

A Novel Image Encryption Algorithm Based on the Two-Dimensional Logistic Map and the Latin Square Image Cipher

M. Machkour  · A. Saaidi · M. L. Benmaati

Received: 19 June 2015 / Revised: 17 September 2015 / Accepted: 23 September 2015 / Published online: 5 October 2015
© 3D Research Center, Kwangwoon University and Springer-Verlag Berlin Heidelberg 2015

Abstract In this paper, we introduce a new hybrid system consisting of a permutation-substitution network based on two different encryption techniques: chaotic systems and the Latin square. This homogeneity between the two systems allows us to provide the good properties of confusion and diffusion, robustness

to the integration of noise in decryption. The security analysis shows that the system is secure enough to resist brute-force attack, differential attack, chosen-plaintext attack, known-plaintext attack and statistical attack. Therefore, this robustness is proven and justified.

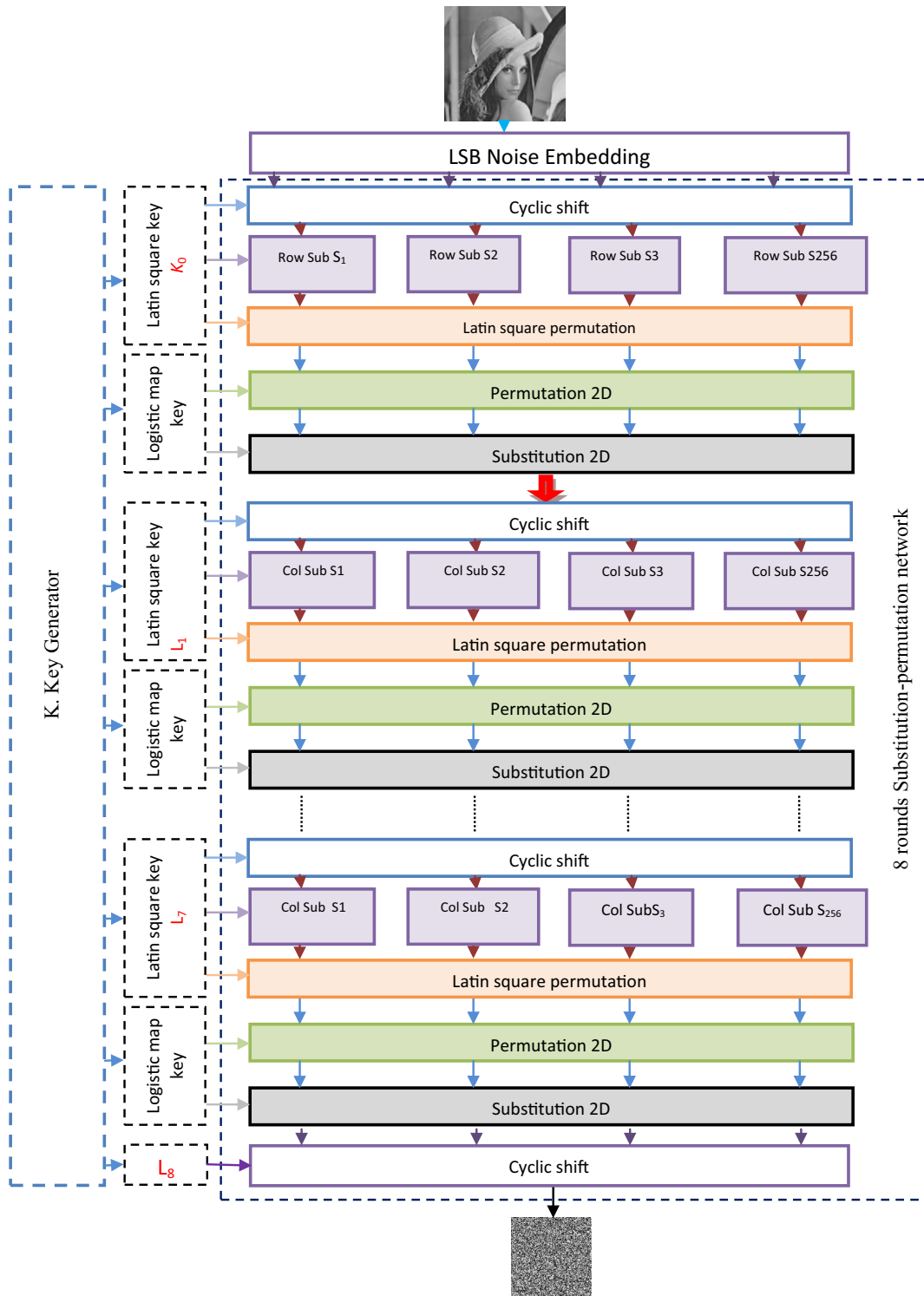
M. Machkour (✉) · M. L. Benmaati
LIROSA, Department of Mathematics and Computer Science, Faculty of Sciences, Abdelmalek Essaâdi University, B.P 2121, M'Hannech II, 93030 Tetouan, Morocco
e-mail: machkour_mohammed@yahoo.fr

M. L. Benmaati
e-mail: mbenmaati@yahoo.fr

A. Saaidi
LSI, Department of Mathematics, Physics and Computer Sciences, Polydisciplinary Faculty of Taza, Sidi Mohamed Ben Abdellah University, B.P 1223, Taza, Morocco
e-mail: abderrahim.saaidi@usmba.ac.ma

A. Saaidi
LIIAN, Department of Mathematics and Computer Science, Faculty of Sciences, Dhar El Mehraz, Sidi Mohamed Ben Abdellah University, B.P 1796, Atlas, Fez, Morocco

Graphical Abstract



Keywords Image encryption · Chaos · Logistic map · Latin square image cipher

Abbreviation List

DC	Cyclicshift
Dcr	Decryption
Ecr	Encryption
GCL	Latin Square Generator
GS	Sequence Generator
PCCL	Column Latin Square Permutation
PCL	Latin Square Permutation
PLCL	Row Latin Square Permutation
SCCL	Column Latin Square Substitution
SLCL	Row Latin Square Substitution

1 Introduction

The development of the IT field especially laptops, media supports and the Internet facilitated the diffusion and exchange of information. Many applications such as military databases, confidential video conferencing, medical imaging system, cable TVs, personal photo album online, etc., require reliable, fast and robust systems for secure transmission. The image encryption is different from text because of some intrinsic features of images, such as data redundancy and the strong correlation between adjacent pixels. Therefore, the traditional systems such as DES [1], AES [2], and Blowfish [3] are not suitable for images and videos encryption. To overcome this problem, many fast systems specially designed for digital images have been proposed in recent years, and their main purposes is to reduce the redundancy in image, to strengthen security and particularly to minimize the encryption time.

Among those existing approaches, Chaos-based scheme, with the desirable properties of pseudo randomness, ergodicity, high sensitivity to initial conditions and parameters, is recognized as great potential for multimedia encryption because such features are considered analogous to those of an ideal cryptosystem. The sequences generated by the chaotic maps are often pseudo-random sequences, and their structures are very complex and difficult to analyze [4–8]. Yue Wu et al. [9] proposed an image encryption

using the two-dimensional logistic chaotic map, which has well-known weaknesses such as the key length which is insufficient, low security, and encryption time which is quite high. It completely loses its chaotic nature and becomes periodic when it is discretized [10] and the initial values can be estimated by a number of existing tools and methods [11, 12]. To overcome these drawbacks, we suggest a hybrid algorithm which consists of a substitution-permutation network of eight rounds based on the two-dimensional logistic map to take advantage of chaotic systems and Latin square [13], which would ensure a good diffusion, strengthen more safety and improve the encryption time considerably. With this homogeneity between the two systems, we can obtain a reliable, robust, and fast encryption algorithm.

The proposed algorithm is designed for medical imagery, because currently, most of Hospital Data Management Systems (HDMSs) and medical equipment's are working in a computer network environment. Medical images are produced and stored in a digital form; moreover, they are exchanged through a computer network. These images are the most important entity in the healthcare diagnostic procedures because they are used to view features of patients such as anatomical cross-sections of internal organs and tissues, in addition they are used for physicians to evaluate the patient's diagnosis and monitor the effects of the treatment. Therefore, protecting medical images from an unauthorized use is an essential requirement.

This article is organized as follows: In the second part, we give an overview of previous works, the third part presents a detailed description of the proposed algorithm, the fourth part contains the software implementation and experimental results (sensitivity towards the secret key, histogram analysis, correlation between adjacent pixels...), and ends with a general conclusion.

2 Previous Works

Due to the high sensitivity of chaotic systems to parameters and initial conditions as well as the availability of many circuit realizations [14, 15], chaos based algorithms are developed and studied as the core of encryption algorithms. Recently, many Chaos-based schemes are proposed: Telem ANK et al.

