An Image Encryption Algorithm with Total Diffusion

Jakub Oravec

Department of Electronics and Multimedia
Communications, Faculty of Electrical Engineering and
Informatics
Technical University Košice
Košice, Slovakia
jakub.oravec@tuke.sk

Ján Turán, Ľuboš Ovseník

Department of Electronics and Multimedia
Communications, Faculty of Electrical Engineering and
Informatics
Technical University Košice
Košice, Slovakia
jan.turan@tuke.sk, lubos.ovsenik@tuke.sk

Abstract—This paper proposes an image encryption algorithm which uses four scans of an image during the diffusion stage in order to achieve total diffusion between intensities of image pixels. The condition of total diffusion is fulfilled by a suitable combination of techniques of ciphertext chaining and plaintext related diffusion. The proposed encryption algorithm uses two stages which utilize chaotic logistic map for generation of pseudo-random sequences. The paper also briefly analyzes approaches described by other researchers and evaluates experimental results of the proposed solution by means of commonly used measures. Properties of our proposal regarding modifications of plain images prior to encryption or modifications of encrypted images prior to decryption are illustrated by two additional experiments. The obtained numeric results are compared with those achieved by other proposals and briefly discussed.

Keywords—ciphertext chaining, image encryption, logistic map, plaintext related diffusion, total diffusion

I. INTRODUCTION

An idea of encryption algorithms designed specifically for encryption of images dates back to the late 1990s when Fridrich described one of the first image encryption techniques [1]. Fridrich's approach introduced two important principles of image encryption algorithms. The first one was an architecture of the algorithm, consisting of two stages. The first stage – *confusion* rearranges image pixels as adjacent image pixels are prone to have high values of correlation. The second stage – *diffusion* computes new pixel intensities which is a crucial operation for establishing resistance against statistical and differential attacks. These two stages were described already in 1949 by Shannon [2].

The second principle proposed in Fridrich's paper is nowadays known as *ciphertext chaining*. Ciphertext chaining is utilized for establishing dependencies between intensities of image pixels. Consider following case: a user tries to encrypt two plain images which differ only in an intensity of one pixel. If the used encryption algorithm would not use ciphertext chaining, the resulting two encrypted images would also differ only in the intensity of one pixel. In the case that used encryption algorithm utilizes ciphertext chaining, the difference

in the intensity of one plain image pixel should spread to multiple pixel intensities in the encrypted images. Ideally, it should affect all pixel intensities in the encrypted images.

The earlier approaches, such as Fridrich's, relied on the ciphertext chaining done during several rounds of encryption. Therefore, the scanning pattern of pixels was not important as at least one of the affected pixels should be moved to the beginning of used scanning pattern in next round of confusion.

The computational difficulty of image encryption algorithms was improved mainly by reducing the amount of rounds performed during encryption. Therefore, it was hard to design suitable scanning pattern, i. e. one that establishes dependencies also between already scanned pixels and pixels which are only going to be scanned. This situation could be solved by using two rounds of ciphertext chaining, with the second round using the last pixel intensity from the first round for the chaining of the first pixel intensity [3]. Fridrich's scheme of ciphertext chaining and the described two round chaining scheme are illustrated in Fig. 1.

Fig. 1. A comparison of two ciphertext chaining schemes

However, also the second ciphertext chaining scheme has some drawbacks. It was pointed out by Solak et al. in 2010 that the combinations of currently scanned and previous pixel intensities establish the dependencies in whole image during encryption, but the operations done during decryption depend only on two intensities for each encrypted pixel [4]. Therefore, if the encrypted image is slightly modified, the decryption with correct key produces plain image with only small differences. This situation is not desired, as it shows that images similar to plain image could be obtained from various encrypted images.

Solak et al. recommended a technique called *plaintext related diffusion* for suppressing above mentioned drawback. Plaintext related diffusion tries to modify currently scanned pixel intensities with intensities of plain image pixels. Because decryption algorithms usually obtain intensities of plain image pixels sequentially (similarly as the ciphertext chaining during encryption), even small differences in obtained pixel intensities could cause big changes in intensities of the following pixels.

In this paper, we aim to combine techniques of ciphertext chaining and plaintext related diffusion for establishing diffusion as it was described by Shannon in [2]: the minimal possible difference in a pair of images (intensity of one pixel is incremented or decremented by one intensity level) should affect all other pixel intensities. Therefore, we denote this type of diffusion for purposes of this paper as a *total diffusion*.

The rest of the paper is organized as follows: the second chapter briefly reviews techniques utilized in other papers. The third chapter provides description of steps used in the proposed image encryption algorithm. The fourth chapter analyzes obtained experimental results. The fifth chapter provides a comparison with other published approaches. The last, sixth chapter concludes the paper with a summary of properties, advantages and disadvantages of the proposed solution.

II. LITERATURE REVIEW

Several techniques used for plaintext related diffusion have been developed since Solak's recommendation. One of the first approaches was described by Kanso and Ghebleh in 2011 [5]. In their algorithm, the amount of iterations of three dimensional Arnold's cat map is affected by intensities of currently processed plain image pixels.

In 2014, Guanghui et al. proposed an image encryption algorithm, which modifies the used key according to the plain image [6]. Authors claim that this technique should suppress the relations between successive iterates generated by chaotic maps. Therefore it should reduce the possibility of phase space reconstruction attacks. However, since today, there are not any particular examples of these attacks, only some ideas [7].

