

An Efficient Image Encryption Scheme Based on a Peter De Jong Chaotic Map and a RC4 Stream Cipher

Gururaj Hanchinamani · Linganagouda Kulkarni

Received: 11 April 2015 / Accepted: 3 July 2015 / Published online: 14 July 2015
© 3D Research Center, Kwangwoon University and Springer-Verlag Berlin Heidelberg 2015

Abstract Security is a vital issue in communication and storage of the images and encryption is one of the ways to ensure the security. This paper proposes an efficient image encryption scheme based on a Peter De Jong chaotic map and a RC4 stream cipher. A Peter De Jong map is employed to determine the initial keys for the RC4 stream generator and also during permutation stage. The RC4 stream generator is utilized to generate the pseudo random numbers for the pixel value rotation and diffusion operations. Each encryption round is comprised of three stages: permutation, pixel value rotation and diffusion. The permutation is based on scrambling the rows and columns, in addition, circular rotations of the rows and columns in alternate orientations. The second stage circularly rotates each and every pixel value by utilizing $M \times N$ pseudo random numbers. The last stage carries out the diffusion twice by scanning the image in two different ways. Each of the two diffusions accomplishes the diffusion in two orientations (forward and backward) with two previously diffused pixels and two pseudo random numbers. The security and performance of the proposed method is assessed thoroughly by using key space, statistical,

differential, entropy and performance analysis. Moreover, two rounds of the call to the encrypt function provide the sufficient security. The experimental results show that the proposed encryption scheme is computationally fast with high security.

Keywords Image encryption/decryption · Peter De Jong map · RC4 stream generator · Statistical attacks · Differential attacks

1 Introduction

The need for the distribution of the digital images over networks has become a crucial part of everyday life. Moreover security is becoming an important apprehension in an increasingly multimedia defined world [1–5]. Hence, in order to maintain the privacy or security, sensitive data needs to be protected before transmission or storage. And, encryption is one of the ways to protect the information, which is a process of disguising a message to hide its substance.

Many traditional ciphers such as the DES, AES, GOST, BLOWFISH etc. were proposed as secure measures. But, due to the bulk volume of data and high correlation among the adjacent image pixels, traditional methods cannot suit for the image encryption. Moreover, it is computationally expensive if we use directly these traditional ciphers to encrypt image data [1–10].

G. Hanchinamani (✉) · L. Kulkarni
BVB College of Engineering & Technology,
Hubli 580031, India
e-mail: gs_hanchinamani@bvb.edu

L. Kulkarni
e-mail: linganagouda@yahoo.co.uk

In recent years, extensive studies have been done in the field of chaotic cryptography. The chaotic systems are characterized by ergodicity, random-like behaviors and sensitive dependence on initial values and parameters. These properties are of immense importance in cryptographic confusion and diffusion processes [5–10]. Hence chaotic based encryption is receiving increasing interest from cryptographers.

Over the past decade, a number of chaotic based image encryption schemes have been proposed, however, each of the image encryption schemes in the literature have their own strengths and constraints in terms of security and performance [1, 2]. This paper proposes a new image encryption scheme based on a Peter De Jong chaotic map and a RC4 stream cipher. The proposed encryption scheme is computationally fast over other schemes and is resistant to a variety of cryptanalytic attacks such as the brute-force attacks, statistical attacks, differential attacks and entropy based attacks.

The novelty of this paper is listed below.

- (i) The Peter de Jong chaotic map and RC4 stream ciphers are explored together for image encryption.
- (ii) Key generation and scheming is carried in an innovative way by initializing the RC4 with chaotic values.
- (iii) The encryption operations are carried with new methods.
- (iv) The proposed scheme is computationally fast with security intact.

The remaining of this paper is structured as follows. In Sect. 2, the literature survey is presented. The Peter De Jong chaotic map and RC4 stream ciphers are discussed in Sect. 3. The proposed encryption scheme is described in Sect. 4. Experimental findings and the security analysis are shown in Sect. 5 to illustrate the effectiveness and validity of the proposed scheme. Final section concludes the paper.

2 Literature Survey

Typically, a chaotic based image encryption scheme consists of two stages: permutation and diffusion [1–6]. The adjacent pixels are decorrelated by employing the permutation and spreading the effect of the individual pixels across the entire image is

achieved by using diffusion. A brief literature survey is presented hereafter.

The authors of [1], proposed a digital image encryption scheme, by employing multiple 1D chaotic map and the cryptographic primitive operations. The chaotic maps used are: logistic map, cubic map, sin map and tent map. The algorithm consists of three components: Key expansion, encryption and decryption. The scheme encrypts 256-bit of the plain image to 256-bit of the cipher image within eight 32-bit registers. The encryption scheme consists of eight w -bit registers $R_i (i = 1, 2, \dots, 8)$, which contain the initial input plain image as well as the output cipher image at the end of the encryption process. The encryption is performed by using a *xor* and pixel value circular rotations. In [2], an encryption scheme based on coupling of chaotic maps and a *xor* operator is presented. The method disturbs the inner binary structure of the image and progressively brings the randomness characteristics. The encryption process is straightly realized on the bits of the plain image. The image is shuffled by using a chaotic function based on linear congruence. The authors of [3] employed multiple chaotic maps to increase the key space to resist the brute force attacks. Nidhi Taneja [4], proposed a structure of combinational domain encryption that encrypts important data in spatial domain and the irrelevant data in wavelet domain. The proposed technique considers the varying data significance and support dissimilar security level to regions of different significance. The scheme makes use of Prewitt edge detector, discrete wavelet transform, Arnold cat map and logistic map.

In [5], an image encryption scheme based on an intertwining chaotic map is proposed to increase the keys pace and security. The substitution process uses six arbitrarily chosen odd integers to permute and then *xored* with the first chaotic key to shuffle and change the image pixel values. The diffusion stage alters the pixel values successively with various operations which include non linear diffusion using the first chaotic value, subdiagonal diffusion of the neighboring pixels and *xoring* with the third chaotic value. The authors of [6], proposed an image encryption scheme based on chaos and multiple S-boxes. A chaotic map is employed to determine the values of S-boxes. The encryption operations are based on the contents of S-boxes. The authors of [7], proposed a parametric switching chaotic system and its corresponding

transforms for image encryption. The scheme integrates the logistic, sin and tent maps into one single system. In [8], an image encryption scheme based on a generalized Arnold cat map is presented. The typical permutation diffusion architecture is employed. Two generalized Arnold cat maps are utilized to generate the chaotic sequences. The scheme has three parts: circular permutation, positive diffusion and opposite diffusion. In [9], an image encryption scheme based on cyclic elliptic curve and a chaotic system is presented. The round keys are generated based on piecewise nonlinear chaotic map. In [10], an image encryption scheme based on Chua chaotic system is presented. The operations are based on indexing and shuffling of the rows and columns. In [11], an image encryption algorithm based on chaos map together with pixel bit is presented. The scheme can implement the position encryption and gray value encryption simultaneously. The authors of [12], proposed an image encryption scheme by using a large pseudorandom permutation which is combinatorially generated from small permutation matrices based on chaotic maps. In [13], discrete exponential chaotic map is explored for image encryption. The permutation is carried with an S-box. In [14], a chaotic block cipher for wireless sensor network is presented and demonstrated better performance over RC5 and RC6 block ciphers. The details of the RC4 are provided in [15].

