Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 932875, 15 pages
http://dx.doi.org/10.1155/2014/932875

Research Article

Color Image Scrambling Technique Based on Transposition of Pixels between RGB Channels Using Knight's Moving Rules and Digital Chaotic Map

Adrian-Viorel Diaconu,[1,2] Alexandru Costea,[3] and Marius-Aurel Costea[1]

[1] *IT&C Department, Lumina—The University of South-East Europe, 021187 Bucharest, Romania*
[2] *ETTI Faculty, University Politehnica of Bucharest, 061071 Bucharest, Romania*
[3] *Faculty of Electronic and Information Military Systems, Military Technical Academy, 050141 Bucharest, Romania*

Correspondence should be addressed to Adrian-Viorel Diaconu; adrian.diaconu@lumina.org

Received 20 November 2013; Accepted 5 March 2014; Published 24 April 2014

Academic Editor: Teh-Lu Liao

Nowadays, increasingly, it seems that the use of rule sets of the most popular games, particularly in new images' encryption algorithms designing branch, leads to the crystallization of a new paradigm in the field of cryptography. Thus, motivated by this, the present paper aims to study a newly designed digital image scrambler (as part of the two fundamental techniques used to encrypt a block of pixels, i.e., the permutation stage) that uses knight's moving rules (i.e., from the game of chess), in conjunction with a chaos-based pseudorandom bit generator, abbreviated PRBG, in order to transpose original image's pixels between RGB channels. Theoretical and practical arguments, rounded by good numerical results on scrambler's performances analysis (i.e., under various investigation methods, including visual inspection, adjacent pixels' correlation coefficients' computation, key's space and sensitivity assessment, etc.) confirm viability of the proposed method (i.e., it ensures the coveted confusion factor) recommending its usage within cryptographic applications.

1. Introduction

It is well known that, clearly, images are considered to contain a huge amount of information (e.g., a family photos might tell not only who are its members but also their rough ages or physical features, etc.) and thus, as an essential and integrated part of the advanced data protection techniques (i.e., image encryption [1–5] and watermarking [6–12]), digital image scramblers are designed to transform clear images into unintelligible ones (i.e., whose inherent information is protected from any unauthorized use).

In recent years, besides classical approaches in the designing of bidimensional bijection based digital image scramblers (e.g., chaos-based [13–17], cellular automata based [18, 19], interpixel displacement or image-blocks transposition based [20–22], spatial transform based [23, 24], and matrix decomposition based [25] scramblers), few new designs based on rule sets of the most popular games (e.g., Sudoku puzzle based [26–28], Chinese chess knight's tour based [29, 30], Rubik's cube principle based [31, 32], and Poker shuffling rules based [33]) found their rightful place.

This paper aims to contribute to the crystallization of reminded paradigm by presenting a new model of digital image scrambler, based on transposition of pixels between RGB channels using knight's moving rules (i.e., from the game of chess) in conjunction with a chaos-based PRBG.

The rest of this paper is organized as follows. Section 2 presents, as comprehensively possible, the design of proposed digital image scrambler. Section 3 showcases scrambler's performance analysis. Finally, Section 4 concludes the work carried out.

2. Algorithm Description/Methodology

This section deals with the presentation, as comprehensively possible, of the designing stages of the proposed digital image scrambler (i.e., use of knight's moving rules, use of digital

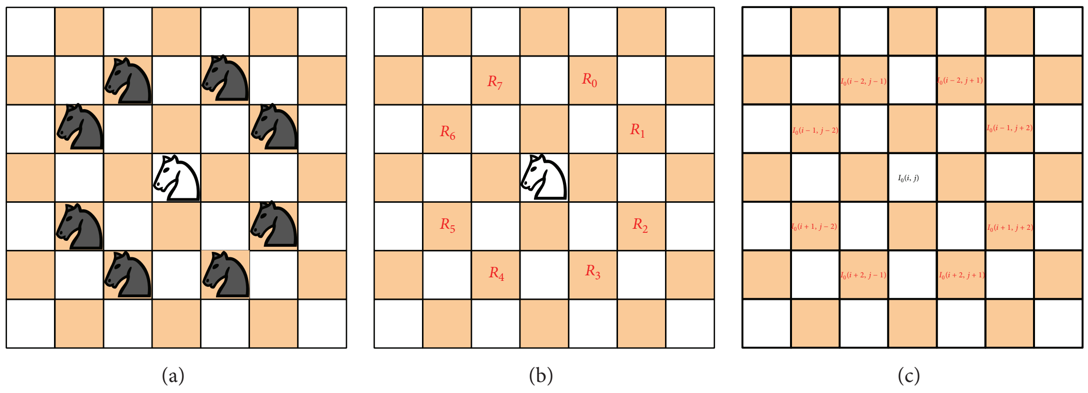

Figure 1: Knight's moving rules. (a) Knight's possible moves, (b) moves' possible annotations (i.e., as rules), and (c) rules' associated coordinates.

chaotic map, and description of the main procedures and algorithms).

2.1. Use of Knight's Moving Rules in Scrambler's Design. Knight's moving rules, unlike those of all other pieces, follow an "elbowed" nonlinear shape (i.e., in form of letter L, in other words, a horizontal displacement of two squares, followed by a vertical displacement of one square, or a vertical displacement of two squares, followed by a horizontal displacement of one square, in either direction). Thus, as shown in Figure 1(a), one can define a set of eight possible moves, each of which can be labeled as suggested in Figure 1(b).

Considering any 24-bit color bitmap I_0, of the size $m \times m \times k$, where $k \in \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$ represents image's color channel, as a chessboard with fixed orientation and any pixel $I_0(i, j, k)$ within this image as knight's initial position, each moving rule (i.e., pixel's new destination) can be mathematically expressed, in relative coordinates, as shown in Figure 1(c).

Since a digital image scrambler aims, among others, to minimize adjacent pixels' correlation we have considered the following slightly different approaches:

(i) in order to ensure the fact that moved pixels' values will be less correlated with those of their new neighboring pixels, any pixel $I_0(i, j, k)$ will not be moved once but n times, using the same moving rule R_x;

(ii) reduced susceptibility of correlation between moved pixel's value and of new neighboring pixels' (i.e., achieved by applying previous approach) will be strengthened by the fact that once pixel's new coordinates (i.e., i and j) have been determined, according to a binary random variable $P_{y,z}$, its value will be simultaneously transposed between color channels.

Obviously a series of further questions arises, for example, "which are n's limits of variations?" and "what happens if pixel's new destination's coordinates exceed image's limits?" or "in what way the random binary variable $P_{y,z}$ influences pixels' transposition between image's color channels?"

The answers to these questions (without any restriction, i.e., one can think of other solutions) are as follows:

(i) from a theoretical perspective, n can accommodate any positive integer value; however, in practical terms (related to PRBG's use, as it will be discussed further within this section), its value will be restricted to an 8-bit maximum value (i.e., $0 \leq n \leq 255$);

(ii) indeed, there is an imminent possibility that pixels' new destination's coordinates exceed image's limits (especially when pixels to be moved are situated at image's extreme limits or when n's value is close to maximum) and therefore, considering the image as being virtually cyclic, i and j coordinates are to be kept within $[1, m]$ interval (i.e., in a relatively plastic but more proper expression, "exiting" through one side of the image, the pixel will "return" within the image on the other side, completing "in a natural way" motion's specific shape);

(iii) when it comes to transpose pixels' values between image's color channels, the basic idea is to ensure that all pixels are removed from their initial color plan (e.g., considering any pixel $I_0(i, j, \mathrm{R})$, i.e., belonging to the red plan, its destination color plane will be forced to green or blue; i.e., $k \in \{\mathrm{G}, \mathrm{B}\}$). This criterion is satisfied by applying, for example (i.e., notwithstanding the fact that other rules may be proposed), transposing rules presented in Figure 2.

2.2. Use of the Chaotic Map in Scrambler's Design. Although any PRBG can be used, for example, [34–40], due to its good cryptographic properties and suitability for cryptographic application, that is, large key space and good randomness features, as proven in [41], PRNG model (1) was used, within designing and testing stages of this work. On PRNG's output sequence of real numbers the multilevel discretization method [42] (e.g., with four thresholds, i.e., 2-bit encoding of each interval) was applied, resulted dibits being spread into two different files (i.e., "Bits_A.txt," containing dibit's first bit, and, resp., "Bits_B.txt," containing dibit's second bit).

