Nonlinear Dyn (2015) 79:2121–2131
DOI 10.1007/s11071-014-1798-y

ORIGINAL PAPER

A fast image encryption algorithm based on chaotic map and lookup table

Pingguang Cheng · Huaqian Yang · Pengcheng Wei · Wei Zhang

Received: 16 March 2014 / Accepted: 10 November 2014 / Published online: 21 November 2014

Abstract At present, a lot of image cryptosystems with permutation/diffusion architecture have been proposed. However, permutation and diffusion are considered as two separate stages, both requiring image-scanning to obtain pixel values. Moreover, because of extraction bits directly from the discrete state value of a chaotic map to generate the pseudorandom binary sequence, the quite time-consuming conversion from floating points to integers cannot be avoided in practical applications. In this paper, a novel image encryption scheme for both combining permutation–diffusion and avoiding conversion of floating-point number is proposed. Firstly, using the lookup table constructed and S-Box of AES, an efficient approach of generating the pseudorandom sequence required by diffusion is proposed. Then, the combined permutation/diffusion architecture is employed to shuffle and change the pixels. Theoretical analyses and computer simulations both confirm that the new algorithm has high security and is very fast for practical image encryption.

Keywords Image encryption · Cryptography · Information security · Chaotic map

P. Cheng (✉) · H. Yang · P. Wei · W. Zhang
Department of Mathematics and Info Engineering,
Chongqing University of Education,
Chongqing 400067, China
e-mail: mailtopartner@163.com

H. Yang
College of Computer Science and Engineering,
Chongqing University, Chongqing 400044, China

1 Introduction

With the rapid growth of image transmission through Internet, the secure transmission of confidential digital images over public channels has become a common interest in both research and application fields. Although some traditional ciphers such as DES and AES are designed with good permutation and diffusion properties, they are generally difficult to handle the image encryption because of some intrinsic properties of images such as bulk data volume and high pixel correlation. Nevertheless, many new image encryption schemes have been proposed in recent years, among which the chaos-based approach appears to be a promising direction [1–15].

In [2], Fridrich suggested that a chaos-based image encryption scheme should compose of the iteration of two processes: permutation and diffusion, namely permutation–diffusion architecture, as shown in Fig. 1.

The permutation stage permutes all the pixels as a whole, without changing their values. In the diffusion stage, the pixel values are modified sequentially so that a tiny change in a pixel spreads out to as many pixels in the cipher-image as possible. To eliminate the correlation between adjacent pixels, the whole permutation/diffusion process repeats for a number of times in order to achieve a satisfactory level of security. In Fig. 1, permutation and diffusion are two separate and iterative stages, and they both require scanning the image in order to obtain the pixel value. Thus, at least twice scanning the same image is required in each round

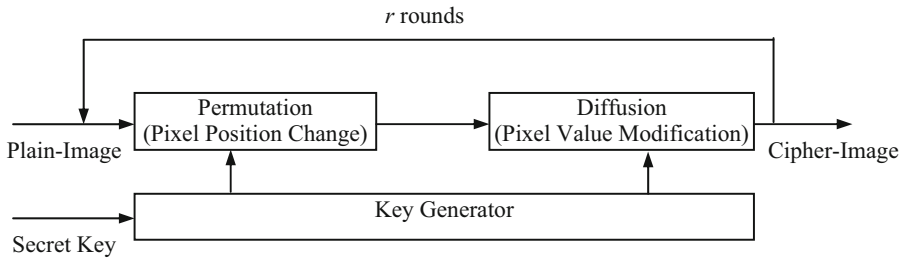

Fig. 1 Architecture of image encryption based on permutation and diffusion

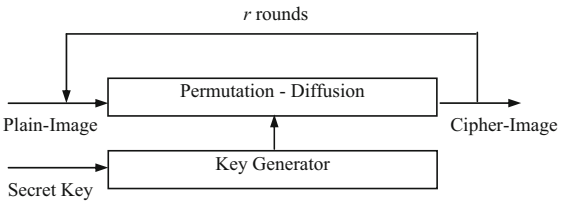

Fig. 2 Architecture of image encryption combining permutation/diffusion

of the permutation–diffusion operation. This scanning process is actually repeated but can be avoided if the permutation and diffusion operations be combined, i.e., via changing the values of the pixels while relocating them as illustrated in Fig. 2. As a result, the image only needs to be scanned once so that the encryption speed and efficiency is significantly improved.

