An image encryption algorithm with a plaintext-related quantisation scheme

Jakub Oravec | Štefanová Ovseník | Ján Papaj

Department of Electronics and Multimedia Communication, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Košice, Slovakia

Correspondence
Jakub Oravec, Department of Electronics and Multimedia Communication, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Námecovej 32, 040 01 Košice, Slovakia.
Email: jakub.oravec@tuke.sk

Funding information
Agentúra na Podporu Výskumu a Vývoja, Grant/Award Number: APVV-17-0208

Abstract
This paper describes an image encryption algorithm that utilises a plaintext-related quantisation scheme. Various plaintext-related approaches from other algorithms are presented and their properties are briefly discussed. Main advantage of the proposed solution is the achievement of a similar behaviour like that of more complex approaches with a plaintext-related technique used in a rather simple step such as quantisation. This design should result in a favourable computational complexity of the whole algorithm. The properties of the proposal are evaluated by a number of commonly used numerical parameters. Also, the statistical properties of a pseudo-random sequence that is quantised according to the plain image pixel intensities are investigated by tests from NIST 800-22 suite. Obtained results are compared to values reported in related works and they imply that the proposed solution produces encrypted images with comparable statistical properties but authors’ design is faster and more efficient.

1 | INTRODUCTION

Development of the first encryption algorithms designed specifically for images dates to the second half of the 1990s [1, 2]. Since the early approaches exploited representation of digital images and perceived image encryption only as a permutation of pixel intensities and a combination with pseudo-random (PR) sequence, they were viewed as a quick and quite effective tool [2]. However, some design flaws were reported since then and the comparison with conventional encryption schemes such as Advanced Encryption Standard (AES) limited the usage of dedicated image encryption algorithms to certain specific applications. These include mainly applications where a change of data type is not possible such as:

- encryption of images before embedding by means of image steganography [3, 4],
- security improvement for medical images [5] and some biometric systems [6],
- performance improvement in some simple biometric systems that do not enable hardware acceleration necessary for more complex encryption schemes [6].

The need for solutions in the mentioned areas helped in the further development of dedicated image encryption approaches. One of their unresolved issues was identified by Solak et al. in 2010 [7] as a relatively weak sensitivity of the image encryption algorithms to small differences in pixel intensities of plain images. It was found out that differential attacks could identify whole steps of some older algorithms or reveal some portions of used key. This problem was later analysed also by Xie et al. in 2017 [8].

Several solutions have been proposed for solving the mentioned sensitivity issue of the image encryption algorithms. The most common approach is to create a dependence between at least one step of the image encryption algorithm and pixel intensities of used plain image. Since various plain images affect the chosen step of image encryption algorithm in a different way, the obtained encrypted images could not be utilised for a successful differential attack. Therefore, these algorithms could be called plaintext-related because of their behaviour.

Majority of the already presented plaintext-related solutions for image encryption could be classified into one of these categories:
• the plain image affects parameters of chaotic map(s),
• the plain image affects element values of generated PR sequences,
• the plain image affects pixel shuffling during confusion stage,
• the plain image affects computation of pixel intensities during diffusion stage.

Several examples of image encryption algorithms are briefly described in Section 2. While some approaches could be easily classified into one of the mentioned categories, other algorithms are more refined. This is also a case for the scheme proposed in this paper which quantises elements of PR sequences according to plain image pixel intensities. Main novelties and advantages of this solution include:

• a custom plaintext-related quantisation technique that should have lower computational complexity than hash functions,
• the quantisation technique able to obtain its input values also from encrypted image without the need of any supplementary information,
• an additional non-linear substitution block that enhances non-linearity even in images with monotonous regions (with small intensity changes).

Since the proposed solution was designed with respect to mentioned applications of dedicated image encryption algorithms, it does not have any major limitations. Plain images that are going to be encrypted could have arbitrary resolution. The colour depths are limited to 8 bits per pixel (greyscale images) or 24 bits per pixel (true-colour images with three colour planes). The key length is fixed at 128 bits and keys need to be entered in a hexadecimal notation. The proposed image encryption algorithm could be used for all mentioned applications—encryption of images containing secret data in steganographic systems, securing sensitive medical images and improving the performance of simple biometric systems by fast encryption of fingerprints or eye irises.

The rest of this paper is organised as follows: Section 2 provides a survey of solutions dealing with the plain image sensitivity issue. Section 3 briefly describes main ideas of the proposed solution and illustrates steps of the designed algorithm. Section 4 discusses obtained experimental results and compares them to numerical values reported in related works. The final, fifth section concludes the paper with an overview of the proposed image encryption algorithm.

2 | RELATED WORKS

One of the first algorithms that addressed issues reported by Solak et al. was a proposal by Kanso and Ghebleh from 2012 [9]. In this case, the number of iterations of used chaotic map is related to the intensity of currently processed image pixel. However, as the amount of iterations increases with higher pixel intensities, this solution is prone to side-channel attacks.

An image encryption scheme that circularly shifts bits of currently processed pixel intensity was presented by Fu et al. in 2013 [10]. Each circular shift depends on an intensity of previous image pixel which raises a concern as there are only 8 possible shifts (including no shift) controlled by 256 intensities of the previous image pixel. Therefore, multiple intensities lead to same shift.

In 2014, Zhang proposed a solution that establishes dependencies between plain images and algorithm during confusion stage (it rearranges pixel intensities) [11]. In order to secure this solution against chosen plaintext attacks, the confusion stage was surrounded by two diffusion stages (usually there is only one) which significantly raised the computational complexity of whole algorithm.

A scheme presented by Norouzi et al. from 2014 [12] utilised the sum of all pixel intensities during the encryption of the first scanned image pixel. However, in images with two or more pixels, there are multiple ways for computing the same sum of pixel intensities. This flaw and other drawbacks were reported in the same year by Zhang et al. [13].

An approach designed by Murillo-Escobar et al. in 2015 [14] modified an element of generated PR sequence by a value obtained from used plain image. However, in this case the decryption algorithm is unable to compute this value. Therefore, Murillo-Escobar et al. decided to store this value in the encrypted image by the usage of steganography. This solution is not safe and also causes distribution of errors via ciphertext chaining. These drawbacks were analysed and presented by Fan et al. in 2018 [15] who also provided a way to break the mentioned image encryption algorithm.

Some of the most recent plaintext-related image encryption algorithms use plain image hashes as parameters of the chaotic maps. While the obtained numerical results for these solutions seem quite good, the usage of hash functions greatly increases computational complexity of the algorithms. The second issue is a belief that more complex chaotic systems provide better behaviour of whole image encryption algorithm. These issues are evident in proposals by Chai et al. from 2017 [16], Wang et al. [17] and Li et al. [18] from 2018 or Sun et al. from 2019 [19].

3 | PROPOSED SOLUTION

The design proposed in this paper takes a different approach for establishing dependencies between pixel intensities of plain images and the image encryption algorithm. Instead of affecting initial value of generated PR sequence(s) or direct usage in confusion or diffusion stage, the plain image intensities are used as parameters for a custom quantisation scheme. This could be perceived as a main novelty of the proposed solution.

Proposed plaintext-related quantisation scheme uses one row of pixel intensities for controlling quantisation of one row of elements from generated PR sequence. The quantised elements are then combined with a row of pixel intensities in the processed image. Since the pixel intensities that control the quantisation are processed prior to pixel intensities that are
going to be combined, the decryption algorithm is able to obtain all necessary values. The used row-wise solution that does not use hash functions should also positively affect the speed of the performed operations.

Since speed is considered as a quite important property of the image encryption algorithms [14, 16, 17, 20], the proposed solution uses a trade-off where the more complicated custom quantisation scheme is used only in one block of the proposed encryption algorithm. This block is then followed by other operations (non-linear substitution and a combination with a PR sequence) that should rule out any possibility of a successful attack.

A simplified block scheme of operations done for encryption and decryption by the proposed solution is shown in Figure 1. The operations such as confusion or diffusion done during the decryption are inverse to operations done during the encryption. Block denoted as “image rearrangement and key processing” rearranges images with multiple colour planes to a two-dimensional matrix, divides entered key into portions used by other blocks and finally generates and quantises the PR sequences.

### 3.1 Preliminaries

This subsection describes some techniques that may be unfamiliar to some readers but they are important for understanding how the proposed solution works.

#### 3.1.1 Logistic map

Logistic map (LM) is a one-dimensional chaotic map that uses one parameter \( r \in (0; 4) \) for mapping of a current iterate \( x_n \in (0; 1) \) into a successive iterate \( x_{n+1} \in (0; 1) \). LM was popularised mainly by a work of May [21]. An equation of the LM can be expressed as (1):

\[
x_{n+1} = r \cdot x_n (1 - x_n),
\]

where \( n = 0, 1, 2, \ldots, N \) is a sequential number of the iterate and \( N \) is a total number of iterates. Computation of the first iterate \( x_1 \) requires an initial value \( x_0 \in (0; 1) \).