Another image encryption algorithm with plaintext related diffusion was described in 2015 by Murillo-Escobar et al. [8]. In their approach, the sum of pixel intensities is used for calculation of an additional parameter for logistic map. However the pixel intensities are combined by a sum which is not a unique operation as multiple plain images could produce the same sum.

A paper by Liu and Miao from 2016 presented a technique that changes the values of map parameters during computation of iterations [9]. Then, the intensities of plain image pixels are used in the diffusion stage for selection of new intensities from

created code book. Probably the biggest drawback of their approach is the difficult way of generation of the code book (it is a large permutation).

Our previous work includes designs of some image encryption algorithms. The effects done by technique of total diffusion could be evaluated by comparison of numeric results with paper [10] which does not utilize this technique.

III. PROPOSED SOLUTION

The proposed image encryption algorithm tries to combine advantages of ciphertext chaining and plaintext related diffusion. The first mentioned technique is used sequentially during encryption and it helps to spread even small differences between intensities of plain image pixels to all pixels. However, it is used in a parallel way during decryption (each pixel uses the previous pixel in the scanning pattern).

The combination of values computed by ciphertext chaining with plain image pixels is implemented in an opposite way (it is parallel during encryption and sequential during decryption). This solution introduces the relations between the intensities of pixels in decrypted versions of images.

Therefore, not only two slightly modified versions of one plain image produce two different encrypted images, but also two slightly modified versions of one encrypted image result in totally different decrypted images.

Our algorithm uses images with arbitrary resolution (including non-square images) and color depths of 8 bits (grayscale) or 24 bits (true color) per pixel. Steps of algorithm could be easily modified for color depths that are integer multiples of 8 bits per pixel (e. g. 32 bits for transparent images). The length of utilized key is 16 bytes, it is entered as a string of hexadecimal characters.

Operations done during encryption are described in following three subchapters. The differences done during decryption are summarized in subchapter D.

A. Pseudo-random Sequence Generation and Key Diffusion

Our algorithm uses totally 10 pseudo-random (PR) sequences for operations done during confusion or diffusion stage. The elements of these sequences are generated according to a key entered by user. For ensuring that even the smallest possible change of the key (value of one key element shifted by one level) would affect all key elements and therefore all stages of the proposed image encryption algorithm, we utilize a technique called *key diffusion*.

The PR sequences are generated by means of *logistic map* (LM), popularized mainly by May [11]. Its equation could be expressed as (1):

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

where n is iteration number, $x_n \in [0, 1]$, $x_0 = 0.5$ is an initial condition and $r \in [0, 4)$.

The desired chaotic behavior of LM is achieved for $r \geq 3.56995$ as it could be seen on a bifurcation diagram plotted in Fig. 2. The illustrated case used initial condition $x_0 = 0.5$, 1,000 iterates of LM, each one was generated after *transient period* with length of 1,000 iterates.

Fig. 2. An example of a bifurcation diagram for the logistic map

The transient period denotes amount of iterates which are discarded for ensuring that generated values could not be predicted. The iterates of chaotic maps produced after transient period should have sufficient “chaotic behavior” as described by Matthews – they should not be periodic, and they should not converge to certain value [12].

The 10 PR sequences are generated with usage of four parameters r_1, r_2, r_3 and r_4 which are obtained from the entered key K by (2):

$$r_i = 3.99 + \frac{i-1}{4 \cdot 10^2} + \frac{K_{1+4(i-1):4i}}{10^4(1+2^{32})}, \quad (2)$$

where $K_{a:b}$ denotes used portion of 16 byte key K (from a^{th} byte to b^{th} byte) and constants of $(i-1)/(4 \cdot 10^2)$ and $10^4(1+2^{32})$ ensures that resulting $r_i \in [3.99 + (i-1)/(4 \cdot 10^2), 3.99 + i/(4 \cdot 10^2)]$.

Two sequences used in confusion stage – seq_{col} and seq_{row} utilize four various values of r_i , which are switched during the transient period. As this period has length of 1,000 iterates, each of used r_i are utilized for 250 iterations during the transient period. The last used r_i is then used in the following iterations of the LM (1). This switching pattern is illustrated in TABLE I.

TABLE I. SWITCHING PATTERN USED FOR GENERATION OF SEQUENCES UTILIZED DURING THE CONFUSION STAGE

sequence / used r_i	seq_{col}	seq_{row}
transient period		
iterations 1 to 250	r_2	r_3
iterations 251 to 500	r_3	r_4
iterations 501 to 750	r_4	r_1
iterations 751 to 1,000	r_1	r_2
sequence elements		
all iterations	r_1	r_2

Sequences seq_{col} and seq_{row} use initial condition $x_0 = 0.5$. For an image with h rows, w columns and d color planes, length of these sequences is $w \cdot d$ and h iterates, respectively.

Because the LM (1) produces iterates $x_n \in [0, 1]$, these need to be quantized before their usage in stages of an image encryption algorithm. Sequences seq_{col}' and seq_{row}' are computed by quantization of sequences seq_{col} and seq_{row} by set of equations (3):

$$\begin{aligned} seq_{col}'(i) &= \lfloor h \cdot seq_{col}(i) \rfloor \\ seq_{row}'(j) &= \lfloor w \cdot d \cdot seq_{row}(j) \rfloor \end{aligned} \quad (3)$$

where $i = 1, 2, \dots, w \cdot d$ and $j = 1, 2, \dots, h$ are indices of sequence elements and h, w and d denote height, width and number of color planes of an image, respectively.