Using random sequences of real numbers generated by the orbits of f_T (1), designed based on a binary composition

P_y \ P_z	0	1
0	GBR	BGR
1	BRG	GRB

(a)

P_y \ P_z	0	1
0	GRB	BRG
1	RBG	GBR

(b)

P_y \ P_z	0	1
0	BRG	GBR
1	BGR	RBG

(c)

FIGURE 2: Correlation between the random binary variable $P_{y,z}$ and pixels' transposing rules, when working on pixels' from (a) red color channel, (b) blue color channel, and (c) green color channel.

```
Require:    m, k    % image's dimensions
Ensure:     c       % a temporary counter, initially set to 0
            KMR     % a matrix of the size $m \cdot m \cdot k$, initially set to 0 (i.e., KMR = zeros $(m, m, k)$)
            RNS     % a matrix of the size $m \cdot m \cdot k$, initially set to 0 (i.e., RNS = zeros $(m, m, k)$)
for $p$ =1 : k
    for $q$ =1 : m
        for $r$ = 1 : n
            $Byte_A = strcat\ (Bits\_A.txt\ (c + s))$,   $s = \overline{1,8}$   % takes 8 bits from file Bits_A
            $Byte_B = strcat\ (Bits\_B.txt\ (c + s))$,   $s = \overline{1,8}$   % takes 8 bits from file Bits_B
            KMR $(i, j, k) = bin2dec\ (Byte_A)$           % update KMR matrix
            RNS $(i, j, k) = bin2dec\ (Byte_B)$           % update RNS matrix
            $c = c + 8$                                   % update the temporary counter
        end
    end
end
```

PROCEDURE 1: Computing KMR and RNS (*Bits_A*, *Bits_B*, *m*, *k*).

(2) of two identical one-dimensional chaotic discrete dynamical systems of form (2.2), in conjunction with the method of discretization previously referenced, $m \cdot m \cdot k \cdot 8$ dibit pairs have been generated (i.e., 6.291.456 bits were written in each file), this number being, as seen, directly proportional to the image dimensions. Consider

$$f_T = f_1\left(x_i^1, r_1\right) * f_2\left(x_i^2, r_2\right) = \frac{f_1\left(x_i^1, r_1\right) + f_2\left(x_i^2, r_2\right)}{1 - f_1\left(x_i^1, r_1\right) \cdot f_2\left(x_i^2, r_2\right)}, \tag{1}$$

where x_0^1, x_0^2 are the initial conditions, r_1, r_2 are the control parameters, and x_i^1, x_i^2 are the two orbits obtained by recurrences $x_{i+1}^1 = f_1(x_i^1, r_1)$ and $x_{i+1}^2 = f_2(x_i^2, r_2)$, for any $i \in \{0, 1, 2, \ldots\}$. Consider

$$a * b = \frac{a + b}{1 - a \cdot b}, \tag{2}$$

$$f_1 : [-1, 1] \longrightarrow [-1, 1],$$

$$f_1(x, r_1) = \frac{2}{\pi}\operatorname{arctg}\left(\operatorname{ctg}\left(r_1 \cdot x\right)\right), \quad r_1 \in [1, 10]$$

$$f_2 : [-1, 1] \longrightarrow [-1, 1],$$

$$f_2(x, r_2) = \frac{2}{\pi}\operatorname{arctg}\left(\operatorname{ctg}\left(r_2 \cdot x\right)\right), \quad r_2 \in [1, 10]. \tag{3}$$

In our scrambler's design and during subsequent tests f_T's initial seeding points, respectively, control parameters' values were chosen as follows: $x_0^1 = 0.687754925117$ and $r_1 = 5.9387250254 21$, respectively, $x_0^2 = -0.013462335467$ and $r_2 = 1.237490188615$.

Under the previous circumstances, two matrices were computed, hereafter referred to as KMR and RNS (i.e., the one associated with knight's moving rules and, respectively, the one associated with the number of steps to be applied on each rule). Procedure 1 describes how KMR and RNS are computed.

KMR and RNS matrices are used to establish values of R_x, $P_{y,z}$, and n, as Figure 3 suggests. It can be noticed that we are dealing with a two-step algorithm. Thus, according to $P_{y,z}$'s value, during the first step, pixel $I_0(i, j, k)$ is moved n_a times using knight's rule R_a and then transposed to other color planes, while, during the second step, with its new

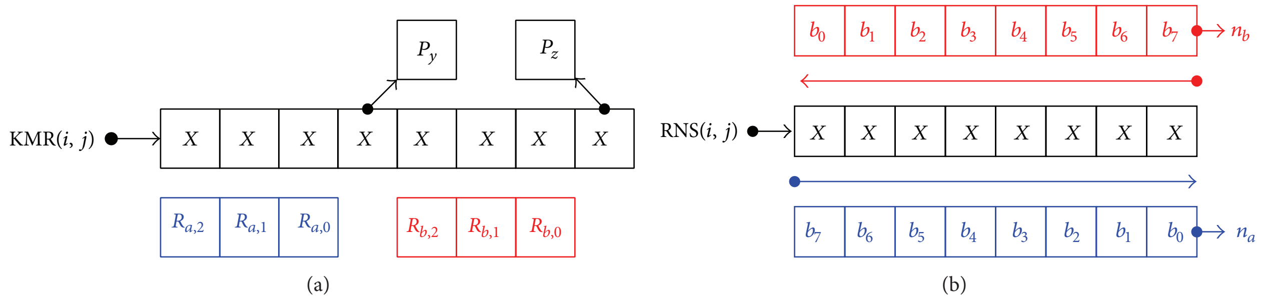
KMR(i, j)
P_y
P_z
X X X X X X X X
$R_{a,2}$ $R_{a,1}$ $R_{a,0}$
$R_{b,2}$ $R_{b,1}$ $R_{b,0}$
(a)
RNS(i, j)
b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7 n_b
b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 n_a
(b)

Figure 3: Method of determining R_x, $P_{y,z}$, and n values: (a) extraction of R_x and $P_{y,z}$ values and (b) establishing the number of steps for each rule.

```
Require:   $i, j, k$   % current pixel's (i.e., the one to be moved) coordinates
$K_{\text{Rule}} = dec2bin(\text{KMR}(i, j, k)$
$R_a = K_{\text{Rule}}\,(1:3)$;   $R_a = bin2dec\,(R_a)$          % compute Knight's rule $R_a$
$R_b = K_{\text{Rule}}\,(5:7)$;   $R_b = bin2dec\,(R_b)$          % compute Knight's rule $R_b$
$P_y = K_{\text{Rule}}\,(4)$;   $P_z = K_{\text{Rule}}\,(8)$          % compute $P_{y,z}$ binary variable, used to establish transposition rule
$n_a = \text{RNS}(i, j, k)$                                  % number of steps to be applied using rule $R_a$
$n_b = dec2bin\,(n_a)$
$n_b = fliplr\,(n_b)$
$n_b = bin2dec\,(n_b)$                                      % number of steps to be applied using rule $R_b$
```

Procedure 2: Extracting features $\left(\text{KMR}\,(i, j, k)_{\substack{1\le i\le m\\ 1\le j\le m\\ 1\le k\le 3}},\ \text{RNS}\,(i, j, k)_{\substack{1\le i\le m\\ 1\le j\le m\\ 1\le k\le 3}}\right)$.

coordinates, the pixel is moved n_b times using knight's rule R_b and then into the another color plane.

Procedure 2 shows how $P_{y,z}$, R_a, n_a, R_b, and n_b are computed, for each image's pixel.

2.3. Practical Example. In order to facilitate understanding of those previously described, the example shown in Figure 4 comes into aid of the reader. For this example, a color bitmap of the size $12 \times 12 \times 3$, a starting pixel $I_0(3, 6, 1)$, $P_{y,z} = \{0, 0\}$, R_3 as first stage rule, R_5 as second stage rule, usage of R_3 three times, and usage of R_5 five times are considered.

2.4. Other Useful Procedures. Before starting with the description of Algorithms 1 and 2 two more procedures must be properly described, namely, the one which deals with computation of pixels' new coordinates, both for scrambling and descrambling algorithms.

Thus, with I_0 representing the pixels' values matrix of a 24-bit color bitmap of the size $m \times m$ and n representing number of steps to be applied with rule R_x, during scrambling process, pixels' new coordinates can be computed using Procedure 3.

On the basis that moving rules are symmetric (e.g., effects of rule R_3 are reversed by applying rule R_7), during descrambling process, pixels' new coordinates can be computed using the same procedure but after modifying one of the function call's parameters, as Procedure 4 suggests.

