A New Color Image Encryption Scheme Based on Chaotic Hénon Map and Lü System

Chong Fu[1(✉)], Gao-yuan Zhang[1], Bei-li Gao[2], Jing Sun[1], and Xue Wang[1]

[1] School of Computer Science and Engineering, Northeastern University, Shenyang 110004, China
fuchong@mail.neu.edu.cn
[2] China Waterborne Transport Research Institute, Beijing 100088, China

Abstract. This paper presents a new efficient chaos-based color image cipher robust against chosen-plaintext attack. The chaotic Hénon map and Lü system are employed to produce the permutation and substitution keystream sequences for image data scrambling and mixing, respectively. In the permutation stage, the positions of colored subpixels in the input image are scrambled using a pixel-swapping mechanism. To strengthen the robustness of the substitution procedure against chosen-plaintext attack, we introduce a novel mechanism for associating the keystream sequence with the plain-image. Compared with other related mechanisms, the new mechanism is implemented during the subpixel values mixing process rather than the keystream generation process. As a result, the keystream sequence can be reused among different rounds of substitution operation, and hence the computational cost is reduced. The suggested mechanism also increases the diffusion intensity and a desired diffusion effect can be achieved with only two rounds of overall permutation-substitution operation. Experimental results demonstrate that the proposed scheme has a satisfactory level of security.

Keywords: Image cipher · Permutation-substitution · Hénon map
Lü system · Plaintext-dependent keystream sequence

1 Introduction

Over the past decade, great concerns have been raised about the security of images transmitted over the Internet. A direct and obvious way to protect image data from unauthorized eavesdropping is to employ an encryption algorithm. Unfortunately, commonly used block ciphers, including DES and AES, characterized by high computational complexity are difficult to meet the increasing demand for real-time communications when dealing with digital images characterized by bulk data capacity. To meet this challenge, many different encryption technologies have been proposed. Among them, the chaos-based technology has suggested a promising direction. In 1998, Fridrich suggested an iterative permutation-substitution model for construction of secure image ciphers [1]. In each round of the cipher, the pixel positions are firstly scrambled in a pseudorandom manner, which leads to a great reduction in the

G. Wang et al. (Eds.): SpaCCS 2017, LNCS 10656, pp. 432–445, 2017.
https://doi.org/10.1007/978-3-319-72389-1_35

correlation among neighboring pixels. Then, the pixel values are altered sequentially and the influence of each pixel is diffused to all its succeeding ones during the modification process. The whole permutation-substitution operation is often iterated for several rounds to ensure the influence of each individual pixel can be spread over the entire cipher-image.

Conventionally, three area-preserving invertible chaotic maps, i.e., the cat map, the baker map, and the standard map, are widely used for image scrambling. Unfortunately, this kind of permutation strategy suffers from two main disadvantages, i.e., the periodicity of discretized version of chaotic maps and only applicable to square images [2–4]. To address these two drawbacks, Fu et al. [5] suggested an image scrambling scheme using a chaotic sequence sorting mechanism. In [6], inspired by the natural ripple-like phenomenon that distorts a reflection on a water surface, Wu et al. suggested a novel scrambling algorithm that shuffle images in an n dimensional (n D) space using wave perturbations.

Recently, it has been reported that many existing image encryption schemes have been successfully broken by using known/chosen-plaintext attacks [7–10]. This is due to the fact that the substitution keystream sequences used in these schemes is solely determined by the secret key. That is, the same keystream sequence is used to encrypt different plain-images unless a different secret key is used. Consequently, the keystream sequence may be determined by encrypting some specially created images (e.g. an image with all pixels having the same value) and then comparing them with their corresponding outputs. Obviously, if a keystream sequence depends on both the secret key and the plaintext, then such analysis may become impractical. For instance, in [11], the keystream elements are extracted from multiple times iteration of the logistic map, and the iteration times are determined by plain-pixel values. In [12, 13], the authors introduced a mechanism that dynamically alters the value of control parameter of the chaotic map under the control of plain-pixel values. Unfortunately, the above mechanisms are implemented during the keystream generation process. This means a new keystream sequence should be produced for each round of substitution operation, thereby increasing the computational cost. To address this problem, this paper suggests a new mechanism that dynamically alters the values of the keystream elements under the control of plain-subpixel values during the subpixel values mixing process. As the keystream generation process has no dependency on the plain-image, the keystream sequence can be reused among different rounds of substitution operation. The suggested mechanism also increases the diffusion intensity and a desired diffusion effect can be obtained with only two rounds of overall permutation-substitution operation. In the permutation stage, the positions of subpixel in each color channel of the plain-image are scrambled across the entire color space using a pixel-swapping strategy under the control of a keystream sequence generated from the Hénon map. Experimental results demonstrate that the proposed scheme has a satisfactory level of security.

