DE GRUYTER OPEN

Journal of ELECTRICAL ENGINEERING, VOL 69 (2018), NO2, 93–105

An image encryption scheme employing key related skipping

Jakub Oravec, Ján Turán, Ľuboš Ovseník, Tomáš Huszaník[*]

This paper describes an image encryption algorithm which utilizes chaotic logistic map. Values generated by this map are used in two steps of algorithm which shuffles image pixels and then changes their intensities. Design of the encryption scheme considers possibility of various attacks, such as statistical, differential or phase space reconstruction attacks. Robustness against last mentioned type of attacks is introduced by selective skipping of values generated by the map. This skipping depends on key entered by user. The paper also verifies properties of proposed algorithm by common measures and by set of statistical tests that examine randomness of computed encrypted images. Results are compared with other approaches and they are also briefly discussed.

Keywords: image encryption, iterate skipping, logistic map, phase space reconstruction attack

1 Introduction

Security of various kinds of data has become an interesting topic in last years. Majority of applications could use already proposed solutions, however there are still some specific tasks which require unique approaches. One example of these is the field of image encryption, where properties of images such as large amount of pixels and high values of inter-pixel correlation are taken into account.

These properties and also the design of encryption algorithms could cause insufficient encryption results. For example, let us illustrate the performance of Advanced Encryption Standard (AES) for purposes of image encryption. In its simplest mode of operation (*Electronic CodeBook*, ECB [1]), the plaintext image which is going to be encrypted is divided into blocks with size of 128 bits (16 grayscale pixels). If plaintext image contains blocks with identical pixel intensities on various locations, the encryption produces a set of blocks with corresponding encrypted intensities on these locations. A demonstration of this property is shown in Fig. 1.

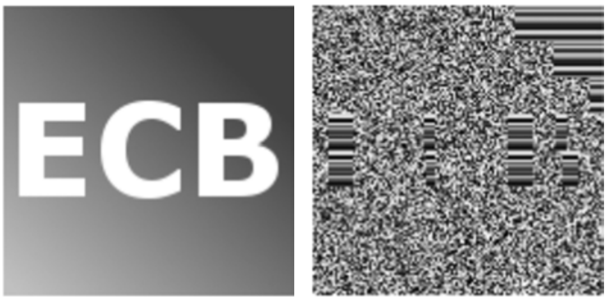


Fig. 1. Illustration of block effect occurring in ECB mode of AES

It could be stated that dark region with small intensity changes in upper right corner of image produces identical blocks in the same corner of encrypted image. Blocks inside white characters, which do not contain background pixels also create visible artefacts. Similar weaknesses could increase possibility of a successful statistical attack.

The dependencies between intensities of image pixels could be introduced by *ciphertext chaining*. This principle is used in another mode of AES, *CipherBlock Chaining* (CBC) [1]. The first block uses so called initialization vector which modifies pixel intensities by means of *eXclusive OR* (XOR). Following blocks use encrypted pixel intensities instead of initialization vector. This approach greatly suppresses possibility of statistical attacks, however, some algorithms could reach similar results with smaller computational complexity.

One of alternative solutions for purposes of image encryption is usage of chaotic encryption algorithms. First proposal of these techniques was described in late 1980s by Matthews [2]. Since then, many approaches were presented including one from Fridrich [3], where the encryption algorithm was divided to two stages. In the first stage, called confusion, the image pixels change their locations in order to minimize correlation between adjacent pixels. After that, diffusion takes place. During this stage, the dependencies between intensities of image pixels are created. Also the values computed by iterating the chaotic map are used for modifying the resulting intensities of pixels in encrypted image.

However, chaotic encryption algorithms have also some drawbacks. First of all, some approaches use two dimensional chaotic maps for pixel shuffling during confusion stage. As majority of these maps was originally proposed for usage on unit square [4, 5], their corresponding discrete versions work only with images which have square

[*] Department of Electronics and Multimedia Communications, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Košice, Slovakia, jakub.oravec@tuke.sk, jan.turan@tuke.sk, lubos.ovsenik@tuke.sk, tomas.huszanik@tuke.sk

DOI: 10.2478/jee-2018–0013, Print (till 2015) ISSN 1335-3632, On-line ISSN 1339-309X © 2018FEI STU

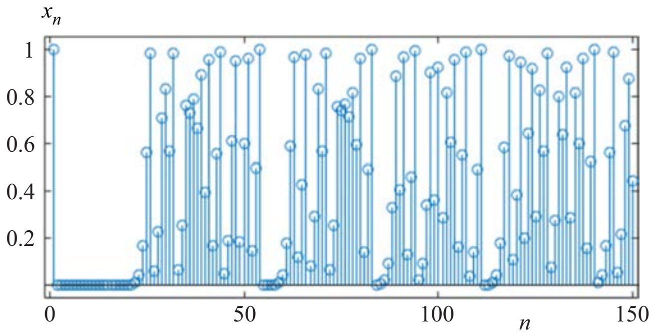


Fig. 2. Transient period occurring during first iterations of logistic map

resolution ($n \times n$ pixels). Also, some maps have so called fixed points, which are locations that do not change their coordinates in following iterations of the map.

Usage of chaotic maps during diffusion could also cause problems when they are employed for generating pseudorandom sequence (PRS) which modifies pixel intensities. In this paper, we would like to point out these problems, describe their importance and propose a solution, which deals with troubles caused by not suitable usage of chaotic maps.

2 Related work

As it was already mentioned, chaotic maps could be used during diffusion stage of image encryption algorithms. One example of these maps is logistic map (LM), which was popularized mainly by May [6]. LM could be given by

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

where $x_n \in [0, 1]$ is value of x in current iteration of map, x_{n+1} is its value in next iteration, n represents sequential number of iteration and $r \in [0, 4)$ denotes parameter of the map. The value of x_0 is called an initial value and set of values of x_n is known as iterations of LM.

The interesting property of LM is its behavior. For values of $r \in [3.56995, 4)$ it is described as chaotic as the following iterations do not clearly converge to any value and they do not show signs of periodicity [2, 6]. Later, the research concerning dynamical degradation of chaos denied these claims and suggested that calculations with finite precision could lead to periodic behavior of chaotic maps [7]. This assumption is used in design phase of image encryption algorithms which use precision at least of 10^{-14} to ensure sufficient values of period.

LM suffers from several drawbacks. Probably the most notable is so-called *transient period*. This period occurs when first iterations are computed. Values of these iterations could be predictable and therefore they could not be used for encryption. For overcoming this problem, some of the first iterations are thrown away. Usually the size of transient period is set as a power of 10 (*eg* 100 or 1000 iterations). An example of this phenomenon is shown in Fig. 2, where first 150 iterations of LM with x_0 set to 0.5 and r equal to $4 - 10^{-14}$ are displayed. The transient period is most notable in first twenty iterations. Patterns occurring in following iterations seem to be similar, but after certain amount of iterations, they start to differ significantly.

Another drawback of chaotic maps used for diffusion of pixel intensities is the possibility of phase space reconstruction. If the attacker is able to extract values of some iterations used during encryption, he may be able to estimate their properties or even values of generated PRS. Fundamental theory for this approach was provided by Takens in [8] and Sauer, Yorke and Casdagli [9]. A brief summary of time delay method is given by Kliková and Raidl [10].

In general, the phase space reconstruction is a problem for all chaotic encryption algorithms where the generated PRS directly influences produced ciphertext. This is the case for proposal by Pareek, Patidar and Sud [11], where a set of chaotic maps was applied depending on chosen key. Iterates of these maps were then used in various calculations. Lui and Wang [12] generate the key stream by piecewise linear map whose iterations are altered by another map. Hence the phase space reconstruction of the second map could reveal values leading to generated key stream. Wang, Teng and Qin [13] directly add values produced by calculations with iterations of chaotic map to image after confusion. Another algorithm which uses generated iterations directly for operation of bitwise XOR is described by Wang and Guo [14].