As each sequence utilizes all four values r_i during the transient period, the resulting iterates depend on all four key portions, therefore the generated sequences are affected even by small modifications of the used key K .

The other 8 PR sequences are used for producing substitution vectors utilized during the diffusion stage. The vectors are obtained from sequences generated by several LMs, all of them use initial value x_0 of 0.5, and their lengths are 10,000 iterates, respectively. These sequences also use four different values of r_i during the transient period and during the generation of sequence elements. The switching pattern for the sequences utilized during the diffusion stage is shown in TABLE II.

TABLE II. SWITCHING PATTERN USED FOR GENERATION OF SEQUENCES UTILIZED DURING THE DIFFUSION STAGE

sequence	used r_i	sequence	used r_i
seq_{dif1}	r_4, r_1, r_2, r_3, r_3	seq_{dif5}	r_1, r_2, r_4, r_3, r_3
seq_{dif2}	r_2, r_4, r_1, r_3, r_3	seq_{dif6}	r_4, r_2, r_1, r_3, r_3
seq_{dif3}	r_1, r_2, r_3, r_4, r_4	seq_{dif7}	r_2, r_3, r_1, r_4, r_4
seq_{dif4}	r_3, r_1, r_2, r_4, r_4	seq_{dif8}	r_3, r_2, r_1, r_4, r_4

The generation of 8 substitution vectors vec_{sub1} to vec_{sub8} starts with quantization of corresponding sequences by (4):

$$seq_{dif(i)}'(j) = \left\lfloor 256 \cdot seq_{dif(i)}(j) \right\rfloor, \quad (4)$$

where $i = 1, 2, \dots, 8$ is an index of substitution vector and $j = 1, 2, \dots, 10^4$ is an index of sequence elements.

Then, the substitution vectors are sequentially filled with integers from set $\{0, 1, \dots, 255\}$. The quantized values of iterates are filled to the vectors only at the first occurrence of the quantized values. Therefore this step produces 8 substitution vectors which are permutations of set $\{0, 1, \dots, 255\}$. An example of two partially filled substitution vectors is presented in TABLE III.

TABLE III. AN EXAMPLE OF TWO SUBSTITUTION VECTORS

sequence / vector	elements
seq_{dif1}'	202, 55, 67, 202, 33, 89, 202, 141, ...
vec_{sub1}	202, 55, 67, 33, 89, 141, ...
seq_{dif2}'	96, 167, 200, 122, 0, 96, 187, 65, ...
vec_{sub2}	96, 167, 200, 122, 0, 187, 65, ...

In the case that any of the substitution vectors is not fully filled with integers from set $\{0, 1, \dots, 255\}$, the unused integers are appended to the end of the vector.

B. Confusion

This stage which rearranges image pixels starts with merging of color planes of plain image P . In the case that P is true color and has h rows, w columns, the first step of confusion stage produces an image C_0 with h rows and $3 \cdot w$ columns. The leftmost w columns are made of red (R) color plane, the middle w columns are taken from green (G) color plane and the rightmost w columns are created from blue (B) color plane. In the case of grayscale image, the plain image P is simply copied to C_0 with h rows and w columns.

Pixel rearrangement itself is done in two steps. The first step uses cyclic shifts for moving pixel intensities in columns of C_0 . The sizes of shifts are determined by elements of sequence seq_{col}' . Resulting image with intensities of pixels shifted in the columns is saved to image C_1 .

The second step shifts pixel intensities in rows of C_1 . The sizes of cyclic shifts are chosen according to sequence seq_{row}' . Then, the resulting image with rearranged pixel intensities is saved to image C_2 .

The steps involving cyclic shifts are illustrated in Fig. 3.

C. Diffusion

Diffusion stage assigns new intensities to pixels of the rearranged image C_2 . The new pixel intensities depend also on intensities of other pixels which is a useful for achieving the property of total diffusion described in chapter I.

Fig. 3. An illustration of steps of the confusion stage

The diffusion stage happens in four steps, each step scans the image in a different way. This technique helps to establish dependencies between intensities of all image pixels. The four steps produce four images D_1 to D_4 and they are described by the following algorithm:

Inputs: rearranged image C_2 , its height h , width w and number of color planes d , generated substitution vectors vec_{sub1} to vec_{sub8} .

Output: encrypted image D_4 .

Step 1: This step diffuses pixel intensities in rows of the image C_2 , starting from the top row, continuing to the bottom row of the image. At first, the image C_2 is copied to image D_1 and then the ciphertext chaining is done by (5):

$$D_1(row, :) = D_1(row, :) + D_1(row-1, :) \pmod{256}, \quad (5)$$

where $row = 1, 2, \dots, h$ is an index of image rows and $:$ means that the operation is done element-wise in each column of the image. The top row ($row = 1$) uses the bottom one ($row = h$) for the addition.

Before continuing to the next row, operation of plaintext related diffusion takes place by applying (6). The resulting values are used as indices for values from substitution vector sub_{vec1} . The substituted values are then copied to the image D_1 .

$$D_1(row, :) = D_1(row, :) \oplus D_1(row+1, :), \quad (6)$$

where $\oplus$ is an operator of bitwise eXclusive OR (XOR). The bottom row ($row = h$) uses the top one ($row = 1$) for the operation.