Though, there exist several image encryption schemes in the literature, each of them have their own strengths and limitations more or less in terms of security level and computational performance. Majority of the schemes produce satisfactory security results but lack in the computational speed. This paper proposes a new image encryption scheme based on a Peter De Jong chaotic map and a RC4 stream cipher. The proposed encryption scheme attains the required level of security with only two rounds and is computationally fast over other schemes in the literature. Moreover, the proposed approach is resistant to a range of cryptanalytic attacks such as the brute-force attacks, statistical attacks, differential attacks and entropy based attacks.

3 Pseudo Random Number Generators

The proposed encryption scheme utilizes a Peter De Jong chaotic map and a RC4 stream cipher for the

pseudo random number generation these are discussed hereafter.

3.1 Peter De Jong Chaotic Map

The 2D Peter De Jong map is a discrete-time dynamical system and is defined as,

$$X_{i+1} = \sin(aY_i) - \cos(bX_i) \quad (1)$$

$$Y_{i+1} = \sin(cX_{i+1}) - \cos(dY_i) \quad (2)$$

where X_i, Y_i are the current chaotic values, X_{i+1}, Y_{i+1} are the next chaotic values and a, b, c, d are the control parameters. The key set for the Peter De Jong map is $\{X_0, Y_0, a, b, c, d\}$. In the proposed scheme, the Peter De Jong chaotic values are used to determine the initial key values for the RC4 stream generator and also during the permutation.

The chaotic output sequence of the Peter De Jong map is assessed by computing the mean and self-correlations. It is observed that the average value of the chaotic sequence is close to 0.5 and the self correlations within the sequence and across the two sequences are very close to 0.

3.2 RC4 Stream Generator

The RC4 stream cipher is employed to generate the pseudo random numbers for the pixel value rotation and diffusion operations and is summarized beneath.

RC4 is a variable key-size stream cipher with byte oriented operations [15]. The algorithm is based on the use of the random permutation. Analysis show that the period of the cipher is likely to be greater than 10^{100} and the cipher run very quickly in software.

The RC4 algorithm is amazingly simple. A variable length key of from 1 to 256 bytes is used to initialize a 256 byte state vector S with elements $S[0], S[1], \dots, S[255]$. At all times, S contains a permutation of all 8-bit numbers from 0 through 255. For encryption operations a byte K is generated from S by selecting one of the 255 entries in a systematic fashion. As each value of K is generated, the entries in S are once again permuted.

Initiation of RC4 stream generation requires a *key* of length *keylen* bytes. *Key* is an array of length *keylen*. The *keylen* is variable and it varies from 1 to 256 bytes. A value of 256 is chosen as *keylen* in the proposed scheme hence there is no effect of mod operation.

Following is the pseudo code for the RC4 stream generation.

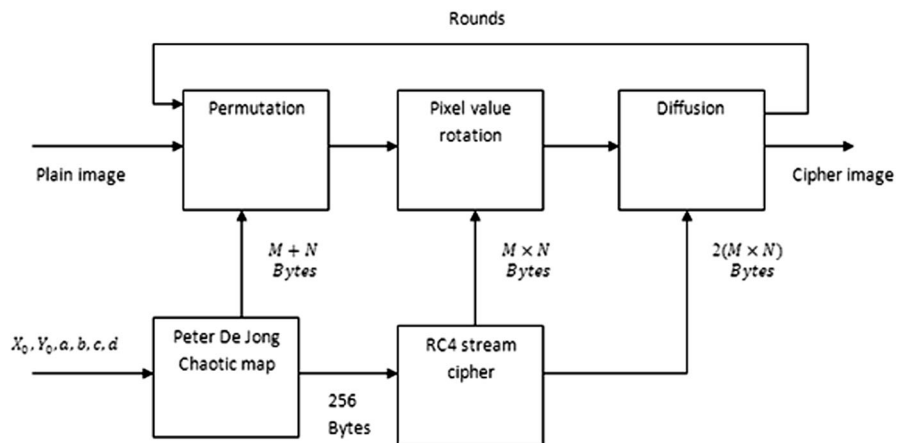
```

Step 1  /* Initialization of S*/
        for i = 0 to 255 do
        {
            S[i] = i;
            T[i] = Key[i mod Keylen];
        }
Step 2  /* Initial permutation of S */
        j=0;
        for i = 0 to 255 do
        {
            j = (j + S[i] + T[i]) mod 256;
            swap(S[i],S[j]);
        }
Step 3  /* Stream generation */
        i=0; j=0;
        while(true)
        {
            i = ( i + 1) mod 256;
            j = ( j + S[i]) mod 256;
            swap(S[i],S[j]);
            t = ( S[i] + S[j] ) mod 256;
            K = S[t];
        }
    
```

4 Proposed Encryption Scheme

The proposed encryption scheme is comprised of a round key generation function and three encryption stages: (i) permutation (ii) pixel value rotation and (iii) diffusion. The architecture of the proposed encryption scheme is shown in Fig. 1 and discussed hereafter.

Fig. 1 Architecture of the proposed encryption scheme



4.1 Round Key Generation and Scheming

In the proposed scheme, the key length of the RC4 stream cipher is chosen as 256 bytes. Hence at each encryption round it requires 256 pseudo random bytes as key, and are generated by employing a Peter De Jong map.

The key set of the Peter De Jong map is $\{X_0, Y_0, a, b, c, d\}$ and which initiates the RC4 stream generation. Hence the key set of the proposed encryption/decryption process is,

$$Key\ set = \{X_0, Y_0, a, b, c, d\} \tag{3}$$

where $\{X_0, Y_0\}$ are the initial values of the chaotic system and $\{a, b, c, d\}$ are the system parameters.

The generated chaotic sequence is a real sequence and is transformed to the integer form with the following transform.

$$T'_i = (T_i \times 10^8) \bmod m \tag{4}$$

where T_i is the real chaotic value, T'_i is the transformed integer value and m is 256 for the gray level image.

The Peter De Jong map is a 2D map and it generate two chaotic sequences X, Y and the following is the key scheming approach employed in the proposed scheme

- Step 1 Generate 128 Peter De Jong chaotic values $\{X\ and\ Y\ sequence\}$
- Step 2 Transform the real values to the integer form
- Step 3 Assign X sequence to $Key[0], \dots, Key[127]$, and Y sequence to $Key[128], \dots, Key[255]$

where X, Y are the Peter De Jong chaotic values and $Key[0, \dots, 255]$ is the initial key for the RC4 stream generator.

