Computing and Informatics, Vol. 38, 2019, 647–678, doi: 10.31577/cai_2019_3_647

IMAGE ENCRYPTION ALGORITHM WITH PLAINTEXT RELATED CHAINING

Ľuboš Ovseník, Ján Turán, Tomáš Huszaník, Jakub Oravec

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

Ondrej Kováč

Department of Technologies in Electronics
Technical University of Košice
Park Komenského 2, 040 01 Košice, Slovakia
e-mail: ondrej.kovac@tuke.sk

Milan Oravec

Department of Safety and Quality of Production
Technical University of Košice
Letná 9, 040 01 Košice, Slovakia
e-mail: milan.oravec@tuke.sk

Abstract. This paper describes a plaintext related image encryption algorithm that utilizes the Mojette transform for computation of bins that are subsequently combined with pixels of the processed image. While the bins are computed solely from pixel intensities of a plain image and also the combination depends only on intensities of plain image pixels, the parameters of bins are rearranged according to used key. This design results in a great sensitivity of the proposed image encryption algorithm to both plain images and keys, which is verified by a set of experiments. The paper also tests the resistance of the proposal against statistical and differential attacks by means of commonly used measures as correlation coefficients, entropy, NPCR and UACI. Furthermore, the paper analyses computation speed reached by the proposed solution. Computed values of all parameters are discussed and then compared with results obtained by some recent plaintext related image encryption algorithms.

Keywords: Image encryption, logistic map, Mojette transform, pixel intensity chaining, plaintext related operation

Mathematics Subject Classification 2010: 94A60, 68U10

1 INTRODUCTION

Security of data transmitted over public networks is an important task these days. One of the possible solutions for establishing security of data is encryption. Conventional encryption algorithms such as Advanced Encryption Standard (AES) were usually designed in a way that is suitable for encryption of rather small blocks of hexadecimal or binary characters [1, 2]. However, this approach is not suitable for all data types. Digital images known for their redundancy (vast amount of pixels) and high correlation between adjacent pixels can be considered as an example.

Encryption of image data by conventional encryption algorithms can lead to various undesirable results. For example, a basic mode of operation for AES, called Electronic CodeBook (ECB), only replaces blocks of plain image data by calculated blocks of encrypted data. In the case that some plain image blocks are the same, all of them are substituted with identical blocks of encrypted data. This situation is shown in Figure 1 where AES in ECB mode was used for encryption of the image with resolution of 256×256 pixels and color depth of 8 bits per pixel. Used encryption key was `0× C4 EB 50 BC 0E C5 EB 50 BC 0E C5 EB 50 BC 0E C5`. This key was acquired from the first 128 bits of binary representation of decimal part of π.

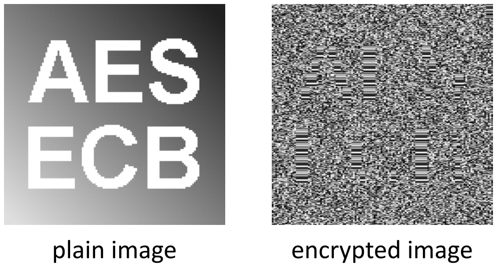


Figure 1. Contours in encrypted image after encryption by ECB mode of AES

As seen in Figure 1, the smooth image areas located in the characters of the plain image, where the intensities of adjacent pixels do not change, produce several identical blocks in the encrypted image. On the other hand, the background of image, which contains a grayscale gradient filling, was substituted with many different blocks. The contrast between these blocks can lead to identification of contours of the characters from the plain image, and that is undesirable.

The example illustrated in Figure 1 can be viewed as an exaggeration as the used plain image is an uncompressed bitmap image. However, also images processed by lossy compression algorithms can contain some identical blocks of pixels. Therefore this problem can arise also in the encrypted versions of such images.

Furthermore, the encryption of images by conventional encryption algorithms can lead to bad performance in terms of computational speed. In the case that used platform does not enable hardware acceleration of encryption [3], the dedicated image encryption algorithms are usually faster [4]. This is due to their better optimization for the task of image encryption. Considering the mentioned properties of images and conventional encryption algorithms, images as a data type require specific and more efficient approach in order to achieve an acceptable performance.

The first dedicated image encryption algorithms were designed in late 1990s. Probably one of the most popular early image encryption algorithms is the Fridrich's algorithm published in 1998 [5], which used chaotic Bakers' map for achieving the desired performance. Application of a chaotic map inspired other researchers [6, 7, 8, 9] and nowadays the majority of image encryption algorithms is based on a suitable chaotic system. Fridrich's paper was also important because it described a useful architecture of image encryption algorithms, consisting of confusion and diffusion stage. This architecture corresponds with ideas of Shannon [10], and with some modifications it is still used nowadays.

First attempts to break chaotic image encryption algorithms can be traced back to early 2000s. However, majority of these papers tried to break only one specific image encryption algorithm [11, 12]. In 2010, Solak et al. proposed a chosen ciphertext attack based on relations between image pixels during decryption [13]. Solak's attack is not only able to break Fridrich's encryption algorithm, but it can be also used for some other designs with the similar architecture [13]. The performance of Solak's attack was later discussed by Xie et al. in [14], where some minor improvements were suggested.

Presentation of the Solak's attack caused changes in the Fridrich's architecture. Most authors tried to establish relations between the steps of image encryption algorithm and used plain image (image before encryption). Therefore these approaches are described as plaintext related image encryption algorithms.

Despite chaotic image encryption algorithms still evolve, the existing algorithms have already found some applications. They usually require sensitive data to remain in the form of an image: image steganography [15, 16, 17], transfer and storage of biometric features [18] or medical images [19].

Image encryption algorithms can be also used for verification of data integrity of images, because any image processing technique applied on an encrypted image leads to the incorrect decryption. If users wish to enhance or modify some image, it needs to be processed before the encryption. Then the decryption of the encrypted image leads to the processed image.

The rest of the paper is organized as follows: Section 2 describes works of other authors in the field of plaintext related chaotic image encryption. Section 3 explains techniques used in our proposal and describes the steps of encryption and decryption algorithms. Section 4 presents obtained experimental results and compares them with the results achieved by some other similar algorithms. Section 5 discusses the properties of the proposed solution and concludes the paper.

2 RELATED WORK

Since the introduction of Solak's attack [13] in 2010, several solutions for suppressing or even eliminating a possibility of a successful attack have been provided. Fu et al. proposed their solution in 2012 [20] where parameter of Chebyshev's map is modified according to intensity of the previously encrypted pixel. However, these parameter modifications can lead to fixed points, where behavior of chaotic maps is constant and therefore unsuitable. Fu et al. used a mechanism that prevents occurrence of fixed points, however this step decreases the overall computational speed.

A proposal by Kanso et al. from 2012 [21] changed the amount of iterations of Arnold's cat map by intensities of currently processed image pixel. As the pixel intensities are divided by 10 and then rounded to the closest smaller integer number, multiple intensities lead to identical amount of iterations. Also, each additional map iteration causes the longer computational time, so timing attacks [22] are possible.

A paper by Fu et al. [23] from 2013 used cyclic shifts of bits from pixel intensities controlled by intensities of previous image pixels. As there are only 8 possible ways of cyclic shift of 8 bits, this solution can lead to equivalent shifts for multiple intensities (there are 256 possible pixel intensities for color depth of 8 bits per pixel).

In 2014, Zhang proposed an encryption algorithm [24] based on a different architecture, where one iteration of diffusion is followed by plaintext related confusion and then second iteration of diffusion. However, this approach can be prone to chosen plaintext attacks [25] in cases when one iteration of diffusion produces a similar results for two images, and the plaintext related confusion – a rearrangement of image pixels only shuffles these slightly different image pixels. Similar problems can happen also with Zhang's later design from 2015 [26].

Murillo-Escobar et al. proposed an image encryption algorithm in 2015 [27] that utilized a sum of plain image pixel intensities for calculation of an initial condition and parameter of logistic map. This solution has two drawbacks: the same sum can be computed from various images and it is not possible to calculate sum used during encryption from the encrypted image. For enabling successful decryption, Murillo-Escobar et al. suggested that this value can be hidden as intensity of one of encrypted image pixels. This approach was broken by Fan et al. in 2018 [28] by usage of chosen and known plaintext attacks.

Three similar plaintext related image encryption algorithms were designed by Chai et al. in 2017 [29], by Wang et al. [30] and Li et al. [31] in 2018. These algorithms use hash functions for calculation of the digests of plain images. The digests are then used for modifying initial conditions or parameters of chaotic maps. Hash functions should provide significantly different digests for similar plain images, however, their usage usually negatively affects the computational speed of the whole algorithm. This drawback is visible mainly in the case of [30], which uses two hash functions.

We already published one paper describing a plaintext related image encryption algorithm [32]. However, as the relation was established individually between each image pixel and each key element by means of Arnold's cat map, it was necessary to provide a key with the length of used plain image. This problem was solved by a key extension algorithm. As the decryption algorithm requires the last elements of the extended key at start of the decryption, the keys for encryption and decryption are different. The fact that approach [32] is asymmetric can be viewed as a drawback.

3 PROPOSED SOLUTION

The algorithm proposed in this paper can be used for encryption of images with arbitrary resolution higher than 16×16 pixels and color depths of 8 or 24 bits per pixel. The restriction placed on image resolution is caused by usage of multiple different image rows for processing of each row of image pixels in the proposed algorithm. The plaintext related operation utilizing the Mojette transform (MoT) requires a matrix of 12×12 pixel intensities. Together with other 2 rows of pixel intensities used for plaintext unrelated operations, one row that chooses the computed Mojette bins and the actually processed row of pixel intensities, the amount of required rows reaches 16. The amount of necessary columns of image pixels, which is 12 was then enlarged to 16 in order to produce so-called square resolution (16×16 pixels). The mentioned color depths are common for grayscale and true color images.

The proposed image encryption algorithm uses key with length of 128 bits, stored in a hexadecimal notation. As it will be shown in Section 4.1, this key length can be considered as sufficient by means of a brute-force attack. Architecture of the proposed algorithm is inspired by Fridrich's approach, however several stages are added as it is shown in Figure 2.