The sequences generated by the LM use a concept of a transient period, when certain amount of iterates is discarded prior to the generation of iterates used as PR sequence elements. This procedure helps to suppress possibly apparent relations between the first few generated iterates. Usual size of the transient period is 1000 iterates.

Chaotic behaviour of the LM [21, 22] can be illustrated by a bifurcation diagram. An example of this diagram for \( r \in (0; 4) \) together with a detail for \( r \in (3; 4) \) is shown in Figure 2. The first bifurcation occurs at \( r \approx 3 \) where successive iterates obtain values from one of two trajectories. As the value of \( r \) increases, more bifurcations and more possible trajectories are present. The point of \( r \approx 3.56995 \) is known as “an onset of chaos” [22] even though there is a region around \( r \approx 3.85 \) where the amount of trajectories decreases. Therefore, many image encryption algorithms which use LM utilise values of \( r \) close to 4.

A quantification of the chaotic behaviour could be done by estimating values of Lyapunov exponents (LEs) \( \lambda \) [22]. Estimation of LEs assumes a little difference between two initial values \( x_0 \) lying on two respective trajectories. In the case that the
difference increases with the computation of successive iterates, the LEs are positive ($\lambda > 0$) and whole system is considered chaotic [22]. The LEs estimated for the LM with $x_0$ of 0.5 and $r \in (3; 4)$ are displayed in Figure 3.

Some of the other drawbacks of the LM such as non-uniform distribution of iterate values, possible relation between pair of successive iterates or occurrence of the fixed points and periodicity were reported in several papers [23, 24]. However, all known flaws were suppressed or bypassed by now [20, 25].

### 3.1.2 Quantisation approaches

Because the iterate values of the LM are real numbers ranging from 0 to 1 and pixel intensities are integers from set \( \{0, 1, 2, \ldots, 255\} \), the data types of these two sets need to be unified. Since operations with integers are considered to be faster, the iterate values need to be quantised, preferably to a set \( \{0, 1, 2, \ldots, 255\} \). The simplest and most straightforward way can be expressed by (2):

\[
x'_n = \lfloor (M + 1) \cdot x_n \rfloor,
\]

where brackets $\lfloor a \rfloor$ denote rounding to the closest integer number equal to or less than $a$, $M$ stands for a maximal possible quantised value, $x'_n \in \{0, 1, 2, \ldots, M\}$ is a sequence of quantised iterates and $x_n \in (0; 1)$ is a sequence of iterates prior to quantisation.

However, as quantisation by using (2) does not solve the issue of non-uniform iterate distribution [20, 25], a more complex quantisation equation (3) was proposed in one of our prior works [25]:

\[
x'_n = \lfloor (M + 1) \cdot (10^S \cdot x_n \text{ (mod 1)}) \rfloor,
\]

where $S$ is a number of the most significant iterate decimal places that are going to be discarded during the quantisation and $a \text{ (mod } b)$ returns remainder after dividing $a$ by $b$. Usual values of $S$ are 4 or 5 decimal places which ensure almost uniform distribution of iterate values [20, 25]. Higher values of $S$ may result in unnecessary increase of computational complexity. In most implementations, the iterate values are double precision data type which enables exact representation of real numbers up to 15 decimal places [20].

The effects of decimal place removal during quantisation are illustrated in Figure 4. The combined histogram contains bins of two colours—dark blue (without any removed decimal places, $S = 0$) and dark red (with 4 removed decimal places, $S = 4$). Please take note that the visible spikes present for $S = 0$ at histogram boundaries are suppressed for $S = 4$. Therefore, the distribution for $S = 4$ could be considered as more uniform.

The removal of the most significant decimal places in iterate values helps not only to balance the distribution of iterate values but also to suppress any possibility of observing relations between the values of successive iterates [25].

### 3.2 Encryption algorithm

The encryption algorithm uses a plain image $P$ and a key $K$ as inputs. Plain images can have arbitrary resolution and colour depths of 8 or 24 bits per pixel. The proposed algorithm uses 128 bit long key, entered as a 32 symbol (0 – F) long hexadecimal string. Resulting encrypted image is stored in a matrix denoted as $E$.

**Step 1:** Plain image processing. At first, the height $h$, width $w$ and number of colour planes $num_{pc}$ of plain image $P$ are obtained. Number of all image pixels $num_{px}$ is acquired simply as $num_{px} = h \cdot w \cdot num_{pc}$. The plain image $P$ is then rearranged to a two-dimensional matrix $P'$ by (4). Please take note, that if the plain image $P$ is greyscale, it is only copied to the matrix $P'$. Rearrangement of true-colour images is illustrated in Figure 5.

\[
P'(l, num_{px} \cdot (k - 1) + cp) = P(l, k, cp),
\]
where \( i = 1, 2, 3, \ldots, b, \ k = 1, 2, 3, \ldots, w \) and \( cp = 1, \ldots, \text{num}_{cp} \) are height, width and colour plane indices. Resulting matrix \( P' \) has \( I \) rows and \( w_{\text{ext}} = w \cdot \text{num}_{cp} \) columns.

**Step 2**: Key processing. Used key \( K \) is divided into eight parts denoted as \( K_1 \) to \( K_8 \) by using (5). Each of these key parts consists of four hexadecimal symbols.

\[
K_i(j) = K(4 \cdot (i - 1) + j),
\]

where \( i = 1, 2, 3, \ldots, 8 \) is a sequential number of key part and \( j = 1, 2, 3, 4 \) is an index of a hexadecimal symbol.

The values of \( K_1 \) to \( K_8 \) are then converted from hexadecimal to decimal notation with usage of a big-endian ordering scheme (the first symbol is the most significant). Obtained values are used for computing eight parameter values \( r_1 \) to \( r_8 \) by (6):

\[
r_j = \begin{cases} 
4 - 10^{-15} \cdot (4^{16} - K_j) & \text{if } i \text{ is odd} \\
4 - 10^{-15} \cdot (2 \cdot 4^{16} - K_j) & \text{if } i \text{ is even}
\end{cases}
\]

where \( i = 1, 2, 3, \ldots, 8 \) is a sequential number of parameter and usage of constants \( 4 \cdot 10^{-15} \) and \( 2 \cdot 4^{16} \) or \( 2 \cdot 4^{16} \) ensures that values of odd and even \( r_j \) are in non-overlapping intervals. This situation enables alternation between odd and even values of \( r_j \) during generation of PR sequences by the LM (1). The variation of parameter values suppresses possible issues with fixed points or periodicity as both these problems are caused by constant parameter values [24].

**Step 3**: Generation of PR sequences. The sequences are generated by iterating the LM (1) with an initial value \( x_0 \) of 0.5 and a transient period of 1000 iterates for all sequences. During the transient period, all four pairs of parameter values \( (r_{2i-1}, r_{2i}); i = 1, 2, 3, 4 \) are used to create an effect known as key diffusion [26]. Key diffusion ensures that even small differences in one key character would affect all generated sequences and therefore also all key dependent blocks of the image encryption algorithm.

The shifting pattern of parameter values and other details of the generated PR sequences are explained in Table 1. The first parameter value of each pair is used for the computation of iterates with odd sequential numbers \( \{n = 1, 3, 5, \ldots\} \) and the second parameter value of each pair is utilised for calculating iterates with even sequential numbers \( \{n = 2, 4, 6, \ldots\} \).

**Step 4**: Quantisation of PR sequences. The quantisation of generated sequence elements belonging to an interval \((0; 1)\) into a set \{0, 1, 2, 3, \ldots, \( M \}\}, where \( M \) is the maximal possible quantised value is done by applying (3). Used values of parameters \( M \) and \( \text{seq}_i \) are included in Table 2. The quantised sequences that are computed from sequences \( \text{seq}_1 \) to \( \text{seq}_6 \) are distinguished by the usage of an apostrophe—they are denoted as \( \text{seq}_1' \) to \( \text{seq}_6' \).

**Step 5**: Processing of the first PR sequence. Only elements of \( \text{seq}_1' \) are quantised according to used plain image due to higher computational complexity of this step. The 65,536 possible element values (the first one is 0) after Step 4 can be represented by 16 bits.

Firstly, two sequences are created from element values of \( \text{seq}_1' \). The first sequence \( \text{seq}_M' \) is obtained from the first eight most significant bits of elements while the second sequence \( \text{seq}_L' \) takes values of eight remaining (less significant) bits of elements. This step could be mathematically expressed as (7):

\[
\text{seq}_M'(n) = \left\lfloor \frac{\text{seq}_1'(n)}{256} \right\rfloor, \\
\text{seq}_L'(n) = \text{seq}_1'(n) \mod 256
\]

where \( n = 1, 2, 3, \ldots, \text{num}_{\text{ext}} \) is an element sequential number.