The above process is repeated for each round, and two rounds of the encrypt function provide the sufficient security.

Key scheming is shown beneath.

Round1 :

$$Key[0..255] = \{X[0], \dots, X[127], Y[0], \dots, Y[127]\}$$

Round2 :

$$Key[0..255] = \{X[128], \dots, X[255], Y[128], \dots, Y[255]\}$$

Round3 :

$$Key[0..255] = \{X[256], \dots, X[383], Y[256], \dots, Y[383]\}$$

and so on..

(5)

4.2 Permutation

The permutation is employed to decorrelate the adjacent pixels of an image. Let I be an original gray image with M rows and N columns and the gray intensities lie in the range 0 to 255. In the process of permutation, firstly, $M + N$ Peter De Jong chaotic values $\{(X_1, \dots, X_M), (Y_1, \dots, Y_N)\}$ are generated by using Eqs. (1, 2). Let $PM = \{X_1, \dots, X_M\}$ and $PN = \{Y_1, \dots, Y_N\}$. Subsequently, PM and PN are sorted, and the positions of the sorted chaotic values in the original chaotic sequence are found and are stored in PM' and PN' . The permutation stage is comprised of the following four steps

- Step 1 Scramble all the rows according to PM'_1, \dots, PM'_M
- Step 2 Scramble all the columns according to PN'_1, \dots, PN'_N
- Step 3 Circularly rotate each and every row in alternate orientations and the amount of rotation is based on PM'_1, \dots, PM'_M . The first row is rotated by PM'_1 , the second row is rotated by PM'_2 , and so on
- Step 4 Circularly rotate each and every column in alternate orientations and the amount of rotation is based on PN'_1, \dots, PN'_N

Permutation is exemplified below for an 4×4 image.

- Step 1 Generate $4 + 4$ chaotic values by using a Peter De Jong map. The first 4 values are used for the row permutation and the next 4 values for the column permutation
- Step 2 Suppose if the first 4 values are $PM = X_1, \dots, X_4 = 0.8, 0.7, 0.2, 0.5$
- Step 3 Sort the values.
 $0.2, 0.5, 0.7, 0.8$
- Step 4 Find the position of the sorted chaotic values in the original chaotic sequence.
 $PM' = 3, 4, 2, 1$
- Step 5 Scramble all the rows based on PM' i.e.,
1st row elements are moved to 3rd row
2nd row elements are moved to 4th row
3rd row elements are moved to 2nd row
4th row elements are moved to 1st row
- Step 6 Similarly scramble all the columns by using PN'
- Step 7 Circularly rotate each and every row in alternate orientations by using PM' i.e.
Circularly rotate 1st row left 3 times
Circularly rotate 2nd row right 4 times
Circularly rotate 3rd row left 2 times
Circularly rotate 4th row right once
- Step 8 Similarly, circularly rotate all the columns in alternate up and down direction by using PN'

The permutation stage involves sorting only $M + N$ values, hence it does not consume more energy.

4.3 Pixel Value Circular Rotations

This stage alters the pixel intensity values by circular rotation and is performed with following steps.

- Step 1 Read the image horizontally and then apply alternate left/right Circular rotation on pixel by pixel basis
- Step 2 Read the image vertically and then apply alternate left/right Circular rotation on pixel by pixel basis

In each of the above two steps, the amount of rotation for the circular rotation is based on $M \times N$ RC4 pseudo random numbers, and are computed with following steps.

- Step 1 Generate $M \times N$ RC4 pseudo random numbers

Step 2 Apply mod 8 for all the values, and replace 0 by a number in the range 1 to 7

The above steps generate $M \times N$ pseudo random numbers in the range 1 to 7, and are used as the rotation amount for the $M \times N$ pixels of the image.

4.4 Diffusion

The diffusion stage is employed to spread the effect of individual pixels over the entire image; i.e., a 1-bit change of the plain image should produce a drastic change in the encrypted image by using the same key. The proposed scheme apply the diffusion twice by scanning the image in two different ways as shown in Fig. 2 and given below.

- Step 1 Scan the image row-wise in alternate orientations and then apply the forward and backward diffusions
- Step 2 Scan the image column-wise in alternate orientations and then apply the forward and backward diffusions

Each of the above two steps carry out the forward and backward diffusions as discussed below.

At first, the 2D image is converted to 1D array $P_{1 \times MN}$ by scanning the image as shown in Fig. 2. Then generate $2(M \times N)$ pseudorandom numbers by utilizing a RC4 stream generator. Assign the first $M \times N$ pseudo random numbers to $R1$ and the next $M \times N$ pseudo random numbers to $R2$. The diffusion operation is performed by using two previously diffused pixels and two RC4 pseudo random numbers as shown in Eqs. (6, 7).

The forward diffusion is realized with the following equation,

$$E_i = (((((P_i + E_{i-2}) \bmod 256) + E_{i-1}) \bmod 256 \oplus R1_i) + R2_i) \bmod 256, i = 1, 2, \dots, MN \tag{6}$$

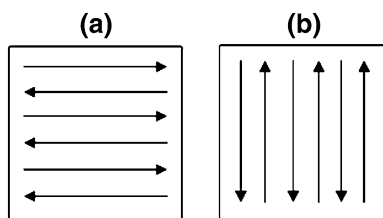


Fig. 2 scanning directions (a) row-wise in alternate orientations (b) column-wise in alternate orientations

where $+$ is the modulo addition, \oplus is the bitwise XOR, E_i is the current pixel to be encrypted, E_{i-1} and E_{i-2} are the formerly encrypted pixels, P_i is the permuted and rotated pixel, $R1_i$ and $R2_i$ are the RC4 pseudo random numbers. E_{-1} and E_0 are considered as the constants.

The backward diffusion is accomplished with Eq. (7) to make the effect of every pixel equal

$$F_i = ((((((E_i + F_{i+2}) \bmod 256) + F_{i+1}) \bmod 256 \oplus R1_i) + R2_i) \bmod 256, i = MN, \dots, 1 \tag{7}$$

where F_i is the current pixel to be encrypted, F_{i+1} and F_{i+2} are the formerly encrypted pixels, E_i is the forward diffused image pixel, $R1_i$ and $R2_i$ are the RC4 pseudo random numbers and E_{MN+1} and E_{MN+2} are considered as the constants. Finally, the encrypted image is obtained after these two diffusion steps.

4.5 Decryption

Decryption involves the reconstruction of the original image by following inverse steps. At first, the diffusion is applied then the pixel values are circularly rotated in reverse direction and finally the inverse permutation is applied.