[16] proposed A robust gray image encryption scheme using chaotic logistic map and artificial neural network (ANN), In the proposed method, an external secret key is used to derive the initial conditions for the logistic chaotic maps which are employed to generate weights and biases matrices of the multilayer perceptron (MLP). During the learning process with the back propagation algorithm, ANN determines the weight matrix of the connections. The plain image is divided into four sub images which are used for the first diffusion stage. The sub images obtained previously are divided into the square sub image blocks. In the next stage, different initial conditions are employed to generate a key stream which will be used for permutation and diffusion of the sub image blocks. Hegui Zhu et al. [17] proposed A novel image encryption–compression scheme using hyper-chaos and Chinese remainder theorem, Specifically, firstly they use 2D hyper-chaos discrete non linear dynamic system to shuffle the plain image, and then they apply Chinese remainder theorem (well known in number theory) to diffuse and compress the shuffled image. Hraoui et al. [18] proposed a mix of a compression technique called SPIHT (Set Partitioning in Hierarchical Trees) and chaotic encryption in order to elaborate a crypto-compressing scheme based on encrypting quantization by a modified logistic map. Kanso et al. [19] introduced a novel image encryption algorithm based on a 3D chaotic map, the proposed algorithm based on three phases: In phase I, the image pixels are shuffled according to a search rule based on the 3D chaotic map. In phases II and III, 3D chaotic maps are used to scramble shuffled pixels through mixing and masking rules.

On the other hand, non-chaotic methods have proved their existence and importance in implementing the confusion and diffusion stages. Such methods usually increase the algorithm complexity to protect against cryptanalysis: Yue Wu et al. [13] introduced a symmetric-key Latin square image cipher (LSIC) for grayscale and color image encryption.

Pareek et al. [20] divided the image into non-overlapping blocks and each block was scrambled using a zigzag-like algorithm. Furthermore, Al-Husainy [21] divided the image into a set of k -bit vectors, each of these vectors was substituted by XORing it with the previous vector and then permuted by circularly right rotating its bits. Pareek et al. [22] divided the image into non-overlapping blocks and for each encryption round

the size of the block changed according to the round key. Within the same block, permutation was performed using a zigzag like algorithm.

The combination of both chaotic and non-chaotic algorithms showed some advantages in many cryptosystems. For example, Li et al. [23] used the 3D Arnold map and a Laplace-like equation to perform permutations and substitutions, respectively. Wang and Yang [24] used the water drop motion and a dynamic lookup table with the help of the logistic map to perform the diffusion and confusion processes. Furthermore, Fouda et al. [25] used a piecewise linear chaotic map to generate pseudo random numbers and these numbers were used in generating the coefficients of the Linear Diophantine Equation (LDE). By sorting the solutions of LDE, large permutations were created and used in scrambling the image pixels. Whereas Zhang et al. [26] used compressive sensing along with Arnold's map in order to encrypt color images into grey images, Zhang et al. [27] used a coupled logistic map, self adaptive permutation, substitution boxes and combined global diffusion to perform the encryption. Finally, AbdEl Haleem et al. [28] used a chess-based algorithm to perform the permutation process and the Lorenz system to perform the substitution process. In summary, permutations and substitutions can be performed using chaotic systems, non-chaotic algorithms or a combination of both.

3 Proposed Method

The proposed algorithm uses two different encryption systems: the two-dimensional logistic map and the Latin square. This complementarity fatherly enhances the security and allows us to benefit from the advantages of each system.

3.1 Two-dimensional Logistic Map

The two-dimensional logistic map is known for its complex behavior of the evolution of basins and attractors [29]. It is more complex than the one-dimensional chaotic behavior.

The logistic map is defined by the system of Eqs. (1) below:

$$\begin{cases} X_{i+1} = r(3Y_i + 1)X_i(1 - X_i) \\ Y_{i+1} = r(3X_{i+1} + 1)Y_i(1 - Y_i) \end{cases} \quad (1)$$

where ‘ r ’ is the parameter of the system (with $r \in [1.1, 1.19]$ it is the interval where the system is chaotic) and (X_i, Y_i) is the point at the i th iteration. This card also depends on the initial conditions (x_0, y_0) to determine its trajectory.

3.2 Latin Square

A Latin square L of N order is a table of $N \times N$ elements filled with N distinct symbols, with each symbol appears exactly once in each row and each column. The name of the Latin square was introduced by the mathematician Leonhard Euler, who used Latin characters as symbols.

Mathematically, we can define a Latin Square of N order via the function $F_L(l, c, i)$ according to Eq. (2):

$$F_L(l, c, i) = \begin{cases} 1 & \text{si } L(l, c, i) = S_i \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$

With l denotes the index of a line of an element of L , $l \in \mathbb{N} = 0, 1, \dots, N - 1$, c denotes the index of a column of an element of L $c \in \mathbb{N}$, i denotes the index of a symbol element in L and S_i is i th symbol in the whole $\mathcal{S} = \{S_0, S_1, \dots, S_{N-1}\}$. This implies that each symbol appears exactly once in each row and each column of L . Figure 1 shows examples of Latin squares of different orders with different symbol sets.

3.3 Detailed Algorithm

The proposed algorithm is of an SPN structure (substitution-permutation network) with eight rounds and an encryption key of 256 bits as shown in Fig. 2. It uses the two-dimensional logistic map and the Latin

square. It consists of the integration of the probabilistic sound encryption LSB (least significant bit-plane) [30], and the SPN stage with five primitives of encryption: cyclic shift, Latin square substitution, Latin square permutation, permutation 2D and substitution 2D. The different steps will be defined in the following sections.

3.3.1 K. Key Generator

The encryption K . key is 256-bits long, and it is used in two different ways:

- Directly in the 2D permutation and 2D substitution, this is called: logistic map key.
- Transformed into Latin square when it is added to the cyclic shift, permutation and Latin square substitution. It will be named Latin square key.

3.3.1.1 Logistic Map Key We cut the encrypting K . key into five parameters namely x_0, y_0, r, T and $A_0 \dots A_8$ as shown in Fig. 3, with (x_0, y_0) and r as initial values of the two-dimensional logistic map expressed in Eq. 1. A and T are parameters of the generator of a linear congruence (see the detailed function in [31]).

3.3.1.2 Latin Square Key We transform the K . encryption key into eight Latin squares key which are 256 bits long, as follows:

- we divide the K . encryption key by using the SubKeyDiv function into eight subkeys of 32 bits: $K = \{K_0, K_1, \dots, K_7\}$
- We generate pairs of pseudo-random sequences: $(Q_1^0, Q_2^0), (Q_1^1, Q_2^1), \dots, (Q_1^8, Q_2^8)$, each pair is

A	B	C
B	C	A
C	A	B

a 3*3

ϙ	Δ	π	⊗
⊗	ϙ	Δ	π
π	⊗	ϙ	Δ
Δ	π	⊗	ϙ

b 4*4

3	4	5	2	1	6	7	9	8
8	6	1	9	3	7	4	2	5
9	7	2	5	8	4	1	3	6
2	1	9	6	4	3	5	8	7
5	8	6	7	2	9	3	4	1
4	3	7	1	5	8	2	6	9
6	2	4	8	7	1	9	5	3
1	9	3	4	6	5	8	7	2
7	5	8	3	9	2	6	1	4