Steps of the proposed image encryption algorithm were carefully chosen for obtaining acceptable performance by means of commonly used measures (presented in Section 4) and also reasonably fast speed of encryption or decryption (investigated in Section 4.4). The following paragraphs describe individual stages of the proposed image encryption algorithm.

Combinations with generated pseudo-random sequences (PRSs) are used for two purposes. The first combination used during encryption helps to achieve better results for plain images with simple scenes, such as "black" images where intensities of all pixels are zero. The second purpose is suppression of the possibility of a suc-

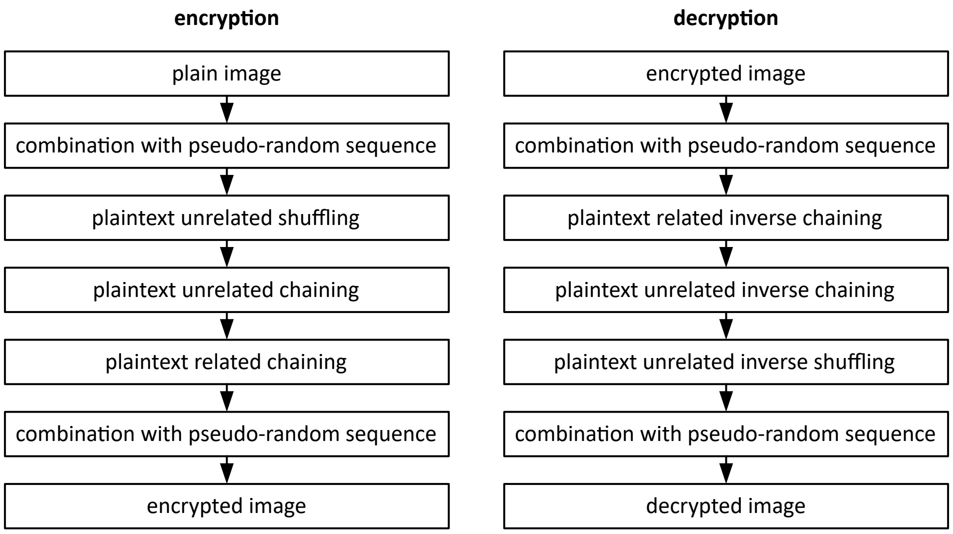

encryption
plain image
combination with pseudo-random sequence
plaintext unrelated shuffling
plaintext unrelated chaining
plaintext related chaining
combination with pseudo-random sequence
encrypted image
decryption
encrypted image
combination with pseudo-random sequence
plaintext related inverse chaining
plaintext unrelated inverse chaining
plaintext unrelated inverse shuffling
combination with pseudo-random sequence
decrypted image

Figure 2. An architecture of the proposed image encryption algorithm

cessful attack. If the plain image is combined with two PRSs before and after the encryption, an attacker would need to guess at least one of the used PRSs before attacking the encryption algorithm. All PRSs used in our encryption algorithm are generated by logistic map (LM) and then processed and quantized.

Confusion stage (shuffling of image pixels) consists of two steps. The first step rearranges image pixels in individual rows of image, while the second step shuffles pixels in individual columns of the image. The rearrangement of image pixels helps to suppress the relations between adjacent image pixels, such as their correlation.

Diffusion stage exploits the fact that the intensities of image pixels were already combined with one PRS. Therefore the diffusion can be achieved simply by performing chaining of pixel intensities. The chaining is done in two steps, the first one is not related to plain image and it is used only for establishing relations between intensities of all image pixels. Second step of the chaining is plaintext related operation based on MoT.

The main contribution of the proposed solution is a description of a novel approach of plaintext related image encryption based on MoT. MoT was chosen because it operates directly with matrices of pixel intensities and therefore it enables relatively simple implementation in the second step of the pixel intensity chaining. Also, MoT has a relatively large amount of parameters even for matrices with small sizes. Finally, as these parameters can be rearranged, the usage of MoT brings a desired nonlinear operation to the proposed image encryption algorithm.

The novel application of relatively simple MoT should lead to a faster performance than that obtained by image encryption algorithms based on more complicated techniques. As it is shown in Section 4.5, the proposed image encryption algorithm is one of the fastest algorithms when compared to similar plaintext related image encryption algorithms. Section 4.5 also discusses properties of the proposed algorithm in contrast with features of other published approaches.

MoT computes bins by using various projections of plain image blocks. Each bin is calculated as a sum of exactly three pixel intensities chosen from various color planes of image (if it has multiple color planes). As image pixels are already rearranged at the point of MoT based chaining, the bin computations use pixel intensities from various locations of the plain image.

Proposed approach utilizes encryption keys also for shuffling parameters of computed Mojette bins. Therefore, the calculated bins depend on both pixel intensities and the key. The rearrangement of Mojette parameters also causes a nonlinearity in calculations, where slight changes in parameters result in big differences in computed bin values. Calculated bins are then combined with rows of the image according to rows that are not yet encrypted, therefore this operation is plaintext related.

3.1 Preliminaries

3.1.1 Logistic Map

Logistic map (LM) is an example of one-dimensional chaotic map controlled by one parameter. LM was popularized mainly by work of May [33]. The equation for calculation of successive iterates generated by LM can be expressed as Equation (1):

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

where $x_{n+1} \in (0, 1)$ is value of a successive iterate, $r \in (0, 4)$ is a parameter of the map and $x_n \in (0, 1)$ is value of a current iterate. Calculations of the first iterate x_1 utilize value x_0 known as an initial condition or initial value.

Chaotic behavior of LM is presented on its bifurcation diagram shown in Figure 3. While the behavior of the map is predictable after the first bifurcation that occurs at $r \sim 3$, after several more bifurcations the predictability gradually decreases. The point where $r \sim 3.56995$ is known as "an onset of chaos" [34] and as a lower bound of parameter r that causes a suitable chaotic behavior of the LM.

LM often uses so-called transient period for providing more complex chaotic behavior of generated sequences. The iterates generated during transient period are used only for modification of initial condition x_0. Usual lengths of transient period are powers of 10, e.g. 1 000 iterates.

3.1.2 Mojette Transform

The Mojette transform (MoT) described in 1995 by Guédon et al. [35] is a discrete two-dimensional transform that sums matrix elements over projection lines [36, 37]. These sums are called bins and projection lines are given by three parameters – b which selects summed elements and p and q which determine discrete projection

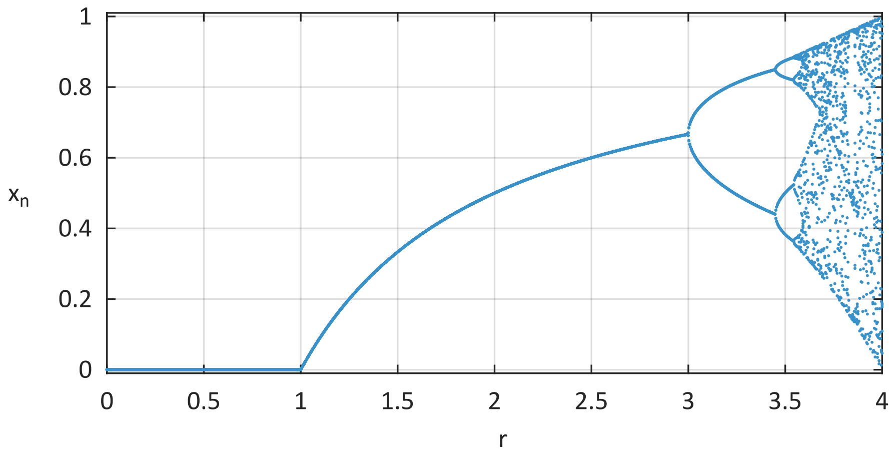


Figure 3. A bifurcation diagram of the logistic map

angle. Mathematically, calculation of Mojette bins for an image is given as Equation (2):

$$bin(b,p,q) = \sum_{k=1}^{w}\sum_{l=1}^{h} Im(l,k)\cdot\delta(b+kq-pl) \tag{2}$$

where $k = 1,2,\dots,w$ is column index of image Im, w is width of the image, $l = 1,2,\dots,h$ is its row index, h is height of the image and $\delta(x)$ is a Kronecker delta function, $\delta(x) = 1$ if $x = 0$, $\delta(x) = 0$ otherwise.

The calculation of Mojette bins for a simple image and a set of two projections is illustrated in Figure 4. Please note that additions use modulo 256 operation in order to preserve the interval of inputs (given by 8 bits – a set $\{0,1,\dots,255\}$) [38].

MoT has various applications including image coding [39, 40]. In our previous work, we found out that MoT can be used for establishing relations between image pixel intensities [41]. This feature is also the goal of diffusion stages of the image encryption algorithms. Furthermore, MoT has a property of redundancy, which causes multiple usages of pixel intensities during bin computations. While this can be viewed as a drawback from the point of computational speed, we utilize it to suppress possibility of differential attacks as each different pixel intensity would affect several bins.

3.2 Encryption Algorithm

The encryption algorithm takes a plain image P and a key K with length of 16 hexadecimal characters as inputs. Resolution of P needs to be at least 16×16 pixels for enabling computations of sufficient number of Mojette bins. The only output is encrypted image E.

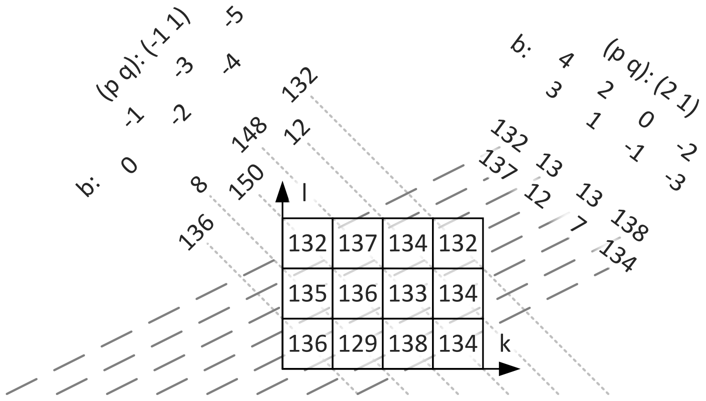


Figure 4. An example of calculated Mojette bins

3.2.1 Generation and Processing of the Pseudo-Random Sequences

Inputs: plain image P, key K.