Several remedies were proposed in order to suppress the possibility of successful phase space reconstruction attack. Zhu [15] tries to improve computational complexity of the attack by combining iterations of four dimensional chaotic map during diffusion. Murillo-Escobar *et al* [16] also rise the complexity of attack by using sum of 50 iterations of LM in one step of their diffusion algorithm. Other approach is presented by Tong *et al* [17], where a new chaotic map based on modified version of LM is designed and its properties are described.

3 Proposed solution

In our approach, we tried to utilize the fact that iterating the map (1) is quite simple operation and hence it is not difficult by means of computational complexity. Therefore, if the amount of computed iterations would be greater, the difference in processing time could be negligible for users. We provide Tab. 1 for illustration of processing times, which are needed for calculations of different amount of iterations N_i by (1). Presented values are arithmetic means of 100 repeated measurements, the value of x_0 was equal to 0.5, parameter r was set to $4 - 10^{-14}$ and first 100 iterations were discarded for overcoming the transient period. Used PC had 2.5 GHz

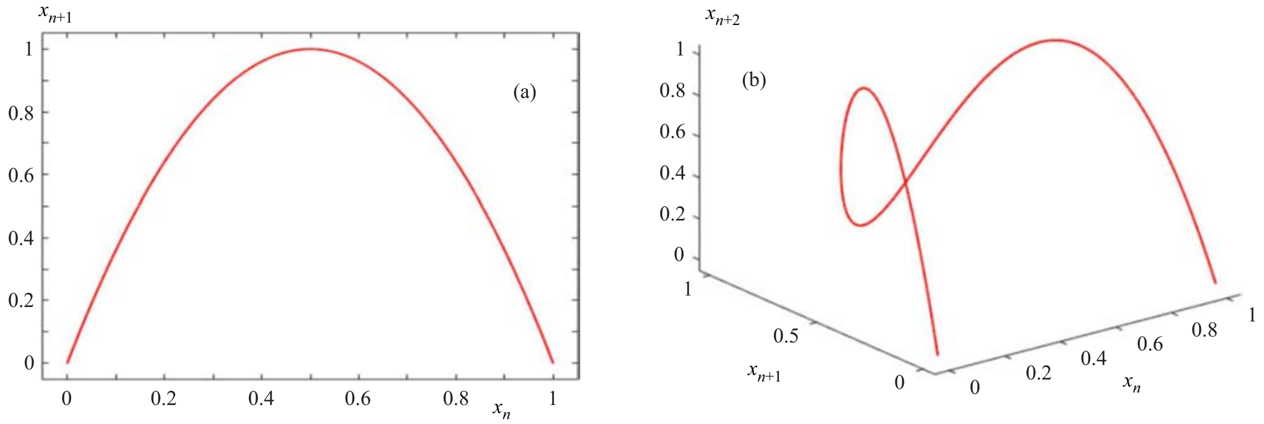


Fig. 3. Poincaré plots for logistic map

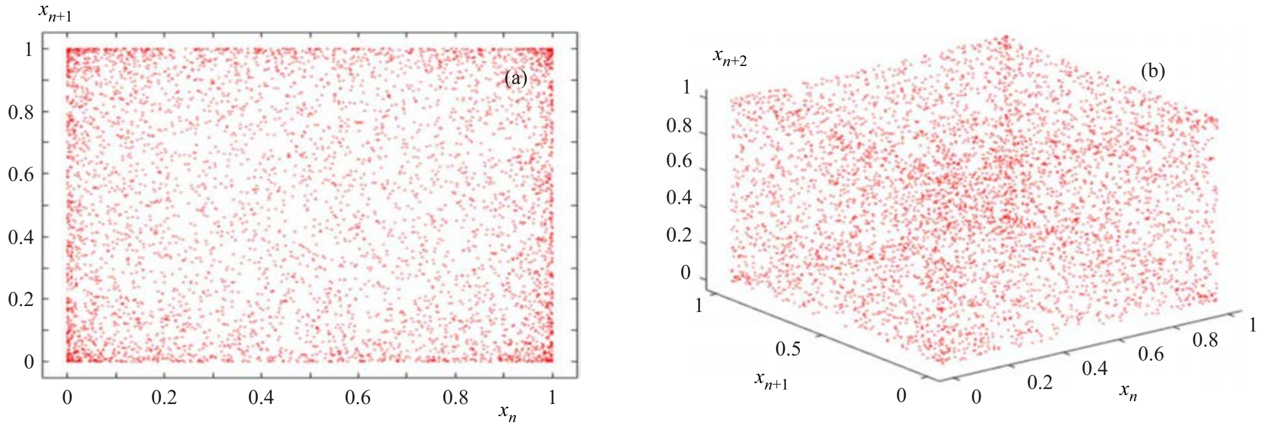


Fig. 4. Examples of Poincaré plots for logistic map with skipped iterations

Table 1. Processing time for iterating logistic map (1): number of iterations N_i and computational time

N_i	time (ms)	N_i	time (ms)
100	0.005	10^5	1.33
1000	0.016	10^6	15.72
10000	0.147	10^7	162.59

CPU, 12 gigabytes of RAM and the simulations were run in MATLAB R2015a.

In our proposal, the greater amount of iterations of LM is used for establishing resistance against phase space reconstruction attack. This is possible by selecting iterations, which would be discarded and hence not used for purposes of encryption. If the attackers try to guess the value of last iterate of LM and use it for reconstruction of the phase space, any skipped iterate would result in inaccurate evaluation of following values. This leads to incorrectly decrypted image.

The properties of map regarding resistance to phase space reconstruction could be shown in a Poincaré plot. These plots use values of consecutive iterations of a map as coordinates of plotted points. Two and three dimensional Poincaré plots for LM given by (1) are shown in Fig. 3. The value of x_0 was fixed at 0.5, while r was equal to $4 - 10^{-14}$. First 100 iterations were discarded to suppress the effects of transient period and subsequent 10 000 iterations were used for the plots.

Poincaré plots of LM explain the difficulty behind evaluation of previously generated iterations. Consider finding out value of only one iterate. The plots show that there are two possible values leading to one value in next iteration of LM. If the amount of evaluated iterations rises to num, the number of possible sequences is then 2^{num}.

As it was already mentioned, our solution discards some of iterations. These are chosen by keys entered by user. Therefore the dependencies between consecutive iterations are broken according to selected keys. This fact is also reflected by Poincaré plots which are shown in Fig. 4. The plots display the same amount of points as those in previous figure. Plots were obtained for PRS used during diffusion stage of encryption which used image *lena*. This image was grayscale with resolution of 512×512 pixels. The selected diffusion key was *0xF0BC940120118097B147F0E35A0EF23E*. Details about encryption algorithm are described in following subparagraphs.

The second set of Poincaré plots shows that phase space reconstruction needs to evaluate more possible

Step 2	pos_{sk} (1:6)	5	10	15	3	2	1
	$size_{sk}$ (1:6)	7	13	15	1	7	6
	pos_{sk} (7:12)	11	13	3	9	2	3
	$size_{sk}$ (7:12)	15	4	2	14	12	1
Step 3	pos_{sk} (1:6)	5	10	15	3	2	1
	$size_{sk}$ (1:6)	7	13	15	1	7	6
	pos_{sk} (7:12)	11	13	$\boxed{19}$	9	$\boxed{18}$	$\boxed{35}$
	$size_{sk}$ (7:12)	15	4	$\boxed{18}$	14	$\boxed{28}$	$\boxed{34}$

Fig. 5. Generation of unique positions and sizes for iterate skipping

points, which lead to a known iterate. The number of these points directly depends on keys selected by user. If these keys would be revealed, the whole algorithm would have similar drawbacks as approaches described in paragraph 2.