On the other hand, some chaos-based image encryption algorithms with permutation–diffusion structure are also attacked [13,16–19,21,22].

The common flaws or deficiencies of these algorithms are summarized as follows:

(1) The keystream for encryption/decryption is independent of the plain-image, and this favors known-plaintext and chosen-plaintext attacks.
(2) In the diffusion process, the pseudorandom binary sequence is extracted directly from the discrete state value of a chaotic map. This means that the conversion from floating points to integers cannot be avoided in practical applications. However, it is found from computer simulations that the conversion process is quite time-consuming [5].

In [5], authors proposed a new fast image encryption. Although it can partly avoid to extract bits operation from the discrete state value of a chaotic map, $N^2/8$ times extracting operation cannot yet be avoided in each encryption round. In this paper, we propose a novel scheme for both combining permutation–diffusion and avoiding the conversion of floating-point number. Firstly, using the lookup table constructed and S-Box of AES, an efficient approach of generating the pseudorandom sequence required by diffusion is proposed. Then, the combined permutation/diffusion architecture is employed to shuffle and change the pixels.

The rest of this paper is organized as follows. In Sect. 2, the process of generating pseudorandom binary sequence is described in detail. Section 3 focuses on the description of the proposed fast image encryption. Performance and security of this scheme are analyzed in Sect. 4. In Sect. 5, a conclusion is drawn.

2 The keystream generator

2.1 Generating the pseudorandom sequences

To avoid the float-point operation of extracting in generating the pseudorandom number sequences, an approach of generating the pseudorandom sequence is proposed in this paper as shown in Fig. 3. The 1D chaotic tent map is defined by the following equation:

$$T_\alpha : x_j = \begin{cases} \frac{x_{j-1}}{\alpha}, & \text{if } \ 0 \le x_{j-1} \le \alpha \\ \frac{1-x_{j-1}}{1-\alpha}, & \text{if } \ \alpha < x_{j-1} \le 1 \end{cases} \tag{1}$$

where $0 < a < 1$. This function maps the interval [0, 1] onto itself with only one parameter a. A sequence formed by iterating T_α from an arbitrary initial point in (0, 1) exhibits chaotic properties [23–26] when T_α is expanding everywhere in the interval (0, 1).

The detail of generating the pseudorandom sequence is described as follow.

Step 1 Partition interval [0, 1] into 256 the same length subintervals sub_i, and $sub_i \in [i \cdot 2^{-8}, (i+1) \cdot 2^{-8}]$, $i = 0, 1, \ldots, 255$, and all subintervals form lookup table LT as shown in Fig. 4.

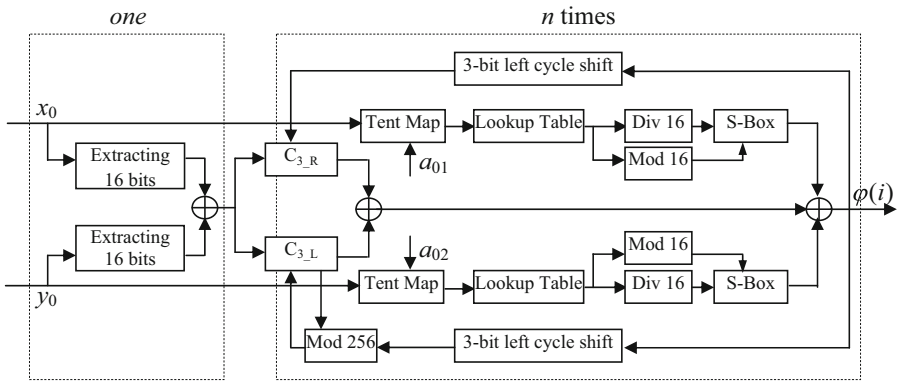

Fig. 3 Architecture of generating the pseudorandom sequences

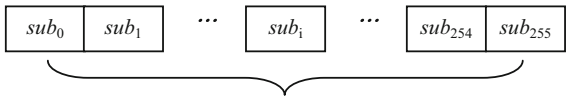

Fig. 4 Lookup table (LT)