There is one exception in the computations done during Step 1. The bottom row ($row = h$) utilizes substitution vector sub_{vec2} instead of vector sub_{vec1} . This is done in order to minimize possibilities of the phase space reconstruction [7], which usually starts with the last processed elements.

Step 2: This step diffuses pixel intensities in rows of the image D_1 , in an opposite pattern (starting from the bottom row, continuing to the top row of the image). At first, the image D_1 is copied to image D_2 and then the ciphertext chaining is done by (7):

$$D_2(row,:) = D_2(row,:) + D_2(row+1,:) \pmod{256}, \quad (7)$$

where $row = h, h-1, \dots, 1$ is an index of image rows. The bottom row ($row = h$) uses the top one ($row = 1$) for the addition.

Before continuing to the next row, plaintext related diffusion is done by (8). The resulting values are used as indices for values from substitution vector sub_{vec3} . The substituted values are then copied to the image D_2 .

$$D_2(row,:) = D_2(row,:) \oplus D_2(row-1,:). \quad (8)$$

The top row ($row = 1$) uses the bottom one ($row = h$) for the operation.

Also, this step utilizes different substitution vector for replacing the values of the last processed row ($row = 1$). In this case, vector sub_{vec4} is used instead of vector sub_{vec3} .

Step 3: This step diffuses pixel intensities in columns of the image D_2 , starting from the leftmost column, continuing to the rightmost column of the image. At first, the image D_2 is copied to image D_3 and then the ciphertext chaining is done by (9):

$$D_3(:,col) = D_3(:,col) + D_3(:,col-1) \pmod{256}, \quad (9)$$

where $col = 1, 2, \dots, w' = w \cdot d$ is an index of image columns and $:$ means that the operation is done element-wise in each row of the image. The leftmost row ($col = 1$) uses the rightmost one ($col = w'$) for the addition.

Before continuing to the next column, operation of plaintext related diffusion takes place by (10). The resulting values are used as indices for values from substitution vector sub_{vec5} . The substituted values are then copied to the image D_3 .

$$D_3(:,col) = D_3(:,col) \oplus D_3(:,col+1). \quad (10)$$

The rightmost column ($col = w'$) uses the leftmost one ($col = 1$) for the operation.

Also, this step utilizes different substitution vector for replacing the values of the last processed column ($col = w'$). In this case, vector sub_{vec6} is used instead of vector sub_{vec5} .

Step 4: This step diffuses pixel intensities in columns of the image D_3 , in an opposite pattern (starting from the rightmost column, continuing to the leftmost column of image). At first, the image D_3 is copied to image D_4 and then the ciphertext chaining is done by (11):

$$D_4(:,col) = D_4(:,col) + D_4(:,col+1) \pmod{256}, \quad (11)$$

where $col = w', w'-1, \dots, 1$ is an index of image columns. The rightmost column ($col = w'$) uses the leftmost one ($col = 1$) for the addition.

Before continuing to the next row, plaintext related diffusion is done by (12). The resulting values are used as indices for values from substitution vector sub_{vec7} . The substituted values are then copied to the image D_4 .

$$D_4(:,col) = D_4(:,col) \oplus D_4(:,col-1). \quad (12)$$

The leftmost column ($col = 1$) uses the rightmost one ($col = w'$) for the operation.

Also, this step utilizes different substitution vector for replacing the values of the last processed column ($col = 1$). In this case, vector sub_{vec8} is used instead of vector sub_{vec7} .

The scanning patterns are illustrated in Fig. 4. Four scans are used in order to achieve property of the total diffusion. While the first two scans are used for establishing dependencies between pixel intensities in individual rows of image, the other two scans spread these intensities to the columns of image. As the intensities that are used for ciphertext chaining in columns of image already depend on the intensities from whole rows of image, each image pixel should be affected by all image pixels.

Fig. 4. Scanning patterns used during the diffusion stage

Steps done during each scan are shown in a block scheme displayed in Fig. 5. It is important to mention that while ciphertext chaining uses previous rows or columns, the plaintext related diffusion utilizes following rows or columns. This approach introduces both the dependencies created during encryption and decryption. Therefore, also slightly modified encrypted images should result in totally different plain images.

Fig. 5. A block scheme of operations done during each scan of the diffusion

D. Differences done during Decryption

The decryption of images utilizes the same PR sequences as the encryption. However, the inverse substitution vectors are created in a different way. At first, all the steps used during

encryption for creating the substitution vectors are done. Then, the elements of substitution vectors are mapped for enabling the inverse substitution: e. g. if a value of 202 is located on 23rd place of the substitution vector, its location is copied to the position according to its value – a value of 23 is copied to 202nd place of the inverse substitution vector. Note that the indices of locations in substitution vectors are from set $\{0, 1, \dots, 255\}$ as the pixels intensities are also from this set.

After the inverse substitution vectors and sequences used during confusion are created, the decryption itself starts with removal of diffusion. This is done also in the same four steps, however their order is reversed, starting with Step 4 and continuing to Step 1. During these steps, the plaintext related diffusion is done before the ciphertext chaining. The operation of bitwise XOR stays the same, but the ciphertext chaining uses subtraction instead of addition.

The inverse confusion is done by two cyclic shifts, starting with shifts in rows before the shifts in columns of image. The sizes of shifts are determined by negative values of sequences seq_{row} and seq_{col} , respectively.

After all these steps, the decrypted image is obtained by reshaping the image with h rows and $w'=w \cdot d$ columns to an image with h rows, w columns and d color planes. The value of d is determined by color depth of the encrypted image.