3.1 Key related skipping of iterations

The discarding of iterations of LM (1) which depends on selected key is done by following algorithm:

Inputs: 12 or 16 byte key K, length of sequence to be generated len_{seq}, parameter $r \in [3.99\,,4)$

Output: sequence of iterations seq_{sk}

Step 1: The length of key L_{key} is determined. Then elements of key K are decomposed into corresponding binary values K_{bin} by means of little endian ordering scheme (the least significant bit is the first).

Step 2: Binary values of K_{bin} are split and processed to vectors which determine positions where iterations are skipped pos_{sk} and amount of the skipped iterations $size_{sk}$

$$pos_{sk}(i) = 1 + \sum_{b=1}^{4} 2^{b-1} K_{\text{bin}}(i, b)\,, \qquad size_{sk}(i) = 1 + \sum_{b=1}^{4} 2^{b-1} K_{\text{bin}}(i, b+4) \tag{2}$$

where $i = 1, 2, \ldots, L_{key}$.

Step 3: The resulting values of positions for iterate skipping pos_{sk} are scanned for purpose of modifying not unique values. These positions and their corresponding amounts of skipped iterations $size_{sk}$ are treated by consecutive additions of 16 until all the positions are unique.

Step 4: The value of initial amount of skipped iterations $size_{sk}(0)$ is computed. As this number needs to be unique for every possible user key, it is calculated by

$$size_{sk}(0) = \sum_{i=1}^{L_{\text{key}}} i\, pos_{sk}(i)\, size_{sk}(i)\,. \tag{3}$$

Step 5: Total number of necessary iterations num_{it} is calculated as

$$num_{it} = len_{\text{seq}} + size_{sk}(0) + \sum_{i=1}^{L_{\text{key}}} \left\lfloor \frac{len_{\text{seq}}}{pos_{sk}(i)} \right\rfloor size_{sk}(i)\,. \tag{4}$$

Step 6: Equation of LM is iterated for num_{it} times with x_0 equal to 0.5 and parameter r. The resulting sequence is denoted as seq_{it}. At this point, the num_{it} is not increased for suppressing the transient period, because $size_{sk}(0)$ is large enough to be used for this purpose.

Step 7: Each element of sequence seq_{it} is scanned. If its sequential number and one of positions from $pos_{sk}(i)$ produces modulus of 0, successive $size_{sk}(i)$ iterations are discarded and therefore not used during encryption or decryption. If the current element is not skipped, it is copied to sequence seq_{sk}.

Effects of some steps of proposed algorithm are illustrated in Fig. 5. Here, r was set as 3.99210459670052, len_{seq} was equal to 262 144 and the key K with length of 12 bytes had value of *0x64C9EE026150EA3C12D8B102*.

Please note that underlining in Fig. 5 indicates positions and sizes of iterate skips which were changed in order to be unique.

Our proposal encrypts and decrypts images with arbitrary resolution and various colour depths such as 8 or 24 bits/pixel. The encryption uses confusion prior to diffusion and decryption applies these stages in opposite order. Algorithms described in the following are intended for usage during encryption, the decryption alternatives are analogous but have several changes mentioned further.

3.2 Confusion

Our approach performs confusion by following algorithm:

Inputs: 16 byte key K_{conf}, plaintext image P

Output: matrix or vector with values after confusion C_{vec}

Step 1: Height h, width w and colour depth d of the plaintext image P are determined.

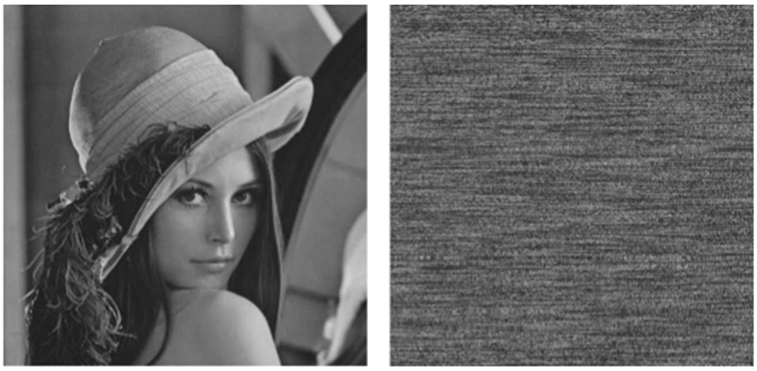
Step 2: The key K_{conf} is divided into two parts. First four bytes are copied into vector r_{add} and the other 12 bytes are passed to vector sk_{conf}. The first vector is used for modification of r_{conf} used for iterating the map (1) and sk_{conf} is utilized for generating positions and sizes of iterate shifting.

Step 3: Value of parameter r_{conf} is calculated as

$$r_{\text{conf}} = 3.99 + \frac{1}{10^2 2^{32}} \sum_{i=1}^{4} 2^{8(i-1)} r_{\text{add}}(i) \tag{5}$$

where constants of 10^2 and 2^{32} are used for ensuring that $r_{\text{conf}} \in [3.99\,,4)$ also after additions of elements of r_{add}. The precision used during these computations is 10^{-14}.

Step 4: Sequence of iterations seq_{conf} is generated by skipping algorithm with parameters $r_{\text{conf}}, sk_{\text{conf}}$ and len_{seq} equal to hw.



plaintext image image after confusion

Fig. 6. Illustration of effects of confusion algorithm

Step 5: Elements of seq_{conf} are split into two subsequences. First h elements are passed to sequence sh_{cols} and the other w elements are copied to sequence sh_{rows}. These sequences are then quantized by (6) and (7). Results of quantization are stored in vectors sh_{colsQ} and sh_{rowsQ}

$$sh_{\mathrm{colsQ}}(i) = \lfloor sh_{\mathrm{cols}}(i)\,(h-1) \rfloor\,, \tag{6}$$

$$sh_{\mathrm{rowsQ}}(j) = \lfloor sh_{\mathrm{rows}}(j)\,(w-1) \rfloor \tag{7}$$

where $i = 1, 2, \dots, h$ and $j = 1, 2, \dots, w$.

Step 6: Pixels in columns of plaintext image P are shuffled by using sh_{colsQ} (8), (9). The shuffling is done in all colour planes. Resulting pixel columns are copied to matrix

$$C_1(l', k, t) = P(l, k, t)\,, \tag{8}$$

$$l' = 1 + \big(l + sh_{colsQ}(k) \pmod{h}\big) \tag{9}$$

where l, l' are old and new indices of image rows, k is index of image columns and $t \in \{1, 2, \dots, d/8\}$ is index of colour plane.

Step 7: Pixels in rows of matrix C_1 are shuffled by using sh_{rowsQ} (10), (11). The shuffling is done in all colour planes. Resulting pixel rows are copied to matrix

$$C_2(l', k', t) = C_1(l', k, t)\,, \tag{10}$$

$$k' = 1 + \big(k + sh_{rowsQ}(l') \pmod{w}\big) \tag{11}$$

where k' is new index of image columns.

Step 8: Elements from matrix C_2 are passed to matrix C_{vec} with $d/8$ rows and hw columns. Hence, each row of this matrix contains shuffled pixels from one colour plane. The pixels are scanned as columns from top left corner to the bottom right one. If plaintext image P is grayscale, the C_{vec} is vector.

Effects of proposed confusion algorithm are displayed in Fig. 6. Used plaintext image *lena* was grayscale with resolution of 512×512 pixels. The value of confusion key was selected as *0x58AFE03564C9EE026150EA3C12D8B102*. For purposes of this example, the vector C_{vec} was reshaped to a matrix with height and width of the plaintext image.

3.3 Diffusion

The diffusion is applied by following these steps:

Inputs: 16 byte key K_{diff}, matrix with values after confusion C_{vec}, height h and width w and colour depth d of the plaintext image P

Output: encrypted image E

Step 1: Sequence of iterations seq_{diff} is generated by skipping algorithm with $r_{\mathrm{diff}} = 4 - 10^{-14}$ and parameters K_{diff} and len_{seq} equal to $\lceil 2hwd/8/7 \rceil$. As the used precision is 10^{-14}, the generated iterations have 14 decimal places.