Output: six processed and quantized PRSs seq'_{1m}, seq'_2 to seq'_5 and seq'_{6m}.

Step 1: Height h, width w and number of color planes num_{cp} of plain image P are determined. These values are used for computation of extended width $w_{ext} = w \cdot num_{cp}$ and total number of pixels $num_{px} = w \cdot h \cdot num_{cp}$.

Step 2: Key K is divided to four parts: K_1 uses first four bytes of K, K_2 utilizes bytes 5 to 8, K_3 is made of bytes 9 to 12 and K_4 uses the last four bytes of K. All four key parts K_1 to K_4 are then converted from hexadecimal to decimal notation.

Step 3: Four key parts K_1 to K_4 are utilized for calculation of four LM (1) parameters r_1 to r_4:

$$r_i = 3.9999 + 25 \cdot 10^{-6} \cdot (i - 1 + 2^{-32} \cdot K_i) \tag{3}$$

where $i = 1, 2, 3, 4$ is an index of parameter and key part, coefficient of $(i-1) \cdot 25 \cdot 10^{-6}$ ensures that each r_i stays in a different interval and constant of 2^{-32} fixes the interval of all possible K_i to $[0, 1)$.

Step 4: Six PRSs seq_1 to seq_6 are generated by six LMs (1). The initial condition x_0 is equal to 0.5 in all cases, values of parameter r are changed during the transient period according to Table 1. The changing of parameter helps to establish relations between all generated sequences and all key parts. Lengths of generated sequences are included in the bottom row of Table 1.

Step 5: The PRSs seq_1 to seq_6 are quantized by applying Equation (4). Quantized sequences are denoted as seq'_1 to seq'_6. Maximal possible value of element after

	Sequence	seq_1	seq_2	seq_3	seq_4	seq_5	seq_6
r used	iterates 1 to 250	r_1	r_2	r_3	r_4	r_2	r_3
during	iterates 251 to 500	r_2	r_3	r_4	r_1	r_1	r_2
transient	iterates 501 to 750	r_3	r_4	r_1	r_2	r_4	r_1
period	iterates 751 to 1 000	r_4	r_1	r_2	r_3	r_3	r_4
r used for following iterates		r_1	r_2	r_3	r_4	r_2	r_3
length of sequence [iterates]		num_{px}	w_{ext}	h	16	16	num_{px}

Table 1. Parameters of logistic maps and lengths of generated sequences

quantization is individual for each sequence and it is determined by Table 2.

$$seq'_i = \left\lfloor (max_i + 1) \cdot \left(10^5 \cdot seq_i \pmod 1\right)\right\rfloor \tag{4}$$

where $i = 1, 2, \ldots, 6$ is an index of sequence and max_i is the maximal possible quantized value for each sequence.

Sequence	seq_1	seq_2	seq_3	seq_4	seq_5	seq_6
max_i	255	$h - 1$	$w_{ext} - 1$	15	15	255

Table 2. Maximal possible values of the sequence elements after quantization

The multiplication of LM iterates by a constant of 10^5 and the modulo operation help to provide more uniform distribution of values, as it is demonstrated on the example in Figure 5. This example used two sequences, both with length of 10^6 iterates, initial condition $x_0 = 0.5$ and parameter $r = 4 - 10^{-15}$. While the sequence denoted as "before processing" was simply generated by LM (1), the second sequence was processed by the multiplication and the modulo operation.

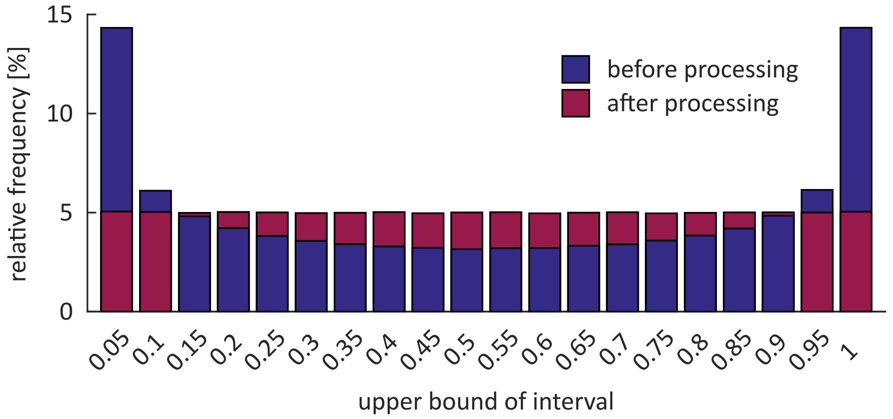


Figure 5. Effect of multiplication and modulo operation on distribution of iterate values

Step 6: Sequences seq'_1 and seq'_6 are rearranged to matrices seq'_{1m} and seq'_{6m} with h rows and w_{ext} columns. The rearrangement uses rows first scanning pattern.

3.2.2 Combination with First Pseudo-Random Sequence

Inputs: plain image P, number of color planes num_{cp}, width w, pseudo-random sequence seq'_{1m}.

Output: extended image P_{ext}.

Step 1: Plain image P is rearranged to extended image P_{ext}. Grayscale images ($num_{cp} = 1$) are simply copied to P_{ext}, while true color images ($num_{cp} = 3$) are decomposed to individual color planes. Columns of red color plane are then copied into columns of P_{ext} with indexes $1 + 3 \cdot (i - 1)$, where $i = 1, 2, \ldots, w$. Columns of green and blue color planes are copied into columns of P_{ext} with indexes $2 + 3 \cdot (i - 1)$, and $3 \cdot i$ where $i = 1, 2, \ldots, w$, respectively.

Step 2: Sequence seq'_{1m} is combined with extended image P_{ext} by means of bitwise eXclusive OR (XOR) using Equation (5). This step is necessary for images with simple scenes (images where most pixel intensities are similar).

$$P_{ext} = P_{ext} \oplus seq'_{1m} \tag{5}$$

where $\oplus$ is an operator of bitwise XOR.

3.2.3 Confusion Stage

Inputs: extended image P_{ext}, extended width w_{ext}, height h, pseudo-random sequences seq'_2 and seq'_3.

Output: extended image P_{ext} with rearranged image pixels.

Step 1: Pixel intensities in columns of extended image P_{ext} are shuffled by a cyclic shift to the bottom side of P_{ext}. Size of shift is individual for each column of P_{ext} and it is determined by corresponding element of sequence seq'_2 (Equation (6)):

$$P_{ext}(l,k) = P_{ext}\left(1 + (l - 1 + seq'_2(k) \pmod{h}), k\right) \tag{6}$$

where $l = 1, 2, \ldots, h$ is row index and $k = 1, 2, \ldots, w_{ext}$ is column index.

Step 2: Pixel intensities in rows of extended image P_{ext} are shuffled by a cyclic shift to the right side of P_{ext}. Size of shift is individual for each row of P_{ext} and it is determined by corresponding element of sequence seq'_3 (Equation (7)):

$$P_{ext}(l,k) = P_{ext}\left(l, 1 + (k - 1 + seq'_3(l) \pmod{w_{ext}})\right). \tag{7}$$

3.2.4 Diffusion Stage – Plaintext Unrelated Chaining

Inputs: extended image P_{ext}, height h, extended width w_{ext}.

Output: extended image P_{ext} with chained intensities of image pixels.

Step 1: Rows of extended image P_{ext} are scanned from the top to the bottom. Pixel intensities in individual rows of P_{ext} are chained with pixel intensities in neighboring rows. The chaining is done by element-wise modulo 256 addition of intensities from the previous scanned row and then bitwise XOR with intensities from the next scanned row, as shown in Figure 6. The top row uses the bottom one as the previous scanned row and the bottom row uses the top row as the next scanned row.

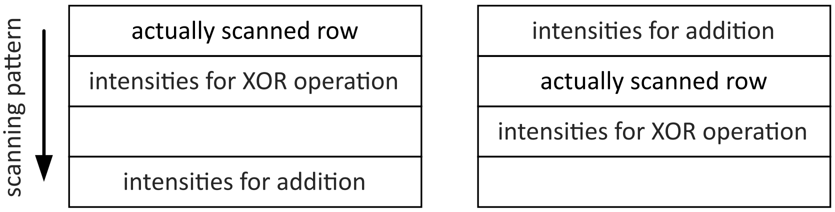


Figure 6. Demonstration of chaining for the first two rows

Step 2: Columns of extended image P_{ext} are scanned from the leftmost one to the rightmost one. Pixel intensities in individual columns of P_{ext} are chained with pixel intensities in neighboring columns. The chaining is done by element-wise modulo 256 addition of intensities from previous scanned column and then bitwise XOR with intensities from next scanned column. The leftmost column uses the rightmost one as the previous scanned column and the rightmost one uses the leftmost one as the next scanned column.

Step 3: Rows of extended image P_{ext} are scanned from the bottom to the top. Pixel intensities in individual rows of P_{ext} are chained by bitwise XOR with intensities from the next scanned row and then element-wise modulo 256 addition of intensities from the previous scanned row. The bottom row uses the top row as the next scanned row and the top row uses the bottom one as the previous scanned row.

Step 4: Columns of extended image P_{ext} are scanned from the rightmost one to the leftmost one. Pixel intensities in individual columns of P_{ext} are chained by bitwise XOR with intensities from next scanned column and element-wise modulo 256 addition of intensities from previous scanned column. The rightmost one uses the leftmost one as next scanned column and the leftmost column uses the rightmost one as previous scanned column.

Four different scans are used for establishing relations between all pixel intensities. An example of this property is shown in Figure 7 where the fill of matrix element means difference between two images. The two operations with neighboring rows or columns are utilized for creating relations during both encryption and decryption.

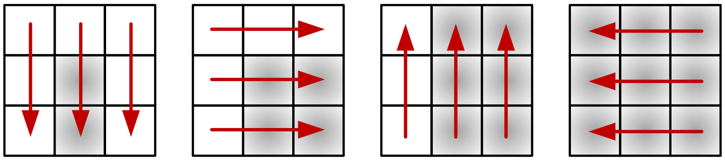


Figure 7. Scanning pattern used during plaintext unrelated chaining

3.2.5 Diffusion Stage – Plaintext Related Chaining

Inputs: extended image P_{ext}, extended width w_{ext}, height h, pseudo-random sequences seq'_4 and seq'_5.