c 9*9

Fig. 1 Examples of Latin squares

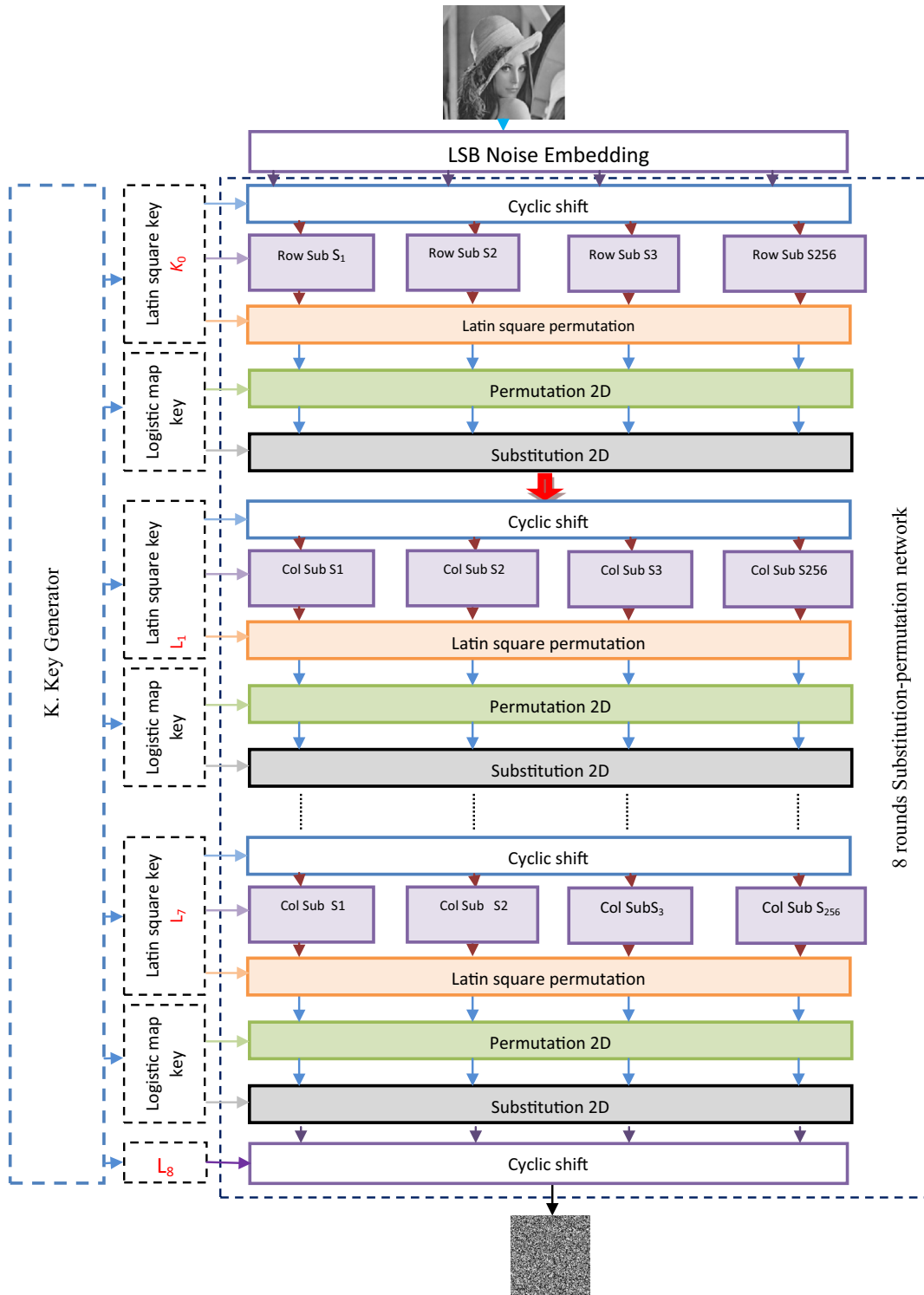


Fig. 2 Encryption images by the proposed algorithm

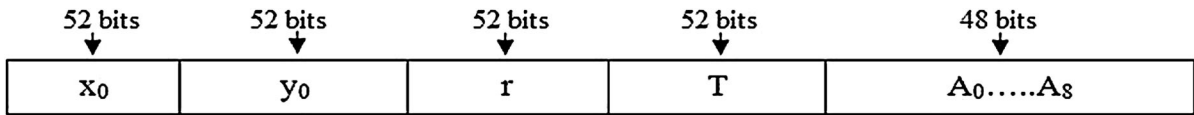


Fig. 3 Logistic map Key

made of 2×256 elements by using PRNGs function (Pseudorandom number generator). These two steps are described in Algorithm 1.

Algorithm 1. Sequence generator $(Q1, Q2) = GS (Key, M)$

```

Require: K a 256-bit key
Require: n is a nonnegative integer
Ensure:  $Q_1 = \{Q_1^0, \dots, Q_1^M\}$  et :  $Q_2 = \{Q_2^0, \dots, Q_2^M\}$  are n-element set of random sequences, each of a length 256.

 $K_0 = K$ 
For  $n = 0: M$  do
     $[k_0, k_1, \dots, k_7] = \text{SubKeyDiv}(K_n)$ 
    for  $i = 0: 8$  do
         $Q_i^0 = \text{PRNG}(K_i)$ 
        For  $j = 1: 63$ 
             $Q_i^j = \text{PRNG}(q^{i(j-1)})$ 
        end for
    end for
     $Q_1^n = [q^0(0:31), q^1(0:31), \dots, q^7(0:31)]$ 
     $Q_2^n = [q^0(32:63), q^1(32:63), \dots, q^7(32:63)]$ 
     $K_{n+1} = [q^0(63), q^1(63), \dots, q^7(63)]$ 
end for
    
```

Finally, we obtain Latin Square keys of 256 bits: L_0, L_1, \dots, L_8 of the 256 bits order by supplying these pseudo-random sequences in Algorithm 2: $n \in \{0, 1, \dots, 8\}$ we have a $L_n = \text{GCL}(Q_1^n, Q_2^n)$.

Algorithm 2. Latin square generator $L = \text{GCL}(Q1, Q2)$

```

Require: Q1 and Q2 are two length-N sequences
Ensure: L is a Latin square of order N

Qseed = SortMap(Q1)
Qshift = SortMap(Q2)
for  $r = 0$  to  $N-1$ 
     $L(r, :) = \text{RowShift}(Qseed, Qshift(r))$ 
end
    
```

$Q1$ and $Q2$ are two sequences produced by a pseudo-random number generator (PRNG), for example the generator of a linear congruence [32]. SortMap (Q) is a function that finds the cartographic index between a Q sequence and its filtered version Q^* in the ascending order, and RowShift (Q, v) makes a cyclic shift of the Q sequence with v elements to the left [13].

3.3.2 Cyclic Shift

In the conventional SPN encryption by blocks [33, 34], the step of shifting normally mixes a clear P message with a key, for example, operation XOR.

We define the cyclic shift as an encryption by transposition [35] on the finite space $\text{GF}(2^8)$ for the image data, as shown in the following equation:

$$y = [x + 1]_{2^8} \tag{3}$$

where 'x' is a byte in the plain image, '1' is the corresponding byte to the Latin key, and 'y' is the result of cyclic shift. $[.]_{2^8}$ denotes the computations over $\text{GF}(2^8)$. The shift Process above may easily be reversed by applying:

$$x = [y + 1]_{2^8} \tag{4}$$

In the image encryption, the plaintext of x byte is a pixel. It is located at the intersection of r th row and c th column hence $x = P(r, c)$. Now, $l = L(r, c)$ is a component located in the corresponding position of the Latin key square L and 'y' byte of cryptogram with $y = C(r, c)$, then the equation is obtained at the pixel level (5):

$$\begin{cases} C(r, c) = [\text{SR}(P(r, c), D) \oplus L(r, c)]_{2^8} \\ P(r, c) = \text{SR}([C(r, c) \oplus L(r, c)]_{2^8}, D) \end{cases} \tag{5}$$

D is the rotation parameter ($D = L(0,0) \bmod 3$), and SR denotes the function of spatial rotation (X, d) which retains an x image according to different values of the direction D as it is defined in the Eq. (6).

$$\begin{cases} X; & \text{if } d = 0 \\ \text{Flip } X \text{ up} \rightarrow \text{down}; & \text{if } d = 1 \\ \text{Flip } X \text{ left} \rightarrow \text{right}; & \text{if } d = 2 \end{cases} \quad (6)$$

Equation 5 is applied to all pixels to obtain the cyclic shift denoted by:

$$\text{DC} = \begin{cases} C = \text{Ecr}_{\text{DC}}(L, P, D) \\ P = \text{Dcr}_{\text{DC}}(L, C, D) \end{cases} \quad (7)$$

3.3.3 Latin Square Substitution

An S-Box in Cryptography is a basic element to perform the substitution. The existence of the bijection FRM/MRI (Forward row mapping, inverse row mapping) and FCM/ICM (Forward column mapping, Inverse column mapping) [13] in a Latin square, we are able to perform the byte substitution in an encrypting image using bijections of rows and columns in a Latin square. The substitution according to a row in a Latin square is denoted by SLCL:

$$\text{SLCL} = \begin{cases} C = \text{Ecr}_s^{\text{ligne}}(L, P) \\ P = \text{Dcr}_s^{\text{ligne}}(L, C) \end{cases} \quad (8)$$

The functions $\text{Ecr}_s^{\text{ligne}}$ and $\text{Dcr}_s^{\text{ligne}}$ are obtained from Eqs. (9) and (10) below:

$$\text{Ecr}_s^{\text{ligne}} = C(r, c) = \begin{cases} \text{FRM}(L, C(r - 1, c), P(r, c)); & \text{if } r \neq 0 \\ \text{FRM}(L, 0, P(r, c)); & \text{if } r = 0 \end{cases} \quad (9)$$

$$\text{Dcr}_s^{\text{ligne}} = P(r, c) = \begin{cases} \text{IRM}(L, C(r - 1, c), C(r, c)); & \text{if } r \neq 0 \\ \text{IRM}(L, 0, C(r, c)); & \text{if } r = 0 \end{cases} \quad (10)$$

Similarly, we use bijections of columns in a Latin square to make substitutions of bytes SCCL.