IV. EXPERIMENTAL RESULTS

All experiments described in this chapter were performed on a PC with 2.5 GHz CPU and 12 GBs of RAM in MATLAB R2015a running under Windows 10 OS.

Images used during experiments are shown in Fig. 6. Their parameters are included in TABLE IV. Please note that image *black* has significantly lower resolution and therefore it is magnified in all following figures. This is done for better assessment of potential patterns in the encrypted images.

Fig. 6. A set of testing images

TABLE IV. PARAMETERS OF IMAGES FROM THE TESTING SET

image / parameter	lena	lenaG	black
resolution (width w x height h) [px]	512x512	256x256	16x8
color depth [bits/px]	24	8	24
number of color planes d	3	1	3

A set of used keys in hexadecimal notation is presented in TABLE V. Please note that the difference between similar keys (K_1 and K_2) is indicated by bold font. Images from Fig. 6 encrypted by K_1 are shown in Fig. 7.

TABLE V. A SET OF USED KEYS

key	value
K_1	0x 74 68 69 73 69 73 61 73 65 63 72 65 74 6B 65 79
K_2	0x 74 69 69 73 69 73 61 73 65 63 72 65 74 6B 65 79
K_3	0x 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Fig. 7. Examples of encrypted images

A. Size of Key Space and Key Sensitivity

As the proposed image encryption algorithm utilizes key with length of 16 bytes, the size of key space could be expressed as $256^{16} = 2^{8 \cdot 16} = 2^{128}$. If we would consider that one decryption of true color image with resolution of 512x512 pixels takes approx. 330 ms, the brute-force attack would take approx. 3.5608×10^{30} years. Therefore we consider this type of attacks infeasible.

Sensitivity of the proposed algorithm to different keys is demonstrated in Fig. 8 where different keys were used for the encryption of image *black*. Then, the encryptions by K_1 and K_2 were compared via the difference image. The figure also shows results of decryption with both correct and incorrect key.

Fig. 8. An illustration of key sensitivity of the proposed algorithm

B. Demonstration of Total Diffusion

The property of total diffusion was evaluated by two experiments:

- image *black* was modified before encryption – the pixel intensity at location (4, 8) in green color plane was changed from 0 to 1, and then it was encrypted by K_1 ,
- image *black* was encrypted by K_1 and the encrypted image was modified before decryption – the pixel intensity at location (4, 8) in green color plane was changed from 60 to 59, then the image was decrypted by K_1 .

The results of the experiments, also with reference (not modified) images and difference images are shown in Fig. 9.

Fig. 9. Effects of the total diffusion

According to Fig. 9, it could be stated that the proposed algorithm satisfies the condition of total diffusion. While the slight change done before encryption is not that visible, it is well demonstrated in the second case. The unmodified encrypted version of image *black* produces original plain image, but the modified version of encrypted image decrypted by a correct key results in a totally different image.

C. Resistance against Statistical Attacks

Properties of image encryption algorithms regarding statistical attacks could be evaluated by several measures: shape and amount of peaks in histograms of images, distribution of points in correlation plots, values of correlation coefficients and values of entropy. The *histograms* for images *lenaG* and its version encrypted by K_1 are shown in Fig. 10.

As the notable peaks present in a histogram of plain image *lenaG* are suppressed by the encryption, the amount of data usable for statistical attack could be considered as minimal.

Correlation plots use intensities of two image pixels as coordinates of plotted points. The two image pixels are chosen as a pair of adjacent pixels. The pixel pairs could be horizontally, vertically or diagonally adjacent. The correlation plots for 1,000 randomly chosen horizontally adjacent pixel pairs from image *lenaG* and its version encrypted by K_1 are shown in Fig. 11.

Fig. 10. Comparison of image histograms

Fig. 11. Comparison of correlation plots

Correlation plots clearly show that the encryption suppresses correlation between intensities of adjacent image pixels in plain image *lenaG*. As the pairs of pixel intensities have greater variance, the distribution of points in the second correlation plot is more uniform.

Numeric parameters – correlation coefficients ρ and entropy H are calculated via (13) and (14), respectively. Correlation coefficients are computed for horizontally (ρ_{hor}), vertically (ρ_{ver}) or diagonally (ρ_{diag}) adjacent pixel pairs. Both the correlation coefficients and entropy are calculated separately for each color plane of tested image.

$$\rho_{[-]} = \frac{\sum_{i=1}^{pp} (in_1(i) - \overline{in_1})(in_2(i) - \overline{in_2})}{\sqrt{\sum_{i=1}^{pp} (in_1(i) - \overline{in_1})^2 \sum_{i=1}^{pp} (in_2(i) - \overline{in_2})^2}}, \quad (13)$$

where $i = 1, 2, \dots, pp$ is an index of a pixel pair, pp is amount of pixel pairs for chosen adjacency, $in_1(i)$ and $in_2(i)$ represent intensities of the first and the second pixel from one pixel pair and $\overline{in}$ denotes an arithmetic mean of vector in .

$$H_{[bits/px]} = - \sum_{in=0}^{2^L-1} p(in) \cdot \log_2(p(in)), \quad (14)$$

where $p(in)$ is a probability of occurrence of a pixel with intensity in and L stands for color depth of tested image (or color plane) in bits per pixel.

The computed values of correlation coefficients ρ and entropy H are included in TABLE VI.