Output: extended image P_{ext} with intensities of image pixels chained according to its pixel intensities.

Step 1: A set of parameters of Mojette bins represented by matrix mot_{par} is rearranged similarly as the extended image P_{ext} during confusion stage. The matrix mot_{par} has 16 rows and 16 columns. Its elements with linear indexes (rows first scanning pattern) are shown in Table 3. Firstly, a cyclic shift shuffles the parameters in columns of mot_{par} according to sequence seq'_4. Then the parameters in rows of mot_{par} are rearranged depending on sequence seq'_5.

Indexes	$(p\ q)$	Set of b
1 to 20	$(-5\ 1)$	$\{-56, -55, -51, -50, -46, -45, -41, -40, \dots, -16, -15, -11, -10\}$
21 to 36	$(-5\ 2)$	$\{-57, -55, -52, -50, -47, -45, -42, -40, \dots, -27, -25, -22, -20\}$
37 to 48	$(-5\ 3)$	$\{-58, -55, -53, -50, -48, -45, -43, -40, -38, -35, -33, -30\}$
49 to 56	$(-5\ 4)$	$\{-59, -55, -54, -50, -49, -45, -44, -40\}$
57 to 96	$(-4\ 1)$	$\{-47, -46 - 45, -44, -43, -42, -41, -40, -39, -38, \dots, -10, -9, -8\}$
97 to 120	$(-4\ 3)$	$\{-53, -50, -49, -47, -46, -45, -44, \dots, -32, -31, -30, -28, -27, -24\}$
121 to 128	$(-4\ 5)$	$\{-59, -55, -54, -50, -49, -45, -44, -40\}$
129 to 168	$(4\ 1)$	$\{-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, \dots, 34, 35, 36\}$
169 to 192	$(4\ 3)$	$\{-9, -6, -5, -3, -2, -1, 0, 1, 2, 3, 4, \dots, 12, 13, 14, 16, 17, 20\}$
193 to 200	$(4\ 5)$	$\{-15, -11, -10, -6, -5, -1, 0, 4\}$
201 to 220	$(5\ 1)$	$\{-1, 0, 4, 5, 9, 10, 14, 15, 19, 20, 24, 25, \dots, 39, 40, 44, 45\}$
221 to 236	$(5\ 2)$	$\{-2, 0, 3, 5, 8, 10, 13, 15, \dots, 28, 30, 33, 35\}$
237 to 248	$(5\ 3)$	$\{-3, 0, 2, 5, 7, 10, 12, 15, 17, 20, 22, 25\}$
249 to 256	$(5\ 4)$	$\{-4, 0, 1, 5, 6, 10, 11, 15\}$

Table 3. A set of parameters utilized for computing Mojette bins

Rearrangement of parameters according to sequences generated by a key establishes the relations between values of computed bins and the key. As the parameters are shuffled, even small change in their indexes should result in a significant difference of computed bin values – this step causes nonlinearity.

The set of parameters was chosen according to several rules – the greatest common denominator of p and q is equal to one and q is always positive. These two conditions were formulated by Guédon et al. [38] and ensure uniqueness of projection angles. Furthermore, as our approach uses MoT for specific purposes, we introduced other conditions:

- $|p| > 3$ ensures that the summed intensities belong to different pixels,
- $|p| \pmod 3 \neq 0$ provides pixel intensities from various color planes,
- values of b were chosen in a way that bins are always made as sums of exactly three intensities.

Step 2: Rows of extended image P_{ext} are scanned from top to bottom. Intensities of an actually scanned row are stored in a vector row_{act}. Intensities of a row that is two rows under the row_{act} are stored in a vector row_{modif}. Then, the following 144 intensities (columns first scanning pattern) are copied to vector vec_{int}. These operations are depicted in Figure 8. If row indexes for row_{modif} and vec_{int} are higher than height h of P_{ext}, the algorithm uses rows from the top of P_{ext}.

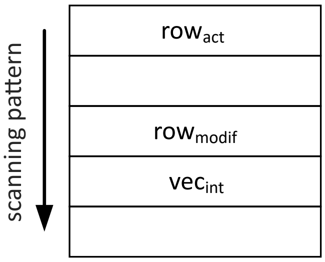


Figure 8. Operations with rows of extended image

Please note that pixel intensities rows of P_{ext} were already shuffled during confusion stage. Therefore the 144 intensities in vec_{int} were chosen from the whole image. Also, the requirement of 144 pixel intensities causes a restriction of minimal image resolution – the size of 16×16 pixels was chosen as the closest square resolution to 16×12 pixels.

The sixteen rows are necessary as one is actually scanned row (row_{act}), another is used for modifications (row_{modif}), two rows are utilized during plaintext unrelated chaining and the remaining 12 rows produce a block of 12×12 pixels required for computation of Mojette bins.

Step 3: Vector vec_{int} is rearranged to matrix mat_{int} with 12×12 elements (rows first scanning pattern). This matrix is used for computation of 256 bins by MoT with parameters from rearranged matrix mot_{par}. Bin values are stored in a vector vec_{bins}. The redundancy property of MoT causes that 256 bins are calculated from various triples of 144 pixel intensities (they are used multiple times).

Step 4: Intensities of row_{act} are element-wise combined with bins vec_{bins} by means of bitwise XOR (Equation (8)). A bin value for each intensity from row_{act} is chosen according to corresponding intensity from row_{modif}. The resulting vector of intensities is stored in the matching row of extended image P_{ext}.

$$P_{ext}(1{:}h, k) = P_{ext}(1{:}h, k) \oplus vec_{bins}\left(row_{modif}(k)\right) \tag{8}$$

where $1{:}h$ denotes sequence $1, 2, \ldots, h$, $k = 1, 2, \ldots, w_{ext}$ is column index and $\oplus$ is an operator of bitwise XOR.

3.2.6 Combination with Second Pseudo-Random Sequence

Inputs: extended image P_{ext}, number of color planes num_{cp}, width w, pseudo-random sequence seq'_{6m}.

Output: encrypted image E.

Step 1: Sequence seq'_{6m} is combined with extended image P_{ext} by means of bitwise XOR using Equation (9). This step suppresses possibility of attacks on the plaintext related diffusion stage.

$$P_{ext} = P_{ext} \oplus seq'_{6m}. \tag{9}$$

Step 2: Extended image P_{ext} is rearranged to encrypted image E. Grayscale images ($num_{cp} = 1$) are simply copied to E, while true color images ($num_{cp} = 3$) are combined from their individual color planes. Columns of red color plane of E are achieved from columns of P_{ext} with indexes $1 + 3 \cdot (i - 1)$, where $i = 1, 2, \ldots, w$. Columns of green and blue color planes of E are obtained from columns of P_{ext} with indexes $2 + 3 \cdot (i - 1)$, and $3 \cdot i$ where $i = 1, 2, \ldots, w$, respectively.

3.3 Decryption Algorithm

Steps of decryption algorithm are analogous to encryption. The opposite order of operations can be seen in Figure 2. One exception of the opposite order is decomposition to two-dimensional extended image E_{ext} and rearrangement to a matrix with one or three color planes. Generation and processing of the PRSs is the same as during encryption. Combinations with PRSs are swapped, the first one is done with sequence seq'_{6m}, while the second one uses seq'_{1m}. Plaintext unrelated inverse chaining utilizes subtraction instead of addition, the usage of all bitwise XOR operations stays the same. Also, the order of subtraction and bitwise XOR operations is reversed. The shifts of image pixels during inverse confusion utilize negative values of elements from sequences seq'_3 and seq'_2.

4 EXPERIMENTAL RESULTS

All experiments described in this section were performed on a PC running MATLAB R2015a on Windows 10 OS, with a 2.5 GHz Intel Core i7-6500U Skylake CPU and 12 GB of RAM. A set of experimental plain images is shown in Figure 9. Their parameters are mentioned in Table 4. Please note that images *black* and *blackG* are magnified in all following figures, as their low resolution enables detection of possible patterns by a naked eye.

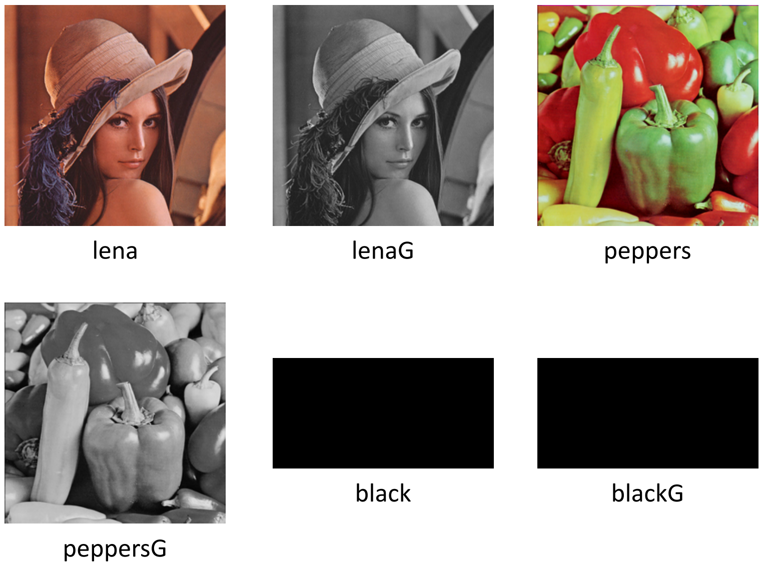


Figure 9. A set of experimental plain images

Image	*lena*	*lenaG*	*peppers*	*peppersG*	*black*	*blackG*
height [px]	512	512	512	512	16	16
width [px]	512	512	512	512	32	32
color depth [bits/px]	24	8	24	8	24	8

Table 4. Parameters of used plain images

A set of three experimental keys in hexadecimal notation is displayed in Table 5. Value of K_1 was obtained from the first 128 bits of binary representation of decimal part of π. The difference between K_1 and K_2 is highlighted by italics.