The remainder of this paper is organized as follows. The proposed color image encryption algorithm is described in detail in Sect. 2. In Sect. 3, the diffusion performance of the proposed cryptosystem is analyzed. In Sect. 4, we analyze the security of the proposed cryptosystem through various statistical analyses, key space analysis and key sensitivity analysis. Finally, conclusions are drawn in the last section.

2 The Image Encryption Scheme

As aforementioned, the chaotic Hénon map and Lü system are employed in our scheme to generate the permutation and substitution keystreams for image data scrambling and masking. A brief introduction of the two models is given below.

The Hénon map [14], introduced by Michel Hénon as a simplified model of the Poincaré section of the Lorenz model, is one of the most studied examples of dynamical systems that exhibit chaotic behavior. The Hénon map takes a point (x_n, y_n) in the plane and maps it to a new point, as described by

$$\begin{cases} x_{n+1} = y_n + 1 - ax_n^2, \\ y_{n+1} = bx_n, \end{cases} \tag{1}$$

where a and b are two parameters. The map is chaotic with classical parameter values $a = 1.4$ and $b = 0.3$. Evidently, the initial conditions (x_0, y_0) of the map are the immediate candidate for the secret key for permutation, as they uniquely determine a chaotic trajectory from which the permutation keystream is extracted.

Mathematically, the Lü system [15] is described by

$$\begin{cases} \dot{x} = a(y - x), \\ \dot{y} = -xz + cy, \\ \dot{z} = xy - bz, \end{cases} \tag{2}$$

where a, b and c are real parameters. Numerical experience shows that the system exhibits chaotic behavior when $a = 36$, $b = 3$ and $c \in (12.7, 17.0) \cup (18.0, 22.0) \cup (23.0, 28.5) \cup (28.6, 29.0) \cup (29.334, 29.345)$. Similarly, the initial conditions (x_0, y_0, z_0) of the system are used as the secret key for substitution.

Without loss of generality, a 24-bit RGB color image of size $W \times H$ is used as an input. The detailed encryption process is described as follows:

Step 1: Load the subpixel data of the input image to a 2-D byte matrix

$$imgData = \begin{bmatrix} p_0 & p_1 & \cdots & p_{3\times W-1} \\ p_{3\times W} & p_{3\times W+1} & \cdots & p_{3\times W\times 2-1} \\ \cdots & \cdots & \cdots & \cdots \\ p_{3\times W\times (H-1)} & p_{3\times W\times (H-1)+1} & \cdots & p_{3\times W\times H-1} \end{bmatrix}$$

Step 2: Generate a chaotic matrix *chaoMat* of the same size as that of *imgData* with each element consisting of two floating-point numbers obtained by iterating map (1).

Step 2.1: Pre-iterate map (1) for T_0 times to avoid the harmful effect of transitional procedure, where T_0 is a constant.

Step 2.2: Continue the iteration for $3 \times W \times H$ times. For each iteration, the current values of the two state variables x and y are stored into a matrix *chaoMat* =

$$\begin{bmatrix} (cm_{x(0)}, cm_{y(0)}) & (cm_{x(1)}, cm_{y(1)}) & \cdots & (cm_{x(3\times W-1)}, cm_{y(3\times W-1)}) \\ (cm_{x(3\times W)}, cm_{y(3\times W)}) & (cm_{x(3\times W+1)}, cm_{y(3\times W+1)}) & \cdots & (cm_{x(3\times W\times 2-1)}, cm_{y(3\times W\times 2-1)}) \\ \cdots & \cdots & \cdots & \cdots \\ (cm_{x[3\times W\times (H-1)]}, cm_{y[3\times W\times (H-1)]}) & (cm_{x[3\times W\times (H-1)+1]}, cm_{y[3\times W\times (H-1)+1]}) & \cdots & (cm_{x(3\times W\times H-1)}, cm_{y(3\times W\times H-1)}) \end{bmatrix}$$

as an element in the order from left to right, top to bottom.

Step 3: Extract a permutation matrix *permMat* =

$$\begin{bmatrix} (pm_{x(0)}, pm_{y(0)}) & (pm_{x(1)}, pm_{y(1)}) & \cdots & (pm_{x(3\times W-1)}, pm_{y(3\times W-1)}) \\ (pm_{x(3\times W)}, pm_{y(3\times W)}) & (pm_{x(3\times W+1)}, pm_{y(3\times W+1)}) & \cdots & (pm_{x(3\times W\times 2-1)}, pm_{y(3\times W\times 2-1)}) \\ \cdots & \cdots & \cdots & \cdots \\ (pm_{x[3\times W\times (H-1)]}, pm_{y[3\times W\times (H-1)]}) & (pm_{x[3\times W\times (H-1)+1]}, pm_{y[3\times W\times (H-1)+1]}) & \cdots & (pm_{x(3\times W\times H-1)}, pm_{y(3\times W\times H-1)}) \end{bmatrix}$$

from *chaoMat* according to

$$\begin{cases} pm_{x(n)} = mod(sig(abs(cm_{x(n)}), \alpha), H), \\ pm_{y(n)} = mod(sig(abs(cm_{y(n)}), \alpha), 3 \times W), \end{cases} \tag{3}$$

where $abs(x)$ returns the absolute value of x, $sig(x, \alpha)$ returns the α most significant decimal digits of x, and $mod(x, y)$ divides x by y and returns the remainder of the division. An α value of 15 is recommended as all the state variables in our scheme are declared as double-precision type, which has 15 or 16 decimal places of accuracy.