$$\text{SCCL} = \begin{cases} C = \text{Ecr}_s^{\text{colonne}}(L, P) \\ P = \text{Dcr}_s^{\text{colonne}}(L, C) \end{cases} \quad (11)$$

$$\begin{aligned} \text{Ecr}_s^{\text{colonne}} &= C(r, c) \\ &= \begin{cases} \text{FCM}(L, P(r, c), C(r, c - 1)); & \text{if } r \neq 0 \\ \text{FCM}(L, P(r, c), 0); & \text{if } r = 0 \end{cases} \end{aligned} \quad (12)$$

$$\begin{aligned} \text{Dcr}_s^{\text{colonne}} &= P(r, c) \\ &= \begin{cases} \text{ICM}(L, C(r, c), C(r, c - 1)); & \text{if } r \neq 0 \\ \text{ICM}(L, C(r, c), 0); & \text{if } r = 0 \end{cases} \end{aligned} \quad (13)$$

3.3.4 Latin Square Permutation

Unlike the byte substitution S-Box, the P-Box (permutation) performs a patch or a byte interference. Each P-Box can also be defined as a bijection [35]. If we consider both the input and output x and y in FRM and IRM as indices, an FRM then defines a mapping $\{0, 1, 2, \dots, 255\}$ to $\{0, 1, 2, \dots, 255\}$ and also an IRM defines the corresponding inverse mapping. We are therefore able to define the Latin square P-box row (PLCL) as follows:

$$\text{PLCL} = \begin{cases} C(r, c_y) = P(r, \text{FRM}(L, r, c_x)) \\ P(r, c_x) = C(r, \text{IRM}(L, r, c_y)) \end{cases} \quad (14)$$

In which c_x and c_y denotes column indices before and after the mapping. Similarly, we can also build the P-column Latin square box (PCCL) by the following equations:

$$\text{PCCL} = \begin{cases} C(r_y, c) = P(r, \text{FCM}(L, r_x, c), c) \\ P(r_x, c) = C(\text{ICM}(L, r_y, c), c) \end{cases} \quad (15)$$

In which r_x and r_y denote the row indices before and after the mapping.

To have the best performances, we build our Latin square permutation by cascading the row permutations PLCLs and the column permutations PCCL as follows:

$$\begin{cases} C(r, c) = C^*(\text{FCM}(L, r, c), c) \\ C^*(r, c) = P(r, \text{FRM}(L, r, c)) \end{cases} \quad (16)$$

In general, we write the function of the Latin square permutation as follows:

$$\text{PCL} = \begin{cases} C = \text{Ecr}_p(L, P) \\ P = \text{Dcr}_p(L, P) \end{cases} \quad (17)$$

3.3.5 2D Logistic Permutation

If we suppose that the size of the clear image P is $M \times N$, then, the total number of pixels by P is $M \times N$. If we Consider the initial value used in a round is (x_0, y_0) , then a sequence of x and y of $M \times N$ length (excluding the initial value) can be generated by the 2D logistic map using Eq. (1). We notice that X_{seq} and Y_{seq} , the sequences of x and y coordinates of the 2D logistic map sequences as the following Eq. (18) shows:

$$\begin{cases} X_{seq} = \{x_1, x_2, \dots, x_{MN}\} \\ Y_{seq} = \{y_1, y_2, \dots, y_{MN}\} \end{cases} \quad (18)$$

We rearrange the items of Xseq and Yseq whose number is $M \times N$ as matrices X and Y, then each row 'r' of X is sorted to build matrix X^{sorted} . There is a bijection between the matrix X and its sorted Version X^{sorted} which is named e_{π_x} [35] as shown in Eq. (19). Similarly, there is also a bijection e_{π_y} between the matrix Y and its sorted version according to the columns Y noted as Y^{sorted} .

$$\begin{cases} X_{r,i}^{sorted} = X_{r,e_{\pi_x}(i)} \\ Y_{i,c}^{sorted} = Y_{e_{\pi_y}(i),c} \end{cases} \quad (19)$$

Therefore, the matrix row of permutation U^x and the matrix column of permutation U^y can be obtained by the Eq. (20).

$$\begin{cases} U^x = [e_{\pi_x}^{r=1}, e_{\pi_x}^{r=2}, \dots, e_{\pi_x}^{r=M}]^T \\ U^y = [e_{\pi_y}^{c=1}, e_{\pi_y}^{c=2}, \dots, e_{\pi_y}^{c=N}] \end{cases} \quad (20)$$

Finally, the 2D logistic permutation is obtained by Algorithm 3 using row and column permutations of the Eq. (20).

Algorithm 3.2D Logistic permutation

Require: 2D plaintext image P, row permutation matrix U_x and column permutation matrix U_y

Ensure: Ciphertext image C

```

for r = 1: M
    for c = 1: N
         $Q_{r,c} = P_{U_{r,c}^x} \setminus \setminus$  (Pixel permutation along x)
    end
end
for r = 1: M
    for c = 1: N
         $C_{r,c} = Q_{r,U_{r,c}^y} \setminus \setminus$  (Pixel permutation along y)
    end
end
    
```

We write the 2D permutation logistic function as follows:

$$P_{2D} = \begin{cases} C = Ecr_{P_{2D}}(P) \\ P = Dcr_{P_{2D}}(L, P) \end{cases} \quad (21)$$

3.3.6 2D Logistic Substitution

In this phase, we change the pixels values while maintaining the reference matrix 'I' which depends on the sequences of the 2D logistic map and which will be calculated as follows:

X and Y are versions of Xseq and Yseq matrices, $Z = X + Y$, Each B block of 4×4 elements of Z will be converted with the following matrix:

$$I = f(B) = \begin{pmatrix} g_N(B_{1,1}) & g_R(B_{1,2}) & g_S(B_{1,3}) & g_D(B_{1,4}) \\ g_R(B_{2,1}) & g_S(B_{2,2}) & g_D(B_{2,3}) & g_N(B_{2,4}) \\ g_S(B_{3,1}) & g_D(B_{3,2}) & g_N(B_{3,3}) & g_R(B_{3,4}) \\ g_D(B_{4,1}) & g_N(B_{4,2}) & g_R(B_{4,3}) & g_S(B_{4,4}) \end{pmatrix} \quad (22)$$

With $g_N(d) = T(d) \bmod F$, $g_R(d) = T(\sqrt{d}) \bmod F$, $g_S(d) = T(d^2) \bmod F$ and $g_D(d) = T(2d) \bmod F$, The function T(d) truncates a decimal d from the 9th digit to 16th digit to form an integer. For example, if $b = 0.1234567890123456789$ then $T(b) = 90,123,456$. Finally the encrypted image is defined by the equation:

$$C = (P + I) \bmod F \quad (23)$$

where F is the number of allowed intensity scales of the plaintext image. For example, $F = 256$ for an 8-bit gray scale image.

We write the 2D logistic substitution function as follows:

$$S_{2D} = \begin{cases} C = Ecr_{S_{2D}}(P) \\ P = Dcr_{S_{2D}}(L, P) \end{cases} \quad (24)$$

Algorithm4. Encryption Algorithm $C = \text{Enc}(P, K)$

Require: K a 256-bit key

Require: P is a 256×256 image block

Ensure: C is a 256×256 image block

```

(Q1, Q2) = GS(K, 8)
for n = 0: 1: 7
    if n == 0
        CLSP = LSBNoiseEmbedding = (P)
        Ln = GCL(Q1n, Q2n)
        Dn = Ln(0, 0)
        CDC = EcrDC(Ln, CLSp, Dn)
        If mod(n, 2) ≠ 0
            CSCL = Ecr8col(Ln, CDC)
        if not
            CSCL = Ecr8ligne(Ln, CDC)
        CPCL = Ecrp(Ln, CSCL)
        CP2D = EcrP2D(CPCL)
        CS2D = EcrS2D(CP2D)
    end
L8 = GCL(Q18, Q28)
D8 = L8(0, 0)
C = EcrDC(L8, CS2D, D8)

```

LSBNoiseEmbedding could be any function introducing a random noise at the level of the least significant bit (LSB).

4 Simulation Results and Security Analysis

Our simulation is performed in a Windows 7 environment, Core2, processor 2.6 and 2 GB of RAM. The images used can be downloaded from the database of the USC-SIPI (<http://sipi.usc.edu/database> 21/09/2013).

4.1 Key Space Analysis

The size of the key space characterizes the capability of resisting brute-force attack. A short key means that the best encryption algorithm can be broken by exhaustive search (also known as brute-force attack) in a reasonable amount of time, while the reverse is not true. In the proposed algorithm, the initial secret keys set $K = 256$ bit. Therefore, the key space is $2^{256} = 1.17 \times 10^{77}$, so the key space is large enough to resist all kinds of brute-force attacks.