**Step 6**: Combination with a quantised plaintext-related matrix. Since sequences \( \text{seq}_M' \) and \( \text{seq}_L' \) represented by row vectors are going to be combined with rearranged plain image \( P' \) that is a matrix, it is useful to convert the two vectors into two matrices. The resulting matrices \( \text{seq}_{Mn}' \) and \( \text{seq}_{Ln}' \) have \( h \) rows and \( w_{\text{ext}} \) columns and use row-first scanning pattern demonstrated in Figure 6 for storing values obtained from the vectors.

---

### Table 1 Parameter switching pattern

| Sequence | 1 to 250 | 251 to 500 | 501 to 750 | 751 to 1000 | after 1000 | Number of iterates |
|----------|----------|------------|------------|-------------|------------|-------------------|
| seq₁     | \( (r_1, r_2) \) | \( (r_1, r_2) \) | \( (r_1, r_2) \) | \( (r_1, r_2) \) | \( (r_1, r_2) \) | num_{px}        |
| seq₂     | \( (r_1, r_2) \) | \( (r_1, r_2) \) | \( (r_1, r_2) \) | \( (r_1, r_2) \) | \( (r_1, r_2) \) | num_{px}        |
| seq₃     | \( (r_2, r_1) \) | \( (r_2, r_1) \) | \( (r_2, r_1) \) | \( (r_2, r_1) \) | \( (r_2, r_1) \) | num_{px}        |
| seq₄     | \( (r_2, r_1) \) | \( (r_2, r_1) \) | \( (r_2, r_1) \) | \( (r_2, r_1) \) | \( (r_2, r_1) \) | num_{px}        |
| seq₅     | \( (r_2, r_1) \) | \( (r_2, r_1) \) | \( (r_2, r_1) \) | \( (r_2, r_1) \) | \( (r_2, r_1) \) | num_{px}        |

### Table 2 Quantisation parameters for generated PR sequences

| Quantised sequence | \( \text{seq}_1' \) | \( \text{seq}_2' \) | \( \text{seq}_3' \) | \( \text{seq}_4' \) | \( \text{seq}_5' \) | \( \text{seq}_6' \) |
|--------------------|------------------|------------------|------------------|------------------|------------------|------------------|
| Maximal quantised value \( M \) | 65535 | 15 | 15 | \( h - 1 \) | \( w_{\text{ext}} - 1 \) | 255 |
| Number of discarded decimal places \( \delta \) | 4 | 4 | 4 | 4 | 4 | 4 |
The combination of plain image and acquired matrices depends on plain image pixel intensities. It is done as a row-wise operation that takes whole rows of rearranged plain image \( P' \) and whole rows of matrices \( seq^{_{\text{Min}}} \) and \( seq^{_{\text{Max}}} \) as inputs. The outputs are rows of modified pixel intensities and they are inserted to matrix \( P' \). The whole operation could be expressed by a set of equations (8):

\[
\begin{align*}
A_1(l, :) &= P'(l - 1, :) \land seq^{_{\text{Min}}}(l, :), \\
A_2(l, :) &= (255 - P'(l - 1, :)) \land seq^{_{\text{Max}}}(l, :), \\
A_3(l, :) &= A_1(l, :) + A_2(l, :) \pmod{256}, \\
P'(l, :) &= P'(l, :) \oplus A_3(l, :)
\end{align*}
\]

where auxiliary matrices \( A_i, i = 1, 2, 3 \) are used only for better clarity; the algorithm computes all four equations at once. Index \( l - 1 \) denotes the previously scanned row, the first row \( (l = 1) \) uses the bottom row \( (l - 1 = b) \) as previously scanned. The symbol \( \land \) is a substitution for all image pixels or sequence elements in a row and operators \( \land \) and \( \oplus \) denote operations of binary AND and binary XOR (eXclusive OR) [27]. The constant of 255 in second equation is used for computing negation of \( P' (l - 1, :) \).

The whole point of operations done by (8) is that the combination of elements of \( seq^{_{\text{Min}}} \) and \( seq^{_{\text{Max}}} \) depends on previously scanned row of rearranged plain image pixel intensities. If quantisation approach (3) with \( S = 4 \) would be used, the quantised PR sequence would look like \( seq^{_{PQ}} \). However, as some of the less significant bits from \( seq^{_{PQ}} \) might be used, the whole quantisation step could be viewed as non-linear.

An example of computation of six intensities in one row of \( P' \) after the plaintext-related quantisation step is illustrated in Table 3.

**Step 7:** Preparation of a substitution table. In order to suppress any indications of linearity in particular types of images (e.g. monotonous images with small intensity changes) after Step 6, the pixel intensities of the matrix with processed image \( P' \) are going to be substituted by values from a substitution table. This table is represented by a matrix \( ST \) with 16 rows and 16 columns. At first, the matrix \( ST \) is filled by a sequence \( \{0, 1, 2, ..., 255\} \) using the row-first scanning pattern demonstrated in Figure 6.

Then the matrix elements are shuffled using two sets of circular shifts. The first set of shifts is done in each of the 16 columns of matrix \( ST \). Shift sizes are determined by elements of sequence \( seq^{_{k}} \). The other set of shifts rearranges each of the 16 rows of the matrix and the size of shifts is controlled by sequence \( seq^{_{b}} \). Both sets of shifts move rearranged elements to the side with increasing coordinate—the first set moves elements to the bottom while the second set moves elements to the right side of the matrix. The whole process of substitution table rearrangement is shown in Figure 7.

The substitution table \( ST \) is then rearranged to a row vector \( ST' \) by the usage of row-first scanning pattern (see Figure 6).

**Step 8:** Non-linear substitution. Pixel intensities of matrix \( P' \) are substituted by (9):

\[
P'(l, k) = ST'(P'(l, k)),
\]

where \( l = 1, 2, 3, ..., b \) and \( k = 1, 2, 3, ..., \ w_{ext} \) are height and width indices and \( b \) and \( w_{ext} \) are height and width of matrix \( P' \).

**Step 9:** Two directional confusion. Image encryption algorithms use a confusion stage for suppressing correlation between adjacent image pixel intensities [2]. The proposed solution utilises two circular shifts for rearranging pixel intensities in a similar fashion as it was done in Step 7 (see example in Figure 7). The first set of shifts is done in individual columns of matrix \( P'(:, k), k = 1, 2, 3, ..., w_{ext} \). The size of shifts is set by elements of sequence \( seq^{_{k}} \).

The second set of shifts is carried out in rows of matrix \( P'(l, :) \), \( l = 1, 2, 3, ..., b \) with shift sizes given by elements of sequence \( seq^{_{b}} \). Circular shifts done in the confusion stage could be expressed mathematically by a set of equations (10):

\[
\begin{align*}
A_4(l, :) &= P'(l + (l - 1 + seq^{_{b}}(l) \pmod{b}) , : ) \\
A_5(:, k) &= A_4(:, 1 + (k - 1 + seq^{_{b}}(k) \pmod{w_{ext}})) \pmod{w_{ext}} \\
P'(l, k) &= A_5(l, k)
\end{align*}
\]

where \( A_4 \) and \( A_5 \) are auxiliary matrices used only for better clarity of the notation, \( l = 1, 2, 3, ..., b \) and \( k = 1, 2, 3, ..., \ w_{ext} \) denote row and column indices, \( b \) and \( w_{ext} \) stand for height and width of matrix \( P' \) and symbol \( : \) denotes all image pixels either

---

**FIGURE 6** Row-first scanning pattern

**TABLE 3** An example of results after plaintext-related quantisation

| Variable | Values before combination with the quantised sequence |
|----------|------------------------------------------------------|
| \( P'(l - 1) \) | 95, 102, 92, 93, 5, 0 |
| \( P'(l) \) | 135, 136, 133, 134, 139, 133 |
| Auxiliary values | |
| \( seq^{_{l}}(l) \) | 11531, 19586, 16721, 11034, 52938, 39615 |
| \( seq^{_{l}}(b) \) | 45, 76, 65, 43, 206, 154 |
| \( seq^{_{l}}(w) \) | 11, 130, 81, 26, 202, 191 |
| \( A_{1}(l) \) | 13, 68, 64, 9, 4, 0 |
| \( A_{2}(l) \) | 0, 128, 1, 2, 202, 191 |
| \( A_{3}(l) \) | 13, 196, 65, 11, 206, 191 |

| Values after the combination | |
| \( P'(l) \) | 138, 76, 196, 141, 69, 58 |
in a row with an index \( l \) or in a column \( k \).

\[
P'(l, :) = P'(l + 1, :) \oplus (P'(l, :) + P'(l - 1, :) \mod 256)
\]

\[
P'( :, k) = P'( :, k + 1) \oplus (P'( :, k) + P'( :, k - 1) \mod 256)
\]

\[
P'(b + 1 - l, :) = P'(b - l, :) \oplus (P'(b + 1 - l, :) + P'(b + 2 - l, :) \mod 256)
\]

\[
P'(w_{\text{ext}} - k, :) = P'(w_{\text{ext}} - k) \oplus (P'(w_{\text{ext}} + 1 - k) + P'(w_{\text{ext}} + 2 - k) \mod 256)
\]

(11)

where \( l = 1, 2, 3, ..., b \) and \( k = 1, 2, 3, ..., w_{\text{ext}} \) are row and column indices, \( b \) and \( w_{\text{ext}} \) denote height and width of matrix \( P' \). If the computed row index is 0 or \( b + 1 \), values of \( b \) and 1 are used. When the computed column index is 0 or \( w_{\text{ext}} + 1 \), values of \( w_{\text{ext}} \) or 1 are utilised.