2.5. Scrambling and Descrambling Algorithms' Description. Depending on random binary variable $P_{y,z}$ value, pixels' color values are transposed between RGB channels using Procedure 5 (i.e., during image's scrambling process) and, respectively, Procedure 6 (i.e., during image's descrambling process).

3. Analysis and Comparison Results

As a general requirement, for any digital image scrambler, the output image should be greatly different in comparison with its plain version (i.e., from statistical point of view). To quantify this requirement, in addition to visual assessment, few statistical estimators can be used, the most commonly ones being presented in the following subsections and according to an already widely used conventional methodology [43, 44].

3.1. Visual Analysis. The purpose of visual testing is to highlight presence of similarities between plain image and its scrambled version (i.e., if the scrambled image does or does not contain any features of the plain image). For the proposed scrambler, same as for all digital image scramblers, expectations are to rearrange plain image's pixels in a deterministic way but with a random-like appearance.

Visual testing was performed on the 512×512 pixels, 24-bit, Lena, Peppers, and Baboon color bitmaps, from the USC-SIPI miscellaneous image dataset [45]. Figure 5 depicts

```
Require:    Bits_A.txt and Bits_B.txt          % files containing bitstreams generated by $f_T$'s trajectories
Compute_KMR_and_RNS (Bits_A, Bits_B, m, k)     % compute KMR and RNS matrices
for $p = 1:k$
    for $q = 1:m$
        for $r = 1:m$
            Extract_features $\left(\text{KMR}(i,j,k)_{\substack{1\le i\le m\\1\le j\le m\\1\le k\le 3}}, \text{RNS}(i,j,k)_{\substack{1\le i\le m\\1\le j\le m\\1\le k\le 3}}\right)$
            $I_0^a(i,j,k) =$ Compute_limits_on_scrambling $\left(I_0(i,j,k)_{\substack{1\le i<m\\1\le j<m}}, R_a, n_a\right)$
            $I_0^b(i,j,k) =$ Compute_limits_on_scrambling $\left(I_0^a(i,j,k)_{\substack{1\le i<m\\1\le j<m}}, R_b, n_b\right)$
            Transpose_pixel_color_values_on_scrambling $\left(I_0(i,j,k), I_0^a(i,j,k), I_0^b(i,j,k), P_y, P_z, k\right)$
        end
    end
end
```

Algorithm 1: Scrambling image $\left(I_0(i,j,k)_{\substack{1\le i\le m\\1\le j\le m\\1\le k\le 3}}\right)$.

```
Require:    Bits_A.txt and Bits_B.txt          % files containing bitstreams generated by $f_T$'s trajectories
Compute_KMR_and_RNS (Bits_A, Bits_B, m, k)     % compute KMR and RNS matrices
for $p = k:-1:1$
    for $q = m:-1:1$
        for $r = m:-1:1$
            Extract_features $\left(\text{KMR}(i,j,k)_{\substack{1\le i\le m\\1\le j\le m\\1\le k\le 3}}, \text{RNS}(i,j,k)_{\substack{1\le i\le m\\1\le j\le m\\1\le k\le 3}}\right)$
            $I_0^b(i,j,k) =$ Compute_limits_on_descrambling $\left(I_0(i,j,k)_{\substack{1\le i<m\\1\le j<m}}, R_b, n_b\right)$
            $I_0^a(i,j,k) =$ Compute_limits_on_descrambling $\left(I_0^b(i,j,k)_{\substack{1\le i<m\\1\le j<m}}, R_a, n_a\right)$
            Transpose_pixel_color_values_on_descrambling $\left(I_0(i,j,k), I_0^a(i,j,k), I_0^b(i,j,k), P_y, P_z, k\right)$
        end
    end
end
```

Algorithm 2: Descrambling image $\left(I_0(i,j,k)_{\substack{1\le i\le m\\1\le j\le m\\1\le k\le 3}}\right)$.

these test plain images, whereas their scrambled versions are showcased in Figure 6. By comparing them, one can say that there is no perceptual similarity (i.e., no visual information can be observed in the processed versions of plain images).

3.2. Adjacent Pixels' Correlation Coefficients' Analysis. It is well known the fact that, generally in plain images, any arbitrarily chosen pixels are strongly correlated with their adjacent ones (either they are diagonally, vertically, or horizontally oriented) [5, 17, 31]. With respect to this idea, an efficient digital image scrambling scheme must minimize this correlation as much as possible.

Whilst Figure 7 showcases the correlation distributions of horizontally (a), vertically (b), and diagonally (c) adjacent pixels for the Baboon plain image, Figure 8 showcases the correlation distributions (i.e., for the same pixels' adjacency cases) for the Baboon scrambled image.

At the same time, all APCCs (computed over 10.000 pairs of adjacent pixels, randomly selected, for each of the testing directions and for each of the color channels) are summarized in Table 1, for each of the test images.

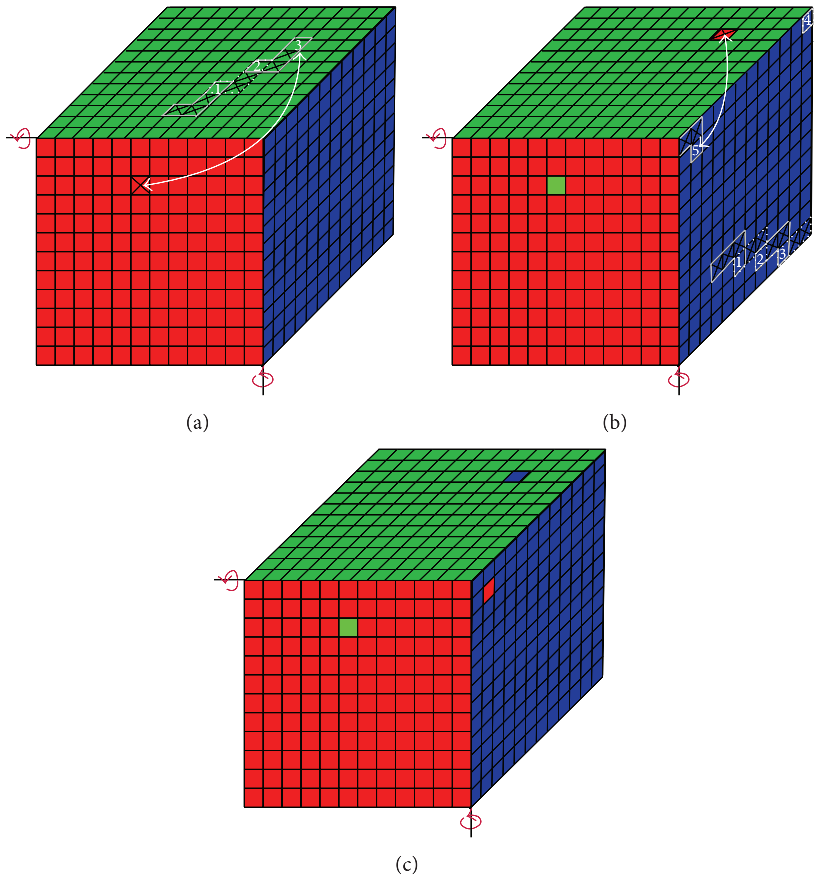

Figure 4: Pixel scrambling/transposition example. (a) First stage's output image, (b) second stage's output image, and (c) final output image.

Figure 5: Test plain images used during algorithm's testing procedures. (a) Lena, (b) Peppers, and (c) Baboon.

It can be easily noticed that neighboring pixels in the plain images are highly correlated; that is, APCCs' values are too high, very close to one. On the contrary, in cases of scrambled images, those values are close to zero, meaning that all neighboring pixels considered in tests are weakly correlated, which is the expected result [5, 46, 47].

3.3. Other Qualitative Measurements' Analysis. Whereas through the visual assessment and APCCs' analysis good scrambling effects are highlighted, based on MSE (mean squared error), NPCR (number of pixel change rate), and UACI (unified average changing intensity) measures, a better and more objective assessment of the proposed scrambler can be accomplished.