Step 2: Each element of seq_{diff} is split to seven parts. These parts are stored in matrix $seq_{\mathrm{diff}M}$

$$seq_{\mathrm{diff}M}(j, i) = \big\lfloor 10^{2j} seq_{\mathrm{diff}}(i) \pmod{100} \big\rfloor \tag{12}$$

where $i = 1, 2, \dots, len_{\mathrm{seq}}$ and $j = 1, 2, \dots, 7$.

Step 3: Matrix $seq_{\mathrm{diff}M}$ is rearranged to vector seq_{diffV} by scanning columns of $seq_{\mathrm{diff}M}$ starting with top left corner continuing to the bottom right one. Then elements of seq_{diffV} are quantized and copied to vector seq_{diffQ}

$$seq_{\mathrm{diffQ}}(i) = round\Big(255 \frac{seq_{\mathrm{diffV}}(i) + 1}{100}\Big) \tag{13}$$

where $i = 1, 2, \dots, 7\,len_{\mathrm{seq}}$. The values of seq_{diffV} use addition of one because (12) produces values from set $\{0, 1, \dots, 99\}$.

Step 4: First iteration of diffusion takes place, separately in each colour plane. Results of ciphertext chaining done by (14) are passed to vector ch_1. Then elements of $seq_{\mathrm{diff}Q}$ modify these values by (15). Computed values representing image pixels after first iteration of diffusion are stored in matrix D_{vec}.

$$ch_1(m, i) = \begin{cases} C_{\mathrm{vec}}(t, 1) + C_{\mathrm{vec}}(t, h\,w) \pmod{256} & \text{if } i = 1\,, \\ C_{\mathrm{vec}}(t, i) + ch_1(t, i-1) \pmod{256} & \text{otherwise,} \end{cases} \tag{14}$$

$$D_{\mathrm{vec}}(t, i) = ch_1(t, i) \oplus seq_{\mathrm{diffQ}}\big(i + h\,w\,(t-1)\big) \tag{15}$$

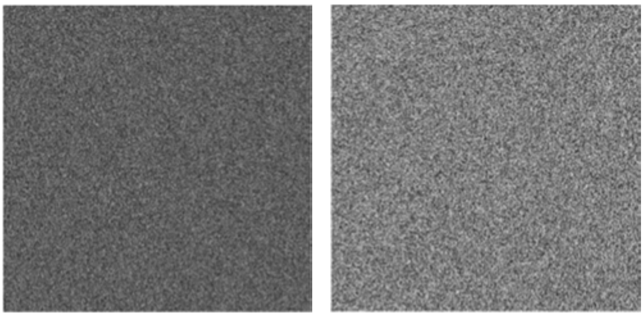


image after confusion

image after diffusion

Fig. 7. Demonstration of properties of diffusion algorithm

where $i = 1, 2, \ldots, 7\,len_{\text{seq}}$, $t \in \{1, 2, \ldots, d/8\}$ is the index of colour plane and $\oplus$ is the operator of bitwise XOR.

Step 5: Second iteration of diffusion takes place, separately in each colour plane. Results of ciphertext chaining done by (16) are passed to vector ch_2. Then elements of seq_{diffQ} modify these values by (17). Computed values representing image pixels after first iteration of diffusion are stored in matrix E_{vec}.

$$ch_2(t,i) = \begin{cases} D_{\text{vec}}(t,1) + D_{\text{vec}}(t,h\,w) \,(\text{mod} 256) & \text{if } i = 1\,, \\ D_{\text{vec}}(t,i) + ch_2(t,i-1) \,(\text{mod} 256) & \text{otherwise,} \end{cases} \tag{16}$$

$$E_{\text{vec}}(t,i) = ch_2(t,i) \oplus seq_{\text{diffQ}}\big(i + h\,w\,(d/8 + (t-1))\big). \tag{17}$$

Step 6: Elements of matrix E_{vec} are passed to matrix E with h rows, w columns and $d/8$ colour planes. The elements are copied separately for each colour plane, they are scanned as columns from top left corner to the bottom right one. Matrix E represents encrypted version of plaintext image P.

Some differences between confusion and diffusion algorithms should be pointed out. Firstly, the diffusion is done in two steps. This is necessary for creating dependencies between higher amount of image pixels. Consider following case: there are two identical plaintext images P_1 and P_2. A difference with minimal size is introduced in one pixel located in bottom right corner of P_2 (intensity of the pixel is either increased or decreased). If encryption algorithm uses only one iteration of diffusion and therefore one iteration of chaining, the difference will not spread to successive image pixels, because it is located at pixel which is scanned as the last. Second iteration of diffusion uses this modified value for computing value representing first pixel, so in this case, the resulting encrypted images E_1 and E_2 should be different.

Secondly, there is a contrast between lengths of key used for iterate skipping during confusion and diffusion. This is caused by different objectives of these stages. While our confusion step shuffles large amount of plaintext image pixels, the diffusion "only" introduces chaotic behavior to results of ciphertext chaining. Therefore, it is important to ensure that sequences generated for confusion are more diverse than those calculated for diffusion.

Finally, diffusion stage utilizes one iterate of LM for modification of seven pixel intensities, while confusion uses one iterate for shifting each column, or row of image pixels. Also the confusion step shuffles image pixels in similar manner in each colour plane, while diffusion uses different values for each colour plane of the processed image.

Impact of our diffusion stage is shown in Fig. 7. The left image was created by applying confusion to plaintext image *lena* (see Fig. 6 for details). Chosen diffusion key had value of *0xF0BC940120118097B147F0E35A0EF23E.*

3.4 Alterations used for decryption

As it was already mentioned, the diffusion is used prior to confusion in case of image decryption. The sequences used for these two operations are created similarly as during encryption. Then the two iterations of diffusion are removed by performing bitwise XOR and inverse version of ciphertext chaining which uses subtraction instead of addition. Confusion stage obtains original positions of image pixels by applying reverse shifts in columns of image before reverse shifts in image rows.

4 Experimental results

All following experiments were performed in MATLAB R2015a running in Windows 10 OS on PC with 2.5 GHz CPU and 12 gigabytes of RAM.

Table 2. Properties of Used Images

property	image			
	Lena	Peppers	Chaos	Black
resolution $w \times h$ (px)	512 ×512	512 ×512	512 ×256	512 ×256
colour depth d (bit/px)	8	24	8	8

Table 3. Used key values

K		value
1	K_{conf}	0x58AFE03564C9EE026150EA3C12D8B102
	K_{diff}	0xF0BC940120118097B147F0E35A0EF23E
2	K_{conf}	0x58AEE03564C9EE026150EA3C12D8B102
	K_{diff}	0xF0BC940120118097B147F0E35A0EF23E
3	K_{conf}	0x58AFE03564C9EE126150EA3C12D8B102
	K_{diff}	0xF0BC940120118097B147F0E35A0EF23E
4	K_{conf}	0x58AFE03564C9EE026150EA3C12D8B102
	K_{diff}f	0xF0BC940120118097B147F0E36A0EF23E

lena peppers
chaos black

Fig. 8. Set of images used during experiments

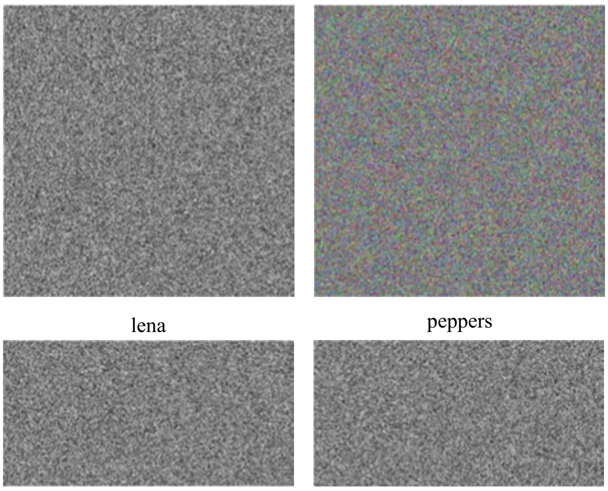

lena peppers
chaos black

Fig. 9. Encrypted versions of used images

The set of used images is shown in Fig. 8. Their parameters are included in Table 2. Keys utilized during experiments are provided in Table 3. Differences between keys are indicated by underlined characters.