Step 4: Generate a chaotic sequence of length $L_{cs} = 3 \times W \times H$ by iterating system (2).

Step 4.1: Pre-iterate system (2) for T_0 times for the same purpose mentioned above. The system can be numerically solved by using fourth-order Runge-Kutta method, as given by

$$\begin{cases} x_{n+1} = x_n + (h/6)(K_1 + 2K_2 + 2K_3 + K_4), \\ y_{n+1} = y_n + (h/6)(L_1 + 2L_2 + 2L_3 + L_4), \\ z_{n+1} = z_n + (h/6)(M_1 + 2M_2 + 2M_3 + M_4), \end{cases} \tag{4}$$

where

$$\begin{cases} K_j = a(y_n - x_n) \\ L_j = -x_n z_n + c y_n \quad (\text{with } j = 1) \\ M_j = x_n y_n - b z_n, \end{cases}$$

$$\begin{cases} K_j = a\left[(y_n + hL_{j-1}/2) - (x_n + hK_{j-1}/2)\right] \\ L_j = -(x_n + hK_{j-1}/2)(z_n + hM_{j-1}/2) + c(y_n + hL_{j-1}/2) \quad (\text{with } j = 2, 3) \\ M_j = (x_n + hK_{j-1}/2)(y_n + hL_{j-1}/2) - b(z_n + hM_{j-1}/2), \end{cases}$$

$$\begin{cases} K_j = a\big[(y_n + hL_{j-1}) - (x_n + hK_{j-1})\big] \\ L_j = -(x_n + hK_{j-1})(z_n + hM_{j-1}) + c(y_n + hL_{j-1}) \\ M_j = (x_n + hK_{j-1})(y_n + hL_{j-1}) - b(z_n + hM_{j-1}), \end{cases} \quad (\text{with } j = 4)$$

and the step h is chosen as 0.005.

Step 4.2: Continue the iteration for $W \times H$ times. For each iteration, the current values of the three state variables x, y and z are in turn stored into an array $subSeq = \{ss_0, ss_1, \ldots, ss_{3\times W\times H-1}\}$.

Step 5: Extract a substitution keystream $subKstr = \{sk_0, sk_1, \ldots, sk_{3\times W\times H-1}\}$ from *subSeq* according to

$$sk_n = mod[sig((abs(ss_n), \alpha), G_L], \tag{5}$$

where G_L is the number of gray levels in the input image (for a 24-bit RGB image, $G_L = 256$).

Step 6: Encipher the subpixel data of the input image, i.e. the matrix *imgData*, using iterative permutation-substitution operations.

Step 6.1: Scramble the subpixels in *imgData* according to the permutation matrix *permMat*, or more specifically, swap each subpixel p_n in *imgData* with another one located at $(pm_{x(n)}, pm_{y(n)})$ in the order from left to right, top to bottom.

Step 6.2: Mask the values of each scrambled subpixel in *imgData* in the order from left to right, top to bottom.

Step 6.2.1: Bit-wise rotate the currently applied keystream element sk_n to the right under the control of the value of the previously operated subpixel p_{n-1}, as described by Eq. (6).

$$\begin{aligned} sk_{n(new)} = [sk_n \ll (\log_2 G_L - mod(p_{n-1}, \log_2 G_L)] | \\ (sk_n \gg mod(p_{n-1}, \log_2 G_L)), \end{aligned} \tag{6}$$

where "$\ll s$" and "$\gg s$" denote a left and right shift by s bit, respectively, and "|" denotes a bit-wise OR operation. For the first subpixel p_0, the initial value p_{-1} can be set as a constant.

Step 6.2.2: Calculate the cipher-subpixel values according to Eq. (7).

$$c_n = sk_{n(new)} \oplus \big[mod\big((p_n + sk_{n(new)}), G_L\big)\big] \oplus c_{n-1}, \tag{7}$$

where p_n is the currently operated subpixel, c_n and c_{n-1} are the output and previous ciphered subpixels, respectively, and $\oplus$ performs bit-wise exclusive OR operation. Similarly, one may set the initial value c_{-1} as a constant.

Step 6.2.3: return to ***Step 6.2.1*** until the values of all the subpixels in *imgData* are mixed.