Key	Value
K_1	0× C4 EB 50 BC 0E C5 EB 50 BC 0E C5 EB 50 BC 0E C5
K_2	0× C4 EB *51* BC 0E C5 EB 50 BC 0E C5 EB 50 BC 0E C5
K_3	0× 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Table 5. A set of experimental keys

Encrypted versions of some plain images from Figure 9 are shown in Figure 10. All encryptions used key K_1.

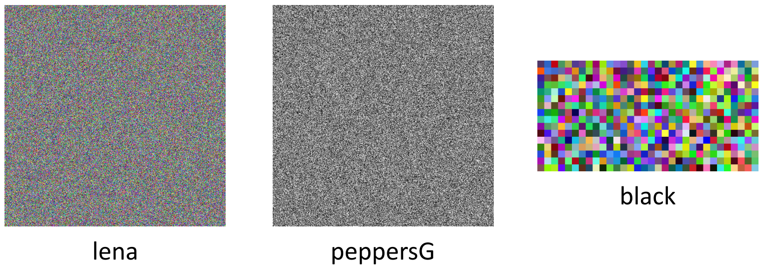


Figure 10. Some examples of encrypted images

4.1 Size of Key Space

The proposed algorithm uses key with length of 16 bytes. Therefore the size of key space can be expressed as $256^{16} = 2^{8 \cdot 16} = 2^{128}$. Considering that one decryption of color image with resolution of 512×512 pixels takes approx. 420 ms (refer to Section 4.4 for details), the brute-force attack would require approx. 2.7192×10^{32} years to complete. Therefore we consider this type of attack as infeasible.

4.2 Key and Plaintext Sensitivity Analysis

4.2.1 Key Sensitivity

Key sensitivity of the proposed algorithm is illustrated in Figure 11. The top row of images was created by encryption of plain image *lena* by three various keys. Left image in the bottom row shows differences between images encrypted by two different keys. The other two images in the bottom row illustrate decryption by the correct key (middle image) and by incorrect key (right image).

4.2.2 Plaintext Sensitivity

Sensitivity of encryption algorithm to slight changes of plaintext (in the form of plain image before encryption or encrypted image before decryption) can be demonstrated by two simple experiments.

The first experiment increases intensity of the last scanned pixel (in the bottom right corner of blue color plane) of image *black*. The pixel intensity is increased by one level from 0 to 1. The left and middle images in Figure 12 illustrate effect of one different pixel intensity before encryption with the same key.

The second experiment starts with encryption of image *black* by key K_1. The result is shown in left image in Figure 12. Then, the encrypted image is modified by increasing the intensity of last scanned image pixel (in the top left corner of red color plane) of encrypted image. The modified image is finally decrypted by key K_1. The decryption without modification should lead to the original plain image *black*, as

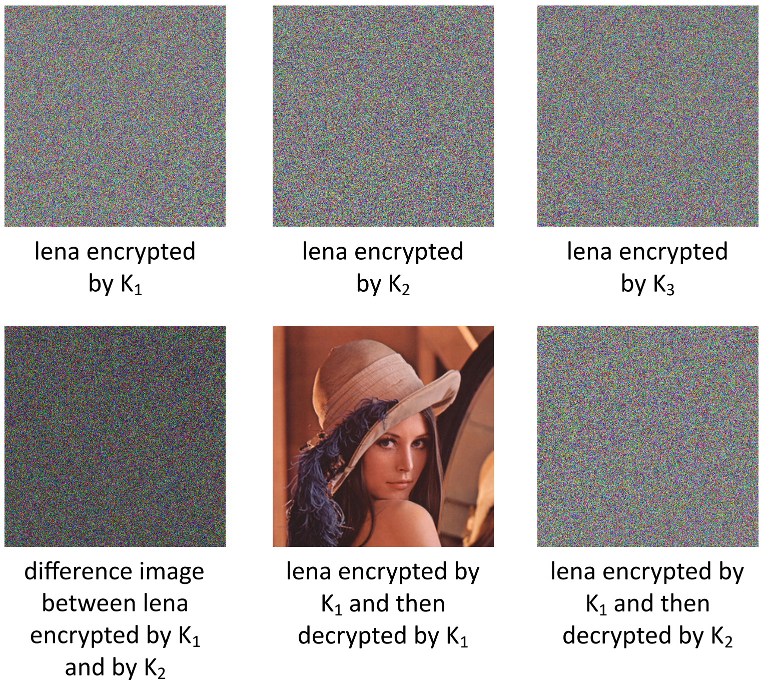

lena encrypted by K_1
lena encrypted by K_2
lena encrypted by K_3
difference image between lena encrypted by K_1 and by K_2
lena encrypted by K_1 and then decrypted by K_1
lena encrypted by K_1 and then decrypted by K_2

Figure 11. Demonstration of key sensitivity

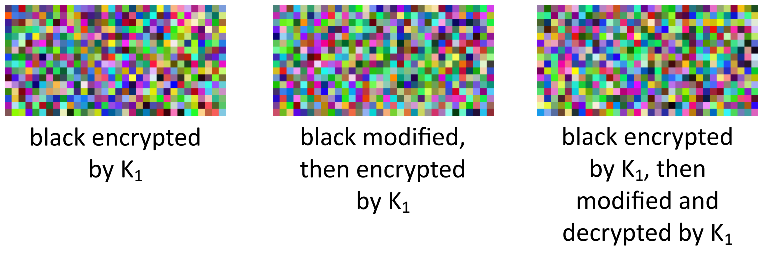

black encrypted by K_1
black modified, then encrypted by K_1
black encrypted by K_1, then modified and decrypted by K_1

Figure 12. Effects of modifications of image before and after encryption

shown in Figure 9. However, as seen in the right image in Figure 12, the decryption of modified image produced a totally different image.

4.3 Robustness Against Certain Types of Attacks

This section investigates the robustness of the proposed image encryption algorithm against commonly used attacks in a field of image encryption. The ways how measures used for assessment of the robustness are computed can be interpreted as examples of certain types of well known attacks.

In the case that the proposed image encryption algorithm improves values of these measures, it can be stated that the algorithm resists not only the mentioned basic attacks, but also some of the more complicated attacks based on these general attacks. A detailed analysis of robustness against all published attacks would be extensive, so this paper investigates only robustness against certain general attacks.

4.3.1 Statistical Attacks

Robustness of image encryption algorithms against statistical attacks can be examined by several measures. First of all, usage of encryption should reduce significant peaks present in the histogram of a plain image. This situation is illustrated in Figure 13 by histograms of plain image *lenaG* and its version encrypted by key K_1.

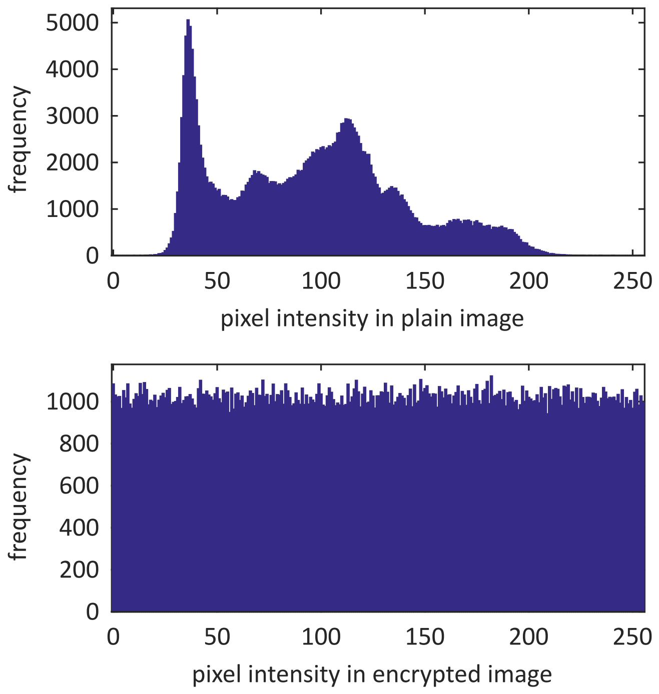


Figure 13. Comparison of image histograms

Secondly, the encryption algorithms should suppress the correlation of adjacent image pixels. This feature can be shown by correlation diagrams, which use intensities of two pixels as coordinates of one plotted point. If pixel intensities are highly correlated, the diagram should have points near line $y = x$. Encryption should result in nearly uniform distribution of points in these diagrams. An example of correlation diagrams for plain image *lenaG* and its version encrypted by key K_1 is shown in Figure 14. Both diagrams contain 1 000 points, which were plotted according to intensities of 1 000 randomly chosen pairs of horizontally adjacent image pixels.

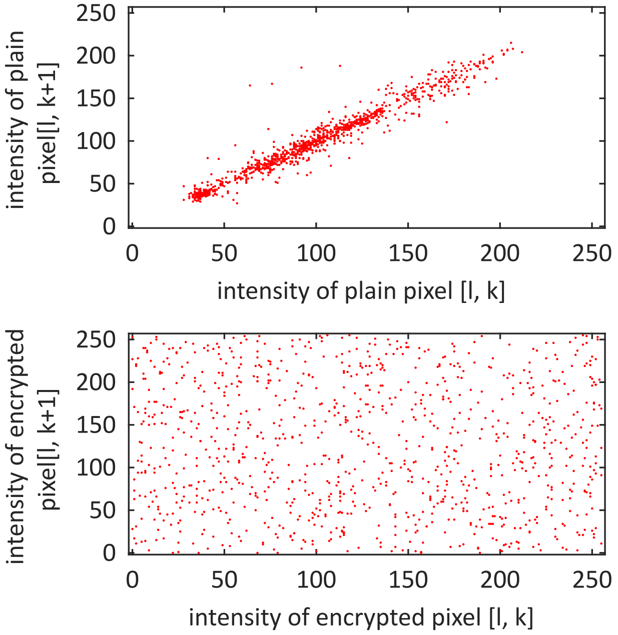


Figure 14. An example of correlation diagrams

Various image encryption algorithms can be compared by numerical parameters. These measures include correlation coefficients ρ, which are calculated separately for horizontally (ρ_h), vertically (ρ_v) and diagonally (ρ_d) adjacent pairs of image pixels and entropy H. Both correlation coefficients and entropy are computed individually for each color plane of the analyzed image.