TABLE VI. COMPUTED VALUES OF CORRELATION COEFFICIENTS AND ENTROPY

parameter and key / image		lena (R)	lena (G)	lena (B)	lenaG	black (R)
ρ_{hor} [-]	plain	0.9749	0.9632	0.9376	0.9468	~1
	K_1	-0.0007	-0.0002	-0.0031	0.0007	0.0630
	K_2	0.0021	-0.0007	0.0005	-0.0048	-0.0996
	K_3	-0.0007	-0.0045	0.0019	-0.0001	0.0921
ρ_{ver} [-]	plain	0.9782	0.9729	0.9515	0.9640	~1
	K_1	0.0013	0.0017	-0.0030	-0.0036	-0.0210
	K_2	-0.0005	-0.0005	0.0009	0.0047	-0.0440
	K_3	-0.0019	-0.0038	-0.0011	-0.0070	0.0731
ρ_{diag} [-]	plain	0.9593	0.9497	0.9212	0.9264	~1
	K_1	0.0006	0.0020	-0.0009	-0.0012	-0.0401
	K_2	0.0014	0.0030	-0.0017	0.0019	0.0149
	K_3	0.0037	-0.0035	-0.0017	0.0035	-0.1374
H [bits/px]	plain	7.5883	7.1060	6.8147	7.1723	0*
	K_1	7.9993	7.9992	7.9992	7.9975	6.4979
	K_2	7.9993	7.9994	7.9993	7.9974	6.5448
	K_3	7.9991	7.9994	7.9994	7.9968	6.5233

* division by zero, entropy is set as 0

Results reported in TABLE VI show that the encryption significantly decreases values of the correlation coefficients.

Also, the coefficients are quite similar even for different keys. It could be concluded that smaller values of resolution produce higher differences in values of correlation coefficients (mostly in case of image *black*).

The values of entropy are quite close to the theoretical boundary of 8 bits per pixel. Also their values do not depend on entropies of color planes from plain images.

D. Resistance against Differential Attacks

Differential attacks observe impact of small modifications done before the encryption on resulting encrypted images. There are two well-known measures used for evaluation of resistance against differential attacks, the first one being *Number per Pixel Change Rate* (NPCR) and the second one is *Unified Average Changing Intensity* (UACI).

Both NPCR and UACI utilize two plain images. The first one, P_1 is chosen by user. The second one, P_2 is an exact copy of P_1 , however an intensity of one image pixel is slightly changed (by one intensity level). These two images are then encrypted by the same algorithm with the same setting and produce images E_1 and E_2 .

NPCR and UACI could be calculated for each color plane of tested image by (15) and (16), respectively:

$$NPCR_{[\%]} = \frac{100}{h \cdot w} \sum_{row=1}^h \sum_{col=1}^w Diff(row, col) \quad (15)$$

$$Diff(row, col) = \begin{cases} 0 & \text{if } E_1(row, col) = E_2(row, col) \\ 1 & \text{otherwise} \end{cases},$$

where row and col are indices of image rows and columns, h and w are height and width of tested image and $Diff$ is a difference matrix.

$$UACI_{[\%]} = \frac{100}{h \cdot w} \sum_{row=1}^h \sum_{col=1}^w \frac{|E_1(row, col) - E_2(row, col)|}{2^L - 1}, \quad (16)$$

where L represents color depth of tested image (or its color plane) in bits per pixel.

Please note that while NPCR only sums the amount of changes between two encrypted images, UACI also takes into account intensities of these changes. Resulting values of NPCR and UACI are included in TABLE VII. These are arithmetic means of 100 repeated measurements, each of those had randomly chosen image pixel with modified intensity.

TABLE VII. COMPUTED VALUES OF NPCR AND UACI

parameter and key / image		lena (R)	lena (G)	lena (B)	lenaG	black (R)
NPCR [%]	K_1	99.6334	99.6227	99.6555	99.6292	99.2188
	K_2	99.6616	99.6410	99.6449	99.6429	99.2188
	K_3	99.6181	99.6502	99.6780	99.6323	99.2188
UACI [%]	K_1	33.4903	33.4740	33.4682	33.4858	33.5141
	K_2	33.4725	33.4663	33.5026	33.5067	33.5080
	K_3	33.4889	33.4770	33.4707	33.4783	33.4957

Computed arithmetic means of NPCR and UACI presented in TABLE VII show that the values for different color planes of one image are quite similar. Moreover, usage of a different key does not change the values much.

NPCR and UACI were thoroughly studied by Wu et al. in [13], where so-called *critical values* of NPCR and UACI were introduced. As these values are 99.6094 % for NPCR and 33.4635 % for UACI, it could be concluded that the proposed algorithm reaches better values than the critical ones in most measurements.

NPCR values of 99.2188 % reported for image *black* are affected by small resolution of the image. As this image has only $16 \cdot 8 \cdot 3 = 384$ pixels in all of its color planes, the value of 99.2188 % points out to fact that only 3 pixels were left with an unchanged intensity after the encryption.

E. Computational Complexity

Computational complexity of the proposed algorithm was investigated by encrypting and decrypting each image from the testing set by all keys. Each of these operations was repeated for 100 times and the arithmetic means of encryption times t_{enc} and decryption times t_{dec} are included in TABLE VIII. This table also shows the processing speeds v_{enc} and v_{dec} , which could be computed by (17):

$$v_{[\text{MB/s}]} = \frac{h \cdot w \cdot d}{2^{20} \cdot t}, \quad (17)$$

where h , w and d are height, width and number of color planes of an image and t is computational time of operation (encryption or decryption) in seconds. Constant of 2^{20} represents amount of bytes in one megabyte.