4.6 Encryption Algorithm

The encryption algorithm is comprised of a main routine for the round key generation and an encrypt routine.

4.6.1 Main Routine for the Round Key Generation

- Step 1 Read the original image and store the pixel intensity values in the matrix $I_{M \times N}$
- Step 2 Initialize a Peter De Jong chaotic map $\{X_0, Y_0, a, b, c, d\}$
- Step 3 Generate 256 Peter De Jong chaotic values for the RC4 initialization by using Eqs. (1, 2)
- Step 4 Transform the real chaotic values to the integer form and assign to RC4 $Key[0, \dots, 255]$ as given in Eq. (5)
- Step 5 Generate $M + N$ Peter de Jong chaotic values and assign to $PM = (X_1, \dots, X_M)$ and $PN = (Y_1, \dots, Y_N)$ for the permutation by using Eqs. (1, 2)
- Step 6 Invoke the encrypt function

Step 7 Repeat step 3 through step 7 for the modified image

Note: Two rounds of the encryption provide the sufficient security.

4.6.2 Encrypt Routine

The encrypt routine is comprised of the following steps.

- Step 1 Obtain the image
- Step 2 Sort PM and PN , find the position of the sorted chaotic values in the original chaotic sequence and store in PM' and PN'
- Step 3 Scramble all the rows according to PM'_1, \dots, PM'_M
- Step 4 Scramble all the columns according to PN'_1, \dots, PN'_N
- Step 5 Circularly rotate each and every row in alternate orientations and the amount of rotation is based on PM'_1, \dots, PM'_M
- Step 6 Circularly rotate each and every column in alternate orientations and the amount of rotation is based on PN'_1, \dots, PN'_N
- Step 7 Generate $M \times N$ RC4 pseudo random numbers for the pixel value rotation
- Step 8 Apply mod 8 for all the values, and replace 0 by a number in the range 1 to 7
- Step 9 Read the image horizontally and then apply alternate left/right circular rotation on pixel by pixel basis
- Step 10 Read the image vertically and then apply alternate left/right circular rotation on pixel by pixel basis
- Step 11 Generate $2 \times (M \times N)$ RC4 pseudo random numbers for the diffusion
- Step 12 Scan the image row-wise in alternate orientations and then apply the forward and backward diffusions by using Eqs. (6, 7)
- Step 13 Scan the image column-wise in alternate orientations and then apply the forward and backward diffusions by using Eqs. (6, 7)

4.7 Decryption Algorithm

The decryption algorithm is comprised of a main routine for the round key generation and a decrypt routine.

4.7.1 Main Routine for the Round Key Generation

- Step 1 Read the encrypted image and store the pixel intensity values in the matrix $EN_{M \times N}$
- Step 2 Initialize a Peter De Jong chaotic map $\{X_0, Y_0, a, b, c, d\}$
- Step 3 Generate $(256 \times \text{Number of rounds})$ Peter De Jong chaotic values for the RC4 initialization by using Eqs. (1, 2)
- Step 4 Generate $[(M + N) \times \text{Number of rounds}]$ Peter De Jong chaotic values for the inverse permutation by using Eqs. (1, 2)
- Step 5 Transform the real chaotic values to the integer form and assign to RC4 $Key[0, \dots, 255]$ in the reverse way of their invocations
- Step 6 Assign the real chaotic values to $PM = \{X_1, \dots, X_M\}$ and $PN = \{Y_1, \dots, Y_N\}$ in the reverse way for the inverse permutation
- Step 7 Invoke the decrypt function
- Step 8 Repeat step 5 through step 8 for the modified image

4.7.2 Decrypt Routine

The decrypt routine is comprised of the following steps.

- Step 1 Obtain the image
- Step 2 Generate $2 \times (M \times N)$ RC4 pseudo random numbers for the reverse diffusion
- Step 3 Scan the image column-wise in alternate orientations and then apply the reverse forward and backward diffusions
- Step 4 Scan the image row-wise in alternate orientations and then apply the reverse forward and backward diffusions
- Step 5 Generate $M \times N$ RC4 pseudo random numbers for the reverse pixel value rotation
- Step 6 Apply mod 8 for all the values, and replace 0 by the same number as in encryption
- Step 7 Read the image vertically and then apply the reverse alternate left/right circular rotation on pixel by pixel basis
- Step 8 Read the image horizontally and then apply the reverse alternate left/right circular rotation on pixel by pixel basis

- Step 9 Sort PM and PN , find the position of the sorted chaotic values in the original chaotic sequence and store in PM' and PN'
- Step 10 Circularly rotate each and every column in reverse alternate orientations and the amount of rotation is based on PN'_1, \dots, PN'_N
- Step 11 Circularly rotate each and every row in reverse alternate orientations and the amount of rotation is based on PM'_1, \dots, PM'_M
- Step 12 Inverse scramble all the columns according to PN'_1, \dots, PN'_N
- Step 13 Inverse scramble all the rows according to PM'_1, \dots, PM'_M

5 Experiments and Security Analysis

5.1 Experimental Setup

The experiments are carried out on a personal computer configured with intel(R) Core(TM) i3-2120 CPU at 3.30 GHz with 2.91 GB of RAM. The code is developed by using C programming language under the Linux platform. The test images are chosen from the *USC-SIPI* image database (sipi.usc.edu/database/) and these are 256×256 gray-scale images. The initial values and parameters of the Peter De Jong chaotic system is randomly set according to Eq. (3) as $\{X_0 = 0.6, Y_0 = 0.4, a = 1.77, b = 1.67, c = -0.85, d = 2.1\}$.

5.2 Visual Assessment

Figure 3 shows the application of the proposed encryption scheme for three different images with varying content. It can be seen that there is no visual information in the encrypted images. The encrypted images are random-like and highly disordered.

5.3 Key-Space Analysis

The chaotic system adopted in this paper is highly sensitive to the initial values and parameters. The key set of the proposed encryption scheme contains 6 parameters $\{X_0, Y_0, a, b, c, d\}$. The key-length of the

proposed scheme is 384 bits by considering 64 bits for each parameter. And the key-space is 2^{384} , which is sufficient to resist the brute-force attacks. Table 1 shows the key-space of the proposed scheme and some of the other approaches in the literature.

5.4 Statistical Analysis

The statistical analysis can be carried out by computing the histograms and the correlations as discussed below.

5.4.1 Histogram Analysis

An image histogram illustrates the frequency distribution of each pixel intensity value [1–10]. Figure 4 shows the histograms of the original and encrypted images. It can be seen that the histograms of the encrypted images are uniformly spread and are wholly different from that of the original image, and bear no statistical resemblance to the original images. Thus the proposed scheme is resistant to the histogram based statistical attacks. The histograms are also uniformly spread for majority of the schemes in the literature.