Step 7: Repeat ***Step 6*** until the influence of each individual subpixel is spread out over the entire cipher-image.

The decryption procedure is similar to that of the encryption process except that some steps are followed in a reversed order. Particularly, the inverse of Eq. (7) is given by

$$p_n = mod\left[\left(sk_{n(new)} \oplus c_n \oplus c_{n-1} + G_L - sk_{n(new)}\right), G_L\right]. \tag{8}$$

As can be seen from the above description of the proposed encryption algorithm, we associate the keystream sequence with the plain-image by bitwise rotating each keystream element before it is applied to a subpixel, and the number of bits to be rotated is determined by the original value of the previously operated subpixel. As the chaotic sequence generation procedure (*Step 4*) and the subsequent keystream sequence quantification procedure (*Step 5*) have no dependencies on the plain-image, the keystream sequence can be reused among different rounds of substitution operation. Moreover, as can be seen from Eq. (7), a subpixel is mixed with the keystream element as well as previous ciphered subpixel, where the latter contains the accumulated effect of all its previous subpixels values. In our scheme, by associating the keystream sequence with the plain-image, the influence of each individual subpixel also acts on keystream elements. As a result, the diffusion intensity is increased and a desired *UACI* performance can be achieved with fewer rounds of overall permutation-substitution operation.

3 Analysis of the Diffusion Performance of the Proposed Scheme

As aforementioned, the substitution procedure serves to spread the influence of individual plaintext bits over as much of the ciphertext as possible. This is of much importance because otherwise the cryptosystem will be vulnerable to chosen-plaintext attack. The differential analysis is the most common way to implement the chosen-plaintext attack. To do this, an opponent may firstly create two plain-images with only one-bit difference, and then encrypt the two images using the same secret key. By observing the differences between the two resulting cipher-images, some meaningful relationship between plain-image and cipher-image could be found out, and it further facilitates in determining the keystream. Obviously, this kind of cryptanalysis may become impractical if a cryptosystem is highly sensitive to plaintext, i.e. changing one bit of the plaintext affect every bit in the ciphertext.

To measure the diffusion property of an image cryptosystem, two criteria, i.e., *NPCR* (the number of pixel change rate) and *UACI* (the unified average changing intensity) are commonly used. The *NPCR* is used to measure the percentage of different pixel numbers between two images. Let $I_1(i, j, k)$ and $I_2(i, j, k)$ be the (i, j)th pixel in kth color channel (k = 1, 2, 3 denotes the red, green, and blue color channels, respectively) of two images I_1 and I_2, the *NPCR* can be defined as:

$$NPCR = \frac{\sum_{k=1}^{3}\sum_{i=1}^{H}\sum_{j=1}^{W} D(i,j,k)}{3 \times H \times W} \times 100\%, \tag{9}$$

where $D(i, j, k)$ is defined as

$$D(i,j,k) = \begin{cases} 0 & \text{if } I_1(i,j,k) = I_2(i,j,k), \\ 1 & \text{if } I_1(i,j,k) \neq I_2(i,j,k). \end{cases} \tag{10}$$

The second criterion, *UACI* is used to measure the average intensity of differences between the two images. It is defined as

$$UACI = \frac{1}{3 \times H \times W}\left[\sum_{k=1}^{3}\sum_{i=1}^{H}\sum_{j=1}^{W} \frac{|I_1(i,j,k) - I_2(i,j,k)|}{G_L - 1}\right] \times 100\%. \tag{11}$$

Clearly, no matter how similar the two input images are, a good image cryptosystem should procedure outputs with *NPCR* and *UACI* values ideally being equal to that of two random images, which are given by

$$NPCR_{\exp ected} = \left(1 - \frac{1}{2^{\log_2 G_L}}\right) \times 100\% \tag{12}$$

and

$$UACI_{\exp ected} = \frac{1}{G_L^2}\left(\frac{\sum_{i=1}^{G_L-1} i(i+1)}{G_L - 1}\right) \times 100\%. \tag{13}$$

For instance, the *NPCR* and *UACI* values for two random color images in 24-bit RGB format are 99.609% and 33.464%, respectively.

The *NPCR* and *UACI* of the proposed cryptosystem are evaluated using five standard 24-bit color test images of size 512 × 512 taken from the USC-SIPI image database. The differential images are created by randomly changing 1-bit in the original ones, as listed in Table 1. The two images in each test pair are encrypted using the same secret key, and their *NPCR* and *UACI* values under different number of cipher rounds are calculated and compared with that of the conventional scheme, as listed in Tables 2 and 3, respectively. As can be seen from Tables 2 and 3, though both the proposed and the conventional schemes take two rounds to obtain a desired *NPCR* value, one more round is needed by the conventional scheme to obtain a desired *UACI* value. Therefore, the proposed substitution strategy provides superior computational efficiency.