MSE is used to evaluate the amount of differences between plain image and its corresponding scrambled one [48]. MSE can be numerically evaluated using (4), with the expected result being a value as high as possible thus denoting

```
Require:    $i, j$   % are current pixel's coordinates, regardless of the color channel in which it lies
Require:    $x$      % represents the moving rule to be applied, where $x \in \{0, 1, 2, 3, 4, 5, 6, 7\}$
Require:    $n$      % represents the number of steps (i.e., how many times the rule is applied)
switch $x$
    case 0
        $i = i - 2 \cdot n$; $j = j + 1 \cdot n$
    case 1
        $i = i - 1 \cdot n$; $j = j + 2 \cdot n$
    case 2
        $i = i + 1 \cdot n$; $j = j + 2 \cdot n$
    case 3
        $i = i + 2 \cdot n$; $j = j + 1 \cdot n$
    case 4
        $i = i + 2 \cdot n$; $j = j - 1 \cdot n$
    case 5
        $i = i + 1 \cdot n$; $j = j - 2 \cdot n$
    case 6
        $i = i - 1 \cdot n$; $j = j - 2 \cdot n$
    otherwise
        $i = i - 2 \cdot n$; $j = j - 1 \cdot n$
end
Remark:     $i, j$   % represent pixel's new coordinates
Ensure:     $\forall i, \forall j \in [1, m]$
switch $i$
    case $i < 0$
        $i = i + 512$
    otherwise
        $i = \mathrm{mod}\,(i, m) + 1$        % ensures $i > 0$
end
switch $j$
    case $j < 0$
        $j = j + 512$
    otherwise
        $j = \mathrm{mod}\,(j, m) + 1$        % ensures $j > 0$
end
```

Procedure 3: Computing limits on scrambling $\left(I_0\,(i, j, :)_{\substack{1\le i<m\\1\le j<m}},\ R_x,\ n\right)$.

```
Require:    $x$   % represents the moving rule applied on scrambling procedure
$x = (\mathrm{mod}\,(x + 4), 8)$
Remark:     $x$   % represents the symmetric moving rule, which is to be applied on descrambling procedure
Compute_limits_on_scrambling $\left(I_0\,(i, j)_{\substack{1\le i<m\\1\le j<m}},\ R_x,\ n\right)$
```

Procedure 4: Computing limits on descrambling $\left(I_0\,(i, j, :)_{\substack{1\le i<m\\1\le j<m}},\ R_x,\ n\right)$.

nonidentical images (i.e., the higher the value of the MSE is, the greater the differences between the two images are). Consider

$$\mathrm{MSE}_C\,(I_P, I_S, C) = \frac{1}{W \times H}\sum_{i=1}^{W}\sum_{j=1}^{H}\left[I_P\,(i, j, C) - I_S\,(i, j, C)\right]^2, \tag{4}$$

where I_P represents plain image's associated matrix, I_S represents scrambled image's associated matrix, W and H represent the image dimensions (i.e., width and height), and C is the color channel (i.e., $C \in \{\mathrm{R}, \mathrm{G}, \mathrm{B}\} \equiv \{1, 2, 3\}$).

First shown in [49] and [50] and afterwards extensively studied [51] and presented in transposed fashion (more suitable for usage within scrambled images' assessment) [17], NPCR and UACI measures are designed to estimate the mean

```
Require:    $P_y, P_z$   % the random binary value which establish the transposing rule
            $k$          % current working color channel
Ensure:     Pixel        % Temporary variable
Pixel = $I_0(i, j, k)$
switch $(P_y, P_z, k)$
    case $(0, 0, 1)$
        $I_0(i, j, k) = I_0^a(i, j, k+1); I_0^a(i, j, k+1) = I_0^b(i, j, k+2); I_0^b(i, j, k+2) =$ Pixel
    case $(0, 1, 1)$
        $I_0(i, j, k) = I_0^b(i, j, k+2); I_0^b(i, j, k+2) =$ Pixel
    case $(1, 0, 1)$
        $I_0(i, j, k) = I_0^a(i, j, k+2); I_0^a(i, j, k+2) = I_0^b(i, j, k+1); I_0^b(i, j, k+1) =$ Pixel
    case $(1, 1, 1)$
        $I_0(i, j, k) = I_0^b(i, j, k+1); I_0^b(i, j, k+1) =$ Pixel
    case $(0, 0, 2)$
        $I_0(i, j, k) = I_0^b(i, j, k-1);\ I_0^b(i, j, k-1) =$ Pixel
    case $(0, 1, 2)$
        $I_0(i, j, k) = I_0^a(i, j, k-1); I_0^a(i, j, k-1) = I_0^b(i, j, k+1); I_0^b(i, j, k+1) =$ Pixel
    case $(1, 0, 2)$
        $I_0(i, j, k) = I_0^a(i, j, k+1); I_0^a(i, j, k+1) =$ Pixel
    case $(1, 1, 2)$
        $I_0(i, j, k) = I_0^a(i, j, k+1); I_0^a(i, j, k+1) = I_0^b(i, j, k-1); I_0^b(i, j, k-1) =$ Pixel
    case $(0, 0, 3)$
        $I_0(i, j, k) = I_0^a(i, j, k-1); I_0^a(i, j, k-1) = I_0^b(i, j, k-2); I_0^b(i, j, k-2) =$ Pixel
    case $(0, 1, 3)$
        $I_0(i, j, k) = I_0^a(i, j, k-2); I_0^a(i, j, k-2) = I_0^b(i, j, k-1); I_0^b(i, j, k-1) =$ Pixel
    case $(1, 0, 3)$
        $I_0(i, j, k) = I_0^a(i, j, k-2); I_0^a(i, j, k-2) =$ Pixel
    case $(1, 1, 3)$
        $I_0(i, j, k) = I_0^b(i, j, k-1); I_0^b(i, j, k-1) =$ Pixel
end
```

PROCEDURE 5: Transposing pixel color values on scrambling $\left(I_0(i, j, k), I_0^a(i, j, k), I_0^b(i, j, k), P_y, P_z, k\right)$.

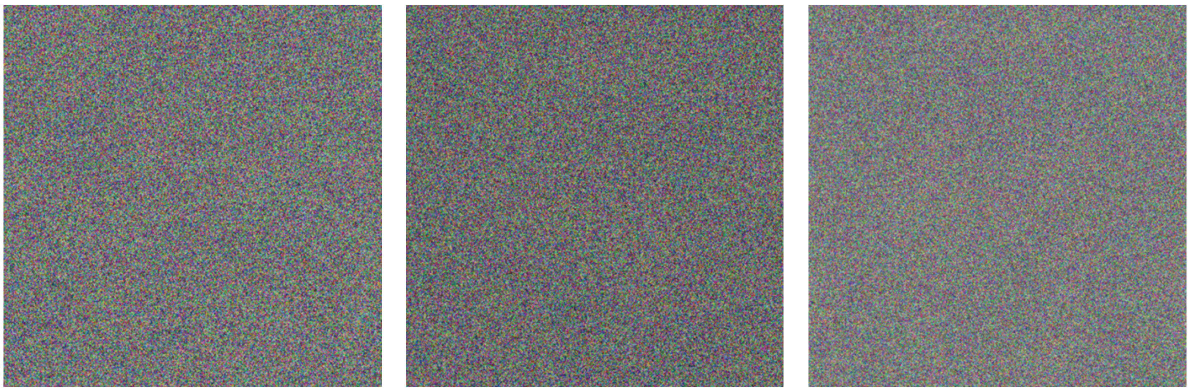
(a) (b) (c)

FIGURE 6: Image scrambling results. (a) Scrambled version of Lena, (b) scrambled version of Peppers, and (c) scrambled version of Baboon.

number of distinct pixels having the same position in the plain image as in corresponding scrambled one, respectively, to estimate the average intensity differences of distinct pixels having the same position in the plain image as in the corresponding scrambledone.