Correlation coefficients ρ can be calculated by Equation (10):

$$\rho = \frac{\sum_{pp=1}^{num_{pp}} \left(vec_1(pp) - \overline{vec_1}\right) \cdot \left(vec_2(pp) - \overline{vec_2}\right)}{\sqrt{\sum_{pp=1}^{num_{pp}} \left(vec_1(pp) - \overline{vec_1}\right)^2 \cdot \sum_{pp=1}^{num_{pp}} \left(vec_2(pp) - \overline{vec_2}\right)^2}} \text{ [-]} \qquad (10)$$

where $pp = 1, 2, \ldots, num_{pp}$ is an index of pixel pair, num_{pp} is total amount of pixel pairs, vectors vec_1 and vec_2 contain intensities of the first and the second pixels from pixel pairs, respectively, and $\overline{vec}$ denotes arithmetic mean of vector vec.

Entropy H is computed by applying Equation (11):

$$H = -\sum_{in=0}^{2^L-1} p(in) \cdot \log_2\left(p(in)\right) \text{ [bits/px]} \qquad (11)$$

where L is color depth of color plane, in denotes intensity of image pixel and $p(in)$ stands for probability of occurrence of pixel with intensity in. The theoretical upper bound of entropy H is given by color depth of the color planes.

Computed values of correlation coefficients ρ and entropy H are presented in columns 4 to 7 of Table 6. Symbol "–" denotes either the only color plane for grayscale images (in the second column) or usage of plain images (in the third column).

4.3.2 Differential Attacks

Differential attacks are used for revealing properties of image encryption algorithms by comparing two encrypted images E_1 and E_2. These two images were created by encryption of plain images P_1 and P_2, where P_2 is slightly modified version of P_1. The modification should be minimal, i. e. intensity of one image pixel from one color plane is changed by one level.

Robustness of image encryption algorithms against differential attacks is tested by two measures: Number of Pixel Change Ratio (NPCR) and Unified Average Changing Intensity (UACI) [42]. While the first one only counts the amount of different pixels, the second one also takes into account the size of differences.

NPCR for images E_1 and E_2 is computed as Equation (12):

$$NPCR = \frac{100}{h \cdot w} \sum_{l=1}^{h} \sum_{k=1}^{w} Diff_{mat}(l,k)\ [\%] \tag{12}$$

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

UACI for the same pair of images is calculated as Equation (13):

$$UACI = \frac{100}{h \cdot w} \sum_{l=1}^{h} \sum_{k=1}^{w} \frac{|E_1(l,k) - E_2(l,k)|}{2^L - 1}\ [\%] \tag{13}$$

where L is color depth of color plane.

Calculated values of NPCR and UACI are included in columns 8 and 9 of Table 6. Each value is an arithmetic mean of 100 repeated measurements with randomly chosen modified pixel intensity in the plain image. Symbol "–" denotes either the only color plane for grayscale images (in second column) or usage of plain images (in third column). NPCR and UACI of plain images could not be computed as the modification of one plain image pixel intensity is not spread to other intensities (images are not encrypted).

The paper by Wu et al., which analyzed NPCR and UACI [42] also mentions so-called expected values of these parameters for certain resolutions of encrypted images. If the computed values of NPCR and UACI are greater than the expected values, it can be concluded that encryption algorithm successfully suppressed the similarity of plain images P_1 and P_2. Wu et al. also defined significance levels α, which can be used for predicting amount of successful differential attacks on a pair of encrypted images.

The computed expected values of NPCR and UACI for a color plane with resolution of 512×512 pixels are 99.6094 % for NPCR and 33.4635 % for UACI [42].

4.3.3 Discussion

Image and Color Plane		Key	ρ_h [–]	ρ_v [–]	ρ_d [–]	H [bits/px]	NPCR [%]	UACI [%]
lena	R	–	0.9749	0.9782	0.9593	7.5889	not computed	
	G		0.9632	0.9729	0.9497	7.106		
	B		0.9376	0.9515	0.9212	6.8147		
	R	K_1	0.0004	0.0013	0.0005	7.9993	99.6101	33.4738
	G		−0.001	−0.0021	−0.0037	7.9994	99.6109	33.4742
	B		−0.0032	−0.002	0.0017	7.9992	99.6103	33.4746
	R	K_2	0.0033	0.0026	−0.001	7.9994	99.6105	33.4742
	G		0.0006	0.0015	−0.0016	7.9993	99.6113	33.4749
	B		0.0016	0.0013	0.003	7.9993	99.6101	33.4743
	R	K_3	0.0028	−0.0012	0.0002	7.9993	99.6101	33.4736
	G		−0.0007	0.002	0.0002	7.9993	99.6109	33.4731
	B		−0.0042	−0.0017	0.0005	7.9993	99.6098	33.4744
lenaG	–	–	0.9679	0.9761	0.955	7.2344	not computed	
		K_1	0.0008	0.0005	−0.0004	7.9992	99.61	33.4714
		K_2	0.0015	0.001	−0.0012	7.9993	99.6099	33.4725
		K_3	0.0051	−0.0004	−0.0001	7.9993	99.6105	33.4704
peppers	R	–	0.9635	0.9663	0.9564	7.3388	not computed	
	G		0.9811	0.9818	0.9687	7.4963		
	B		0.9665	0.9664	0.9475	7.0583		
	R	K_1	0.0007	−0.005	0.0009	7.9994	99.6112	33.4719
	G		−0.0009	0.0006	−0.0018	7.9993	99.6106	33.4727
	B		0.0013	−0.0011	0.0026	7.9993	99.611	33.4712
	R	K_2	−0.0007	0.0004	−0.0026	7.9994	99.6114	33.4707
	G		−0.0028	−0.0015	0.0031	7.9993	99.6103	33.4718
	B		−0.0005	−0.0019	0.0011	7.9993	99.6108	33.4723
	R	K_3	0.0034	−0.0005	0.0006	7.9992	99.61	33.471
	G		−0.0025	−0.0014	−0.0007	7.9994	99.6102	33.4708
	B		−0.0035	0.0003	0.0006	7.9992	99.6109	33.4714
peppersG	–	–	0.9768	0.9792	0.9639	7.5944	not computed	
		K_1	0.0051	0.0007	−0.0001	7.9991	99.6106	33.4721
		K_2	0.0019	−0.0011	−0.0008	7.9991	99.6121	33.4729
		K_3	−0.0002	−0.0006	−0.0016	7.9993	99.6117	33.4735

Table 6. Achieved numerical parameters

Results presented in Table 6 show that encryption by the proposed algorithm helps to decrease values of correlation coefficients ρ. Also, the values of ρ are quite similar for different keys. The biggest difference of ρ caused by usage of other key is present for image *peppersG* where key K_1 produces $\rho_h = 0.0051$, while other keys produce values of 0.0019 and -0.0002, respectively. However, the high value of ρ_h is balanced by values of ρ_v and ρ_d, which are better for K_1. Values of ρ for other two images (*black* and *blackG*) are affected by their low resolution and the simplicity of their scene (all image pixels have zero intensity). Encryption of image *blackG* by key K_1 resulted in $\rho_h = -0.0191$, $\rho_v = -0.0616$ and $\rho_d = -0.0181$.

The encryption also increases entropy H of the images. Obtained values are close to theoretical bound of 8 bits per pixel and they are more uniform among various color planes. Usage of different keys does not significantly affect the values of H. Image *blackG* encrypted by key K_1 produced $H = 7.5929$ bits per pixel.

Values of NPCR and UACI are similar for various combinations of color planes and keys applied on individual images. Images *black* and *blackG* have greater variance of these values as their low resolution makes even slight differences in amount of changes (NPCR) or their intensity (UACI) more noticeable. Examples of NPCR and UACI for image *blackG* encrypted by key K_1 are 99.6719 % and 33.4304 %, respectively.

All arithmetic means of NPCR and UACI for images *lena*, *lenaG*, *peppers* and *peppersG* are higher than the expected values. However, some of the 100 measurements have lower values than the expected value, which causes limited confidence about possibility of a successful differential attack. Figure 15 illustrates 100 repeated measurements of NPCR for image *lenaG* encrypted by key K_1. The displayed significance level $\alpha = 0.001$ results in a confidence level of 99.9 % (NPCR > 99.5717 %). Therefore 1 out of 1 000 predictions of a possible differential attack can be wrong [42].

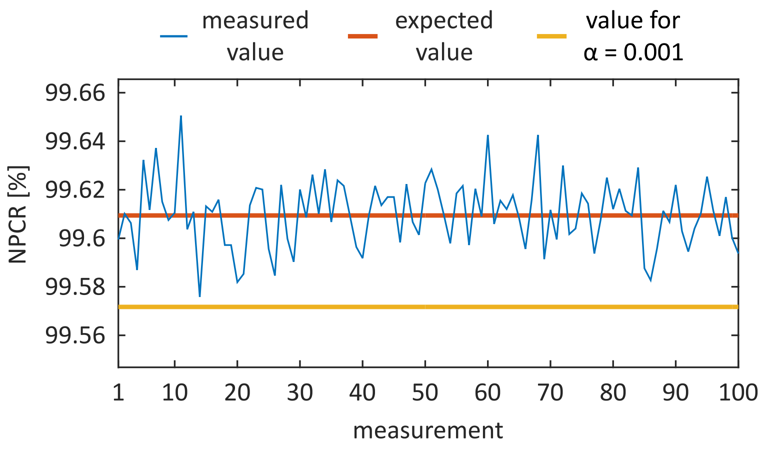


Figure 15. The illustration of 100 measured NPCR values

The properties of the proposed image encryption algorithm regarding commonly used attacks in the field of image encryption can be evaluated from several aspects. Firstly, the key space of the proposed algorithm is big enough for preventing brute-force attacks. Secondly, the design of the proposed algorithm ensures certain level of sensitivity to both used keys and plain or encrypted images. Also, the architecture of the proposed algorithm contains a plaintext related stage, which eliminates application of Solak's attack [13]. Finally, the robustness against statistical and differential attacks can be evaluated by values of correlation coefficients ρ, entropy H, NPCR and UACI.

Therefore it can be concluded that the proposed image encryption algorithm is robust against all currently known attacks in the field of image encryption.