4 Security Analysis

In this section, thorough security analysis has been carried out, including the most important ones like brute-force analysis, statistical analysis and key sensitivity analysis, to demonstrate the high security of the proposed scheme.

Table 1. Differential images used in *NPCR* and *UACI* tests

Test image name	Color component	Pixel position (x, y)	Pixel value	
			Original	Modified
F16	R	(160, 271)	193	192
House	B	(43, 118)	130	129
Montreal	R	(309, 135)	3	4
Peppers	B	(249, 410)	131	132
Sailboat	G	(469, 406)	187	186

Table 2. Results of *NPCR* test

Test image name	No. of cipher rounds (proposed scheme)			No. of cipher rounds (conventional scheme)		
	1	2	3	1	2	3
F16	0.32416	**0.99605**	0.99605	0.32416	**0.99634**	0.99609
House	0.61242	**0.99600**	0.99608	0.61242	**0.99622**	0.99613
Montreal	0.62026	**0.99612**	0.99600	0.62026	**0.99620**	**0.99613**
Peppers	0.64716	**0.99610**	0.99616	0.64716	**0.99598**	**0.99606**
Sailboat	0.34846	**0.99603**	0.99609	0.34846	**0.99625**	**0.99607**

Table 3. Results of *UACI* test

Test image name	No. of cipher rounds (proposed scheme)			No. of cipher rounds (conventional scheme)		
	1	2	3	1	2	3
F16	0.04068	**0.33441**	0.33477	0.00508	0.33612	**0.33461**
House	0.30731	**0.33446**	0.33493	0.00480	0.33718	**0.33447**
Montreal	0.07786	**0.33477**	0.33397	0.00973	0.33615	**0.33472**
Peppers	0.32461	**0.33440**	0.33444	0.00253	0.33270	**0.33420**
Sailboat	0.17489	**0.33456**	0.33492	0.08755	0.33404	**0.33467**

4.1 Brute-Force Analysis

In cryptography, a brute-force attack is a cryptanalytic attack that attempts to break a cipher by systematically checking all possible keys until the correct one is found. A key should therefore be long enough that this line of attack is impractical – i.e., would take too long to execute. As mentioned above, the initial states of the Hénon map, (x_0, y_0), and the Lü system, (x_0, y_0, z_0), are used as the permutation and substitution keys, respectively. As aforementioned, all the state variables in our algorithm are declared as 64-bit double-precision type, which gives 53 bits of precision. The two keys are independent of each other, and therefore the key length of the proposed scheme is $53 \times 5 = 265$ bits. Generally, cryptographic algorithms use keys with a length greater than 100 bits are considered to be "computational security" as the number of operations required to try all possible 2^{100} keys is widely considered out of reach for conventional digital computing techniques for the foreseeable future. Therefore, the proposed scheme is secure against brute-force attack.

4.2 Statistical Analysis

Frequency distribution of pixel values. A good image cryptosystem should flatten the frequency distribution of cipher-pixels values as such information has the potential to be exploited in a statistical attack. The frequency distribution of pixel values in an image can be easily explored by using histogram analysis. An image histogram is a graph showing the number of pixels in an image at each different intensity value found in that image. The histograms of the three color channels of the "Montreal" test image and its output cipher-image produced by the proposed scheme are shown in Fig. 1. It's clear from Figs. 1(l)–(n) that the pixel values of all the three color components of the output cipher-image are fairly evenly distributed over the whole intensity range, and therefore no information about the plain-image can be gathered through histogram analysis.

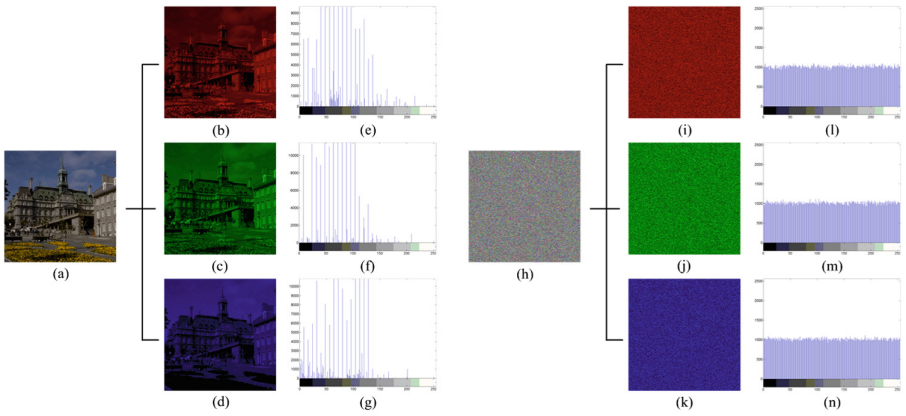

Fig. 1. Histogram analysis. (a) and (h) are the test image and its output cipher-image, respectively. (b)–(d) and (i)–(k) are the three color channels of (a) and (h), respectively. (e)–(g) and (l)–(n) are the histograms of (b)–(d) and (i)–(k), respectively.