5.4.2 Correlation Analysis

An image encryption scheme should produce the encrypted image with no correlations among the adjacent pixels [1–7]. The correlation coefficient between the adjacent pixels in an image is determined as in Eqs. (8–11)

$$E(x) = \frac{1}{N} \sum_{i=1}^N x_i \quad (8)$$

$$D(x) = \frac{1}{N} \sum_{i=1}^N (x_i - E(x))^2 \quad (9)$$

$$\text{cov}(x, y) = \frac{1}{N} \sum_{i=1}^N (x_i - E(x))(y_i - E(y)) \quad (10)$$

$$r_{xy} = \frac{\text{cov}(x, y)}{\sqrt{D(x)}\sqrt{D(y)}} \quad (11)$$

where x and y are the adjacent pixels of the original or encrypted images, $E(x)$ is the mean value, $D(x)$ is the

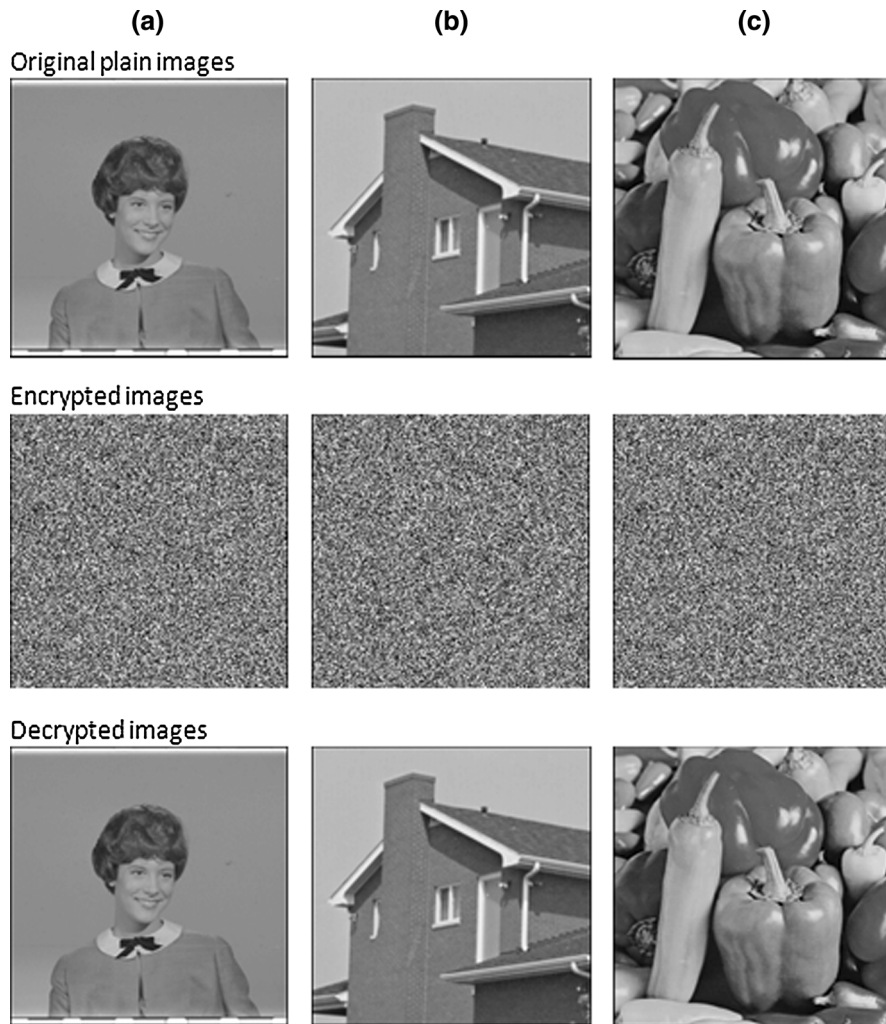


Fig. 3 Original images, encrypted images and decrypted images with the proposed scheme **a** Girl **b** House **c** Peppers

Table 1 Key space of the proposed method and some of the other methods in the literature

Method	Key space
Proposed algorithm	2^{384}
Ref. [1]	2^{349}
Ref. [2]	2^{128}
Ref. [3]	2^{400}
Ref. [4]	2^{135}
Ref. [5]	2^{216}
Ref. [6]	2^{240}

deviation with respect to the mean, $cov(x, y)$ is the covariance among the adjacent pixels, and r_{xy} is the correlation coefficient.

Correlations of the original and encrypted images are computed by randomly selecting 2048 pairs of the adjacent pixels in three different orientations: horizontal, vertical, and diagonal. Table 2 shows the outcome of the correlation coefficient calculation. It is observed that the adjacent pixels in the original image are extremely correlated to each other; however the correlation coefficients for the encrypted images are very close to zero. Hence the proposed scheme is resistant to the correlation based statistical attacks. The comparison of the correlation outcome with the other schemes is given in Table 3. It can be seen that correlation results of the proposed scheme is extremely close to zero and comparable to the other schemes in the literature.

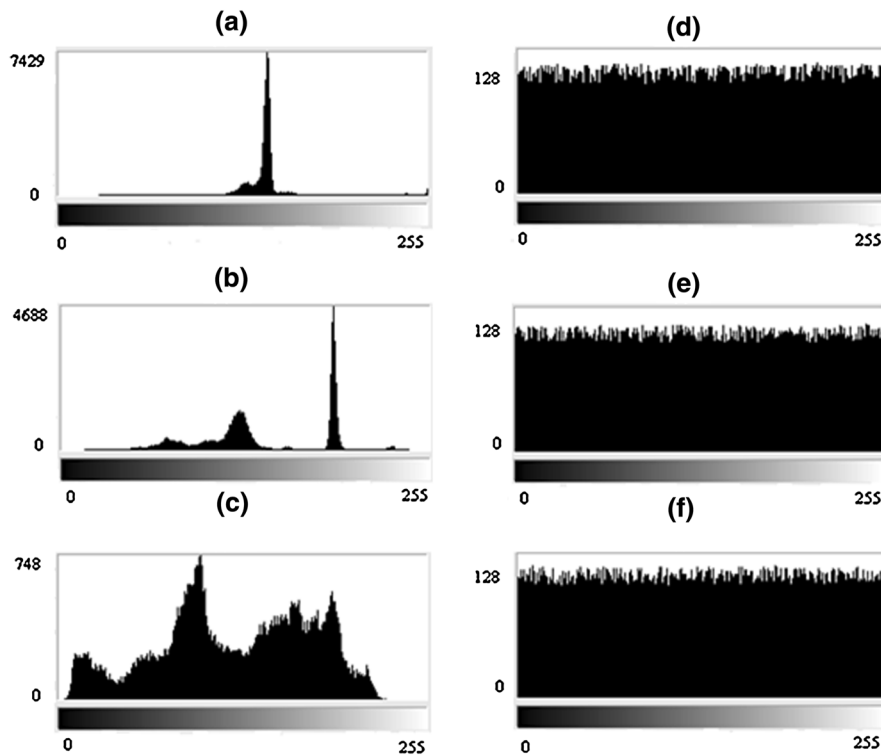


Fig. 4 Histograms of the original and encrypted images (a–c) Histograms of the original images Girl, House and Peppers (d–f) Histograms of the respective encrypted images

5.5 Sensitivity Analysis

An image encryption scheme is expected to be sensitive to both the secret key and the plain image and is demonstrated beneath.