TABLE VIII. CALCULATED TIMES AND SPEEDS OF OPERATIONS

parameter and key / image		lena	lenaG	black
t_{enc} [ms]	K_1	366.43	40.84	10.76
	K_2	366.97	40.47	10.76
	K_3	366.91	40.48	10.70
t_{dec} [ms]	K_1	332.42	37.40	10.23
	K_2	333.36	37.82	10.17
	K_3	334.02	37.41	10.16
v_{enc} [MB/s]	K_1	2.0468	1.5305	0.0340
	K_2	2.0438	1.5442	0.0340
	K_3	2.0441	1.5440	0.0342
v_{dec} [MB/s]	K_1	2.2562	1.6713	0.0119
	K_2	2.2498	1.6524	0.0120
	K_3	2.2454	1.6705	0.0120

Values presented in TABLE VIII show that the duration of any operation (encryption or decryption) does not depend on the used key, as the differences between times obtained for various keys are minimal. Also, it could be stated that the decryption is slightly faster than the encryption. Please note that the smaller values of image resolution results in decrease of the processing speeds.

V. COMPARISON WITH OTHER APPROACHES

This chapter compares the numeric results of the proposed algorithm with results reported by Kanso and Ghebleh in [5], Guanghui et al. in [6], Murillo-Escobar et al. in [8] and Liu and Miao in [9]. Results for the last algorithm used for comparison, described in paper [10] were calculated in a new set of experiments.

The comparison of results is not an easy task, as various papers contain results for various images. While [5] and [8] use version of image *lena* with resolution of 512x512 pixels and color depth of 24 bits per pixel, [6] and [9] use image *lenaG* with resolution of 256x256 pixels and color depth of 8 bits per pixel. As it was already mentioned, the results for paper [10] were obtained by performing new set of experiments. These experiments used both desired images.

The comparison of results is included in TABLE IX and TABLE X. The proposed algorithm used key K_1 for all measurements. In the case of true color images, all presented values except for encryption times t_{enc} are computed for their red color plane.

TABLE IX. COMPARISON OF VALUES OF CORRELATION COEFFICIENTS AND ENTROPY

algorithm / parameter	ρ_{hor} [-]	ρ_{ver} [-]	ρ_{diag} [-]	H [bits/px]
true color image (<i>lena</i>)				
proposed	-0.0007	0.0013	0.0006	7.9993
ref. [5] (2011)	-0.0029	-0.0150	0.0129	7.9997
ref. [8] (2015)	0.0135	-0.0835	-0.0170	7.9974
ref. [10] (2017)	0.0032	-0.0019	0.002	7.9993
grayscale image (<i>lenaG</i>)				
proposed	0.0007	-0.0036	-0.0012	7.9975
ref. [6] (2014)	-0.0278	-0.0065	-0.0074	7.9895
ref. [9] (2016)	0.0021	0.0046	0.0033	7.9995
ref. [10] (2017)	-0.0071	0.0072	0.0054	7.997

TABLE X. COMPARISON OF VALUES OF NPCR, UACI AND ENCRYPTION TIMES

algorithm / parameter	NPCR [%]	UACI [%]	t_{enc} [ms]
true color image (<i>lena</i>)			
proposed	99.6334	33.4903	366.43
ref. [5] (2011)	99.62	33.51	~390.6
ref. [8] (2015)	99.63	33.31	243.2
ref. [10] (2017)	99.6139	33.4647	338.14
grayscale image (<i>lenaG</i>)			
proposed	99.6292	33.4858	40.84
ref. [6] (2014)	99.66	33.57	unknown
ref. [9] (2016)	99.63	33.71	65.69
ref. [10] (2017)	99.6201	33.4692	37.75

Results shown in TABLE IX and TABLE X lead to several conclusions:

- the proposed algorithm achieves the best values of correlation coefficients,
- because the values of correlation coefficients for the grayscale image achieved by the proposed algorithm and algorithm without total diffusion [10] are different, it could be concluded that the technique of total diffusion helps to obtain better values of correlation coefficients,
- the values of entropy are close to the best obtained values ([5] for true color images and [9] for grayscale images),
- while the results of NPCR are comparable with the best values for both true color and grayscale images, values of UACI are comparable only for true color images,
- obtained values of UACI for grayscale images are significantly worse,
- usage of total diffusion helps to increase values of NPCR and UACI, however, not so significantly as in the case of correlation coefficients,
- reached encryption times are quite slow for the true color images, however, they are better than those of the fastest approach for grayscale images,
- the difference of encryption times between the proposed solution and algorithm without total diffusion [10] is approx. 8 %.

VI. CONCLUSION

This paper dealt with an image encryption algorithm which utilizes technique of plaintext related diffusion combined with suitable scanning pattern in order to establish total diffusion between pixel intensities of encrypted image. The desired properties of utilized pseudo-random sequences were achieved by usage of chaotic logistic map.

The amount of applications for image encryption algorithms is still rising. Probably the most obvious one is usage of encryption prior to information hiding via steganography [14, 15]. The image encryption algorithms could be also used for modification of image data into a set of samples with more uniform distribution, which could be helpful for some processing operations [16].