**Step 10:** Four directional diffusion. The diffusion stage introduces dependencies between intensities of the rearranged pixel intensities [2]. In the case of the proposed solution, the pixel intensities in the matrix \( P' \) are processed in four steps. Each step scans matrix elements in a different direction, beginning with top to bottom, continuing with left to right and bottom to top and finishing with right to left. This procedure is important for the propagation of the difference between similar plain images. The worst scenario for the proposed solution (the difference between two plain images marked by orange spot is located at bottom right corner) is shown in Figure 8. However, also in this case the different matrix elements affect all matrix elements after the final step of diffusion stage.

The four steps of diffusion procedure could be given as a set of equations (11). Please bear in mind that each equation uses both rows or columns that are going to be scanned (with incremented index) and also rows or columns that were already scanned (with decremented index). This bidirectional approach spreads the differences during both encryption and decryption.

The equations from set (11) use two mathematical operations in order to distinguish two vectors—one operation (modulo 256 addition) for a vector that was already scanned and the second one (combination by means of XOR) for vector that is going to be scanned.

**Step 11:** Combination with a PR matrix. This step increases the difficulty of all potential attacks as attackers need to crack this step before they could analyse any other steps. The combination uses sequence \( \text{seq}_{16} \) which is rearranged to a matrix \( \text{seq}_{16} \) using the row-first scanning pattern (see Figure 6).

Then the combination is carried out by (12):

\[
P'(l, k) = P'(l, k) \oplus \text{seq}_{16}(l, k),
\]

(12)

where \( l = 1, 2, 3, ..., b \) and \( k = 1, 2, 3, ..., w_{\text{ext}} \) are height and width indices.

**Step 12:** Encrypted image processing. At this point, the matrix \( P' \) represents encrypted image. However, it needs to be rearranged from matrix representation to an image with one or three colour planes. A procedure that is inverse to (4) can be expressed as (13):

\[
E\left( l, 1 + \left\lfloor \frac{k-1}{\text{num}_{\text{cp}}} \right\rfloor, 1 + (k-1 \mod \text{num}_{\text{cp}}) \right) = P'(l, k),
\]

(13)

where \( \text{num}_{\text{cp}} \) stands for number of colour planes.


### TABLE 4  A set of used keys

| Key | Value |
|-----|-------|
| $K_1$ | 0x C9 0F DA A2 21 68 C2 34 C6 62 88 80 DC 1C D1 |
| $K_2$ | 0x C9 0F DA A2 21 68 C2 34 C5 C6 62 88 80 DC 1C D1 |
| $K_3$ | 0x 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |

#### 3.3 Decryption algorithm

The steps of decryption algorithm are analogous to the steps done during encryption. At first, Steps 1–4 are carried out. Then Step 11 is done, followed by an inverse of Step 10. The inverse version of this step means that the equations are computed in reversed order, starting with the last one from (11) and continuing to the first one. Also, the sign between two expressions used for the modulo operation changes from “+” to “−”. The last difference for Step 10 is the reversal of the indices as $l$ begins with $h, h−1, h−2$ and ends with $1$. Similarly, values of $k$ go from $w_{ext}, w_{ext}−1, w_{ext}−2$ to $1$.

After inverse diffusion stage, the decryption algorithm continues with inverse confusion stage. In this case the order of equations in the set (10) is reversed and elements of sequences $seq'_4$ and $seq'_5$ are multiplied by a factor of $−1$.

Then, the substitution table is prepared by Step 7 and inverse substitution maps coordinates of table elements to pixel intensities in Step 8. After that, Step 5 takes place. Since the row of pixel intensities with coordinate $l = h−1$ contains the same values as those used by encryption algorithm for operations with the last row of intensities ($l = h$), the decryption algorithm is able to utilise the second to last row of pixel intensities for operations done with the last row of pixel intensities. The computations by (8) are carried out by similar fashion, only coordinate $l$ has reversed elements, ranging from $h, h−1, h−2$ to $1$.

The result after combination with the sequence quantised according to a plain image is the decrypted image stored in a matrix $D$. With usage of the same key for both encryption and decryption, the decrypted image is identical to plain image ($D = P$).

### 4 EXPERIMENTAL RESULTS

All experiments done with the proposed image encryption algorithm were performed on a PC with 2.5 GHz CPU, 12 gigabytes of RAM running MATLAB R2015a on Windows 10 OS. Used set of experimental keys is presented in Table 4. Key $K_1$ contains the first 128 bits of binary expansion of number $\pi$. Please take note that the highlighted difference between keys $K_1$ and $K_2$ is present only in one hexadecimal symbol with the smallest possible amplitude.

A set of experimental images chosen from USC-SIPI database [28] is shown in Figure 9. The first three rows of images represent plain images. Results of encryption with various keys are illustrated in the last row. Parameters of the plain images are described in Table 5.

![lena](lena.png)
![lenaG](lenaG.png)

![peppers](peppers.png)
![peppersG](peppersG.png)

![baboong](baboong.png)
![baboongG](baboongG.png)

![lena encrypted with $K_1$](lena_encrypted_K1.png)
![lenaG encrypted with $K_1$](lenaG_encrypted_K1.png)

A magnified detail of 16×8 image pixels from top left corner of plain image lena and its version encrypted with key $K_3$ is demonstrated in Figure 10. This example points out that even monotonous regions of plain images are turned in more detailed areas by the encryption, even with not complex keys.
TABLE 5 Parameters of used plain images

| Image  | Height \( h \) [px] | Width \( w \) [px] | Colour depth [bits/px] |
|--------|---------------------|-------------------|------------------------|
| lena   | 512                 | 512               | 24                     |
| lenaG  | 512                 | 512               | 8                      |
| peppers| 512                 | 512               | 24                     |
| peppersG| 512                | 512               | 8                      |
| baboonG| 512                 | 512               | 8                      |
| boatG  | 512                 | 512               | 8                      |

4.1 | Key space size

As the proposed image encryption algorithm uses 128 bit long keys, the size of whole key space could be expressed as \(2^{128}\). The brute-force attack on a greyscale image with a resolution of 512×512 pixels would take approximately \(1.3488 \times 10^{30}\) years as decryption with one key requires approximately 125 ms (see Section 4.6 for details).

4.2 | Sensitivity to used key

An example of key sensitivity of the proposed image encryption algorithm is shown in Figure 11. At first, plain image *lena* was encrypted with key \(K_1\). Then it was decrypted twice, one time by correct key \(K_1\), then by a slightly different key \(K_2\).

While the image decrypted with correct key is identical to plain image *lena*, the image decrypted with incorrect key does not provide any information about content or scene of the plain image.

4.3 | Statistical properties of the sequence quantised according to a plain image

Since one of the generated PR sequences is quantised with a respect to pixel intensities of used plain image, it may be interesting to investigate the statistical properties of this PR sequence. Therefore, an NIST 800-22 test suite [29] used for choosing AES candidates was selected to evaluate properties of matrix \(A_3\) from (8). Mentioned matrix is later combined with processed image \(P'\) in Step 6 of the encryption algorithm (see Section 3.2 for details).

NIST recommends at least \(10^8\) bits of input data for obtaining plausible test results [29]. The input data is represented by 100 binary sequences, each of them is \(10^6\) bits long. These sequences are filled by elements of matrix \(A_3\) from (8) that was computed during encryption of a zero intensity image with key \(K_3\). The intensities of all image pixels in zero intensity images are equal to 0.

Elements of matrix \(A_3\) are scanned by the row-first pattern (Figure 6) and then they are decomposed to eight bits using the big-endian ordering scheme. In order to provide big enough data sample, the zero intensity image has a resolution of 3840×2160 pixels and colour depth of 24 bits per pixel. Testing sequences use the first \(10^8\) scanned bits from computed matrix \(A_3\).

The results of performed statistical tests are included in Table 6. Required pass rates and default values of additional parameters such as block sizes are discussed in [29].

The obtained results presented in Table 6 show that even in the case of the zero intensity plain image and zero key \((K_3)\), the resulting PR sequence quantised according to the plain image has appropriate statistical parameters by means of the NIST 800-22 test suite. Therefore, any concerns about suitability of the proposed plaintext-related quantisation scheme are not confirmed.