The differences between keys were made intentionally on certain places. K_2 produces different r_{conf} than K_1 and usage of K_3 or K_4 instead of K_1 results in different sequences of iterations generated during confusion or diffusion stage.

Encrypted versions of images from Fig. 8 are displayed in Fig. 9. Key used for this purpose was K_1.

4.1 Size of key space and analysis of key sensitivity

Key space includes all possible combinations of keys. Because the confusion key K_{conf} and diffusion key K_{diff} do not depend on each other, the size of key space could be computed as $num_{\text{keys}} = 256^{16} \times 256^{16} = 2^{128+128} = 2^{256}$.

The dependence on used key is illustrated in Fig. 10. Left column of this figure shows results of encryption of *lena* by keys K_1, K_3 and K_4. Right column contains examples of difference image and decryptions by correct and incorrect key.

4.2 Statistical attacks

This kind of attacks tries to search for dependencies between plaintext images and their corresponding encrypted versions. In order to prevent possibility of statistical attacks, the diffusion stage should disturb these relations.

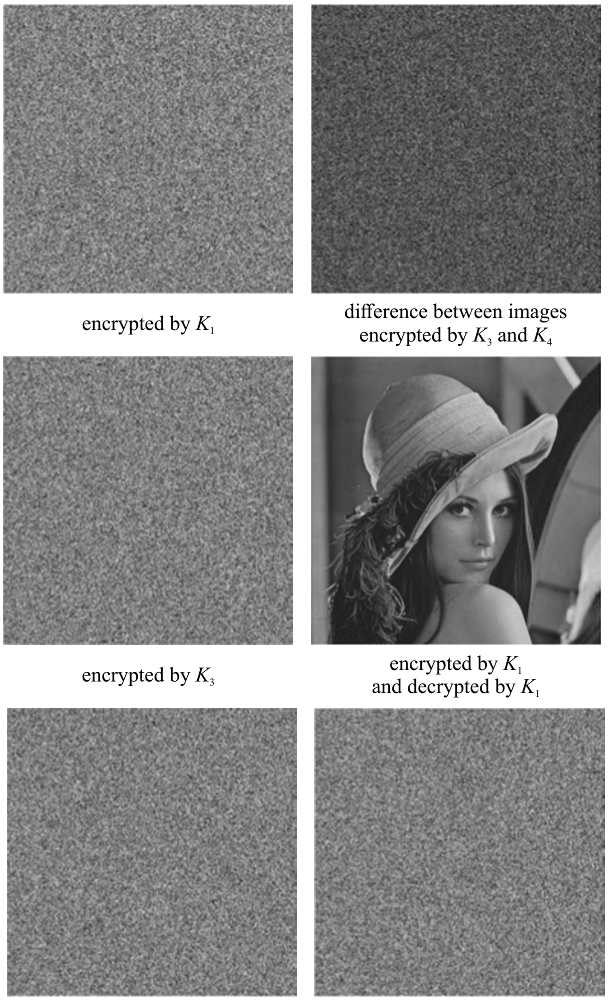

encrypted by K_1 difference between images encrypted by K_3 and K_4
encrypted by K_3 encrypted by K_1 and decrypted by K_1
encrypted by K_4 encrypted by K_1 and decrypted by K_4

Fig. 10. An example of key sensitivity

The resistance against statistical attacks could be evaluated by various measures, such as shape of histograms, correlation diagrams, correlation coefficients or the value of entropy.

Histograms of plaintext image *lena* and its version encrypted by K_1 are shown in Fig. 11. The second histogram does not contain so notable peaks as the histogram of plaintext image, which means that small amount of information usable for statistical attacks could be extracted from these histograms.

Table 4. Computed values of numeric parameters

image and key	ρ_{hor} (-)	ρ_{ver} (-)	ρ_{diag} (-)	H (bits/px)
		plain-text images		
Lena	0.9730	0.9752	0.9562	7.2344
Peppers				
(R)	0.9536	0.9642	0.9446	7.3388
(G)	0.9731	0.9785	0.9609	7.4962
(B)	0.9649	0.9661	0.9339	7.0583
Chaos	0.9748	0.9767	0.9469	2.0180
Black		division by zero		0
		encrypted images		
Lena, K_1	0.0054	0.0017	0.0036	7.9993
Lena, K_2	-0.0019	-0.00169	0.0031	7.9993
Peppers				
(R), K_1	-0.0057	-0.0018	0.0006	7.9993
(R), K_2	-0.0023	-0.0014	-0.0038	7.9993
(G), K_1	0.0053	0.0052	-0.0008	7.9992
(G), K_3	-0.0027	-0.0016	0.0049	7.9992
(B), K_1	-0.0006	-0.0033	0.0025	7.9984
(B), K_4	0.0006	-0.0012	0.0035	7.9986
Chaos				
K_1	0.0000	-0.0041	-0.0035	7.9986
K_3	0.0059	0.0017	0.0014	7.9987
Black				
K_1	-0.0045	-0.0066	0.0001	7.9986
K_4	-0.0008	0.0020	0.0068	7.9987

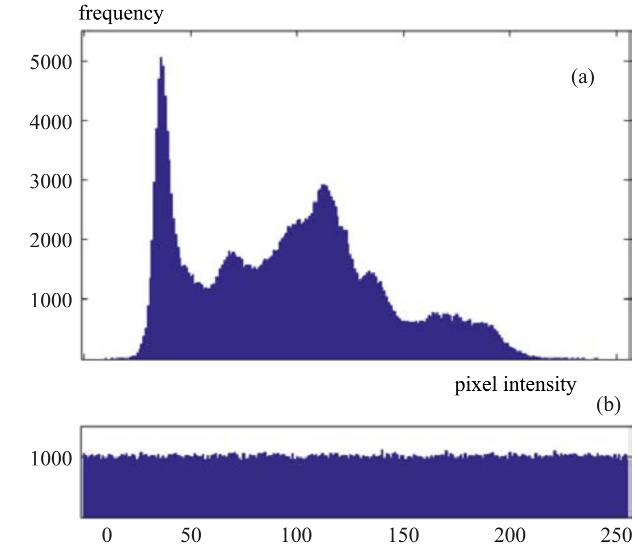


Fig. 11. Comparison of image histograms: (a) – plain-text image, (b) – encrypted by K_1

Correlation diagrams display similarity between adjacent image pixels. There are three possible kinds of pixel adjacency: horizontal, vertical or diagonal one. The diagrams for 1 000 randomly chosen pairs of horizontally adjacent pixels from plaintext image *lena* and its version encrypted by K_1 are shown in Fig. 12. As the intensities of adjacent pixels in plaintext image are similar, the diagram contains many points which lie near line $y = x$. These relations are violated by diffusion stage and therefore the diagram of encrypted image shows more uniform distribution of plotted points.

Correlation coefficients ρ provide numeric values that represent correlation among adjacent pixels in image. They are computed for horizontally, vertically and diagonally adjacent pairs of pixels, separately in each colour plane. Computations of correlation coefficients use following equations

$$\rho = \frac{\text{cov}_{P,E}}{\sqrt{\sigma_P^2\, \sigma_E^2}} \quad [-], \tag{18}$$

$$\text{cov}_{P,E} = \sum_{l=1}^{h}\sum_{k=1}^{w}\big(P(l,k) - \bar{P}\big)\big(E(l,k) - \bar{E}\big), \tag{19}$$

$$\sigma_I^2 = \sum_{l=1}^{h}\sum_{k=1}^{w}\big(I(l,k) - \bar{I}\big)^2, \tag{20}$$

$$\bar{I} = \frac{1}{h\,w}\sum_{l=1}^{h}\sum_{k=1}^{w} I(l,k), \tag{21}$$

where P is plaintext image and E is its corresponding encrypted version, σ_I^2 is dispersion of arbitrary image I, l and k are pixel coordinates, h and w denote image height and width and $\bar{l}$ represents arithmetic mean of pixel intensities in image I.