4.2 Statistical Analysis

4.2.1 Histogram Analysis

An image histogram illustrates that how pixels in an image are distributed by plotting the number of pixels at each gray scale level. The distribution of cipher-text is of much importance. More specifically, it should hide the redundancy of plain-text and should not leak any information about the plain-text or the relationship between plain-text and cipher-text. Table 1 shows the histograms of plain-images and its ciphered images generated by the proposed scheme respectively. It's clear from that the histograms of the cipher-images are fairly uniform and significantly different from that of the plain image and hence do not provide any clue to employ statistical attack.

4.2.2 Information Entropy Analysis

In information theory, entropy is the most significant feature of disorder, or more precisely unpredictability. To calculate the entropy $H(X)$ of a source x , we have:

$$H(X) = \sum_{i=1}^n P_r(x_i) \log_2 \frac{1}{P_r(x_i)} \quad (25)$$

where X denotes the test image, x_i denotes the i th possible value in X , and $P_r(x_i)$ is the probability of $X = x_i$, that is, the probability of pulling a random pixel in X and its value is x_i . For a truly random source emitting 2^N symbols, the entropy is $H(X) = N$. therefore, for a ciphered image with 256 gray levels, the entropy should ideally be $H(X) = 8$. If the output of a cipher emits symbols with entropy less than 8, there exists certain degree of predictability, which threatens its security. The entropies for plain image and ciphered images using various images are calculated in Table 2. Apparently, the proposed algorithm is much closer to the ideal situation. This means that information leakage in the encryption process is negligible and the cryptosystem is secure against entropy attack.

Table 3 shows the comparison of proposed algorithm with existing algorithm. As compared to entropy of encrypted images in Ref. [13, 36–39] resultant

Table 1 Resultant Encrypted Images and its histogram of proposed method


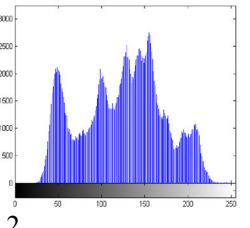
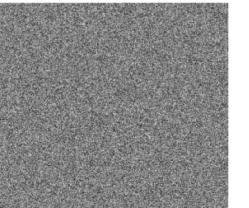
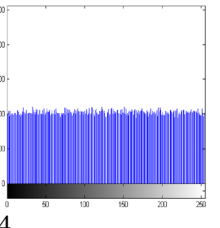
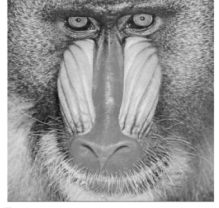
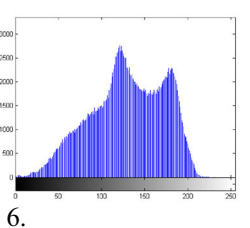
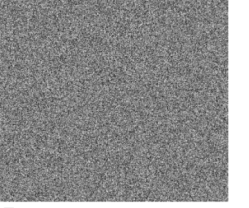
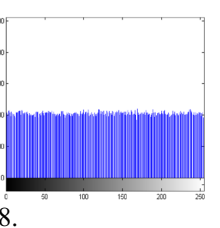

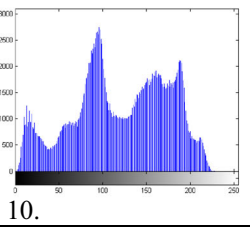
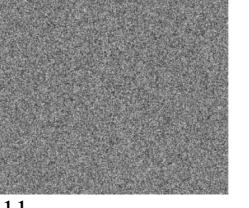
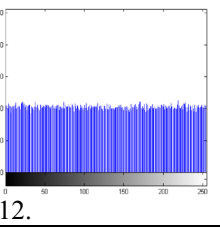

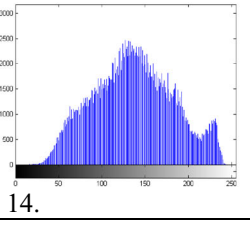
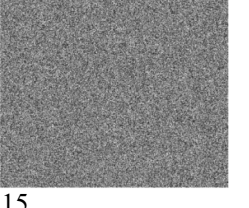
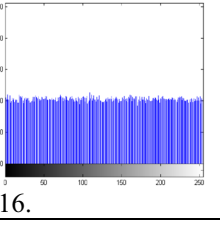

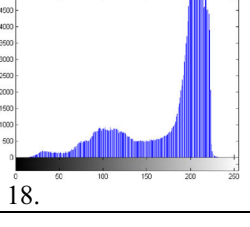
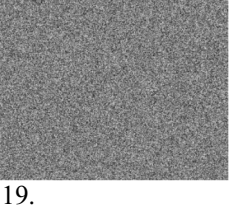
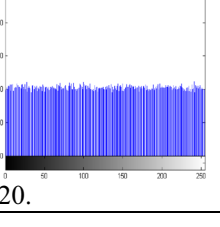

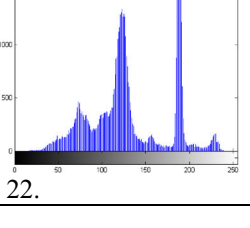
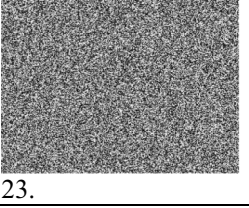
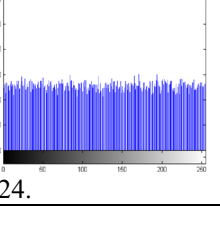
Input image	Histogram	Encrypted image	Histogram
 <p>1.</p>	 <p>2.</p>	 <p>3.</p>	 <p>4.</p>
 <p>5.</p>	 <p>6.</p>	 <p>7.</p>	 <p>8.</p>
 <p>9.</p>	 <p>10.</p>	 <p>11.</p>	 <p>12.</p>
 <p>13.</p>	 <p>14.</p>	 <p>15.</p>	 <p>16.</p>
 <p>17.</p>	 <p>18.</p>	 <p>19.</p>	 <p>20.</p>
 <p>21.</p>	 <p>22.</p>	 <p>23.</p>	 <p>24.</p>

Table 2 Entropy

Input image	Size	Plain image entropy	Encrypted image entropy
Lena	512 × 512	7.445	7.9993
Peppers	512 × 512	7.593	7.9992
Plane	256 × 256	6.955	7.9993
Babon	512 × 512	7.358	7.9993
Elaine	512 × 512	7.506	7.9993

Table 3 Encrypted image entropy of proposed algorithm with existing algorithms

Image	Proposed	[36]	[37]	[38]	[13]	[39]
Lena	7.9993	7.9993	7.9990	7.9974	7.9971	7.9970
Babon	7.9993	7.9992	7.989	7.9993	7.9997	7.9969

Table 4 Correlation coefficients of two adjacent pixels in plain-image and ciphered-images of proposed method

Image	Plain image			Encrypted image		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Lena	0.971926	0.985028	0.959328	-0.000169	0.001153	-0.000989
Peppers	0.976772	0.979206	0.963936	0.001754	0.000818	0.002377
Plain	0.966316	0.964127	0.937024	0.000540	-0.000361	0.001619
Babon	0.866542	0.758734	0.726188	0.002181	0.001928	0.002138
Elaine	0.975653	0.973016	0.969246	0.000720	0.002119	0.000457

entropy of our proposed method is more nearer to entropy of random image.

4.2.3 Analysis of Correlation of Adjacent Pixels

There exists a high correlation between pixels of an image which is called intrinsic feature. Thus, a secure encryption scheme should remove it to improve resistance against statistical analysis. To test the correlation between two-adjacent pixels in plain-image and cipher image, we randomly select 1000 pairs of two-adjacent pixels (in vertical, horizontal, and diagonal direction) from plain-image and cipher-image, and calculate the coefficient of each pair by Eq. (26):

$$r_{xy} = \frac{cov(x, y)}{\sqrt{D(x)}\sqrt{D(y)}} \tag{26}$$

where

$$cov(x, y) = \frac{1}{N} \sum_{i=1}^N (x_i - E(x))(y_i - E(y)) \tag{27}$$

$$D(x) = \frac{1}{N} \sum_{i=1}^N (x_i - E(x))^2 \tag{28}$$

$$E(x) = \frac{1}{N} \sum_{i=1}^N x_i \tag{29}$$

where x and y are gray-scale values of two-adjacent pixels in the image.

Table 4 shows the correlation coefficients of two horizontally adjacent pixels, two vertically adjacent pixels and two diagonally adjacent pixels in the plain and the cipher-image. It is clear that the strong correlation between adjacent pixels in plain image is greatly reduced in the cipher image produced by the proposed scheme, which is a private high-level security.