The properties of the proposed algorithm were evaluated by a set of experiments. It was clearly demonstrated that the proposed algorithm is sensible even to the slightest modification of plaintext images before encryption or encrypted images before decryption. As the changes caused by dependencies created between pixel intensities spread to whole image, it could be concluded that the proposed algorithm fulfills the condition of total diffusion as it was conceived by Shannon in [2].

Comparison of experimental results with those obtained by other researches show that the proposed algorithm reaches better values of correlation coefficients. Also, the values of entropy and NPCR are comparable. However, while values of UACI for true color image are still comparable, the values of UACI for grayscale images are worse.

The other drawback of the proposed algorithm is its computational complexity shown by longer durations of encryption or decryption. This could be caused by higher amount of scans done during the diffusion stage. However, all of the scans are necessary for fulfilling the condition of total diffusion. In the future, we plan to investigate other possible scanning patterns of pixel intensities, which could be useful for minimizing the computational complexity.

ACKNOWLEDGMENT

This work was supported by following research grants: KEGA 023TUKE-4/2017 and APVV-0025-12.

REFERENCES

- [1] 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.
- [2] C. E. Shannon, "Communication theory of secrecy systems," *The Bell System Technical Journal*, 1949, vol. 28, no. 4, pp. 656-715. ISSN: 0005-8580. DOI: 10.1002/j.1538-7305.1949.tb00928.x.
- [3] J. Oravec, J. Turán, L. Ovseník, "Image encryption algorithm based on squared sine logistic map," *Carpathian Journal of Electronic and Computer Engineering*, 2017, vol. 10, no. 2, pp. 8-14. ISSN: 1844-9689.
- [4] E. Solak, C. Çokal, O. T. Yildiz, T. Biyikoğlu, "Cryptanalysis of Fridrich's chaotic image encryption," *International Journal of Bifurcation and Chaos*, 2010, vol. 20, no. 5, pp. 1405-1413. ISSN: 0218-1274. DOI: 10.1142/S0218127410026563.
- [5] A. Kanso, M. Ghebleh, "A novel image encryption algorithm based on a 3D chaotic map," *Communications in Nonlinear Science and Numerical Simulation*, 2012, vol. 17, no. 2, pp. 2943-2959. ISSN: 1007-5704. DOI: 10.1016/j.cnsns.2011.11.030.
- [6] C. Guanghui, H. Kai, Z. Yizhi, Z. Jun, Z. Xing, "Chaotic image encryption based on running-key related to plaintext," *The Scientific World Journal*, 2014, vol. 15, no. 1, pp. 1-9. ISSN: 2356-6140. DOI: 10.1155/2014/490179.
- [7] B. Kliková, A. Raidl, "Reconstruction of phase space of dynamical systems using method of time delay," *Proceedings of WDS'11*, Prague (Czech Republic), 2011, pp. 83-87. ISBN: 978-8-073-78186-6.
- [8] M. A. Murillo-Escobar, C. Cruz-Hernández, F. Abundiz-Pérez, R. M. López-Gutiérrez, O. R. Acosta Del Campo, "A RGB image encryption algorithm based on total plain image characteristics and chaos," *Signal Processing*, 2015, vol. 109, no. C, pp. 119-131. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2014.10.033.
- [9] L. Liu, S. Miao, "A new image encryption algorithm based on logistic chaotic map with varying parameter," *SpringerPlus*, 2016, vol. 5, no. 1, pp. 289-300. ISSN: 2193-1801. DOI: 10.1186/s40064-016-1959-1.
- [10] J. Oravec, J. Turán, L. Ovseník, "Image encryption algorithm based on squared sine logistic map," *Carpathian Journal of Electronic and Computer Engineering*, 2017, vol. 10, no. 2, pp. 8-14. ISSN: 1844-9689.
- [11] 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.
- [12] R. Matthews, "On the derivation of a 'chaotic' encryption algorithm," *Cryptologia*, 1989, vol. 8, no. 1, pp. 29-42. ISSN: 0161-1194. DOI: 10.1080/0161-118991863745.

- [13] Y. Wu, J. Noonan, S. Agaian, "NPCR and UACI randomness tests for image encryption," *Journal of Selected Areas in Telecommunications*, 2011, vol. 2, no. 4, pp. 31-38. ISSN: 1925-2676.
- [14] J. Oravec, 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.
- [15] V. Hajduk, D. Levický, "Cover Selection Steganography with Intra-Image Scanning," *Proceedings of Radioelektronika 2018*, Prague (Czech Republic), 2018, pp. 1-4. ISBN: 978-1-5386-2485-2.
- [16] O. Kováč, J. Mihalík, I. Gladišová, "Convolution implementation with a novel approach of DGHM multiwavelet image transform," *Journal of Electrical Engineering*, 2017, vol. 68, no. 6, pp. 455-462. ISSN: 1335-3632. DOI: 10.1515/jee-2017-0080.

BIOGRAPHIES

Jakub Oravec (Ing.) received Ing. (MSc.) degree in 2015 at Department of Electronics and Multimedia Communications, Faculty of Electrical Engineering and Informatics, Technical University Košice. Since September 2015 he has been at Technical University, Košice as PhD. student. His research interests include steganography, digital watermarking and digital image processing.

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

Ľuboš Ovseník (doc., Ing., PhD.) received Ing. (MSc.) degree in radioelectronics from Technical University, Košice in 1990. He received PhD. degree in electronics from Technical University, Košice in 2002. Since February 1997, he has been at the same university as Associate Professor for electronics and information technology. His general research interests include optoelectronics, digital signal processing, fiber optics communications and fiber optics sensors.