Defined for each of image's color channels [49–51], NPCR indicator can be numerically evaluated using (5) and considering two random images (i.e., the plain image is completely different in comparison with its scrambled version) its maximum expected value [17] is found to be $\varepsilon[\text{NPCR}_C] = 99.6093\%$. Consider

$$\text{NPCR}_C(I_P, I_S, C) = \left(\frac{1}{W \times H}\sum_{i=1}^{W}\sum_{j=1}^{H} D(i, j, C)\right) \times 100\%, \quad (5)$$

```
Require:    $P_y, P_z$   % the random binary value which establish the transposing rule
            $k$          % current working color channel
Ensure:     Pixel        % Temporary variable
Pixel = $I_0(i, j, k)$
switch $(P_y, P_z, k)$
    case $(0, 0, 1)$
        $I_0(i, j, k) = I_0^b(i, j, k+2); I_0^b(i, j, k+2) = I_0^a(i, j, k+1); I_0^a(i, j, k+1) =$ Pixel
    case $(0, 1, 1)$
        $I_0(i, j, k) = I_0^b(i, j, k+2); I_0^b(i, j, k+2) =$ Pixel
    case $(1, 0, 1)$
        $I_0(i, j, k) = I_0^b(i, j, k+1); I_0^b(i, j, k+1) = I_0^a(i, j, k+2); I_0^a(i, j, k+2) =$ Pixel
    case $(1, 1, 1)$
        $I_0(i, j, k) = I_0^b(i, j, k+1); I_0^b(i, j, k+1) =$ Pixel
    case $(0, 0, 2)$
        $I_0(i, j, k) = I_0^b(i, j, k-1); I_0^b(i, j, k-1) =$ Pixel
    case $(0, 1, 2)$
        $I_0(i, j, k) = I_0^b(i, j, k+1); I_0^b(i, j, k+1) = I_0^a(i, j, k-1); I_0^a(i, j, k-1) =$ Pixel
    case $(1, 0, 2)$
        $I_0(i, j, k) = I_0^a(i, j, k+1); I_0^a(i, j, k+1) =$ Pixel
    case $(1, 1, 2)$
        $I_0(i, j, k) = I_0^b(i, j, k-1); I_0^b(i, j, k-1) = I_0^a(i, j, k+1); I_0^a(i, j, k+1) =$ Pixel
    case $(0, 0, 3)$
        $I_0(i, j, k) = I_0^b(i, j, k-2); I_0^b(i, j, k-2) = I_0^a(i, j, k-1); I_0^a(i, j, k-1) =$ Pixel
    case $(0, 1, 3)$
        $I_0(i, j, k) = I_0^b(i, j, k-1); I_0^b(i, j, k-1) = I_0^a(i, j, k-2); I_0^a(i, j, k-2) =$ Pixel
    case $(1, 0, 3)$
        $I_0(i, j, k) = I_0^a(i, j, k-2); I_0^a(i, j, k-2) =$ Pixel
    case $(1, 1, 3)$
        $I_0(i, j, k) = I_0^b(i, j, k-1); I_0^b(i, j, k-1) =$ Pixel
end
```

Procedure 6: Transposing pixel color values on descrambling $(I_0(i, j, k), I_0^a(i, j, k), I_0^b(i, j, k), P_y, P_z, k)$.

Table 1: Adjacent pixels' correlation coefficients' analysis.

Pixels' adjacency	Image status	Color channel	Test images		
			Lena	Peppers	Baboon
Horizontally	Original	R	0.9785	0.9677	0.8722
		G	**0.9747**	**0.9815**	**0.7743**
		B	0.9695	0.9661	0.8857
	Scrambled	R	0.0063	0.0009	0.0011
		G	**0.0110**	**0.0044**	**−0.0009**
		B	0.0104	0.0012	0.0015
Vertically	Original	R	0.9899	0.9695	0.9241
		G	**0.9877**	**0.9831**	**0.8680**
		B	0.9842	0.9696	0.9088
	Scrambled	R	0.0004	−0.0017	0.0002
		G	**−0.0064**	**−0.0022**	**0.0032**
		B	0.0003	−0.0020	0.0004
Diagonally	Original	R	0.9685	0.9565	0.8624
		G	**0.9622**	**0.9719**	**0.7441**
		B	0.9533	0.9527	0.8420
	Scrambled	R	−0.0020	0.0071	−0.0013
		G	**0.0166**	**0.0007**	**0.0016**
		B	0.0049	0.0035	−0.0007

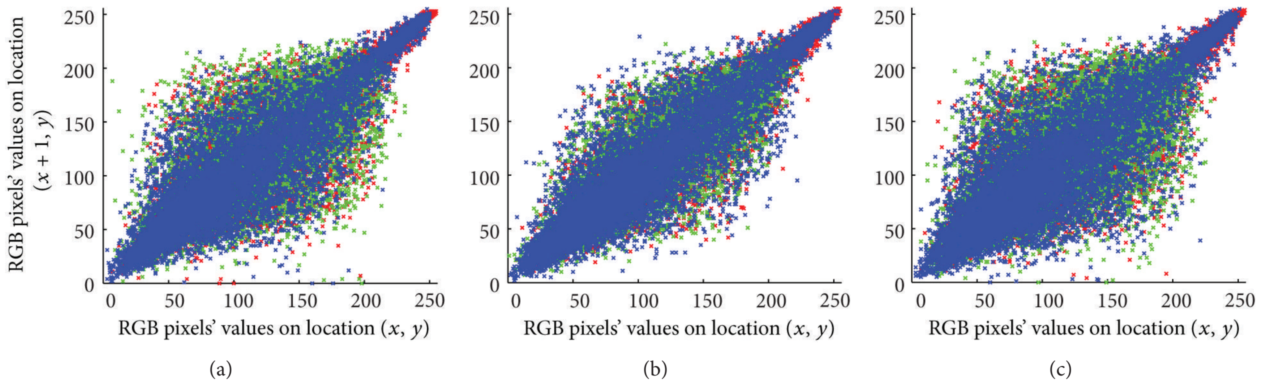

Figure 7: Correlation distributions of adjacent pixels in Baboon plain image. (a) Horizontally adjacent pixels, (b) vertically adjacent pixels, and (c) diagonally adjacent pixels.

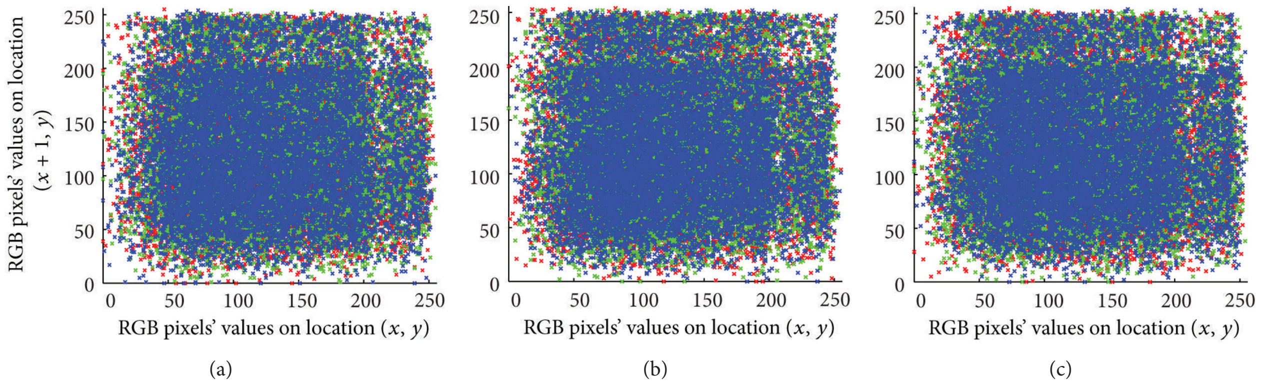

Figure 8: Correlation distribution of adjacent pixels in Baboon scrambled image. (a) Horizontally adjacent pixels, (b) vertically adjacent pixels, and (c) diagonally adjacent pixels.

where $D(i, j, C)$ represents the matrix associated with the differences between I_P's and I_S's color channels, with its structure being computed according to the following relationship:

$$D(i, j, C) = \begin{cases} 0, & \text{if } I_P(i, j, C) = I_S(i, j, C), \\ 1, & \text{if } I_P(i, j, C) \neq I_S(i, j, C). \end{cases} \quad (6)$$

Defined for each of image's color channels [49–51], UACI indicator can be numerically evaluated using (7) and considering two random images (i.e., the plain image is completely different in comparison with its scrambled version) its maximum expected value [17] is found to be $\varepsilon[\mathrm{UACI}_C] = 33.4635\%$. Consider

$$\mathrm{UACI}_C(I_P, I_S, C) = \left(\frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} \frac{|I_P(i, j, C) - I_S(i, j, C)|}{255} \right) \times 100\%. \quad (7)$$

Analyzing NPCR, UACI, and MSE indicators' values, for each of the test images, as they are summarized in Table 2, one can say that the proposed scrambling scheme approaches the performance of an ideal digital image scrambler. However, the attention is drawn to UACI indicator's values, not very close to the maximum expected value. Reasoning for this is the fact that the proposed digital image scrambler keeps the same set of possible values of pixels in the scrambled image as in the plain image. Obviously, this limits the maximum difference values that could be achieved between plain image's pixels and scrambled image's corresponding ones [17]. For example, in case of the Peppers plain image, maximum differences that could be achieved on each color channel are upper bounded by pixels maximum values (i.e., 231 for R channel, 239 for G channel, and 229 for B channel), instead of the theoretical one of 255.