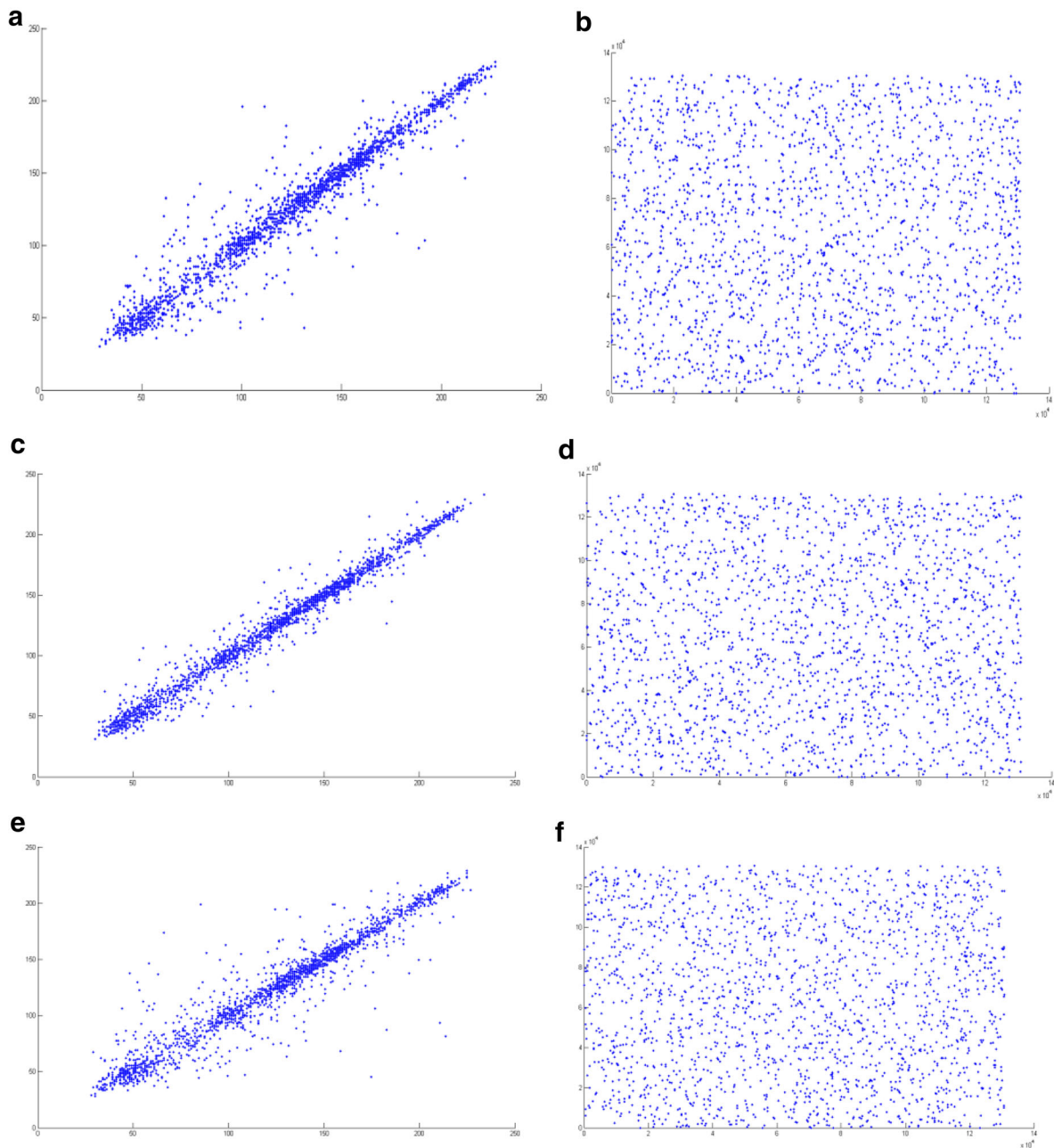


Fig. 4 Correlations of two adjacent pixels for Lena image of size 512×512 **a** horizontal direction of the plain image, **b** horizontal direction of the cipher image, **c** vertical direction of

the plain image, **d** vertical direction of the cipher image, **e** diagonal direction of the plain image, and **f** diagonal direction of the cipher image

Additionally, the correlation distribution of “Lena” and its ciphered image in each direction are plotted as shown in Fig. 4. Figure 4a–c shows the correlation distributions of the plain image, while Fig. 4d–f shows the correlation distributions of the ciphered image. The

strong correlation between adjacent pixels of the plain image is evident as all the dots are congregated along the diagonal in Fig. 4a–c. However, in Fig. 4d–f, the dots are scattered over the entire plane, which indicates that the correlation is greatly reduced in the ciphered

Table 5 Comparison of the correlation coefficients of Lena

		Horizontal	Vertical	Diagonal
Lena	Original	0.9355	0.9592	0.9087
Encrypted image	[36]	0.002016	-0.000916	0.001650
	[37]	0.02046	0.01748	0.02317
	[13]	0.005336	-0.0027616	0.0016621
	[39]	0.0020	-0.0007	-0.0014
	Proposed method	-0.000169	0.001153	-0.000989

Table 6 NPCR and UACI of proposed method

Image	NPCR	UACI
Lena	0.9979	0.3339
Peppers	0.9981	0.3332
Babon	0.9980	0.3337
Elaine	0.9979	0.3336

Table 7 Comparison results of the NPCR and UACI scores of the Lena image

Image	Methods	NPCR (%)	UACI (%)
Lena	[37]	99.61	33.48
	[38]	99.60	33.41
	[13]	99.60	99.66
	[39]	99.65	33.48
	Proposed algorithm	99.79	33.39

image. Table 5 shows the comparison with the schemes in Refs. [13, 36, 37, 39], the proposed scheme has smaller values in horizontal and diagonal directions.

4.3 Sensitivity Analysis

A well-designed encryption algorithm should be highly sensitive to plain-image and keys, so a slight change in plain-image or keys will make the cipher-image quite different. If an encryption scheme contains no confusion or diffusion stage, it would easily be destroyed by differential attacks. In order to confirm whether the proposed encryption algorithm is sensitive to plain image and keys, this paper brings out two tests: Number of pixels change rate (NPCR) and Unified average changing intensity (UACI) [40]. The equation to calculate NPCR is Eq. (30):

$$NPCR = \frac{\sum_{i,j} D(i,j)}{M \times N} \times 100\% \tag{30}$$

where, M stands for images width, N stands for images height, $C_1(i, j)$ means the gray-scale value of cipher-image in position (i, j), and $C_2(i, j)$ means the gray-scale value of the new cipher-image which is the encryption result of modified plain image that has just one different pixel to the original plain-image and where $D(i, j)$ defined as follows:

$$D(i,j) = \begin{cases} 1 & \text{if } C_1(i,j) \neq C_2(i,j) \\ 0 & \text{if } C_1(i,j) = C_2(i,j) \end{cases} \tag{31}$$

UACI can be calculated by the equation Eq. (32):

$$UACI = \frac{1}{M \times N} \sum_{i,j} \frac{|C_1(i,j) - C_2(i,j)|}{255} \times 100\% \tag{32}$$

When one bit of a pixels gray-scale value in the plain image is changed, then a new plain image is generated from the original one. Encrypt the two images with the same secret keys, and then take cipher images into Eqs. (30) and (32) and results are shown in Table 6. From this results we can find that our algorithm is very sensitive to tiny changes in the plain image, even if there is only one bit difference between two plain images, the decrypted images will be completely different. Table 7 gives the comparison of performance of UACI and NPCR when encrypting the image of Lena with Ref. [13, 37–39].

4.3.1 Key Sensitivity Analysis

An ideal image encryption procedure should be sensitive to the secret key in both encryption and decryption processes. It means that a change in a single bit of the secret key should produce a completely different encrypted image. Simulation results with respect to encryption and decryption stages are shown in Figs. 5 and 6. The encryption keys in our simulations are listed below in the HEX format:

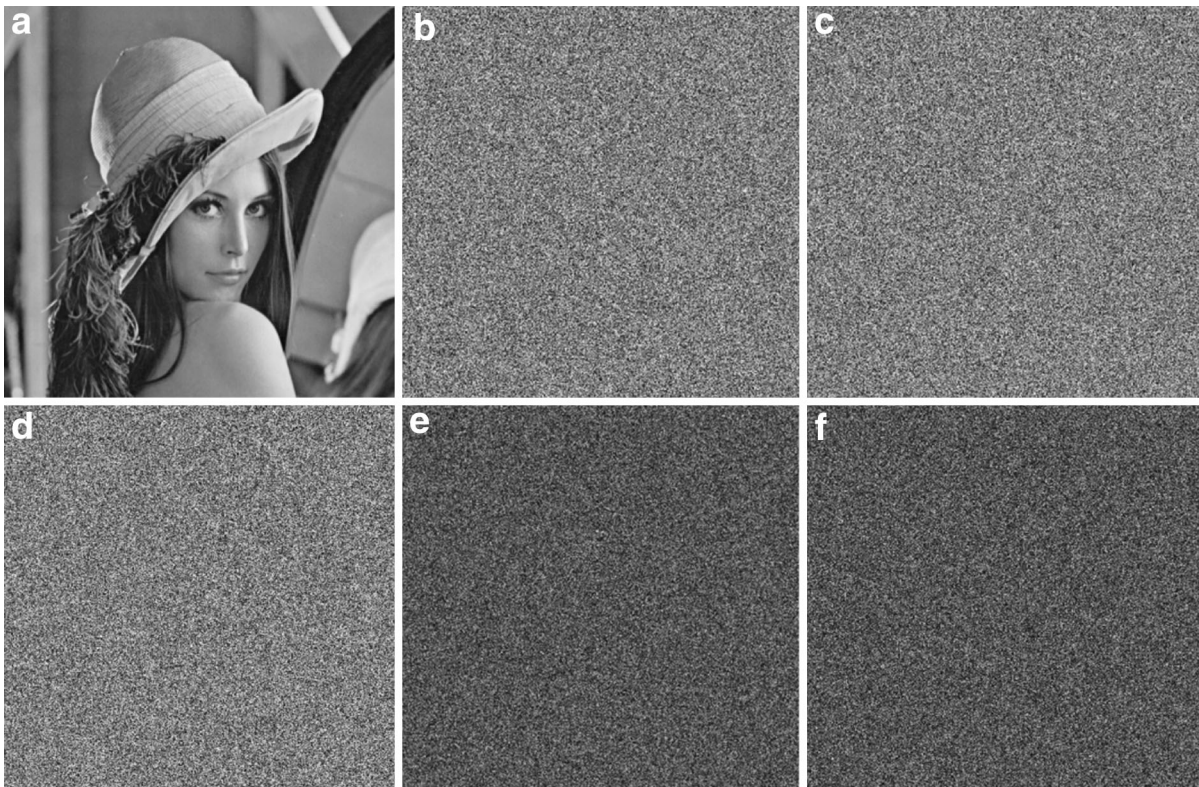


Fig. 5 Key sensitivity analysis at the encryption stage. **a** plaintext Lena, **b** Encrypted image by K1, **c** Encrypted image by K2, **d** Encrypted image by K3, **e** Ciphertext difference (C1–C2) and **f** ciphertext difference (C1–C3)

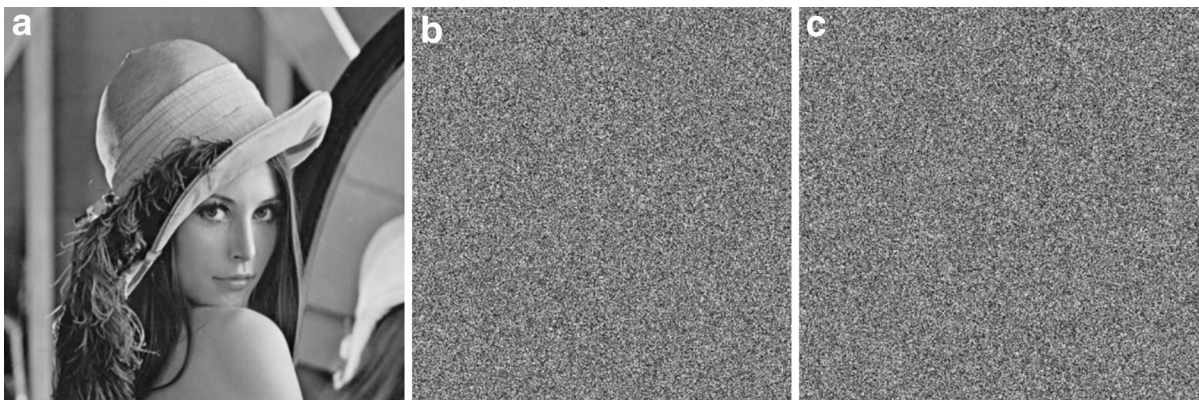


Fig. 6 Key sensitivity analysis at the decryption stage. **a** Decrypted image by K1, **b** Decrypted image by K2, **c** Decrypted image by K3

K1 = D9218FA02ED9AD9E1ED4FA1288EB2D
 D3FFDCFDDB01D8F21FA2E4A79F2FE2E0F5
 K2 = D9218FA12ED9AD9E1ED4FA1288EB2D
 D3FFDCFDDB01D8F21FA2E4A79F2FE2E0F5

K3 = D9218FA02ED9AD9E1ED4FA1288EB2D
 D3FFDCFDDB01D8F21FA2E4A79FAFE2E0F5

As can be seen, K1 differs K2 only for one bit, K2 differs K3 only for one bit, and K1 differs K3

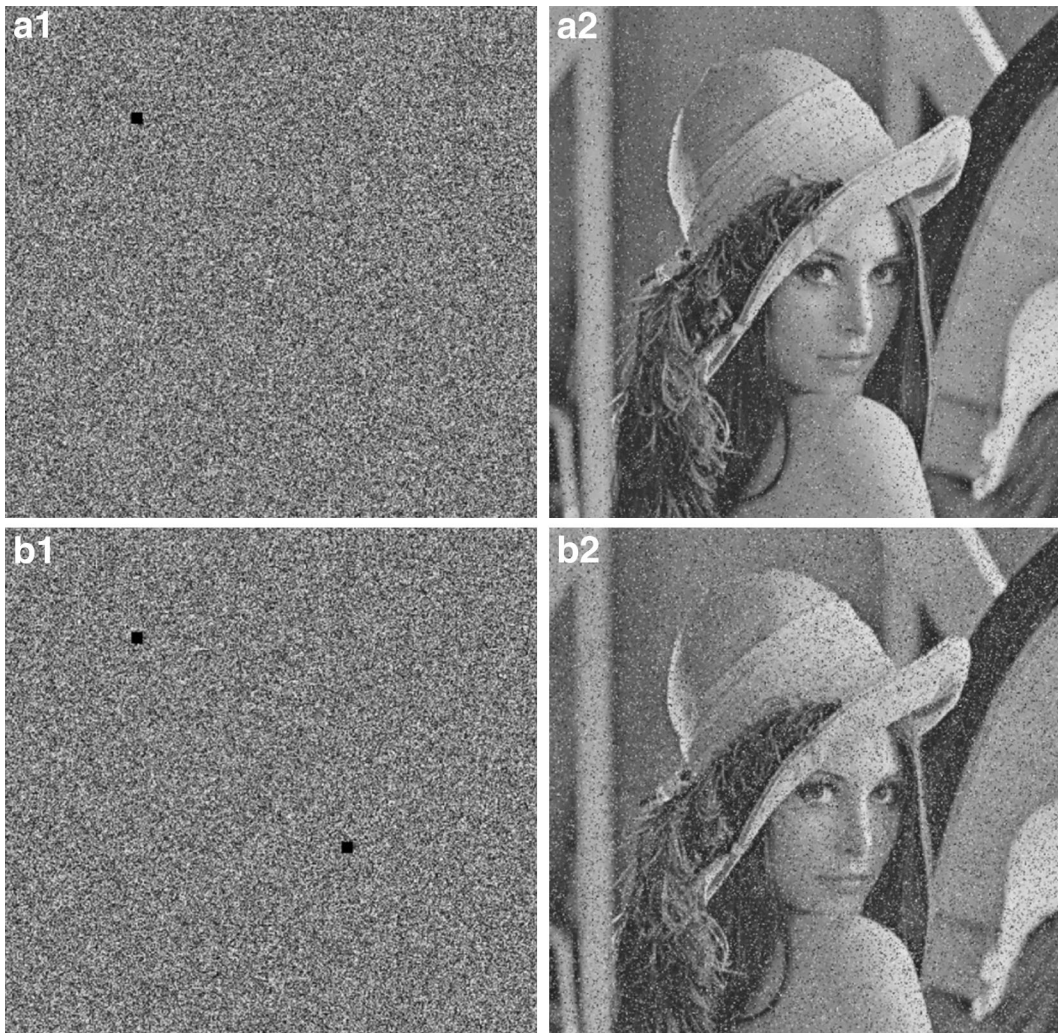


Fig. 7 shows the integration of noise in the encrypted image with a different degree. **a** Encrypted image with noise n_1 and its decrypted image, **b** Encrypted image with noise n_2 and its decrypted image

only for two bits. Although the hamming distances between K_1 , K_2 and K_3 are very small, i.e. they are very similar to each other, their corresponding cipher text images C_1 , C_2 and C_3 have significant differences. These can be verified by the fact that their differences, i.e. $|C_1 - C_2|$, $|C_1 - C_3|$, are random-like images as shown in Fig. 5. The example shows the proposed algorithm is sensitive to the encryption keys in the encryption stage.