The distribution of pixel values can be further quantitatively determined by calculating the information entropy of the image. Information entropy, introduced by Claude E. Shannon in his classic paper "A Mathematical Theory of Communication", is a key measure of the randomness or unpredictability of information content. The information entropy is usually expressed by the average number of bits needed to store or communicate one symbol in a message, as described by

$$H(S) = -\sum_{i=1}^{N} P(s_i) \log_2 P(s_i), \tag{14}$$

where S is a random variable with N outcomes $\{s_1,\ldots, s_N\}$ and $P(s_i)$ is the probability mass function of outcome s_i. It's obvious from Eq. (14) that the entropy for a random source emitting N symbols is $\log_2 N$. For instance, for a ciphered image with 256 color levels per channel, the entropy should ideally be 8, otherwise there exists certain degree of predictability which threatens its security.

The information entropies of above five test images and their output cipher-images are calculated, and the results are listed in Table 4. As can been seen from Table 4, the entropy of all the output cipher-images are very close to the theoretical value of 8. This means the proposed scheme produces outputs with perfect randomness and hence is robust against frequency analysis.

Table 4. Information entropies of the test images and their output cipher-images.

Test image name	Plain-image	Cipher-image
F16	6.663908	7.999784
House	7.485787	7.999748
Montreal	4.826442	7.999748
Peppers	7.669826	7.999757
Sailboat	7.762170	7.999741

Correlation between neighboring pixels. The simplest way to investigate the relationship between two neighboring pixels in an image is to use a scatter plot, which is carried out as follows. First, randomly select S_n pairs of neighboring pixels in each direction from the image. Then, the selected pairs is displayed as a collection of points, each having the value of one pixel determining the position on the horizontal axis and the value of the other pixel determining the position on the vertical axis.

Figures 2(a)–(c) and (d)–(f) show the scatter diagrams for horizontally, vertically and diagonally neighboring pixels in the red channel of the "Montreal" test image and its output cipher-image with S_n = 5000, respectively. Similar results can be obtained for the other two color channels. As can be seen from Fig. 2, most points in (a)–(c) are clustered around the main diagonal, whereas those in (d)–(f) are fairly evenly distributed. The results indicate that the proposed scheme can effectively eliminate the correlation between neighboring pixels in an input image.

To further quantitatively measure the correlation between neighboring pixels in an image, the correlation coefficients r_{xy} for the sampled pairs are calculated according to the following three formulas:

$$r_{xy} = \frac{\frac{1}{S_n}\sum_{i=1}^{S_n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\left(\frac{1}{S_n}\sum_{i=1}^{S_n}(x_i - \bar{x})^2\right)\left(\frac{1}{S_n}\sum_{i=1}^{S_n}(y_i - \bar{y})^2\right)}}, \tag{15}$$

$$\bar{x} = \frac{1}{S_n}\sum_{i=1}^{S_n} x_i, \tag{16}$$

$$\bar{y} = \frac{1}{S_n}\sum_{i=1}^{S_n} y_i, \tag{17}$$

where x_i and y_i form the ith pair of neighboring pixels.

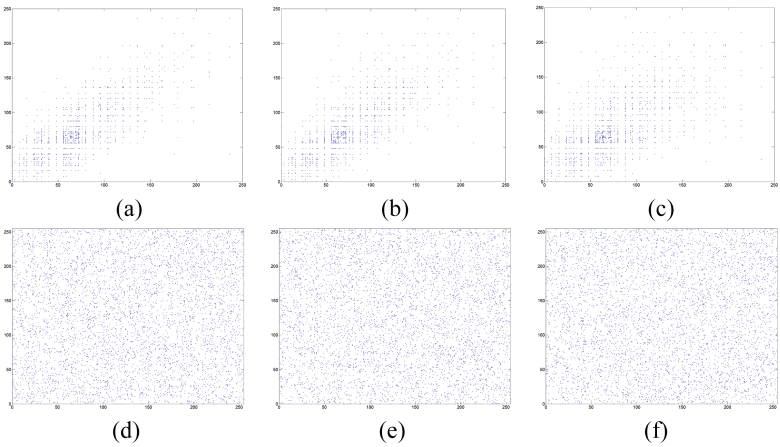

Fig. 2. Graphical analysis for correlation of neighboring pixels. (a)–(c) and (d)–(f) are scatter diagrams for horizontally, vertically and diagonally neighboring pixels in the red channel of the "Montreal" test image and its output cipher-image, respectively. (Color figure online)

Table 5 lists the calculated correlation coefficients for neighboring pixels in the three color channels of the five test images and their output cipher-images. As can be seen from Table 5, the correlation coefficients for neighboring pixels in all the three color channels of the output cipher-images are very close to zero, and it further supports the conclusion drawn from Fig. 2.