4.4 | Robustness against statistical attacks

Robustness of the proposed image encryption algorithm against statistical attacks could be evaluated by several measures. First of all, encryption should suppress significant peaks present in a histogram of a plain image. This test was performed with a plain image *lenaG* and encryption was done with key \(K_1\). Resulting combined histogram of images before and after encryption is shown in Figure 12.

Statistical data presented in histograms could be expressed by two measures. The first one, histogram variance \(\text{var} \_i\) expresses differences between individual bin values for each colour plane (14):

\[
\text{var} \_i = \frac{1}{2^L \cdot 2^L} \sum_{i=1}^{2^L} \sum_{j=1}^{2^L} \left( \frac{\text{hg}(i) - \text{hg}(j)}{2} \right)^2,
\]

where \(L\) is a colour depth of analysed colour plane and \(i\) and \(j\) denote bin indices of histogram \(hg\). Since the histograms of
TABLE 6 Obtained results for statistical tests from NIST 800-22 suite

| Test                                      | Required pass rate | Proportion of successful results | Final test result |
|-------------------------------------------|--------------------|----------------------------------|-------------------|
| Frequency (monobit)                       | 96/100             | 99/100                           | Passed            |
| Frequency within a block (M = 128 bits)   | 96/100             | 100/100                          | Passed            |
| Runs                                      | 96/100             | 99/100                           | Passed            |
| Longest run of ones in a block            | 96/100             | 97/100                           | Passed            |
| Binary matrix rank                        | 96/100             | 99/100                           | Passed            |
| Discrete Fourier transform (spectral)     | 96/100             | 98/100                           | Passed            |
| Non-overlapping template matching (first P-value; m = 9 bits) | 96/100             | 100/100                          | Passed            |
| Overlapping template matching (m = 9 bits) | 96/100             | 99/100                           | Passed            |
| Maurer’s “universal statistic”            | 96/100             | 97/100                           | Passed            |
| Linear complexity (M = 500 bits)          | 96/100             | 98/100                           | Passed            |
| Serial (first P-value; m = 16 bits)       | 96/100             | 98/100                           | Passed            |
| Approximate entropy (m = 10 bits)         | 96/100             | 98/100                           | Passed            |
| Cumulative sums (first P-value)           | 96/100             | 98/100                           | Passed            |
| Random excursions (first P-value)         | 55/58              | 58/58                            | Passed            |
| Random excursions variant (first P-value) | 55/58              | 56/58                            | Passed            |

FIGURE 12 A comparison of histograms for plain and encrypted image

encrypted images should be balanced, the encryption should decrease value of histogram variance [30].

The second measure used for the interpretation of histogram data is the value \( \chi^2 \) obtained after chi-squared test. This test computes difference between the actual distribution of pixel intensities and the expected one [31, 32]. The value of \( \chi^2 \) could be calculated by (15):

\[
\chi^2 = \sum_{i=1}^{M} \frac{(h_g(i) - h_e)^2}{h_e},
\]

(15)

where \( h_g \) is the expected value of histogram bins and \( b \) and \( w \) stand for height and width of analysed colour plane.

Lower values of \( \chi^2 \) are favourable as they point to more uniform distribution of pixel intensities in histograms. Also, there are some critical values of \( \chi^2 \) that confirm a hypothesis that histogram distribution is uniform. These depend on used significance level \( \alpha \) and number of degrees of freedom \( n_{df} \) (255 for colour planes with 8 bit colour depth). The critical value of \( \chi^2 \) for commonly used values of \( \alpha = 0.05 \) and \( n_{df} = 255 \) is equal to 293.25 [32].

The resulting values of histogram variance \( \text{var} \) and \( \chi^2 \) obtained for the set of experimental images are included in columns 4 and 5 of Table 7. Letters R, G and B indicate red, green or blue colour plane of used image. Symbol “—” denotes greyscale images or plain images that were not encrypted.

\[
\rho \left( \frac{cov(in_1, in_2)}{\sigma(in_1) \cdot \sigma(in_2)} \right) = \frac{\sum_{p=1}^{num_{pp}} (in_1(pp) - \overline{in_1}) \cdot (in_2(pp) - \overline{in_2})}{\sqrt{\sum_{p=1}^{num_{pp}} (in_1(pp) - \overline{in_1})^2 \cdot \sum_{p=1}^{num_{pp}} (in_2(pp) - \overline{in_2})^2}},
\]

(16)

where \( cov(a, b) \) represents covariance of sequences \( a \) and \( b \), \( \sigma(a) \) is a standard deviation of sequence \( a \), \( pp = 1, 2, 3, ..., num_{pp} \) is an index of pixel pair, \( num_{pp} \) stands for amount of pixel pairs, sequences \( \overline{in_1} \) and \( \overline{in_2} \) contain pixel intensities from pixel pairs and \( \overline{a} \) denotes arithmetic mean of sequence \( a \).

Please take note that the usage of proposed image encryption algorithm resulted in nearly uniform distribution of values among histogram bins. Also, the encryption significantly decreases the values of histogram variance \( \text{var} \) and \( \chi^2 \). All values of \( \chi^2 \) for encrypted images are lower than the critical value of 293.25 for \( \alpha = 0.05 \) and \( n_{df} = 255 \). Therefore, the possibility of a successful statistical attack could be considered as significantly suppressed.
| Image and colour plane | Key  | \(\text{var} \ [-]\) | \(\chi^2 \ [-]\) |
|------------------------|------|----------------|-----------------|
| lena                   |      |                |                 |
| R                      | -    | 510,371        | 127592.6758     |
| G                      |      | 1,290,286      | 322571.5586     |
| B                      |      | 1,908,534      | 477133.4238     |
| R                      | \(K_1\) | 1099           | 274.7324        |
| G                      |      | 990            | 247.4785        |
| B                      |      | 1014           | 253.584         |
| R                      | \(K_2\) | 1059           | 264.8379        |
| G                      |      | 1107           | 276.7461        |
| B                      |      | 1048           | 262.1016        |
| R                      | \(K_3\) | 1053           | 263.1758        |
| G                      |      | 977            | 244.1387        |
| B                      |      | 1027           | 256.7188        |
| lenaG                  |      | 1,039,126      | 259781.3926     |
| R                      | \(K_1\) | 857            | 214.1992        |
| G                      | \(K_2\) | 1102           | 275.3945        |
| B                      | \(K_3\) | 938            | 234.5391        |
| peppers                |      |                |                 |
| R                      | -    | 852,749        | 213187.2168     |
| G                      |      | 1,273,532      | 318382.9297     |
| B                      |      | 1,965,713      | 491428.1777     |
| R                      | \(K_1\) | 1166           | 191.4492        |
| G                      | \(K_2\) | 1153           | 288.207         |
| B                      | \(K_3\) | 998            | 249.3906        |
| R                      | \(K_1\) | 1058           | 264.4453        |
| G                      | \(K_2\) | 1039           | 259.8203        |
| B                      | \(K_3\) | 1035           | 258.8496        |
| R                      | \(K_3\) | 1017           | 254.2871        |
| G                      |      | 1049           | 262.3691        |
| B                      |      | 1018           | 254.5723        |
| peppersG               |      | 478,900        | 119725.1152     |
| R                      | \(K_1\) | 1063           | 263.7051        |
| G                      | \(K_2\) | 890            | 222.4434        |
| B                      | \(K_3\) | 972            | 243.0508        |
| baboonG                |      | 750,764        | 187690.9043     |
| R                      | \(K_1\) | 998            | 249.5273        |
| G                      | \(K_2\) | 852            | 213.0703        |
| B                      | \(K_3\) | 1137           | 284.2168        |
| boatG                  |      | 718,875        | 179718.6504     |
| R                      | \(K_1\) | 1069           | 267.2324        |
| G                      | \(K_2\) | 890            | 222.3926        |
| B                      | \(K_3\) | 1065           | 266.2051        |

Secondly, the statistical properties of encrypted images can be investigated by correlation diagrams. These diagrams use certain amount of randomly chosen adjacent image pixel intensities. Usual amount of pixel intensity pairs is 1000. The pixel intensities in a pair are adjacent either in horizontal, vertical or diagonal direction. Then, the two pixel intensities in a pair are used as two coordinates of a plotted point in the correlation diagram.

An example of two correlation diagrams for plain image \(\text{lenaG}\) and its version encrypted with key \(K_1\) are shown in...
Robustness against differential attacks

Figure 13. Both diagrams contain 1000 points made from 1000 randomly chosen pairs of pixel intensities adjacent in a horizontal direction.

While the first diagram shows that pixel intensity pairs in the plain image are quite correlated (the points are located around line \( y = x \)), the second diagram has more uniform distribution of points. This means that the encryption helps to decrease correlation between adjacent pixel intensities.

Thirdly, there are some numerical parameters that reflect robustness against statistical attacks. The correlation coefficients \( \rho \) are computed for a certain number of intensities of horizontally (\( \rho_h \)), vertically (\( \rho_v \)) or diagonally (\( \rho_d \)) adjacent pixel pairs. Usually, 1000 randomly chosen pairs of adjacent pixel intensities are used. The correlation coefficients are calculated separately for each colour plane of the investigated image by (16).