5.5.1 Key Sensitivity

Key sensitivity implies that a minute change of the key should produce a drastic change in the encrypted image by using the same plain image [1–10]. The key sensitivity test has been conducted with the following steps.

- Step 1 Encrypt the plain image with a key K_1 to produce a cipher image C_1
- Step 2 Create a key K_2 from key K_1 with a small change; Encrypt the plain image again with key K_2 to produce the cipher image C_2
- Step 3 Compare the two cipher images C_1 and C_2 to observe the number of differing pixels

To quantitatively test the influence of minute alterations in the key, two common measures are

used, i.e., number of pixels change rate ($NPCR$) and unified average changing intensity ($UACI$), they are determined as in Eqs. (12–14)

$$NPCR = \frac{\sum_{i,j} D(i,j)}{M \times N} \times 100\% \tag{12}$$

$$D(i,j) = \begin{cases} 1, & \text{if } C_1(i,j) \neq C_2(i,j) \\ 0, & \text{Otherwise} \end{cases} \tag{13}$$

$$UACI = \frac{1}{M \times N} \left[\sum_{ij} \frac{C_1(i,j) - C_2(i,j)}{255} \right] \times 100\% \tag{14}$$

where C_1 and C_2 are the two encrypted images with slight different keys K_1 and K_2 . $C_1(i, j)$ and $C_2(i, j)$ are the pixel intensity values at the location (i, j) . D is a bipolar array with the same size as C_1 and C_2 and its contents are either 0 or 1 based on Eq. (13).

The key sensitivity is assessed by testing one parameter at a time with a minute change in the key. The key set of the proposed scheme contain six parameters $\{X_0, Y_0, a, b, c, d\}$. The original key is

Table 2 Correlation coefficients of the adjacent pixels in different orientations for the original and encrypted images

Image	Correlation coefficients for the encrypted images			Correlation coefficients for the original images		
	Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
Lena	-0.000319	0.001201	0.005271	0.968683	0.943269	0.933408
Girl	-0.000085	0.000394	-0.002232	0.921996	0.974345	0.900548
House	-0.002920	0.001395	0.007965	0.941437	0.978137	0.931483
Peppers	-0.000347	0.009052	0.000524	0.966791	0.960712	0.940809
Truck	-0.004059	0.005093	0.001668	0.879504	0.946834	0.856167
Tree	0.006592	-0.005950	0.004847	0.935594	0.968341	0.917945

Table 3 Comparison of the correlation coefficients of the proposed scheme with the other methods

Method	Correlation coefficients		
	Horizontal	Vertical	Diagonal
Proposed algorithm	-0.000319	0.001201	0.005271
Ref. [1]	0.003200	0.002424	0.001520
Ref. [2]	-0.001700	0.005900	-0.001900
Ref. [3]	0.006300	0.005900	0.007300
Ref. [4]	-0.044500	0.004100	Not specified
Ref. [5]	0.004800	0.002700	0.003600
Ref. [6]	0.003206	-0.000289	0.001167

altered with minute changes and six different keys are generated. The keys can be summarized as, the original key,

$$K_1 = \{0.6, 0.4, 1.77, 1.67, -0.85, 2.1\}$$

and the slightly altered keys,

$$K_2 = \{0.600000001, 0.4, 1.77, 1.67, -0.85, 2.1\}$$

$$K_3 = \{0.6, 0.400000001, 1.77, 1.67, -0.85, 2.1\}$$

$$K_4 = \{0.6, 0.4, 1.770000001, 1.67, -0.85, 2.1\}$$

$$K_5 = \{0.6, 0.4, 1.77, 1.670000001, -0.85, 2.1\}$$

$$K_6 = \{0.6, 0.4, 1.77, 1.67, -0.850000001, 2.1\}$$

$$K_7 = \{0.6, 0.4, 1.77, 1.67, -0.85, 2.100000001\}$$

Table 4 shows the result of the key sensitivity test for six different parameters. It can be seen that the *NPCR* and *UACI* values are close to their ideal values of 99.6 and 33.4 %. Thus the proposed scheme has high key sensitivity. The key sensitivity results in the

Table 4 Key sensitivity results with different parameters of the chaotic map

Parameter changed	Key used	<i>NPCR</i> (%)	<i>UACI</i> (%)
X_0	K_2	99.600220	33.418064
Y_0	K_3	99.601746	33.496555
a	K_4	99.592590	33.496502
b	K_5	99.613953	33.598839
c	K_6	99.626160	33.642021
d	K_7	99.629211	33.500935

literature are satisfactory and are demonstrated in the visual form hence are not shown here.

In addition, the key sensitivity results are also shown pictorially in Fig. 5. The encrypted images with the correct key (K_1) and the incorrect key (K_2) are shown in Fig. 5b, c. Even though they look similar, they are entirely different from each other. This can be seen by observing the difference image given in Fig. 5d.

Furthermore, the key sensitivity test is also analyzed for the decryption process. The decryption is

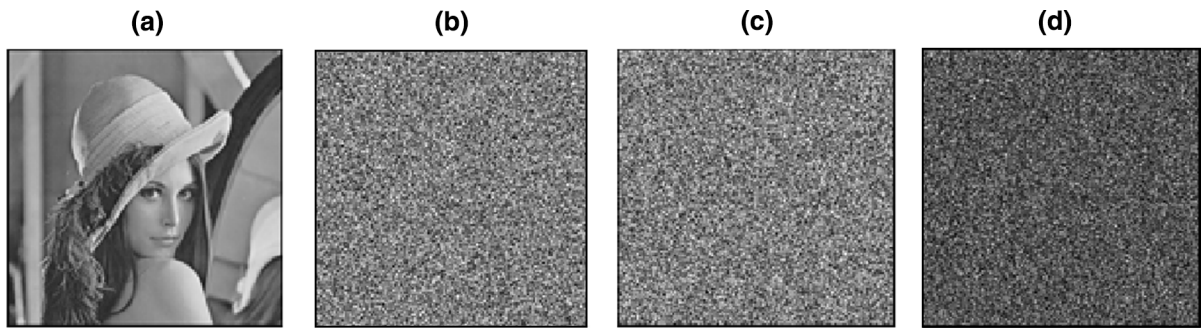


Fig. 5 Key sensitivity test for the encryption process for the Lena image **a** original image **b** encrypted image with the correct key **c** encrypted images with slightly different key **d** difference image between **b** and **c**



Fig. 6 Key sensitivity test for the decryption process for the Lena image. **a** original image **b** decryption with the correct key **c** decryption with the slight changed key

performed with the correct key (K_1) and the slight altered key (K_2). The Fig. 6b shows the decrypted image with the correct key and Fig. 6c shows the decrypted image with slight altered key. The decrypted image with slight altered key is random-like and thus it is observed that the correct decryption cannot be attained if there is a small alteration in the key.