Table 5. Correlation coefficients for neighboring pixels in the test images and their output cipher-images.

Test image name	Direction	Plain-image			Cipher-image		
		R	G	B	R	G	B
F16	Horizontal	0.9516	0.9703	0.9290	–0.0054	0.0141	0.0045
	Vertical	0.9735	0.9565	0.9636	–0.0031	–0.0156	–0.0273
	Diagonal	0.9271	0.9350	0.9109	0.0176	–0.0029	–0.0032
House	Horizontal	0.9575	0.9463	0.9691	0.0186	–0.0101	–0.0144
	Vertical	0.9513	0.9368	0.9735	–0.0216	0.0149	–0.0028
	Diagonal	0.9196	0.8933	0.9468	–0.0196	0.0045	0.0321
Montreal	Horizontal	0.8927	0.8862	0.9517	0.0263	0.0086	–0.0004
	Vertical	0.8870	0.8808	0.9418	–0.0204	–0.0016	0.0040
	Diagonal	0.7994	0.8024	0.9117	0.0130	0.0056	0.0045
Peppers	Horizontal	0.9691	0.9755	0.9674	–0.0064	–0.0027	–0.0247
	Vertical	0.9656	0.9824	0.9666	–0.0091	0.0072	–0.0116
	Diagonal	0.9583	0.9644	0.9465	–0.0116	–0.0333	–0.0074
Sailboat	Horizontal	0.9537	0.9677	0.9698	0.0034	0.0006	–0.0227
	Vertical	0.9558	0.9696	0.9704	0.0065	0.0056	–0.0125
	Diagonal	0.9422	0.9530	0.9515	0.0030	–0.0063	–0.0214

Table 6. Decryption keys used for key sensitivity test

Figure	Decryption key	
	Permutation part	Substitution part
3(b)	$x_0 = 0.825201589901473$ $y_0 = -0.206378406313441$	$x_0 = -5.11799279781735$, $y_0 = 7.69311459457213$ $z_0 = 25.4568382928529$
3(c)	$\boldsymbol{x_0 = 0.825201589901472}$ $y_0 = -0.206378406313441$	$x_0 = -5.11799279781735$, $y_0 = 7.69311459457213$ $z_0 = 25.4568382928529$
3(d)	$x_0 = 0.825201589901473$ $\boldsymbol{y_0 = -0.206378406313442}$	$x_0 = -5.11799279781735$, $y_0 = 7.69311459457213$ $z_0 = 25.4568382928529$
3(e)	$x_0 = 0.825201589901473$ $y_0 = -0.206378406313441$	$\boldsymbol{x_0 = -5.11799279781736}$, $y_0 = 7.69311459457213$ $z_0 = 25.4568382928529$
3(f)	$x_0 = 0.825201589901473$ $y_0 = -0.206378406313441$	$x_0 = -5.11799279781735$, $\boldsymbol{y_0 = 7.69311459457212}$ $z_0 = 25.4568382928529$
3(g)	$x_0 = 0.825201589901473$ $y_0 = -0.206378406313441$	$x_0 = -5.11799279781735$, $y_0 = 7.69311459457213$ $\boldsymbol{z_0 = 25.4568382928528}$

4.3 Key Sensitivity Analysis

To evaluate the key sensitivity property of the proposed scheme, the "Montreal" test image is firstly encrypted using a randomly generated secret key: Hénon map with initial conditions ($x_0 = 0.825201589901473$, $y_0 = -0.206378406313441$) and Lü system with initial condition ($x_0 = -5.11799279781735$, $y_0 = 7.69311459457213$,

z_0 = 25.45683 82928529), and the resulting cipher-image is shown in Fig. 3(a). Then the cipher image is tried to be decrypted using six decryption keys, one of which is exactly the same as the encryption key and the other five have only one-bit difference to it, as listed in Table 6. The resulting deciphered images are shown in Figs. 3(b–g), respectively, from which we can see that even an almost perfect guess of the key does not reveal any information about the original image. It can, therefore, be concluded that the proposed scheme fully satisfies the key sensitivity requirement.