Entropy H is a measure that was developed for evaluating randomness of data flows [18]. In case of image encryption, the theoretical bound of entropy which would indicate perfectly random image is determined by colour depth of analyzed image. Therefore the bound for grayscale image is 8 bits/pixel. Entropy of true colour images could be calculated separately for each of their colour planes or for all three planes arranged in one sequence of pixels. In this paper, we provide results for each of the colour planes, denoted as R (red), G (green) and B (blue).

Entropy H of image or colour plane with colour depth of 8 bits/pixel could be calculated as

$$H = -\sum_{i=0}^{255} p(in)\, \log_2\big(p(in)\big) \text{ [bits/pixel]} \tag{22}$$

where $p(in)$ denotes frequency of pixels with intensity in.

Values of numeric parameters – correlation coefficients ρ and entropy H calculated for various combinations of plaintext images and keys are included in Tab. 4.

There are several remarks about values presented in Tab. 4. First of all, values of correlation coefficients and entropy differ depending on the used key. Some of these differences are quite big (values for black encrypted by K_1 *vs* values encrypted by K_4). The positive thing is

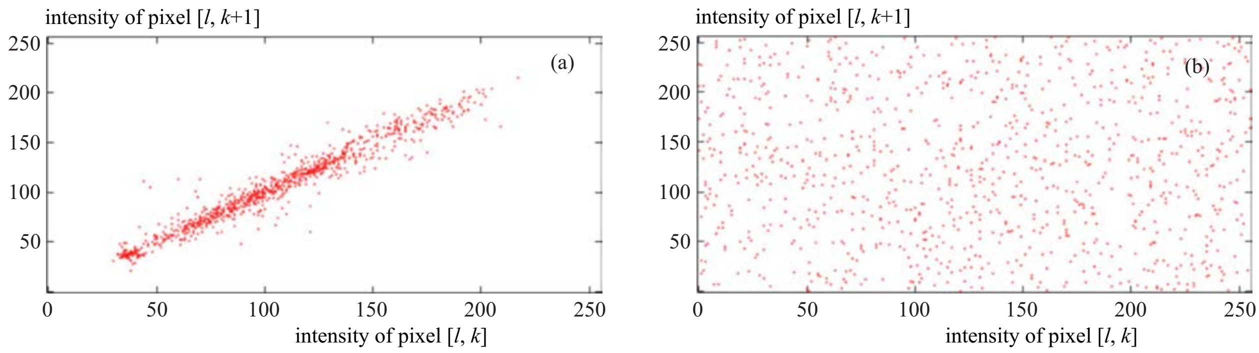


Fig. 12. Correlation diagrams for tested image

flower

tarn

Fig. 13. Thumbnails of images used for randomness evaluation

that absolute values of all obtained correlation coefficients for encrypted images are smaller than 0.01. Secondly, the entropy H is greater than 7.998 bits/pixel for all mentioned combinations.

Table 5. Achieved values of NPCR and UACI

image and key	NPCR	UACI
Lena		
K_1	99.8023	33.4692
K_2	99.7986	33.4747
Peppers		
$(R), K_1$	99.8200	33.4577
$(R), K_2$	99.7791	33.4720
$(G), K_1$	99.8231	33.4632
$(G), K_3$	99.8014	33.4569
$(B), K_1$	99.8196	33.4677
$(B), K_4$	99.7985	33.4763
Chaos		
K_1	99.8156	33.4643
K_3	99.8302	33.4713
Black		
K_1	99.7942	33.4736
K_4	99.7969	33.4789

4.3 Differential attacks

Differential attacks try to reveal properties of encryption algorithm by performing encryption on images with only small differences. A quite usual approach is the introduction of minimal difference between two plaintext images P_1 and P_2 and performing encryption with same parameters which produces images E_1 and E_2. This approach is applied also in two parameters that evaluate performance of encryption algorithms [19]. If used plaintext images are true colour, the difference is made only in one of their colour planes.

First of these parameters is NPCR (*Number of Pixel Change Ratio*) which sums the amount of image pixels with different intensities and divides this sum with total number of pixels in tested image. Hence, NPCR of image or colour plane with colour depth of 8 bits/pixel is given by

$$NPCR = \frac{100}{h\,w}\sum_{l=1}^{h}\sum_{k=1}^{w} Diff_{\mathrm{mat}}(l,k)\ \ [\%]\,, \tag{23}$$

$$Diff_{\mathrm{mat}}(l,k) = \begin{cases} 0 & \text{if } E_1(l,k) = E_2(l,k)\,, \\ 1 & \text{otherwise} \end{cases} \tag{24}$$

where $Diff_{\mathrm{mat}}(l,k)$ is difference matrix.

Second parameter, UACI (*Unified Average Changing Intensity*) considers also the sizes of intensity changes.

Table 6. Evaluation of randomness for two images

image	Flower		Tarn	
test	P-value	proportion	P-value	proportion
Frequency (Monobit)	0.141256	128/128	0.484646	128/128
Frequency within a block ($m = 128$)	0.862344	127/128	0.213309	127/128
Cumulative sums				
(forward)	0.723129	127/128	0.082177	128/128
(reverse)	0.772760	127/128	0.100508	127/128
Runs	0.500934	128/128	0.364146	127/128
Longest run in a block	0.195163	128/128	0.97606	128/128
Binary matrix rank	0.922036	127/128	0.264458	127/128
Spectral (FFT)	0.888137	126/128	0.110952	127/128
Non-overlapping template (first P-value for $m = 9$)	0.299251	127/128	0.551026	127/128
Overlapping template (first P-value for $m = 9$)	0.602458	127/128	0.484646	125/128
Maurer's universal test	0.723129	125/128	0.900104	126/128
Approximate entropy ($m = 10$)	0.819544	126/128	0.875539	127/128
Random excursions (first P-value)	0.559523	80/80	0.350485	82/86
variant (first P-value)	0.484646	80/80	0.509162	84/86
Serial 1 ($m = 16$)	0.468595	126/128	0.756476	127/128
Serial 2 ($m = 16$)	0.452799	128/128	0.468595	128/128
Linear complexity ($M = 500$)	0.804337	124/128	0.922036	128/128

Value of UACI for image or colour plane with colour depth of 8 bits/pixel is computed by

$$UACI = \frac{100}{h\,w} \sum_{l=1}^{h} \sum_{k=1}^{w} \frac{|E_1(l,k) - E_2(l,k)|}{2^d - 1} \quad [\%] \tag{25}$$

where d represents colour depth of image or its colour plane.

Computed values of NPCR and UACI for various combinations of plaintext images and keys are shown in Tab. 5. The values are arithmetic means of 100 repeated measurements.

Results shown in Tab. 5 lead to several conclusions. All values of NPCR exceed 99.77 % and UACI achieves at least level of 33.45 %. These values depend on the used key, as it was notable also for correlation coefficients and entropy.

4.4 Evaluation of randomness

Performance of an encryption algorithm could be measured by several sets of statistical tests. Selected portion of tests used for choosing candidate algorithms for AES [20] was later released as NIST 800-22 suite [21]. We used this set of tests for evaluating performance of proposed algorithm.

The NIST 800-22 suite calculates one P-value for each of performed tests. The encrypted images could be considered as strings of random bits if their corresponding P-values achieve values greater than chosen significance level α. The default value of α is 0.01 [21].