4.4 Analysis of Computational Speed

Encryption and decryption times are other important properties of the image encryption algorithms. The speed of performed operations can be also evaluated via encryption speed v_{enc} or decryption speed v_{dec}, which take into account the resolution and color depth of images (Equation (14)):

$$v_{oper} = \frac{h \cdot w \cdot d}{2^{20} \cdot t_{oper}} \text{ [MB/s]} \tag{14}$$

where h, w and d are height, width and color depth of image, 2^{20} is a constant representing amount of bytes in a megabyte and t_{oper} is time in seconds needed for an operation (either encryption or decryption).

Considering different specifications of various computers used for experiments, the speed of image encryption algorithms can be given by the number of processor cycles required for operations with one byte of data. These values are denoted as cyc_{enc} for encryption of one image byte and cyc_{dec} for decryption of one image byte. The number of cycles necessary for an operation (either encryption or decryption) with one byte of data cyc_{oper} can be computed by Equation (15):

$$cyc_{oper} = \frac{f_{cpu}}{v_{oper}} \text{ [cycles/B]} \tag{15}$$

where f_{cpu} is processor clock frequency measured in Hz and v_{oper} is speed of the investigated operation given in B/s. Please note that the computed values of cycles required for operations with one byte expect that only one core of the processor is utilized (at 100 %) for the purposes of the image encryption algorithms.

Times, speeds and numbers of processor cycles for operations with images and keys from the experimental set are presented in Table 7. The times are arithmetic means of 100 repeated measurements. Speeds and numbers of the processor cycles were computed from these means.

Results shown in Table 7 lead to several conclusions. Firstly, the speed of operations does not depend on the key used. Secondly, both the encryption or

Image	Key	t_{enc} [ms]	t_{dec} [ms]	v_{enc} [MB/s]	v_{dec} [MB/s]	cyc_{enc} [cycles/B]	cyc_{dec} [cycles/B]
lena	K_1	436.2245	420.5649	1.7193	1.7833	1454.08	1401.9
	K_2	436.3845	420.3934	1.7187	1.784	1454.59	1401.35
	K_3	436.7181	421.081	1.7174	1.7811	1455.69	1403.63
lenaG	K_1	126.2499	122.7206	1.9802	2.0371	1262.5	1227.23
	K_2	126.9473	122.6332	1.9693	2.0386	1269.49	1226.33
	K_3	126.905	122.6084	1.97	2.039	1269.04	1226.09
peppers	K_1	436.3264	419.8888	1.7189	1.7862	1454.42	1399.62
	K_2	436.4397	421.8564	1.7185	1.7779	1454.76	1406.15
	K_3	435.8724	419.6389	1.7207	1.7873	1452.9	1398.76
peppersG	K_1	127.1529	122.8078	1.9661	2.0357	1271.55	1228.08
	K_2	126.8485	122.7695	1.9709	2.0363	1268.46	1227.72
	K_3	127.3672	122.8902	1.9628	2.0343	1273.69	1228.92
black	K_1	2.557	2.6487	0.5729	0.553	4363.76	4520.8
	K_2	2.5451	2.6251	0.5756	0.558	4343.29	4480.29
	K_3	2.535	2.6628	0.5779	0.5501	4326.01	4544.63
blackG	K_1	1.5223	1.5587	0.3207	0.3133	7795.45	7979.57
	K_2	1.5124	1.564	0.3229	0.3122	7742.34	8007.69
	K_3	1.4991	1.5803	0.3257	0.309	7675.78	8090.61

Table 7. Measured times, speeds and the numbers of the processor cycles

decryption speeds and the number of the processor cycles required for processing of one byte of data for images with the same resolution and color depth are very similar. Also, the speeds for images with a higher resolution (such as *lena* and *peppers*) are lower than the speeds for images with a lower resolution. This can be caused by generating of longer PRSs since LM (1) is a recursive function where each currently generated iterate (x_{n+1}) requires the previous iterate (x_n) for its computations. The lower speeds result in an increased number of the processor cycles required for processing of one byte of data.

The values for images *black* and *blackG* are distorted by rather low resolution of the images (32×16 pixels). For images with resolution of 512×512 pixels, the encryption speeds v_{enc} are approx. 1.7 MB/s for true color images and approx. 1.95 MB/s for grayscale images. The decryption speeds v_{dec} are slightly higher.

4.5 Comparison with Other Approaches

The comparison of numeric parameters obtained by image encryption algorithms is not an easy task, as researchers tend to use different plain images and various measures. This section summarizes results reported in papers [20, 21, 24, 26, 27, 29, 30, 31, 32], which all describe plaintext related image encryption algorithms. The comparison of parameters presented in Table 8 is divided into two parts – the first part includes values obtained by papers that used true color image *lena*, while the second part shows values from papers that utilized grayscale image *lenaG*.

Results for this paper were obtained by using key K_1. Some papers used plain images with different resolution: refs. [20, 30] used resolution of 256×256 pixels, while refs. [24, 26] used resolution of 357×317 pixels. Algorithm published in [27] was already broken in [28]. An approach presented in [32] is asymmetric (meaning that the encryption and decryption keys are different).

The numbers of the processor cycles required for encryption of one byte cyc_{enc} were calculated with respect to both resolutions of used images and specifications of the computers used. All compared approaches obtained the encryption times used for calculation of cyc_{enc} in various versions of the MATLAB computational environment. None of the compared algorithms mentioned the usage of multiple processor cores.

If some paper presented multiple values for one parameter, the best results were chosen for the comparison. The most outstanding value was highlighted by italics for each parameter.

Approach	ρ_h [–]	ρ_v [–]	ρ_d [–]	H [bits/px]	NPCR [%]	UACI [%]	cyc_{enc} [cycles/B]
red color plane of true color image *lena* (512×512 pixels)							
proposed	*0.0004*	*0.0013*	*0.0005*	7.9993	99.6101	33.4738	1454.08
ref. [21]	−0.0029	−0.015	0.0129	*7.9997*	99.62	*33.51*	∼2270
ref. [27]	0.0135	not reported		7.9974	*99.63*	33.31	*648.53*
grayscale image *lenaG* (512×512 pixels)							
proposed	*0.0008*	*0.0005*	*−0.0004*	7.9992	99.61	33.4714	1262.5
ref. [20]	0.0088	−0.0087	−0.006	7.9902	*99.62*	33.46	9063.44
ref. [24]	−0.0046	−0.0511	−0.0168	*7.9993*	99.6101	33.4679	8230.32
ref. [26]	−0.0084	0.0041	−0.0463	7.9984	99.6077	33.4441	3366.6
ref. [29]	0.0044	0.0151	0.0012	*7.9993*	*99.62*	33.45	15120.97
ref. [30]	−0.0037	−0.0029	0.0047	7.9975	99.5956	*33.5512*	43151.97
ref. [31]	0.0013	0.0008	0.0066	*7.9993*	99.6107	33.4436	5185.19
ref. [32]	0.0042	0.0022	−0.0045	7.9992	99.1802	33.3483	*795.17*

Table 8. Comparison of numerical parameters

As seen in Table 8, the proposed solution provides the best values of correlation coefficients ρ among presented values. The results for entropy H are similar, and in most cases close to the theoretical bound of 8 bits per pixel. The highest entropy value for red color plane of color image *lena* is achieved by approach [21], while highest entropy values for grayscale image *lenaG* are obtained by algorithms [24, 29, 31].

Majority of NPCR and UACI results are higher than the expected values. The best value of NPCR for color images was obtained by design [27], however this algorithm was already broken [28]. NPCR values for grayscale images are similar, except for proposal [32]. The best values were produced by approaches [20, 29]. The best UACI values were achieved by algorithm [21] for true color images and by approach [30] for grayscale images.

Reported numbers of the processor cycles required for encryption of one byte cyc_{enc} are greatly influenced by the architecture of the algorithms. This fact is most visible in the approaches [29, 30], which used hash functions and the algorithm [27], which inserted plaintext related parameter into the encrypted image. While hash functions significantly slow down the whole algorithms, the other solution needs a relatively small number of processor cycles for the encryption. However, as it was already pointed out, this solution was already broken [28]. Another approach with different properties was described in [32], where an asymmetric image encryption algorithm was proposed. Except these solutions, the design proposed in this paper is the fastest.

5 CONCLUSIONS

This paper described an image encryption algorithm, which employed the Mojette transform for plaintext related chaining of pixel intensities. The values of Mojette bins, which are going to be combined with image pixels are chosen by pixel intensities that are yet not encrypted, therefore this proposal is plaintext related. Moreover, the Mojette bins are computed also from pixel intensities of plain images.

Parameters of Mojette bins were chosen in a way that each bin is affected by pixel intensities from various color planes. Also, the pixel intensities that are summed by the Mojette transform are chosen from different columns of the processed images.

As the combinations of images and computed Mojette bins are done with whole rows of pixel intensities, the proposed solution is among the fastest plaintext related image encryption designs. Also, the experimental results show that the algorithm proposed in this paper reaches excellent values of correlation coefficients. The obtained values of entropy, NPCR and UACI are comparable with other proposals.

Probably the biggest drawback of the proposed algorithm is the fact that not all of 100 performed NPCR and UACI measurements are higher than the expected values. This disadvantage could be improved in the future.

Acknowledgment

The work was supported by the research grant APVV-17-0208.

REFERENCES

[1] Zhang, Y.-Q.—Wang, X.-Y.: A Symmetric Image Encryption Algorithm Based on Mixed Linear-Nonlinear Coupled Map Lattice. Information Sciences, Vol. 273, 2014, pp. 329–351, doi: 10.1016/j.ins.2014.02.156.

[2] Hua, Z.—Zhou, Y.: Image Encryption Using 2D Logistic-Adjusted-Sine Map. Information Sciences, Vol. 339, 2016, pp. 237–253, doi: 10.1016/j.ins.2016.01.017.

[3] Gueron, S.: Intel Advanced Encryption Standard (AES) New Instructions Set. Cited 2019-04-07. Available online at: `https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf`.

[4] Liu, L.—Miao, S.: A New Image Encryption Algorithm Based on Logistic Chaotic Map with Varying Parameter. SpringerPlus, Vol. 5, 2016, No. 1, pp. 289–300, doi: 10.1186/s40064-016-1959-1.