Similar results can also be obtained in the decryption stage as shown in Fig. 6. We decrypt the same cipher text image C_1 using the encryption keys K_1 , K_2 and K_3 , respectively. As can be seen, using the correct encryption key K_1 , decrypted image D_1 perfectly

reconstructs the original plaintext image. However, decrypted images D_2 and D_3 using key K_2 and K_3 are random-like ones which do not contain any information related to the original plaintext image.

4.3.2 Noise Robustness Analysis

A good cipher should also be able to tolerate a certain amount of noise, e.g. noise in a channel or decoding errors. The proposed Latin square image cipher adopts an asymmetric structure for encryption and decryption, and one noisy pixel in cipher image will only propagate in a factor of two in each round. As shown in

Fig. 7 we find that in both cases, the decrypted image is still visible.

5 Conclusion

An appropriate image encryption system must be able to withstand all types of known attacks and cryptanalysis, and its performance must be independent whether from the encryption key or from the encrypted image itself. In addition, this system must have good properties of confusion and diffusion. Our system meets all of these criteria with the various stages that constitute it and has distinctive characteristics. The proposed algorithm is purely defined in integers, and thus it can be easily implemented in software and hardware without causing finite precision or discretization problems, the proposed algorithm constructs all encryption primitives based on one key generator, including cyclic shift, substitution and permutation, and thus the proposed method attains high sensitivities to any key change. The proposed algorithm arranges all encryption primitives in the framework of substitution–permutation network, and thus it attains good confusion and diffusion properties. The proposed method also integrates probabilistic encryption allowing to encrypt one plaintext image into different ciphertext images with one encryption key and the decryption stage is robust against a certain level of errors.

Security analysis including statistical attack analysis and differential attack analysis are performed numerically and visually. All the experimental results show that the proposed encryption scheme is secure thanks to its large key space, its high sensitivity to the cipher keys and plain-images. The results show that the proposed scheme leads to a higher security level in terms of NPCR, UACI and entropy of the cipher-images.

References

- Said, F., Yasser, A., & Abdo, A. (2011). How good is the DES algorithm in image ciphering? *International Journal of Advanced Networking and Applications*, 2(5), 796–803.
- Manoj, B., Manjula, N. (2012) Image encryption and decryption using AES. *International Journal of Engineering and Advanced Technology (IJEAT)* ISSN: 2249–8958, 1(5).
- Irfan, L., Burhanuddin, C., Aamna, P. et al. (2012) Image encryption and decryption using blowfish algorithm, National Conference on Emerging Trends in Information Technology.
- Lian, S. (2009). A block cipher based on chaotic neural networks. *Neurocomputing*, 72, 1296–1301.
- Guan, Z., Huang, F., & Guan, W. (2005). Chaos-based image encryption algorithm. *Physics Letters A*, 346, 153–157.
- Pareek, N., Patidar, V., & Sud, K. (2006). Image encryption using chaotic logistic map. *Image and Vision Computing*, 24, 926–934.
- Lian, S. (2009). Efficient image or video encryption based on spatiotemporal chaos system. *International Journal of Chaos Solitons Fractals*, 40(15), 2509–2510.
- Fu, C., Zhu, Z. (2008) A chaotic image encryption scheme based on circular bit shift method. The 9th International Conference for Young Computer Scientists, pp. 3057–3061.
- Yue, W., Gelan, Y., Huixia, J., et al. (2012). Image encryption using the two-dimensional logistic chaotic map. *Journal of Electronic Imaging*, 21(1), 013–014.
- Li, S., Mou, X., Cai, Y., et al. (2003). On the security of a chaotic encryption scheme: problems with computerized chaos in finite computing precision. *Computer Physics Communications*, 153(1), 52–58.
- Alvarez, G., Li, S., & Hernandez, L. (2007). Analysis of security problems in a medical image encryption system. *Computers in Biology and Medicine*, 37(3), 424–427.
- lvarez, G., Montoya, F., Romera, M., et al. (2003). Cryptanalysis of a discrete chaotic cryptosystem using external key. *Physics Letters A*, 319(3–4), 334–339.
- Yue, W., Yicong, Z., Joseph, P., et al. (2014). Design of image cipher using latin squares. *Information Sciences*, 264, 317–339.
- Radwan, A., Soliman, A., & Sedeek, E. L. (2004). Realization of the modified Lorenz chaotic system. *Chaos Soliton Fractals*, 21, 553–561.
- Radwan, A., Soliman, A., & Elwakil, A. (2007). 1-D digitally-controlled multi-scroll chaos generator. *International Journal of Bifurcation and Chaos (IJBC)*, 17(1), 227–242.
- Telem, A. N. K., Segning, C. M., Kenne, G., Fotsin, H. B. (2014) A simple and robust gray image encryption scheme using chaotic logistic map and artificial neural network. *Advances in Multimedia* (2014): 19
- Zhu, H., Zhao, C., & Zhang, X. (2013). A novel image encryption–compression scheme using hyper-chaos and Chinese remainder theorem. *Signal Processing: Image Communication*, 28(6), 670–680.
- Hraoui, S., Gmira, F., Saaidi, A., et al. (2015). Chaos based crypto-compression using SPIHT coding. *International Journal of Imaging and Robotics*, 15(2), 68–78.
- Kanso, A., & Ghebleh, M. (2012). A novel image encryption algorithm based on a 3D chaotic map. *Communications in Nonlinear Science and Numerical Simulation*, 17(7), 2943–2959.
- Pareek, N. K., Patidar, V., & Sud, K. K. (2013). Diffusion–substitution based gray image encryption scheme. *Digital Signal Process*, 23(3), 894–901.
- Al-Husainy, M. A. F. (2012). A novel encryption method for image security. *International Journal of Security and Its Applications*, 6(1), 1–8.
- Pareek, N. K., Patidar, V., & Sud, K. K. (2011). Substitution-diffusion based image cipher. *International Journal of*

- Network Security and Its Applications (IJNSA)*, 3(2), 149–160.
23. Li Z, Liu X (2010) The image encryption algorithm based on the novel diffusion transformation. In *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE)* (pp. 345–348).
 24. Wang, X., & Yang, L. (2012). A novel chaotic image encryption algorithm based on water wave motion and water drop diffusion models. *Optics Communication*, 285(20), 4033–4042.
 25. Fouda, J. S. A. E., Effa, J. Y., Sabat, S. L., & Ali, M. (2014). A fast chaotic block cipher for image encryption. *Communications in Nonlinear Science and Numerical Simulation*, 19(3), 578–588.
 26. Zhang, A., & Zhou, N. (2013). Color image encryption algorithm combining compressive sensing with Arnold transform. *Journal of Computers*, 8(11), 2857–2863.
 27. Zhang, Y., & Xiao, D. (2014). Self-adaptive permutation and combined global diffusion for chaotic color image encryption. *International Journal of Electronics and Communications (AEÜ)*, 68, 361–368.
 28. AbdElHaleem, S. H., Radwan, A. G., Abd-El-Hafiz, S. K. (2014) A chess-based chaotic block cipher. In *IEEE 12th International Conference on New Circuits and Systems* (pp. 405–408).
 29. Fournier-Prunaret, D., Lopez-Ruiz, R. (2003) Basin bifurcations in a two-dimensional logistic map, EprintarXiv:nlin/0304059.
 30. Luo, W., Huang, F., & Huang, J. (2010). Edge adaptive image steganography based on lsb matching revisited. *Information Forensics and Security. IEEE Transactions*, 5(2), 201–214.
 31. Menezes, A., Van Orschoot, P., & Vanstone, S. (1997). *Handbook of Applied Cryptography*. Boca Raton: Chapman and Hall CRC.
 32. Press, W. (2007). *The art of scientific computing. Numerical recipes*. New York: Cambridge University Press.
 33. Data encryption standard (1977) Federal Information Processing Standards Publication 46.
 34. Advanced encryption standard (2001) Federal Information Processing Standards Publication 197.
 35. Stinson, D. (2006). *The CRC Press series on discrete mathematics and its applications, cryptography: Theory and practice*. New York: Chapman & Hall/CRC.
 36. Zhi-liang, Z., Wei, Z., Wong, K., & Hai, Y. (2011). A chaos-based symmetric image encryption scheme using a bit-level permutation. *Information Sciences*, 181, 1171–1186.
 37. Wang, X., & Lei, Y. (2012). A novel chaotic image encryption algorithm based on water wave motion and water drop diffusion models. *Optics Communications*, 285, 4033–4042.
 38. Zhang, G., & Liu, Q. (2011). A novel image encryption method based on total shuffling scheme. *Optics Communications*, 284, 2775–2780.
 39. Wang, X., Zhang, Y., & Bao, X. (2015). A novel chaotic image encryption scheme using DNA sequence operations. *Optics and Lasers in Engineering*, 73, 53–61.
 40. Wua, Y., Joseph, P. N., & Agaianb, S. (2011). Shannon entropy based randomness measurement and test for image encryption. *Information Sciences*, 00, 1–23.