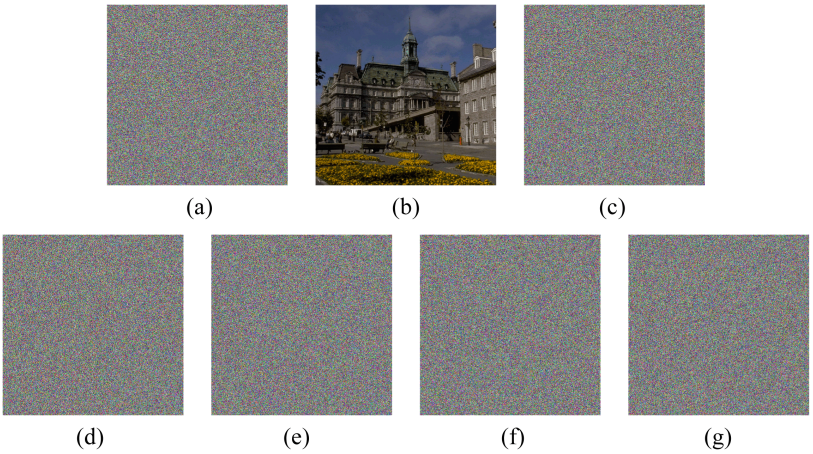

Fig. 3. Results of key sensitivity test

5 Conclusions

This paper has proposed a new permutation-substitution type color image cipher based on chaotic Hénon Map and Lü System. To confuse the relationship between the ciphertext and the secret key, the positions of colored subpixels in the input image are scrambled using a pixel-swapping mechanism, which avoids the two main problems encountered when using the discretized version of area-preserving chaotic maps. To strengthen the robustness of the substitution procedure against chosen-plaintext attack, we introduced a new mechanism for associating the keystream with the plain-image by dynamically altering the values of the keystream elements under the control of plain-subpixels values during the subpixel values mixing process. Compared with other related mechanisms, the proposed mechanism allows the keystream sequence to be reused among different rounds of substitution operation as keystream generation process has no dependency on the plain-image. The proposed mechanism also helps increase the diffusion intensity and the experimental results indicate that the proposed scheme takes only two cipher rounds to achieve both desired *NPCR* and *UACI* values. We have carried out an extensive security analysis, which demonstrates the satisfactory security level of the new scheme. It can therefore be concluded that the proposed scheme provides a good candidate for online secure image communication applications.

Acknowledgments. This work was supported by the Fundamental Research Funds for the Central Universities (No. N150402004), and the Online Education Research Fund of MOE Research Center for Online Education (Qtone Education) (No. 2016YB123).

References

1. Fridrich, J.: Symmetric ciphers based on two-dimensional chaotic maps. Int. J. Bifurc. Chaos **8**, 1259–1284 (1998)
2. Fu, C., Meng, W., Zhan, Y., et al.: An efficient and secure medical image protection scheme based on chaotic maps. Comput. Biol. Med. **43**, 1000–1010 (2013)
3. Chen, J., Zhu, Z., Fu, C., et al.: Reusing the permutation matrix dynamically for efficient image cryptographic algorithm. Signal Process. **111**, 294–307 (2015)
4. Wong, K.W., Kwok, B.S.H., Law, W.S.: A fast image encryption scheme based on chaotic standard map. Phys. Lett. A **372**, 2645–2652 (2008)
5. Fu, C., Lin, B., Miao, Y., et al.: A novel chaos-based bit-level permutation scheme for digital image encryption. Opt. Commun. **284**, 5415–5423 (2011)
6. Wu, Y., Zhou, Y., Agaian, S., et al.: A symmetric image cipher using wave perturbations. Signal Process. **102**, 122–131 (2014)
7. Li, C., Li, S., Lo, K.T.: Breaking a modified substitution–diffusion image cipher based on chaotic standard and logistic maps. Commun. Nonlinear Sci. Numer. Simul. **16**, 837–843 (2011)
8. Li, C., Zhang, L.Y., Ou, R., et al.: Breaking a novel colour image encryption algorithm based on chaos. Nonlinear Dyn. **70**, 2383–2388 (2012)
9. Li, C., Xie, T., Liu, Q., et al.: Cryptanalyzing image encryption using chaotic logistic map. Nonlinear Dyn. **78**, 1545–1551 (2014)
10. Li, C., Liu, Y., Zhang, L.Y., et al.: Breaking a chaotic image encryption algorithm based on modulo addition and XOR operation. Int. J. Bifurc. Chaos **23**, 1350075 (2013)
11. Wang, Y., Wong, K.W., Liao, X., et al.: A chaos-based image encryption algorithm with variable control parameters. Chaos, Solitons Fractals **41**, 1773–1783 (2009)
12. Fu, C., Chen, J., Zou, H., et al.: A chaos-based digital image encryption scheme with an improved diffusion strategy. Opt. Express **20**, 2363–2378 (2012)
13. Chen, J., Zhu, Z., Fu, C., et al.: An improved permutation-diffusion type image cipher with a chaotic orbit perturbing mechanism. Opt. Express **21**, 27873–27890 (2013)
14. Cvitanović, P., Gunaratne, G.H., Procaccia, I.: Topological and metric properties of Hénon-type strange attractors. Phys. Rev. A **38**, 1503 (1988)
15. Lü, J., Chen, G.: A new chaotic attractor coined. Int. J. Bifurc. Chaos **12**, 659–661 (2002)