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

[6] Chen, G.—Mao, Y.—Chui, C. K.: A Symmetric Image Encryption Scheme Based on 3D Chaotic Cat Maps. Chaos, Solitons and Fractals, Vol. 21, 2004, No. 3, pp. 749–761, doi: 10.1016/j.chaos.2003.12.022.

[7] Pareek, N. K.—Patidar, V.—Sud, K. K.: Image Encryption Using Chaotic Logistic Map. Image and Vision Computing, Vol. 24, 2006, No. 9, pp. 926–934, doi: 10.1016/j.imavis.2006.02.021.

[8] Wong, K.-W.—Kwok, B. S.-H.—Law, W.-S.: A Fast Image Encryption Scheme Based on Chaotic Standard Map. Physics Letters A, Vol. 372, 2008, No. 15, pp. 2645–2652, doi: 10.1016/j.physleta.2007.12.026.

[9] Ye, G.—Wong, K.-W.: An Efficient Chaotic Image Encryption Algorithm Based on a Generalized Arnold Map. Nonlinear Dynamics, Vol. 69, 2012, No. 4, pp. 2079–2087, doi: 10.1007/s11071-012-0409-z.

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

[11] Li, S.—Zheng, X.: Cryptanalysis of a Chaotic Image Encryption Method. Proceedings of the 2002 IEEE International Symposium on Circuits and Systems (ISCAS 2002), Phoenix-Scottsdale, May 2002, pp. 708–711, doi: 10.1109/ISCAS.2002.1011451.

[12] Rhouma, R.—Belghith, S.: Cryptanalysis of a New Image Encryption Algorithm Based on Hyper-Chaos. Physics Letters A, Vol. 372, 2008, No. 38, pp. 5973–5978, doi: 10.1016/j.physleta.2008.07.057.

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

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

[15] Broda, M.—Hajduk, V.—Levický, D.: Universal Statistical Steganalytic Method. Journal of Electrical Engineering, Vol. 68, 2017, No. 2, pp. 117–124, doi: 10.1515/jee-2017-0016.

[16] Oravec, J.—Turán, J.: Substitution Steganography with Security Improved by Chaotic Image Encryption. Proceedings of the 2017 IEEE 14th International Scientific

Conference on Informatics (Informatics 2017), Poprad, November 2017, pp. 284–288, doi: 10.1109/INFORMATICS.2017.8327261.

[17] Fang, D.—Sun, S.: A New Scheme for Image Steganography Based on Hyperchaotic Map and DNA Sequence. Journal of Information Hiding and Multimedia Signal Processing, Vol. 9, 2018, No. 2, pp. 392–399.

[18] Abundiz-Pérez, F.—Cruz-Hernández, C.—Murillo-Escobar, M. A.—López-Gutiérrez, R. M.—Arellano-Delgado, A.: A Fingerprint Image Encryption Scheme Based on Hyperchaotic Rössler Map. Mathematical Problems in Engineering, Vol. 2016, 2016, Art. No. 2670494, pp. 1–15, doi: 10.1155/2016/2670494.

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

[20] Fu, C.—Chen, J.-J.—Zou, H.—Meng, W.-H.—Zhan, Y.-F.—Yu, Y.-W.: A Chaos-Based Digital Image Encryption Scheme with an Improved Diffusion Strategy. Optics Express, Vol. 20, 2012, No. 3, pp. 2363–2378, doi: 10.1364/OE.20.002363.

[21] Kanso, A.—Ghebleh, M.: A Novel Image Encryption Algorithm Based on a 3D Chaotic Map. Communications in Nonlinear Science and Numerical Simulation, Vol. 17, 2012, No. 7, pp. 2943–2959, doi: 10.1016/j.cnsns.2011.11.030.

[22] Arroyo, D.—Rhouma, R.—Alvarez, G.—Li, S.—Fernandez, V.: On the Security of a New Image Encryption Scheme Based on Chaotic Map Lattices. Chaos: An Interdisciplinary Journal of Nonlinear Science, Vol. 18, 2008, No. 3, pp. 1–8, doi: 10.1063/1.2959102.

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

[24] Zhang, Y.: A Chaotic System Based Image Encryption Algorithm Using Plaintext-Related Confusion. TELKOMNIKA Indonesian Journal of Electrical Engineering and Computer Science, Vol. 12, 2014, No. 11, pp. 7952–7962.

[25] Fan, H.—Li, M.: Cryptanalysis and Improvement of Chaos-Based Image Encryption Scheme with Circular Inter-Intra-Pixels Bit-Level Permutation. Mathematical Problems in Engineering, Vol. 2017, 2017, Art. No. 8124912, 11 pp., doi: 10.1155/2017/8124912.

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

[27] Murillo-Escobar, M. A.—Cruz-Hernández, C.—Abundiz-Pérez, F.—López-Gutiérrez, R. M.—Acosta Del Campo, O. R.: A RGB Image Encryption Algorithm Based on Total Plain Image Characteristics and Chaos. Signal Processing, Vol. 109, 2015, pp. 119–131, doi: 10.1016/j.sigpro.2014.10.033.

[28] Fan, H.—Li, M.—Liu, D.—An, K.: Cryptanalysis of a Plaintext-Related Chaotic RGB Image Encryption Scheme Using Total Plain Image Characteristics. Multimedia Tools and Applications, Vol. 77, 2018, No. 15, pp. 20103–20127, doi: 10.1007/s11042-017-5437-8.

[29] Chai, X.—Gan, Z.—Zhang, M.: A Fast Chaos-Based Image Encryption Scheme with a Novel Plain Image-Related Swapping Block Permutation and Block Diffusion. Multimedia Tools and Applications, Vol. 76, 2017, No. 14, pp. 15561–15585, doi: 10.1007/s11042-016-3858-4.

[30] Wang, X.—Zhu, X.—Wu, X.—Zhang, Y.: Image Encryption Algorithm Based on Multiple Mixed Hash Functions and Cyclic Shift. Optics and Lasers in Engineering, Vol. 107, 2018, pp. 370–379, doi: 10.1016/j.optlaseng.2017.06.015.

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

[32] Oravec, J.—Turán, J.—Ovseník, Ľ.—Ivaniga, T.—Solus, D.—Márton, M.: Asymmetric Image Encryption Approach with Plaintext-Related Diffusion. Radioengineering, Vol. 27, 2018, No. 1, pp. 281–288, doi: 10.13164/re.2018.0281.

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

[34] Gleick, J.: Chaos. Vintage Books, London, UK, 1998.

[35] Guédon, J.-P.—Barba, D.—Burger, N.: Psychovisual Image Coding via an Exact Discrete Radon Transform. Proceedings of Visual Communications and Image Processing '95 (VCIP 95), Taipei, April 1995. Proceedings of the SPIE, Vol. 2501, 1995, pp. 562–572, doi: 10.1117/12.206765.

[36] Guédon, J.-P.: The Mojette Transform. Wiley, London, UK, 2009.

[37] Turán, J.—Szoboszlai, P.—Vásárhelyi, J.: Mojette Transform Software – Hardware Implementations and Its Applications. Infocommunications Journal (Híradástechnika), Vol. 3, 2011, No. 1, pp. 39–47.

[38] Guédon, J.-P.—Normand, N.: The Mojette Transform: The First Ten Years. In: Andres, E., Damiand, G., Lienhardt, P. (Eds.): Discrete Geometry for Computer Imagery (DGCI 2005). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3429, 2005, pp. 79–91, doi: 10.1007/978-3-540-31965-8_8.

[39] Verbert, P.—Ricordel, V.—Guédon, J.-P.: Analysis of Mojette Transform Projections for an Efficient Coding. Proceedings of Workshop on Image Analysis for Multimedia Interactive Services 2004 (WIAMIS 2004), Lisboa, April 2004, pp. 1–4.

[40] Oravec, J.—Ovseník, Ľ.—Turán, J.: An Iterative Approach for Image Coding Using Mojette Transform. Proceedings of the 2016 26th International Conference Radioelektronika (Radioelektronika 2016), Košice, April 2016, pp. 1–4, doi: 10.1109/RADIOELEK.2016.7477351.

[41] Oravec, J.—Turán, J.—Ovseník, Ľ.: LSB Steganography with Usage of Mojette Transform for Secret Image Scrambling. Proceedings of 23rd International Conference on Systems, Signals and Image Processing (IWSSIP 2016), Bratislava, May 2016, pp. 1–4, doi: 10.1109/IWSSIP.2016.7502754.

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

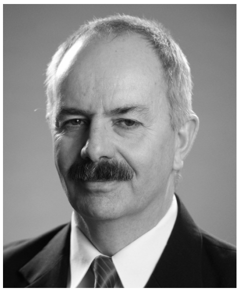
Ľuboš Ovseník received his M.Sc. and Ph.D. degrees from the Department of Electronics and Multimedia Communications, Technical University of Košice in 1990 and 2002. He works at the Technical University of Košice, currently as Associate Professor. His research interests are fiber optic communication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

Ján Turán received Ing. (M.Sc.) degree in physical engineering with honours from the Czech Technical University, Prague, Czech Republic, in 1974, and R.N.Dr. degree in experimental physics with honours from Charles University, Prague, Czech Republic, in 1980. He received a C.Sc. (Ph.D.) and Dr.Sc. (D.Sc.) 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.

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

Jakub Oravec received his M.Sc. and Ph.D. degrees from the Department of Electronics and Multimedia Communications, Technical University of Košice in 2015 and 2019. The topic of his dissertation was researching transform techniques in the new fields of image processing. His research interests included image encryption, steganography and digital image processing.

Ondrej Kováč received his M.Sc. and Ph.D. degrees from the Department of Electronics and Multimedia Communications, Technical University of Košice in 2011 and 2015. He works at the Technical University of Košice, currently as Assistant Professor. His Ph.D. thesis was focused on texture generation, 3D modeling and coding of human head.



Milan Oravec received Ing. (M.Sc.) and C.Sc. (Ph.D.) degrees from the Department of Safety and Quality of Production, Technical University of Košice in 1984 and 1992. He works at the Technical University of Košice, currently as Full Professor. His research interests include prevention of industrial accidents and safety assessments of critical infrastructure systems.