Ideal values of correlation coefficients \( \rho \) for encrypted images are close to 0. Plain images have values of \( \rho \) close to 1, which means that their pixel intensities are rather closely correlated.

Another parameter that is utilized for evaluating the robustness against statistical attacks is called entropy \( H \). Entropy is computed for individual colour planes of images by (17): 

\[
H_{\text{bins/pix}} = - \sum_{m=0}^{2^L-1} p(m) \log_2(p(m)), \tag{17}
\]

where \( m \) denotes intensity of image pixels, \( L \) stands for colour depth of analysed colour plane and \( p(m) \) represents probability of occurrence for a pixel with intensity \( m \).

The theoretical boundary of entropy for images is the same as used colour depth. In this case, the investigated image could be considered as a source of random information. Practical values of entropy for encrypted images are close to their colour depth.

Obtained values of correlation coefficients \( \rho \) and entropy \( H \) are included in columns 4 to 7 of Table 8. Correlation coefficients were computed from 1000 randomly chosen pairs of adjacent pixel intensities. Characters R, G and B stand for red, green or blue colour plane of images. The symbol “-” is used either for greyscale images (they have only one colour plane) or plain images that were not encrypted.

4.5 Robustness against differential attacks

The point of differential attacks is a comparison of encrypted images that were obtained by encryption of similar plain images by the same key. The best practice is usage of two plain images \( P_1 \) and \( P_2 \) that differ only in an intensity of one image pixel (in one colour plane) by one intensity level. In this case, the resulting encrypted images can be denoted simply as \( E_1 \) and \( E_2 \).

There are two commonly used measures that evaluate the robustness against differential attacks. First one, Number of Pixel Change Ratio (NPCR) sums up the number of pixels that have different intensities in a pair of corresponding colour planes of encrypted images \( E_1 \) and \( E_2 \). NPCR is computed separately for each colour plane of an encrypted image by (18):

\[
\text{NPCR[%]} = \frac{100}{b \cdot w} \sum_{l=1}^{b} \sum_{k=1}^{w} \text{Diff}_m(l,k), \tag{18}
\]

\[
\text{Di f f}_m(l,k) = \begin{cases} 
0 & \text{if } E_1(l,k) = E_2(l,k) \\
1 & \text{if } E_1(l,k) \neq E_2(l,k)
\end{cases},
\]

where \( b \) and \( w \) stand for colour plane height and width, \( l = 1, 2, 3, \ldots, b \) and \( k = 1, 2, 3, \ldots, w \) are height and width indices, \( \text{Di f f}_m \) is a difference matrix and \( E_1 \) and \( E_2 \) stand for respective colour planes of two encrypted images.

The second parameter, Unified Average Changing Intensity (UACI) operates with intensities of differences between colour planes of the encrypted images. UACI is calculated for individual colour planes by (19):

\[
\text{UACI[%]} = \frac{100}{b \cdot w} \sum_{l=1}^{b} \sum_{k=1}^{w} \left| \frac{E_1(l,k) - E_2(l,k)}{2^L - 1} \right|, \tag{19}
\]

where \( L \) denotes colour depth of investigated colour plane.

Values of NPCR and UACI obtained by (18) and (19) are affected by location of pixel with modified intensity and also by direction of the change (the intensity could be decremented or incremented). In order to suppress significantly good or bad values, a set of repeated measurements with various locations of changed intensity and different directions of intensity change is used for the calculation of an arithmetic mean. The means presented as final NPCR and UACI values are usually computed from a set of 100 measurements.
While the numerical values of NPCR and UACI can be used for comparison of various image encryption algorithms, sometimes their significance is not clear enough. The importance of certain levels of NPCR and UACI values was illustrated by Wu et al. in 2011 [33] in a paper that specified expected values of NPCR and UACI for some commonly used resolutions of test images. Expected values of NPCR and UACI for a colour plane (or a greyscale image) with a resolution of 512×512 pixels and colour depth of 8 bits per pixel are 99.61094% for NPCR and (33.3115%, 33.6156%) for UACI.

| Image and colour plane | Key | \( \rho_s [-] \) | \( \rho_v [-] \) | \( \rho_d [-] \) | \( H \) [bits/px] | NPCR [%] | UACI [%] |
|-----------------------|-----|-----------------|-----------------|-----------------|-----------------|---------|---------|
| lena                  | R   | 0.9739          | 0.9725          | 0.9539          | 7.5889          | Not computed |
|                       | G   | 0.9734          | 0.9719          | 0.953           | 7.106           |         |         |
|                       | B   | 0.9731          | 0.9723          | 0.9546          | 6.8147          |         |         |
|                       | R  \( K_1 \) | -0.0007         | -0.0008         | -0.0012         | 7.9992          | 99.6125  | 33.4811 |
|                       | G  \( K_2 \) | 0.0009          | -0.0002         | -0.0011         | 7.9993          | 99.6138  | 33.4796 |
|                       | B  \( K_3 \) | 0.0007          | -0.0006         | 0.001           | 7.9993          | 99.6128  | 33.4848 |
|                       | R  \( K_1 \) | 0.0008          | 0.0012          | 0.0002          | 7.9993          | 99.6131  | 33.4726 |
|                       | G  \( K_2 \) | -0.0014         | 0.0009          | 0.0007          | 7.9992          | 99.614   | 33.4859 |
|                       | B  \( K_3 \) | -0.0013         | 0.0007          | 0.0019          | 7.9993          | 99.6133  | 33.4755 |
|                       | R  \( K_1 \) | -0.0007         | -0.0014         | -0.0019         | 7.9993          | 99.6145  | 33.4752 |
|                       | G  \( K_2 \) | -0.0012         | -0.0009         | 0.0022          | 7.9993          | 99.6125  | 33.4801 |
|                       | B  \( K_3 \) | -0.0008         | -0.0018         | 0.002           | 7.9993          | 99.6139  | 33.4792 |
| lena\(_G\)            |     | 0.9731          | 0.9761          | 0.9561          | 7.2344          |         |         |
|                       | \( K_1 \) | -0.0012         | -0.0009         | 0.0003          | 7.9994          | 99.6132  | 33.4766 |
|                       | \( K_2 \) | -0.001          | 0.0011          | -0.0008         | 7.9992          | 99.614   | 33.4828 |
|                       | \( K_3 \) | 0.0002          | 0.0016          | 0.0012          | 7.9994          | 99.6143  | 33.4814 |
| peppers               | R   | 0.9578          | 0.9654          | 0.9478          | 7.3388          |         |         |
|                       | G   | 0.9616          | 0.9681          | 0.9439          | 7.4963          |         |         |
|                       | B   | 0.9541          | 0.9641          | 0.9545          | 7.0583          |         |         |
|                       | R  \( K_1 \) | -0.0021         | 0.0012          | 0.0014          | 7.9992          | 99.6138  | 33.4767 |
|                       | G  \( K_2 \) | -0.0011         | -0.0018         | -0.0014         | 7.9992          | 99.6123  | 33.48   |
|                       | B  \( K_3 \) | 0.0015          | 0.0002          | 0.0014          | 7.9993          | 99.6133  | 33.4817 |
|                       | R  \( K_1 \) | 0.001           | 0.0013          | -0.0008         | 7.9993          | 99.6144  | 33.4774 |
|                       | G  \( K_2 \) | -0.0017         | 0.0002          | 0.0006          | 7.9993          | 99.6146  | 33.4827 |
|                       | B  \( K_3 \) | 0.0008          | -0.0015         | 0.0019          | 7.9993          | 99.6131  | 33.4796 |
|                       | R  \( K_1 \) | -0.0002         | 0.0001          | 0.0012          | 7.9993          | 99.6136  | 33.4742 |
|                       | G  \( K_2 \) | -0.0004         | 0.0018          | 0.0006          | 7.9993          | 99.6157  | 33.4796 |
|                       | B  \( K_3 \) | 0.001           | -0.0004         | -0.0008         | 7.9993          | 99.6129  | 33.4758 |
| peppers\(_G\)         |     | 0.9705          | 0.9726          | 0.9656          | 7.5944          |         |         |
|                       | \( K_1 \) | 0.0001          | 0.0005          | 0.0012          | 7.9993          | 99.6128  | 33.4772 |
|                       | \( K_2 \) | 0.0015          | -0.0011         | -0.0004         | 7.9994          | 99.6131  | 33.4784 |
|                       | \( K_3 \) | -0.0009         | -0.0008         | -0.0002         | 7.9993          | 99.6139  | 33.4755 |
| baboon                |     | 0.8506          | 0.7134          | 0.6809          | 7.3579          |         |         |
|                       | \( K_1 \) | -0.0004         | -0.0029         | 0.0006          | 7.9993          | 99.6138  | 33.4783 |
|                       | \( K_2 \) | 0.0013          | -0.0009         | -0.0021         | 7.9994          | 99.6141  | 33.4813 |
|                       | \( K_3 \) | -0.0015         | 0.0001          | 0.0012          | 7.9992          | 99.6132  | 33.4766 |
| boat\(_G\)            |     | 0.9772          | 0.9668          | 0.9554          | 7.4847          |         |         |
|                       | \( K_1 \) | -0.0013         | 0.0012          | 0.0004          | 7.9993          | 99.6136  | 33.4803 |
|                       | \( K_2 \) | 0.002           | 0.0009          | -0.0001         | 7.9994          | 99.6128  | 33.4789 |
|                       | \( K_3 \) | -0.0009         | 0.0019          | 0.0023          | 7.9993          | 99.6143  | 33.4814 |
Table 9: Calculated values for computational complexity