Tests from NIST 800-22 suite require certain amount of data for their calculations. The results of tests are also given as proportion of subsequences which pass the test and total amount of these subsequences (usually set to 100 or 128). NIST recommends length of the subsequences n at least equal to 10^6. Therefore, the minimal size of data used for testing randomness of image encryption algorithm is 100×10^6 bits, which is approximately 11.921 megabytes.

We used two digital images for purposes of testing, both with resolution of $3\,264 \times 1\,836$ pixels and colour depth of 24 bits/pixel. Thumbnails of these images are shown in Fig. 13. The images were encrypted by K_1.

For enabling possibility of repeatable experiments, we present uploaded plaintext images as [22]. These images are provided free of charge for scientific purposes. Binary strings of required length were obtained from encrypted versions of images by decomposition to their colour planes. Pixel intensities of these planes were then scanned as columns of pixels starting from top left corner and continuing to the bottom right one. Finally, each of

Table 7. Time elapsed during operations and corresponding processing speeds

operation	encryption		decryption	
image and key	time (ms)	v_{proc} (MB/s)	time (ms)	v_{proc} (MB/s)
Lena				
K_1	235.008	1.06379	245.545	1.01814
K_2	234.496	1.06611	245.989	1.01631
Peppers				
K_1	711.998	1.05337	739.847	1.01372
K_2	713.108	1.05174	741.713	1.01117
Chaos				
K_1	115.636	1.08098	122.399	1.02125
K_3	116.169	1.07602	122.139	1.02343
Black				
K_1	115.636	1.08535	122.067	1.02402
K_4	116.169	1.07981	122.239	1.02259

these intensities was decomposed to 8 bits by little endian ordering scheme.

Resulting values for tests from NIST 800-22 suite are presented in Tab. 6. The number of subsequences was set to 128, their length n was selected as 10^6, and the tests used default values of auxiliary variables (m and M). The value of significance level α was left unchanged at 0.01, so the encrypted image passes tests if P-values are greater than 0.01. The proportions of subsequences which are needed for passing the test are calculated either as 123 of 128 and 76 of 80 or 82 of 86 subsequences for Random excursions tests. More details about used tests and subsequences which are skipped in some of tests are provided in [21].

Results in Tab. 6 illustrate possible differences between P-values of various encrypted images. However, the pass proportion and therefore also the result of testing stays similar in case of both used images.

4.5 Computational Complexity

The amount of time needed for operations with our algorithm was tested on PC with 2.5 GHz CPU and 12 gigabytes of RAM which ran MATLAB R2015a under Windows 10 OS. Values given in Tab. 7 are arithmetic means of 100 repeated measurements.

The processing speed v_{proc} is computed simply as size of image divided by processing time t_{proc} given in seconds

$$v_{\text{proc}} = \frac{h\,w\,d}{2^{20}\,t_{\text{proc}}} \quad (MB/s) \tag{26}$$

where h, w and d are height, width and colour depth of processed image. Constant of 2^{20} represents amount of bits in one megabyte of data (1 megabyte = 1 048 576 bits).

Processing speeds presented in Tab. 7 illustrate impact of chosen combination of image and key on the duration of performed operation. All encryption speeds were higher than 1.05 MB/s, while decryption speeds were slightly slower, they barely rose above value of 1 MB/s. The results also imply that higher resolution causes decrease of the processing speeds.

4.6 Comparison with other approaches

Comparison of numeric results of various proposals is quite difficult task mainly due to different images used during experiments. Also some papers provide results separately for each colour plane of true colour images, while other papers contain arithmetic means of computed measures. Calculated processing speeds depend on configuration of used PC and also on environment where the experiments were conducted.

Table 8. Comparison of numeric parameters

reference or image	colour plane	ρ_{hor} (-)	ρ_{ver} (-)	ρ_{diag} (-)	H (bits/px)
Lena256-8	-	0.000797	-0.000823	0.001173	7.99701
ref. [15]	-	0.001005	-0.000850	0.000897	7.99770
Lena 256-24	R	-0.003044	-0.001364	0.002850	7.99729
	G	-0.00287	-0.004494	0.006668	7.99691
	B	-0.003423	-0.003736	-0.005474	7.99685
ref. [17]	R	0.003153	0.009650	0.017851	7.999987*
	G	0.006827	0.004191	0.012953	
	B	0.001469	0.003467	0.009146	
Lena 512-24	R	-0.001531	-0.001737	0.003448	7.99913
	G	0.003782	-0.001290	-0.001861	7.99938
	B	-0.001807	-0.000638	0.001907	7.99932
ref. [16]	R	0.0135	unknown		7.99740
	G	-0.0835			7.99750
	B	-0.017			7.99690

* mean for all colour planes

Table 9. Values of NPCR, UACI and processing speeds

reference or image	colour plane	NPCR (%)	UACI (%)	v_{proc} (MB/s)
Lena 256-8	-	99.7984	33.4820	1.075115
ref. [15]	-	99.6094	33.4635	1.953125
Lena 256-24	R	99.8044	33.4744	1.072387
	G	99.7903	33.4818	
	B	99.7865	33.4642	
ref. [17]	R	99.4048	33.2243	0.451431
	G	99.3306	33.3345	
	B	99.4845	33.2893	
Lena 512-24	R	99.8058	33.4621	1.056691
	G	99.8169	33.4477	
	B	99.8168	33.4810	
ref. [16]	R	99.63	33.31	3.083882
	G	99.60	33.34	
	B	99.61	33.43	

For comparing results of our proposal, we encrypted different versions of plaintext image *lena* utilized by other authors. Grayscale version with resolution of 256 × 256 pixels used in [15] is denoted in this subparagraphr as *lena*256-8. True colour image with the same resolution was used in [17] and here it is called *lena*256-24. Paper [16] used true colour version with resolution of 512 × 512 pixels, which is marked here as *lena*512-24. All encryptions done by our algorithm utilized key K_1.

Correlation coefficients ρ and values of entropy H for algorithms described in [15–17] and for the proposed approach are included in Tab. 8. The correlation coefficients were calculated separately for each colour plane (R stands for red, G for green and B for blue colour plane) of encrypted images.

The results show that our solution obtains correlation coefficients comparable with those yielded by [15, 16]. In some cases they are slightly better (colour planes of *lena*256-24 *vs* those of [17]). Values of entropy are better when compared with [17], but otherwise they are slightly worse.

Values of NPCR, UACI and computed processing speeds of encryption v_{proc} are shown in Tab. 9. While NPCR and UACI are calculated separately for each colour plane, processing speed measures encryption of all three colour planes.

It could be concluded that our solution reaches values of NPCR and UACI which are comparable to those achieved by proposal described in [15]. Comparison with the other two approaches shows that our solution is better by means of NPCR and UACI values. On the other hand, our proposal is the second slowest of mentioned algorithms. Approach from [16] is three times faster, but the solution described in [17] is still more than two times slower than our proposal.

5 Conclusion

This paper proposed chaotic image encryption algorithm which utilizes skipping of iterations for providing higher robustness against phase space reconstruction attack. As the iterate skipping depends solely on key chosen by user, the details of this process are clear only when key is revealed.

Described encryption algorithm could be used in various applications which require secure transmission of image data. Some examples of those include sending pictures over internet, safe storage of image data, or providing higher amount of security in steganographic systems which use rather simple data hiding techniques [23, 24].

Properties of our proposal such as robustness against statistical or differential attacks were verified by experiments with set of four images and four keys. The set of images contained true colour and grayscale images with square and not square resolutions. Randomness of generated encrypted images was evaluated by tests from NIST 800-22 suite. Because all resulting P-values were higher than required significance level and also all proportions reached desired values, the tests were succesfully passed. Comparison with results achieved by other researchers shows that our algorithm reaches similar or even slightly better values of correlation coefficients and entropy, while the values of NPCR and UACI are usually clearly better. These advantages are balanced by slower processing speed, although one of the other approaches is even slower.