5.5.2 Plain Image Sensitivity

Plain image sensitivity implies that, a 1-bit alteration of the plain image should produce a drastic change in the encrypted image by using the same key [1–10]. The plain-image sensitivity analysis is conducted with the following steps.

- Step 1 Encrypt the original image to produce a cipher image C_1
- Step 2 Alter 1-bit of the original image at any arbitrary location, and encrypt again to produce the cipher image C_2

Table 5 Plain image sensitivity test

Bit position	Changed pixel values	$NPCR$ (%)	$UACI$ (%)
0	222	99.629211	33.397148
1	221	99.577332	33.499115
2	219	99.588013	33.485855
3	215	99.632263	33.466526
4	207	99.630737	33.402138
5	255	99.668884	33.514412
6	159	99.606323	33.481953
7	95	99.601746	33.480759
Average values		99.616815	33.465988

- Step 3 Compare the two cipher images C_1 and C_2 to observe the number of differing pixels

Two common measures $NPCR$ and $UACI$ are used to test the influence of 1-bit alteration in the plain image. Table 5 shows the outcome of the plain image sensitivity test for an arbitrarily chosen position (60,

Table 6 Comparison of the plain image sensitivity results with the other schemes in the literature

Method	Plain image sensitivity	
	<i>NPCR</i> %	<i>UACI</i> %
Proposed algorithm	99.616815	33.465988
Ref. [1]	99.610000	33.450000
Ref. [2]	99.610000	33.450000
Ref. [3]	99.609200	33.489100
Ref. [4]	60.00–94.00	15.00–38.00
Ref. [5]	99.562100	33.432400
Ref. [6]	99.605745	33.473353

195). The pixel intensity value of the selected position is 223. The *NPCR* and *UACI* values are observed by changing one bit at a time from the lower bit to the upper bit. The average *NPCR* and *UACI* values are found to be 99.616815 and 33.465988 % and these values are close to their ideal values of 99.6 and 33.4 %. Thus, the proposed encryption scheme has higher plain image sensitivity and is resistant to the differential attacks. Table 6 lists the plain image sensitivity results of the proposed scheme and the other schemes in the literature. The plain image sensitivity results of the proposed scheme are comparable to other schemes and are close to the ideal values.

5.6 Information Entropy Analysis

Information entropy is a measure of the amount of randomness in information content, and is determined as given below [1–5],

$$H(S) = \sum_{i=0}^{n-1} P(S_i) \log_2 \frac{1}{P(S_i)} \tag{15}$$

where S_i represents the pixel values, $P(S_i)$ is the probability of the symbol S_i and n is the total number of symbols and it is 256 for the gray scale images. Suppose that, the source emits 2^8 symbols with the same probability, i.e. $S = (S_0, S_1, S_2, \dots, S_{255})$ after evaluating Eq. (15), we obtain its entropy $H(S) = 8$. Table 7 shows the outcome of the entropy calculation. It can be seen that the entropies of the encrypted images are extremely close to the ideal value of 8. Hence the information outflow in the proposed encryption process is insignificant and is resistant against the entropy based attacks. The comparison of

Table 7 Entropy values for original and encrypted images for different images

Image	Entropy	
	Original image	Encrypted image
Lena	7.426985	7.997279
Girl	5.590780	7.996859
House	6.503890	7.997036
Peppers	7.576389	7.997114
Truck	6.527547	7.997415
Tree	7.313894	7.997020

Table 8 Comparison of the entropy values of the proposed scheme with the other methods

Method	Entropy
Proposed algorithm	7.997279
Ref. [1]	7.996991
Ref. [3]	7.993000
Ref. [5]	7.994000
Ref. [6]	7.999369
Ref. [2, 4]	Not specified

the entropy outcome with the other schemes is listed in Table 8. The entropy outcome of the proposed scheme is close to the ideal value of 8 and is comparable to other schemes in the literature.

5.7 Peak Signal to Noise Ratio (PSNR) Analysis

The objective evaluation of an encryption scheme can be performed by computing the *PSNR*. In this evaluation, the original plain image is considered as a signal and the encrypted image is considered as a noise [4]. The *PSNR* is calculated as,

$$PSNR = 20 \times \log_{10} \left(\frac{255}{\sqrt{MSE}} \right) dB \tag{16}$$

where *MSE* is the mean square error and is computed as,

$$MSE = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (|I(i,j) - I'(i,j)|)^2 \tag{17}$$

where $I(i, j)$ is the pixel value of the original plain image and $I'(i, j)$ is the pixel value of the encrypted image at location (i, j) . The *PSNR* results for the

different test images are listed in Table 9. It can be seen that the *PSNR* values are lower, which indicates the complexity in getting the original plain image from the encrypted image for adversaries. The comparison of the *PSNR* outcome with the other schemes is listed in Table 10. The *PSNR* outcome of the proposed scheme is lower and is comparable to other schemes in the literature.

Table 9 The *PSNR* values for different test images

Image	<i>PSNR</i> (dB)	Image	<i>PSNR</i> (dB)
Lena	9.215507	Peppers	8.924724
Girl	10.029229	Truck	9.950633
House	9.211928	Tree	8.115029

Table 10 Comparison of the *PSNR* values of the proposed scheme with the other methods

Method	<i>PSNR</i> (dB)
Proposed algorithm	8.115029–10.029229
Ref. [4].	8.2462–9.7322
Ref. [1–3, 5, 6]	Not specified

Table 11 Comparison of the computational time with the other schemes in the literature

Method	Computational time in seconds	Time cost	Programming language	Operating system	System
Proposed algorithm	0.000006	10^{-5}	C	Linux	Intel core i3 at 3.30 GHz
Ref. [1]	0.093	10^{-1}	Visual C++	Windows XP	Intel dual core at 2.7 GHz
Ref. [2]	0.90	10^0	C	Linux	Pentium M at 1.7 GHz
Ref. [3]	0.00297	10^{-2}	Matlab	Windows Vista	Pentium Core 2 Duo at 3 GHz
Ref. [4]	0.2	10^0	Matlab	Windows	Pentium P4 dual core at 2.16 GHz
Ref. [5]	0.8645	10^0	Matlab	Windows	Pentium Core 2 Duo at 3 GHz
Ref. [6]	0.001396	10^{-2}	C++	Windows	Intel core i5 at 3.10 GHz