3.4. Key Space and Key Sensitivity Analysis. A good digital image scrambler should have a sufficiently large key space to resist brute-force attacks, mostly if its usage as part of images' encryption schemes is desired (i.e., as part of the two fundamental techniques used to encrypt a block of pixels, i.e., permutation stage, which aims to ensure the coveted confusion properties).

As previously shown, the proposed digital image scrambler operates based on values of the bitstreams generated by a multithresholded digital chaotic map. Thus, taking into consideration the two seeding points, respectively, the two control parameters of the digital chaotic generator (namely,

Table 2: Difference measures between original and scrambled images.

Image	Color channel	Measures		
		NPCR (%)	UACI (%)	MSE
Lena	R	99.6059	32.9915	$3.7081 \cdot 10^3$
	G	**99.6013**	**30.7161**	$\mathbf{3.0761 \cdot 10^3}$
	B	99.5735	30.2703	$3.0090 \cdot 10^3$
Peppers	R	99.5174	28.5412	$2.6343 \cdot 10^3$
	G	**99.3126**	**31.9327**	$\mathbf{3.3309 \cdot 10^3}$
	B	99.2966	28.9468	$2.7533 \cdot 10^3$
Baboon	R	99.5007	25.2236	$2.1084 \cdot 10^3$
	G	**99.4621**	**24.5648**	$\mathbf{1.8140 \cdot 10^3}$
	B	99.5296	26.8748	$2.3806 \cdot 10^3$

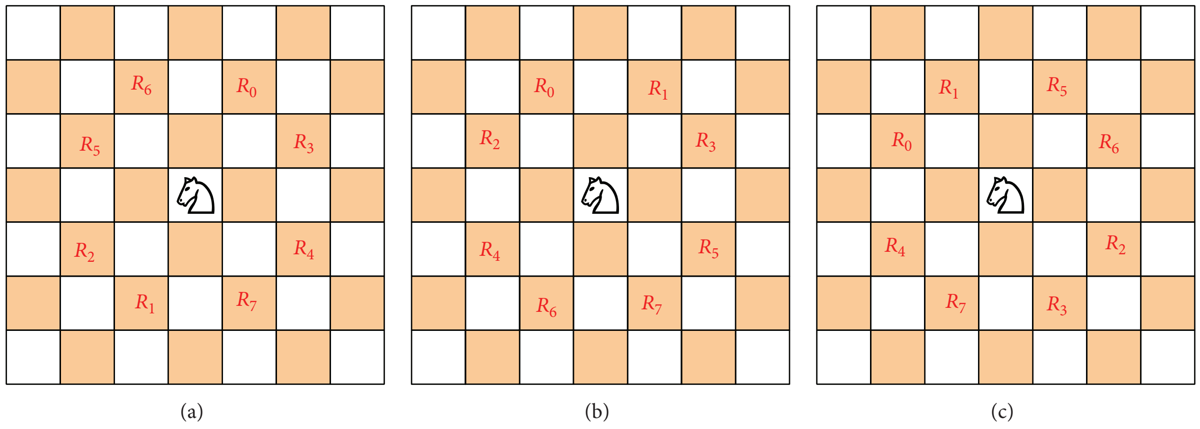

Figure 9: Three more cases, out of 8!, of possible knight's moving rules annotations.

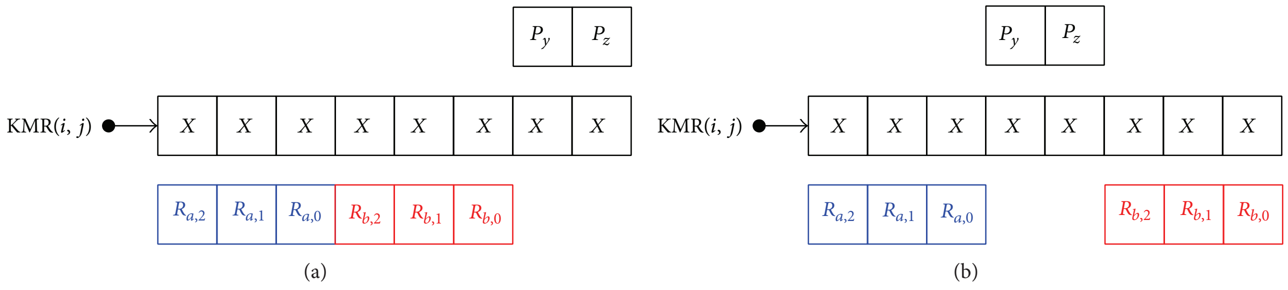

Figure 10: Two more cases, out of 8!, of possible feature extraction vector's elements configuration.

x_0^1, x_0^2, r_1, r_2), with a precision of up to 10^{-15}, and the key space of a possible encryption scheme (which uses proposed digital image scrambler scheme within permutation stage) is significantly improved by the key space of the proposed scrambler, namely, by a factor of 10^{60} (hereafter referred to as improvement factor and abbreviated as IF).

On its turn, this key space improvement factor can be raised if some artifacts are considered, such as (but not limited to) the following:

(i) if knight's moving rules are to be permuted on user choice (e.g., any of the three more cases, out of 8!, listed in Figure 9 may be used) then IF increases with $10^{4.6055}$;

(ii) if features extraction vector's elements are to be permuted on user choice (e.g., any of the two more cases, out of 4! (i.e., if bits defining each rule are taken grouped) or out of 8! (i.e., if bits defining each rule are taken random), listed in Figure 10) and may be used then IF increases with another $10^{1.3802}$ or $10^{4.6055}$;

(iii) if color channels interchanging rules are to be permuted on user choice, considering that there are $(C_{3!}^4)^3 = 3375$ possible ways to do it, then IF increases with $10^{3.5283}$.

Thereby, proposed scrambler's key space can achieve a value of $10^{69.5140}$ (for the first case mentioned in (ii)) or

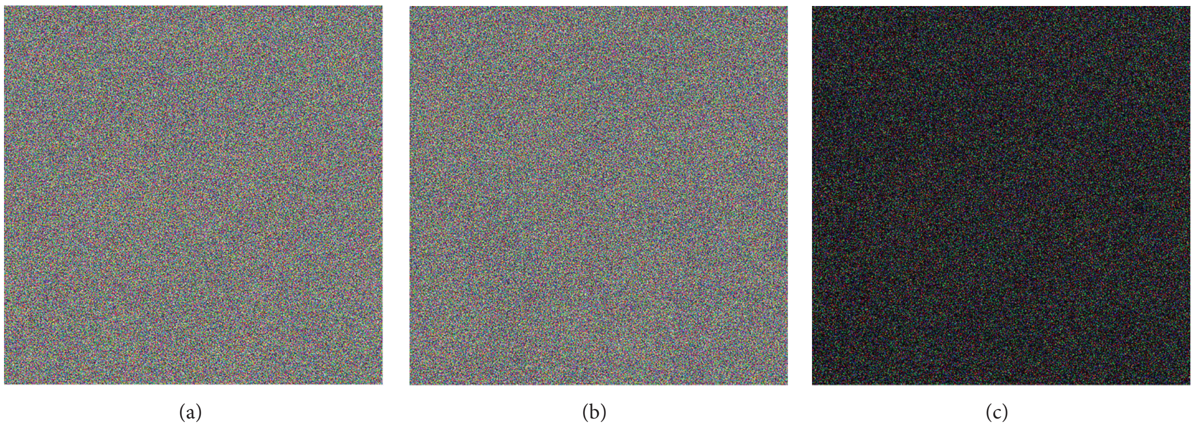
FIGURE 11: Key sensitivity of the proposed digital image scrambler. (a) Scrambled image using K_S, (b) scrambled image using K_D, and (c) image representing differences between (a) and (b).