The processing speed as the biggest drawback of our proposal could be improved in future. However, as the described approach would always require more generated iterations than other algorithms, it may not reach the values achieved by the fastest designs. Still, there are some available options which would help to increase the processing speed, such as more effective usage of computer memory or enabling higher performance of computations. First approach merges some algorithm steps, *eg* generated iterations do not need to be stored in memory before their quantization, instead they could be passed to next iteration and could be stored in memory after the quantization. Second way is much more uncertain, because the iterations with more decimal places could be split into more values, but the higher precision usually causes increased computational complexity. Therefore careful considerations need to be made for improving processing speed.

Acknowledgements

This work was supported by following research grants: KEGA 023TUKE-4/2017 and VEGA 1/0772/17.

References

[1] ST Special Publication 800-38A: Recommendation for Block Cipher Modes of Operation: Methods and Techniques. 2001 66 p.[Online] Cited 2017-10-19.Available at: http://nvlpubs.nist.gov /nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf.

[2] R. Matthews, "On the Derivation of a 'Chaotic' Encryption Algorithm", *Cryptologia*, 1989, vol. 8, no. 6, pp. 29-41.ISSN: 0161-1194.DOI: 10.1080/0161-118991863745.

[3] J. Fridrich, "Symmetric Ciphers based on Two-Dimensional Chaotic Maps", *International Journal of Bifurcation and Chaos*, 1998, vol. 8, no. 6, pp. 1259–1284.ISSN: 0218-1274. DOI: 10.1142/S021812749800098X.

[4] V. I. Arnold and A. Avez, *Ergodic Problems of Classical Mechanics*, New Jersey: W.A.Benjamin, 1968.

[5] B. Chirikov, *Research Concerning the Theory of Non-Linear Resonance and Stochasticity*, Geneva: CERN, 1971.

[6] R. May, "Simple Mathematical Models with Very Complicated Dynamics", *Nature*, 1976, vol. 261, no. 5560, pp. 459–467.ISSN: 0028-0836.DOI: 10.1038/261459a0.

[7] S. Li, G. Chen and X. Mou, "On the Dynamical Degradation of Digital Piecewise Linear Chaotic Maps", *International Journal of Bifurcation and Chaos*, 2005, vol. 15, no. 10, pp. 3119-3151. ISSN: 0218-1274. DOI: 10.1142/S0218127405014052.

[8] F. Takens, "On the Numerical Determination of the Dimension of an Attractor, Dynamical Systems and Bifurcations", *Lecture Notes Mathematics* vol. 1125, B. L. J.Braaksma, H. W. Broer, F. Takens (Eds), Berlin, Heidelberg: Springer, 1985, pp. 99–106. ISBN: 978-3-540-15233-0.DOI: 10.1007/BFb0075637.

[9] T. Sauer, J. A. Yorke and M. Casdagli, "Embedology", *Journal of Statistical Physics*, 1991, vol. 65, no. 3-4, pp. 579–616. ISSN: 0022-4715.DOI: 10.1007/BF01053745.

[10] B. Kliková and A. Raidl, "Reconstruction of Phase Space of Dynamical Systems using Method of time Delay", *Proceedings of WDS'11. Praha (Czech Republic)*, 2011, pp. 83–87.ISBN: 978-8-073-78186-6.

[11] N. K. Pareek, V. Patidar and K. K. Sud, "Cryptography using Multiple One-Dimensional Chaotic Maps", *Communications Nonlinear Science and Numerical Simulation*, 2005, vol. 10, no. 7, pp. 715–723. ISSN: 1007-5704.DOI: 10.1016/j.cnsns.2004.03.006.

[12] H. Liu and X. Wang, "Colour Image Encryption based on One-Time Keys and Robust Chaotic Maps", *Computers & Mathematics with Applications*, 2010, vol. 59, no. 10, pp. 3320–3327. ISSN: 0898-1221.DOI: 10.1016/j.camwa.2010.03.017.

[13] X. Wang, L. Teng and X. Qin, "A Novel Colour Image Encryption Algorithm based on Chaos", *Signal Processing*, 2012, vol. 92, no. 4, pp. 1101–1108. ISSN: 0165-1684. DOI: 10.1016/j. sigpro.2011.10.023.

[14] X. Wang and K. Guo, "A New Image Alternate Encryption Algorithm based on Chaotic Map", *Nonlinear Dynamics*, 2014, vol. 76, no. 4, pp. 1943–1950. ISSN: 0924-090X.DOI: 10.1007/s11071-014-1259-7.

[15] C. Zhu, "A Novel Image Encryption Scheme based on Improved Hyper-Chaotic Sequences", *Optics Communications*, 2012, vol. 285, no. 1, pp. 29–37.ISSN: 0030-4018. DOI: 10.1016/j. optcom.2011.08.079.

[16] M. A. Murillo-Escobar, C. Cruz-Hernández, F. Abundiz-Pérez *et al*, "A RGB Image Encryption Algorithm based on Total Plain Image Characteristics and Chaos", *Signal Processing*, 2015, vol. 109, no. C, pp. 119–131. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2014.10.033.

[17] X. J. Tong, Z. Wang, M. Zhang *et al*, "An Image Encryption Algorithm based on the Perturbed High-Dimensional Chaotic Map", *Nonlinear Dynamics*, 2015, vol. 80, no. 3, pp. 1493–1508. ISSN: 0924-090X. DOI: 10.1007/s11071-015-1957-9.

[18] C. E. Shannon, "A Mathematical Theory of Communication", *The Bell System Technical Journal*, 1948, vol. 27, no. 3, pp. 379–423. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[19] Y. Wu, J. Noonan and S. Agaian, "NPCR and UACI Randomness Tests for Image Encryption", *Journal of Selected Areas Telecommunications*, 2011, vol. 2, no. 4, pp. 31–38. ISSN: 1925-2676.

[20] ST Internal Report 6483: Randomness Testing of the Advanced Encryption Standard Finalist Candidates, 2000 15 p. [Online] Cited 2017-11-02. Available at: http://ws680.nist.gov/publication/get_pdf.cf?pub_id/alpha 151216.

[21] ST Special Publication 800-22 Rev.1a: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. 2001 131 p. [Online] Cited 2017-11-02. Available at: http:http://nvlpubs.nist.gov/nistpubs/Legacy/SP/ nistspecialpublication800-22r1a.pdf.

[22] High Resolution Images used for an Experiment with NIST 800-22 Suite, [Online] Cited 2017-11-13. Available at: http://imagedb.wz.sk/ sets/flowerandtarn.zip.

[23] J. Oravec and J. Turán, "Substitution Steganography with Security Improved by Chaotic Image Encryption", Proceedings of Informatics 2017. Poprad (Slovakia), pp. 284–288. ISBN: 978-1-538-60888-3 DOI: 10.1109/Informatics.2017.8327261.

[24] V. Hajduk, M. Broda, O. Kováč and D. Levický, "Image Steganography with QR Code and Cryptography", *Proceedings of Radioelektronika* 2016. Košice (Slovakia), pp. 350–353. ISBN: 978-15-0901-673-0. DOI: 10.1109/RADIOELEK.2016.7477370.

Received 22 November 2017

Jakub Oravec received his Ing (MSc) degree from Department of Electronics and Multimedia Communications (DEMC), Technical University of Košice in 2015. Now he continues as PhD student. His research interests include image encryption, steganography and digital image processing.

Ján Turán received Ing (MSc) 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 Charles University, Prague, Czech Republic, in 1980. He received a CSc (PhD) and DrSc (DSc) degrees in radioelectronics from 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.

Ľuboš Ovseník received his Ing (MSc) and PhD degrees from DEMC, Technical University of Košice in 1990 and 2002. He works at the Technical University of Košice, currently as an Associate Professor. His research interests include photonics, fiber optic communication systems and sensor networks.

Tomáš Huszaník received Ing (MSc) degree in 2017 at DEMC, Technical University of Košice. Since September 2017 he continues as PhD student. His research interests include all optical networks and degradation mechanisms in all optical WDM systems.