Table 12 Time complexity of the proposed algorithm

No.	Phases	Unit time	Time complexity
1	Round key generation	One round key generation	$O(1)$
2	Permutation	One row/column sorting and permutation	$O(M \log M + N \log N)$
3	Pixel value rotation	One pixel value rotation	$O(MN)$
4	Diffusion	One stage diffusion	$O(MN)$

5.8 Computational Speed Analysis

The running time of an encryption scheme can be determined by several factors such as the operating system, programming language, system configuration etc. The proposed encryption scheme is coded in C language under the Linux environment. The time taken to encrypt a 256×256 gray scale image is 0.000006 s and for decryption it is the same. Moreover two rounds of the encryption function provide sufficient security. Table 11 shows the computational time requirement of the proposed scheme and the other schemes in the literature. The proposed scheme requires only 0.000006 s and the time cost is 10^{-5} . The time cost of [1–6] varies from 10^0 to 10^{-2} . The time cost of the proposed scheme is also observed on system with differing speeds. The time cost of the proposed scheme at 2.7 and 1.7 GHz are found to be 10^{-5} and 10^{-4} respectively. Hence the proposed scheme is faster than [1–6]. The time complexity of the proposed algorithm consists of four terms as specified in Table 12. The time complexities of the proposed scheme and other schemes are listed in Table 13. It can be seen that the time complexity of the proposed scheme is lower and it is faster than [1–6].

Table 13 Comparison of the time complexities with the other schemes in the literature

Method	Time complexity
Proposed algorithm	$O(1) + O(M \log M + N \log N) + O(MN) + O(MN)$
Ref. [1]	$O(M) + O(MN) + O(MN) + O(MN)$
Ref. [2]	$O(MN) + O(MN) + O(MN) + O(MN)$
Ref. [3]	$O(MN) + O(MN) + O(MN) + O(MN)$
Ref. [4]	$O(MN) + O(MN) + O(MN) + O(MN)$
Ref. [5]	$O(MN) + O(MN) + O(MN) + O(MN) + O(MN)$
Ref. [6]	$O(1) + O(MN) + O(MN) + O(MN)$

6 Conclusions

In this work, an efficient image encryption scheme based on a Peter De Jong chaotic map and a RC4 stream cipher is presented. The proposed scheme has adequate key space of 2^{384} , hence the brute-force attacks are infeasible. The correlations are close to zero in all the three directions and the histogram is almost uniformly distributed, thus the statistical attacks are resisted. Furthermore, the *NPCR* and *UACI* values are close to their ideal values of 99.6 and 33.4 % for both the key sensitivity and plain image sensitivity tests, hence the scheme is resistant to the differential attacks. In addition, the average entropy attained is 7.997120, which is close to the ideal value of 8, and hence the information leakage is negligible. The *PSNR* is lower. The proposed scheme is coded in C under the Linux environment and the speed attained is 0.000006 s for 256×256 images. Moreover only two rounds of the encryption function provide sufficient security; hence the proposed encryption scheme is computationally fast with security intact.

References

1. El-Latif, A. A. A., Li, L., Zhang, T., Wang, N., Song, X., & Niu, X. (2012). Digital image encryption scheme based on multiple chaotic systems. *Sensing and Imaging*, 13(2), 67–88. doi:10.1007/s11071-012-0409-z.
2. Francois, M., Grosjes, T., Barchiesi, D., & Erra, R. (2012). A new image encryption scheme based on a chaotic function. *Signal Processing*, 27(3), 249–259. doi:10.1016/j.image.2011.11.003.
3. Sam, S., Devaraj, P., & Bhuvaneshwaran, R. S. (2012). A novel image cipher based on a mixed transformed logistic maps. *Multimedia Tools and Applications*, 56(2), 315–330. doi:10.1007/s11042-010-0652-6.
4. Taneja, N., Raman, B., & Gupta, I. (2012). Combinational domain encryption for still visual data. *Multimedia Tools*

- and Applications, 159(3), 775–793. doi:10.1007/s11042-011-0775-4.
5. Sam, S., Devaraj, P., & Bhuvaneshwaran, R. S. (2012). An intertwining chaotic maps based image encryption scheme. *Nonlinear Dynamics*, 69(4), 1995–2007. doi:10.1007/s11071-012-0402-6.
 6. Zhang, X., Mao, Y., & Zhao, Z. (2014). An efficient chaotic image encryption based on alternate circular S-boxes. *Nonlinear Dynamics*, 78(1), 359–369. doi:10.1007/s11071-014-1445-7.
 7. Zhou, Y., Bao, L., & Chen, C. L. P. (2013). Image encryption using a new parametric switching chaotic system. *Signal Processing*, 93(11), 3039–3052. doi:10.1016/j.sigpro.2013.04.021.
 8. Ye, G., & Wong, K. W. (2012). An efficient chaotic image encryption algorithm based on a generalized Arnold map. *Nonlinear Dynamics*, 69, 2079–2087. doi:10.1007/s11071-012-0409-z.
 9. El-Latif, A. A. A., Li, L., & Niu, X. (2014). A new image encryption scheme based on cyclic elliptic curve and chaotic system. *Multimedia Tools and Applications*, 70, 1559–1584. doi:10.1007/s11042-012-1173-2.
 10. Huang, C. K., Liao, C. W., Hsu, S. L., & Jeng, Y. C. (2013). Implementation of gray image encryption with pixel shuffling and gray-level encryption by single chaotic system. *Telecommunication Systems*, 52(2), 563–571. doi:10.1007/s11235-011-9461-0.
 11. Ye, Guodong. (2010). Image scrambling encryption algorithm of pixel bit based on chaos map. *Pattern Recognition Letters*, 31(5), 347–354. doi:10.1016/j.patrec.2009.11.008.
 12. Yoon, J. W., & Kim, H. (2010). An image encryption scheme with a pseudorandom permutation based on chaotic maps. *Communications in Nonlinear Science and Numerical Simulations*, 15(12), 3998–4006. doi:10.1016/j.cnsns.2010.01.041.
 13. Zhang, L., Liao, X., & Wang, X. (2004). An image encryption approach based on chaotic maps. *Chaos, Solitons & Fractals*, 24(3), 759–765. doi:10.1016/j.chaos.2004.09.035.
 14. Liu, Y., Tian, S., Hu, W., & Xing, C. (2012). Design and statistical analysis of a new chaotic block cipher for wireless sensor networks. *Communications in Nonlinear Science and Numerical Simulation*, 17(8), 3267–3278. doi:10.1016/j.cnsns.2011.11.040.
 15. Wong, K. K., Carter, G., Dawson, E. (2010). An analysis of the RC4 family of stream ciphers against algebraic attacks. *Proceedings 8th Australasian information security conference*, 103, pp. 67–74.