| Image       | Key | $t_{enc}$ [ms] | $t_{dec}$ [ms] | $v_{enc}$ [MB/s] | $v_{dec}$ [MB/s] | $cyc_{enc}$ [cycles/B] | $cyc_{dec}$ [cycles/B] |
|-------------|-----|----------------|----------------|------------------|-----------------|------------------------|------------------------|
| true-colour images (resolution of 512×512 pixels) |     |                |                |                  |                 |                        |                        |
| lena        | $K_1$ | 479.69         | 472.17         | 1.5635           | 1.5884          | 1524.89               | 1500.99                |
|             | $K_2$ | 480.73         | 473.04         | 1.5601           | 1.5855          | 1528.2                | 1503.74                |
|             | $K_3$ | 477.07         | 471.33         | 1.5721           | 1.5912          | 1516.55               | 1498.32                |
| peppers     | $K_1$ | 480.32         | 472.28         | 1.5615           | 1.5880          | 1526.9                | 1501.34                |
|             | $K_2$ | 478.76         | 470.27         | 1.5666           | 1.5948          | 1521.94               | 1494.94                |
|             | $K_3$ | 479.21         | 474.3          | 1.5651           | 1.5813          | 1523.38               | 1507.75                |
| greyscale images (resolution of 512×512 pixels) |     |                |                |                  |                 |                        |                        |
| lenaG       | $K_1$ | 131.99         | 129.4          | 1.894            | 1.932           | 1258.78               | 1234.04                |
|             | $K_2$ | 132.34         | 128.71         | 1.8891           | 1.9424          | 1262.09               | 1227.44                |
|             | $K_3$ | 132.89         | 129.14         | 1.8812           | 1.9358          | 1267.35               | 1231.6                 |
| peppersG    | $K_1$ | 131.72         | 128.86         | 1.898            | 1.94           | 1256.14               | 1228.92                |
|             | $K_2$ | 131.86         | 128.61         | 1.8959           | 1.9439          | 1257.55               | 1226.5                 |
|             | $K_3$ | 132.01         | 128.22         | 1.8938           | 1.9498          | 1258.94               | 1222.78                |
| baboonG     | $K_1$ | 131.84         | 128.49         | 1.8963           | 1.9457          | 1257.31               | 1225.38                |
|             | $K_2$ | 132.33         | 128.03         | 1.8892           | 1.9526          | 1262.03               | 1221.01                |
|             | $K_3$ | 132.13         | 128.25         | 1.8921           | 1.9493          | 1260.05               | 1223.12                |
| boatG       | $K_1$ | 132.38         | 129.45         | 1.8885           | 1.9313          | 1262.48               | 1234.51                |
|             | $K_2$ | 132.55         | 128.88         | 1.8861           | 1.9398          | 1264.06               | 1229.09                |
|             | $K_3$ | 132.79         | 128.92         | 1.8827           | 1.9392          | 1266.36               | 1229.48                |

Since the mentioned expected values are valid for a significance level $\alpha = 0.001$, resulting confidence level for resistance against differential attacks is $100 \cdot (1 - 0.001) = 99.9\%$ [33].

Calculated values of NPCR and UACI are presented in columns 8 and 9 of Table 8. The presented values are arithmetic means of 100 repeated measurements. Letters R, G and B indicate red, green or blue colour plane of used image. Symbol "-" denotes greyscale images or plain images that were not encrypted.

4.6 | Measurement of computational complexity

The computational complexity of image encryption algorithms could be evaluated via several measures. The first one is a measurement of time necessary for an operation with an image (the operation being either encryption or decryption). In the rest of this paper, these times are denoted as $t_{enc}$ and $t_{dec}$. The presented values are arithmetic means of a set of 100 repeated measurements with the same key in order to decrease impact of some faster or slower measurements.

The encryption times $t_{enc}$ and decryption times $t_{dec}$ do not take parameters of used image as resolution or colour depth into consideration. Therefore, encryption speed $v_{enc}$ and decryption speed $v_{dec}$ are computed from $t_{enc}$ or $t_{dec}$ via (20):

$$v_{oper} [\text{MB/s}] = \frac{b \cdot w \cdot \text{num}_{ip}}{2^{20} \cdot t_{oper}},$$

where $b$, $w$ and $\text{num}_{ip}$ stand for image height, width and number of its colour planes, a constant of $2^{20}$ represents number of bytes in a megabyte and $t_{oper}$ is a time necessary either for one encryption ($t_{enc}$) or one decryption ($t_{dec}$). The $t_{oper}$ is measured in seconds.

Since the PCs used for testing of image encryption algorithms have different hardware components, the configuration of used PC can be taken in account by calculating the number of CPU cycles necessary for one operation $cyc_{oper}$ by (21):

$$cyc_{oper} [\text{cycles/B}] = \frac{f_{CPU} \cdot t_{oper}}{b \cdot w \cdot \text{num}_{ip}},$$

where $f_{CPU}$ represents CPU frequency. Please take note that the number of CPU cores is not considered as the used computational environment (MATLAB R2015a) utilises only one CPU core.

Obtained encryption and decryption times $t_{enc}$, $t_{dec}$, encryption and decryption speeds $v_{enc}$, $v_{dec}$ and the number of CPU cycles necessary for encryption and decryption of one image byte $cyc_{enc}$, $cyc_{dec}$ are included in Table 9. The encryption and decryption times $t_{enc}$, $t_{dec}$ are arithmetic means of a set of 100 repeated measurements.

The computational complexity could be expressed also in big $O$ notation. Encryption of image with $\text{num}_{ip}$ pixels with the proposed algorithm takes approximately $O(36 \cdot \text{num}_{ip})$ operations. In this case, the most complex operation is division [34].
Comparison of computational complexity by means of big $O$ values could be tricky since many papers do not report exact values or they do not report them at all. Generally speaking, the complexity of many image encryption algorithms is linear (the order of complexity is $O(n)$) and it rises with number of processed image pixels. This was reported also in some of the referenced papers [25, 30].

### 4.7 Discussion

Numerical values presented in Tables 8 and 9 point to several conclusions. First of all, the usage of encryption decreases values of correlation coefficients $\rho$. Also their values for individual images are similar, therefore the usage of different keys does not have a significant impact on statistical properties of the resulting encrypted images. Furthermore, correlation coefficients are similar also for all four testing images. These results illustrate that the proposed image encryption algorithm obtains balanced results for various plain images.

Achieved values of entropy $H$ show that the encrypted images could be considered more random than the plain images. The values for encrypted images are also much closer to the theoretical bound of 8 bits per image pixel.

The presented values of NPCR and UACI fulfil the condition of expected values set by the paper of Wu et al. [33]. The differences between values obtained with usage of various keys or different plain images are rather small. Also, the results do not show any pattern—there is not a case where one of the colour planes achieves best score for all three keys.

Encryption and decryption times $t_{enc}$, $t_{dec}$ are consistent with different plain images with identical parameters (resolution and colour depth) and various keys. One of the significant differences can be found between the duration of encryption and decryption. This situation is caused by a more complex approach for non-linear substitution during the encryption. As the decryption algorithm has the element index already loaded when it obtains substituted pixel intensity, the decryption is a bit faster than encryption.

The second difference is more visible by comparing the values of encryption and decryption speeds $v_{enc}$, $v_{dec}$ and corresponding numbers of CPU cycles $\sigma_{enc}$, $\sigma_{dec}$. Presented results show that the speeds are decreasing for larger images. This is caused by a rather complex PR generation and quantisation procedures. Used techniques suppress the flaws of LM but on the other hand they take more time than all other steps of the whole algorithm. Therefore, generation and quantisation of longer PR sequences for larger images would decrease speed of the proposed image encryption algorithm.

### 4.8 Comparison with similar approaches

This section compares obtained numerical parameters for the proposed image encryption algorithm with values reported in other papers [9–11, 14, 16–18]. The comparison presented in Table 10 was done with plain images lena and lenaG since they were used also by majority of similar algorithms. Values for true-colour image lena represent only its red colour plane as some papers did not provide data for other colour planes. The proposed solution used key $K_1$ for all calculations.