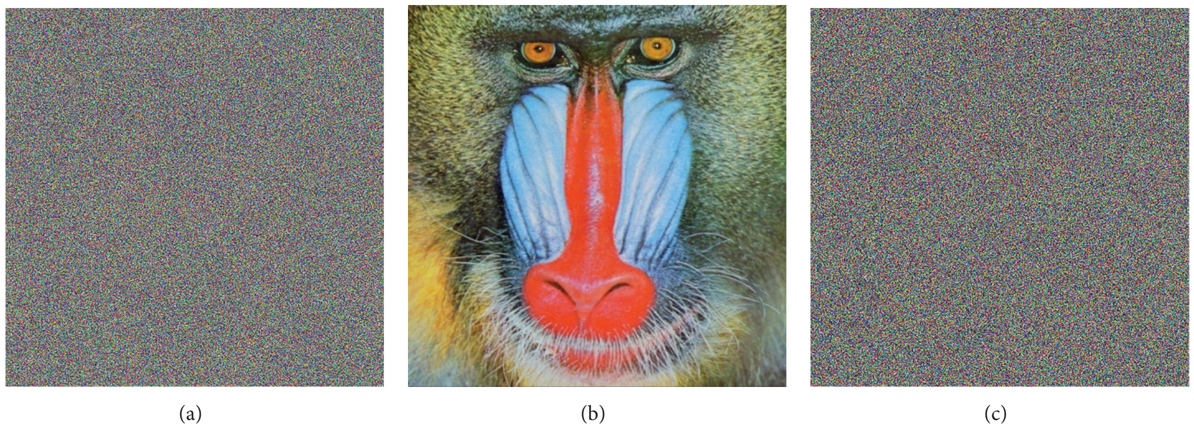
FIGURE 12: Key sensitivity of the proposed digital image scrambler. (a) Scrambled image using K_S, (b) descrambled image using K_S, and (c) descrambled image using K_D.

of $10^{72.7393}$ (for the second case mentioned in (ii)), which is sufficiently large to recommend usage of the proposed scrambler within any image encryption scheme.

In addition, in order to approach the performance of an ideal image encryption algorithm, the proposed digital image scrambler should have high sensitivity to scrambling key (i.e., a slight key change should lead to significant changes in the scrambled images during scrambling procedure, respectively, significant changes in the descrambled images during descrambling procedure).

Figure 11 shows the key sensitivity of the proposed digital image scrambler, when operating over Baboon plain image with two different scrambling keys (differentiated from each other by the LSB of initial seeding points, i.e., by $\pm 1 \cdot 10^{-15}$ variations of x_0^1 and x_0^2). As can be seen, one bit change in the scrambling key leads to two different scrambled images, whose difference is also random-like. When image is descrambled using an incorrect key (e.g., with same variation conditions of x_0^1 and x_0^2) the result is also a random-like image, as shown in Figure 12.

The two keys used for key sensitivity analysis of the proposed digital image scrambler are

$$\begin{aligned} K_S &= \{x_0^1 = 0.687754925117, r_1 = 5.93872502542\ , \\ &\quad x_0^2 = -0.013462335467, r_1 = 1.237490188615\}, \\ K_D &= \{x_0^1 = 0.687754925118,\ r_1 = 5.93872502542, \\ &\quad x_0^2 = -0.013462335466, r_1 = 1.237490188615\}. \end{aligned} \tag{8}$$

3.5. Performances' Comparison with Other Digital Image Scrambling Schemes. Performances' comparison was done taking into consideration some of the most recent works [17, 19, 27, 52]. Table 3 summarizes NPCR's, UACI's, APCCs', and speeds' measure mean values, computed over the 512 × 512 pixels, 24-bit, and color bitmaps, for all subjected scrambling algorithms.

Whilst the proposed digital image scrambler presents similar or better results, in comparison with other proposed

Table 3: Mean values of NPCR, UACI, APCCs, and speed measures.

Measure	Digital image scrambling algorithm				
	Dăscălescu and Boriga [17]	Dalhoum et al. [19]	Wu et al. [27]	Ye [52]	Ours
NPCR	99.431	90.126	94.163	93.676	99.489
UACI	25.032	NaN	NaN	NaN	29.006
APCC					
Horizontal	0.0398	0.0767	0.0020	0.1059	0.0049
Vertical	0.0448	0.1007	0.0022	0.1094	0.0018
Speed (KB/s)	$12.5 \cdot 10^3$	NaN	193.502	NaN	117.028

ones, when it comes to general qualitative measures (i.e., NPCR, UACI, and APCCs) it seems to lack speed (i.e., from processing time point of view), a fact which can be upheld to MATLAB's very slow for-loop execution. But using parallel computations or (and) other programming languages the processing speed can be largely enhanced.

All the above results (i.e., numerical results presented in the entire Section 3) were generated with the proposed scrambling scripts for (de)shuffling written on MATLAB 7.3.0 and run on an Intel Pentium Dual CPU T3200 at 2.00 GHz personal computer.

4. Conclusions

In the present paper, the study of a newly designed digital image scrambler (as part of the two fundamental techniques used to encrypt a block of pixels, i.e., the permutation stage) that uses knight's moving rules (i.e., from the game of chess), in conjunction with a chaos-based PRBG, in order to transpose original image's pixels between RGB channels, was presented.

Theoretical and practical arguments, rounded by very good numerical results on scrambler's performance analysis (i.e., under various investigation methods, including visual inspection, adjacent pixels' correlation coefficients' computation, key's space, sensitivity assessment, etc.), confirm viability of the proposed scrambling method (i.e., it ensures the coveted confusion factor) recommending its usage within cryptographic applications, thus contributing to the crystallization of reminded new paradigm (i.e., in the field of images' encryption algorithms designing branch, namely, designing of new digital image scramblers based on rule sets of the most popular games).

As future work, actual implementation on FPGA (focused on the optimization of proposed algorithm, for parallel computing) is concerned.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] A. Musheer and F. Omar, "A multi-level blocks scrambling based chaotic image cipher," in *Proceedings of the 3rd International Conference on Contemporary Computing*, pp. 171–182, Noida, India, August 2010.

[2] S. Rakesh, A. A. Kaller, B. C. Shadakshari et al., "Image encryption using block based uniform scrambling and chaotic logistic mapping," *International Journal on Cryptography and Information Security*, vol. 2, no. 1, pp. 49–57, 2012.

[3] Q. Zhang, X. Xue, and X. Wei, "A novel image encryption algorithm based on DNA subsequence operation," *The Scientific World Journal*, vol. 2012, Article ID 286741, 10 pages, 2012.

[4] C. K. Huang and H. H. Nien, "Multi chaotic systems based pixel shuffle for image encryption," *Optics Communications*, vol. 282, no. 11, pp. 2123–2127, 2009.

[5] S. Etemadi Borujeni and M. Eshghi, "Chaotic image encryption design using tompkins-paige algorithm," *Mathematical Problems in Engineering*, vol. 2009, Article ID 762652, 22 pages, 2009.

[6] H.-F. Huang, "Perceptual image watermarking algorithm based on magic squares scrambling in DWT," in *Proceedings of the 5th International Joint Conference on International Conference on Networked Computing, International Conference on Advanced Information Management and Service, and International Conference on Digital Content, Multimedia Technology and Its Applications*, pp. 1819–1822, Seoul, Republic of Korea, August 2009.

[7] R. Ye, "A novel image scrambling and watermarking scheme based on orbits of Arnold Transform," in *Proceedings of the Pacific-Asia Conference on Circuits, Communications and Systems*, pp. 485–488, Chengdu, China, May 2009.

[8] C. R. Wei, J. J. Lid, and G. Y. Liang, "A DCT-SVD domain watermarking algorithm for digital image, based on Moore-model cellular automata scrambling," in *Proceedings of the IEEE International Conference on Intelligent Computing and Integrated Systems (ICISS '10)*, pp. 104–108, Guilin, China, October 2010.

[9] L. Fang and W. YuKai, "Restoring of the watermarking image in Arnold scrambling," in *Proceedings of the 2nd International Conference on Signal Processing Systems*, vol. 1, pp. 771–774, Dalian, China, July 2010.

[10] Y. Li and X. Hao, "A blind watermarking algorithm based on image scrambling and error correct coding preprocessing," in *Proceedings of the International Conference on Electrical and Control Engineering*, pp. 4231–4233, Yichang, China, September 2011.

[11] G. Liu, H. Liu, and A. Kadir, "Wavelet-based color pathological image watermark through dynamically adjusting the embedding intensity," *Computational and Mathematical Methods in Medicine*, vol. 2012, Article ID 406349, 10 pages, 2012.

[12] V. Seenivasagam and R. Velumani, "A QR code based zero-watermarking scheme for authentication of medical images in

teleradiology cloud," *Computational and Mathematical Methods in Medicine*, vol. 2013, Article ID 516465, 16 pages, 2013.

[13] S. Li, Y. Zhao, and B. Qu, "Image scrambling based on chaotic sequences and Vigenere cipher," *Multimedia Tools and Applications*, vol. 66, no. 3, pp. 573–588, 2013.