Correlation coefficients $\rho$ favour the proposed solution for both true-colour and greyscale images. Some approaches (e.g. [18]) reported almost similar results; however, the combinations of $\rho$ are more balanced for the proposed solution.

Our proposal records the highest values also for entropy $H$. While the values reported by [9, 14] for red colour plane or true-colour image lena are significantly lower, the values for greyscale image lenaG achieved by [11, 16, 18] are close to the proposed solution.

Presented values of NPCR and UACI could be perceived as a weakness of the described solution. While they are higher than the expected values [33], they are not as good as the values reported in other papers. Properties of the proposed solution regarding differential attacks could be improved by the addition of some blocks, for example, the combination with more PR sequences or more complex diffusion stage. However, ideas like
these could raise computational complexity and lead to a solution with less balanced performance.

Another advantage of the proposed solution is rather small number of CPU cycles necessary for operations with one byte of image data. Only one algorithm [14] reported better result; however, it was already broken [15] mainly due to its simplicity. Furthermore, the second fastest solution for greyscale images [18] uses approximately 4 times more CPU cycles for encryption than the proposed algorithm.

5 | CONCLUSION

This paper described an image encryption algorithm that utilises a plaintext-related quantisation scheme for the processing of one PR sequence. Main novelty of the proposed solution is the mentioned quantisation scheme which could be perceived as a two step procedure. While the first step suppresses known drawbacks of the LM that generates elements of PR sequences, the second step computes final element values with respect to previously processed pixel intensities. This solution enables decryption without any additional information. Also, due to the processing of whole rows of pixel intensities at once, the proposed solution is quite efficient.

Furthermore, the properties of the described solution were evaluated by commonly used numerical parameters with some well known testing images. Obtained results were then compared with the values reported for some other algorithms that also use plaintext-related designs. While numerical results show that our proposal is not as robust against differential attacks as some other algorithms, it still reaches expected values of used parameters. The robustness against statistical attacks is by means of numerical values; the highest among reported results.

The partial conclusions presented in this paper imply that the usage of a plaintext-related technique during quantisation stage results in a same behaviour as is achieved by applying plaintext-related approaches in other stages and whole image encryption algorithm remains effective. Since the utilisation of plaintext-related solution in some complex stages such as diffusion could negatively affect the computational complexity of the whole image encryption algorithm, we would like to investigate the possibilities of introducing plaintext-related techniques to other simple stages of image encryption algorithms in our future work.

ACKNOWLEDGEMENTS

This work was supported by Slovak Research and Development Agency research grant no. APVV-17-0208.

ORCID

Jakub Oravec https://orcid.org/0000-0002-0484-226X
Luboš Ovseník https://orcid.org/0000-0002-5594-6157
Ján Petaj https://orcid.org/0000-0002-5428-4008

REFERENCES

1. Pichler, F., Schraiger, J.: Finite dimensional generalized baker dynamical systems for cryptographic applications. In: Proceedings of the International Conference on Computer Aided Systems Theory EUROCAST 1995, pp. 465–476. Springer, Berlin (1995)
2. Fridrich, J.: Symmetric ciphers based on two-dimensional chaotic maps. Int. J. Bifurcation Chaos 8(6), 1259–1284 (1998)
3. Oravec, J., Turán, J.: Substitution steganography with security improved by chaotic image encryption. In: Proceedings of International Conference on Informatics 2017, pp. 284–288. ACM, New York (2017)
4. Fang, D., Sun, S.: A new scheme for image steganography based on hyperchaotic map and DNA sequence. J. Inf. Inf. Hiding Multimedia Signal Process 9(2), 392–399 (2018)
5. Chen, X., Hu, C.-J.: Adaptive medical image encryption algorithm based on multiple chaotic mapping. Saudi J. Biol. Sci. 24(8), 1821–1827 (2017)
6. Abundiz-Pérez, F., et al.: A fingerprint image encryption scheme based on hyperchaotic Rössler map. Math. Prog. Eng. 1–15 (2016)
7. Solak, E., et al.: Cryptanalysis of Fridrich’s chaotic image encryption. Int. J. Bifurcation Chaos 20(5), 1405–1413 (2010)
8. Xie, F.Y., et al.: On the cryptanalysis of Fridrich’s chaotic image encryption scheme. Signal Process. 132(C), 150–154 (2017)
9. Kanso, A., Ghabchel, M.: A novel image encryption algorithm based on a 3D chaotic map. Commun. Nonlinear Sci. Numer. Simul. 17(7), 2943–2959 (2012)
10. Fu, C., et al.: A chaos-based image encryption scheme with a plaintext related diffusion. Proc. Int. Conf. on Information, Communications & Signal Processing ICICS 2013, pp. 1–5. Springer, Cham (2013)
11. Zhang, Y.: A chaotic system based image encryption algorithm using plaintext-related confusion. Indones. J. Electrical Eng. Comput. Sci. 12(11), 7952–7962 (2014)
12. Norouzi, B., et al.: A simple, sensitive and secure image encryption algorithm based on hyperchaotic system with only one round diffusion process. Multimedia Tools Applic. 71(3), 1469–1497 (2014)
13. Zhang, Y., et al.: Breaking an image encryption algorithm based on hyperchaotic system with only one round diffusion process. Nonlinear Dyn. 76(3), 1645–1650 (2014)
14. Muriño-Escobar, M.A., et al.: A RGB image encryption algorithm based on total plain image characteristics and chaos. Signal Process. 109, 119–131 (2015)
15. Fan, H., et al.: Cryptanalysis of a plaintext-related chaotic RGB image encryption scheme using total plain image characteristics. Multimedia Tools Applic. 77(15), 20103–20127 (2018)
16. Chai, X., et al.: A fast chaos-based image encryption scheme with a novel plain image-related swapping block permutation and block diffusion. Multimedia Tools Applic. 76(14), 15561–15585 (2017)
17. Wang, X., et al.: Image encryption algorithm based on multiple mixed hash functions and cyclic shift. Opt. Lasers Eng. 107, 370–379 (2018)
18. Li, Z., et al.: A novel plaintext-related image encryption scheme using hyper-chaotic system. Nonlinear Dyn. 94(2), 1319–1333 (2018)
19. Sun, S., Guo, Y., Wu, R.: A novel image encryption scheme based on 7D hyperchaotic system and row-column simultaneous swapping. IEEE Access 7, 28539–28547 (2019)
20. Ovseník, L., et al.: Image encryption algorithm with plaintext related chaining. Comput. Inform. 38(3), 647–678 (2019)
21. May, R.R.: Simple mathematical models with very complicated dynamics. Nature 261(5560), 459–467 (1976)
22. Gleick, J.: Chaos: Making a New Science. Vintage Books, London (1998)
23. Arroyo, D., Alvarez, G., Fernandez, V.: On the inadequacy of the logistic map for cryptographic applications. Paper presented at 10th Spanish meeting on cryptography and information security, Salamanca, Spain, September 2008, pp. 1–6
24. Persohn, K.J., Povinelli, R.J.: Analyzing logistic map pseudorandom number generators for periodicity induced by finite precision floating-point representation. Chaos, Solitons Fractals 45(3), 238–245 (2012)
25. Oravec, J., et al.: A chaotic image encryption algorithm robust against the phase space reconstruction attacks. Acta Polytech. Hung. 16(3), 37–57 (2019)
26. Oravec, J., Turán, J., Ovseník, L.: Image encryption technique with key diffusion by coupled map lattice. In: Proceedings of International Conference on Radioelektronika 2018, pp. 1–6. IEEE, Piscataway (2018)
27. Steingartner, W., Galinec, D.: The rôle of categorical structures in infinitesimal calculus. J. Appl. Math. Comput. Mech. 12(1), 107–119 (2013)
28. The USC-SIPI image database. http://sipi.usc.edu/database/. Accessed 4 Sep 2020
29. NIST SP 800-22 rev. 1a: A statistical test suite for random and pseudorandom number generators for cryptographic applications. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf. Accessed 4 Sep 2020
30. Ying-Qian, Z., Xing-Yuan, W.: A symmetric image encryption algorithm based on mixed linear–nonlinear coupled map lattice. Inf. Sci. 273, 329–351 (2014)
31. Khanzadi, H., et al.: Image encryption using random bit sequence based on chaotic maps. Arab. J. Sci. Eng. 39(2), 1039–1047 (2014)
32. ur Rehman, A., et al.: An efficient mixed inter-intra pixels substitution at 2 bits-level for image encryption technique using DNA and chaos. Optik 153, 117–134 (2018)
33. Wu, Y., Noonan, J.P., Agaian, S.: NPCR and UACI randomness tests for image encryption. J. Sel. Areas Telecommun. 2(4), 31–38 (2011)
34. Steingartner, W., Novitzká, V.: Coalgebras for modelling observable behaviour of programs. J. Appl. Math. Comput. Mech. 16(2), 145–157 (2017)