[14] G. Ye, "Image scrambling encryption algorithm of pixel bit based on chaos map," *Pattern Recognition Letters*, vol. 31, no. 5, pp. 347–354, 2010.

[15] C. Guang-Hui, H. Kai, and T. Wei, "Image scrambling based on Logistic uniform distribution," *Acta Physica Sinica*, vol. 60, no. 11, Article ID 110508, 7 pages, 2011.

[16] G. Zhang and Q. Liu, "A novel image encryption method based on total shuffling scheme," *Optics Communications*, vol. 284, no. 12, pp. 2775–2780, 2011.

[17] A. C. Dăscălescu and R. E. Boriga, "A novel fast chaos-based algorithm for generating random permutations with high shift factor suitable for image scrambling," *Nonlinear Dynamics*, vol. 74, no. 1-2, pp. 307–318, 2013.

[18] R. Ye and H. Li, "A novel image scrambling and watermarking scheme based on cellular automata," in *Proceedings of the International Symposium on Electronic Commerce and Security*, pp. 938–941, Guangzhou, China, August 2008.

[19] A. L. A. Dalhoum, B. A. Mahafzah, A. A. Awwad, I. Aldamari, A. Ortega, and M. Alfonseca, "Digital image scrambling using 2D cellular automata," *IEEE Transactions on Multimedia*, vol. 19, no. 4, pp. 28–36, 2012.

[20] R. Mathews, A. Goel, P. Saxena, and V. P. Mishra, "Image encryption based on explosive inter-pixel displacement of the RGB attributes of a pixel," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, pp. 41–44, San Francisco, Calif, USA, October 2011.

[21] P. Jagadeesh, P. Nagabhushan, and R. P. Kumar, "A novel color image scrambling technique based on transposition of image-blocks between RGB color components," *International Journal of Research in Engineering & Advanced Technology*, vol. 1, no. 2, pp. 1–8, 2013.

[22] P. Jagadeesh, P. Nagabhushan, and R. P. Kumar, "A novel image scrambling technique based on information entropy and quad tree decomposition," *International Journal of Computer Science Issues*, vol. 10, no. 2, pp. 285–294, 2013.

[23] G. Ye, X. Huang, and C. Zhu, "Image encryption algorithm of double scrambling based on ASCII code of matrix element," in *Proceedings of the International Conference on Computational Intelligence and Security (CIS '07)*, pp. 843–847, December 2007.

[24] K. T. Lin, "Hybrid encoding method by assembling the magic-matrix scrambling method and the binary encoding method in image hiding," *Optics Communications*, vol. 284, no. 7, pp. 1778–1784, 2011.

[25] D. van de Ville, W. Philips, R. van de Walle, and I. Lemahieu, "Image scrambling without bandwidth expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 892–897, 2004.

[26] Y. Zou, X. Tian, S. Xia, and Y. Song, "A novel image scrambling algorithm based on Sudoku puzzle," in *Proceedings of the 4th International Congress on Image and Signal Processing (CISP '11)*, vol. 2, pp. 737–740, Shanghai, China, October 2011.

[27] Y. Wu, S. S. Agaian, and J. P. Noonan, "Sudoku associated two dimensional bijections for image scrambling," http://arxiv.org/abs/1207.5856.

[28] Y. Y. Wang, D. Wan, and H. Y. Sheng, "An encryption algorithm by scrambling image with sudoku grids matrix," *Advanced Materials Research*, vol. 433, pp. 4645–4650, 2012.

[29] J. Delei, B. Sen, and D. Wenming, "An image encryption algorithm based on Knight's tour and slip encryption filter," in *Proceedings of the International Conference on Science and Software Engineering*, vol. 1, pp. 251–255, Wuhan, China, December 2008.

[30] Z. K. Lei, Q. Y. Sun, and X. X. Ning, "Image scrambling algorithms based on knight-tour transform and its applications," *Journal of Chinese Computer Systems*, vol. 5, article 044, 2010.

[31] K. Loukhaoukha, J.-Y. Chouinard, and A. Berdai, "A secure image encryption algorithm based on Rubik's cube principle," *Journal of Electrical & Computer Engineering*, vol. 2012, Article ID 173931, 13 pages, 2012.

[32] A.-V. Diaconu and K. Loukhaoukha, "An improved secure image encryption algorithm based on Rubik's cube principle and digital chaotic cipher," *Mathematical Problems in Engineering*, vol. 2013, Article ID 848392, 10 pages, 2013.

[33] X. Wang and J. Zhang, "An image scrambling encryption using chaos-controlled Poker shuffle operation," in *Proceedings of the IEEE International Symposium on Biometrics and Security Technologies (ISBAST '08)*, pp. 1–6, Islamabad, Pakistan, April 2008.

[34] A.-C. Dăscălescu and R. Boriga, "A new method to improve cryptographic properties of chaotic discrete dynamical systems," in *Proceedings of International Workshop on Information Security, Theory and Practice-in Conjunction with the 7th International Conference for Internet Technology and Secured Transactions*, pp. 60–65, London, UK, December 2012.

[35] A. Luca, A. Ilyas, and A. Vlad, "Generating random binary sequences using tent map," in *Proceedings of the 10th International Symposium on Signals, Circuits and Systems (ISSCS '11)*, pp. 1–4, Iai, Romania, June 2011.

[36] N. K. Pareek, V. Patidar, and K. K. Sud, "A random bit generator using chaotic maps," *International Journal of Network Security*, vol. 10, no. 1, pp. 32–38, 2010.

[37] Q. Zhou, X. Liao, K.-W. Wong, Y. Hu, and D. Xiao, "True random number generator based on mouse movement and chaotic hash function," *Information Sciences*, vol. 179, no. 19, pp. 3442–3450, 2009.

[38] A.-C. Dăscălescu, R. E. Boriga, and C. Răcuciu, "A new pseudorandom bit generator using compounded chaotic tent maps," in *Proceedings of the 9th IEEE International Conference on Communications*, pp. 339–342, Bucharest, Romania, June 2012.

[39] X.-Y. Wang and Y.-X. Xie, "A design of pseudo-random bit generator based on single chaotic system," *International Journal of Modern Physics C*, vol. 23, no. 3, Article ID 1250024, 2012.

[40] A. Kanso and N. Smaoui, "Logistic chaotic maps for binary numbers generations," *Chaos, Solitons and Fractals*, vol. 40, no. 5, pp. 2557–2568, 2009.

[41] A.-C. Dăscălescu, R. E. Boriga, and A.-V. Diaconu, "Study of a new chaotic dynamical system and its usage in a novel pseudorandom bit generator," *Mathematical Problems in Engineering*, vol. 2013, Article ID 769108, 10 pages, 2013.

[42] A.-V. Diaconu, "Multiple bitstreams generation using chaotic sequences," *The Annals of the University Dunarea De Jos of Galati Fascicle III*, vol. 35, no. 1, pp. 37–42, 2012.

[43] B. Furht and D. Kirovski, *Multimedia Security Handbook*, CRC Press, 2004.

[44] A. J. Menezes, P. C. Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[45] http://sipi.usc.edu/database/.

[46] A. N. Pisarchik and M. Zanin, "Image encryption using chaotic logistic map," *Physica D*, vol. 237, no. 20, pp. 2638–2648, 2008.

[47] S. E. Borujeni and M. Eshghi, "Design and simulation of encryption system based on PRNG and Tompkins-Paige permutation algorithm using VHDL," in *Proceedings of International Conference on Robotics, Vision, Information and Signal Processing*, pp. 63–67, Penang, Malaysia, 2007.

[48] D. Arroyo, R. Rhouma, G. Alvarez, S. Li, and V. Fernandez, "On the security of a new image encryption scheme based on chaotic map lattices," *Chaos*, vol. 18, no. 3, Article ID 033112, 2008.

[49] G. Chen, Y. Mao, and C. K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," *Chaos, Solitons and Fractals*, vol. 21, no. 3, pp. 749–761, 2004.

[50] Y. Mao, G. Chen, and S. Lian, "A novel fast image encryption scheme based on 3D chaotic baker maps," *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, vol. 14, no. 10, pp. 3613–3624, 2004.

[51] Y. Wu, J. P. Noonan, and S. Agaian, "NPCR and UACI randomness tests for image encryption," *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications*, pp. 31–38, 2011.

[52] G. Ye, "Image scrambling encryption algorithm of pixel bit based on chaos map," *Pattern Recognition Letters*, vol. 31, no. 5, pp. 347–354, 2010.