Step 2 Extract 16 bits (1st to 16th bits after the decimal point) from initial values x_0, y_0 of tent maps, respectively, and be denoted as c_1 and c_2.

Step 3 XOR c_1 and c_2, left 8 bits of XOR result are denoted as c_{3_L}, and right 8 bits are denoted as c_{3_R}, respectively.

Step 4 Iterate the Eq. (1) once with two different initial values and control parameters, and get two state values x_i and y_i.

Step 5 Locate the subinterval index of x_i and y_i in lookup table LT and denote the index as j_1 and j_2, respectively

Step 6 Compute j_{1_1}, j_{1_2}, j_{2_1} and j_{2_2} as follow:

$$j_{1\_1} \leftarrow j_1 \text{ div } 16; \quad j_{1\_2} \leftarrow j_1 \bmod 16;$$
$$j_{2\_1} \leftarrow j_2 \text{ div } 16; \quad j_{2\_2} \leftarrow j_2 \bmod 16;$$

Step 7 Generate 8 bits pseudorandom sequence $\varphi(i)$ according to the following formula:

$$\varphi(i) \leftarrow \text{Sbox}\left[j_{1\_1}\right]\left[j_{1\_2}\right] \oplus \text{Sbox}\left[j_{2\_1}\right]\left[j_{2\_2}\right] \oplus c_{3\_L} \oplus c_{3\_R} \quad (2)$$

where the Sbox is S-box used in AES algorithm as in Fig. 5.

Step 8 : Performs operations as follow:

$$c_{3\_R} \leftarrow cycL(3, \varphi(i));$$
$$c_{3\_L} \leftarrow (c_{3\_L} + cycL(3, \varphi(i)) \bmod 256$$

where $cycL(x, y)$ denotes the x-bit left cyclic shift on the pseudorandom sequence y.

By repeating the operations Steps 4–8, a pseudorandom sequence with a desired length, $(\varphi(1), \varphi(2), \ldots, \varphi(i), \ldots, \varphi(n))$, is obtained.

2.2 Randomness of the generated sequence

The National Institute of Standards and Technology (NIST) provides 16 statistical tests to detect deviations of a binary sequence from randomness in SP800-22 document [27]. Each statistical test is formulated to test a specific *null hypothesis* (H_0). The null hypothesis under test is that the sequence being tested is *random*. The test statistic is used to calculate a p value that summarizes the strength of the evidence against the null hypothesis. For these tests, each p value is the probability that a perfect random number generator would have produced a sequence less random than the sequence that was tested, given the kind of non-randomness assessed by the test. A significance level (a) can be chosen for the tests. If p value $\geq$a, then the null hypothesis is accepted; i.e., the sequence appears to be random. If p value $<$a, then the null hypothesis is rejected; i.e., the sequence appears to be non-random. Typically, a is chosen in the range [0.001, 0.01]. In our experiment, 1,000 sequences, each of 1,000,000 bits, are generated using our scheme, and they all pass the statistic tests. The p values for various tests are listed in Table 1. In test, the initial values and control parameters of Eq. (1) are chosen randomly as $x_0 = 0.1345645961$, $y_0 = 0.9432234875$, $a_{01} = 0.4565625849$, $a_{02} = 0.2435724359$, respectively. If there is more than one statistical value in a test, the test is marked with an asterisk and the average value is computed.

Fig. 5 S-box in AES algorithm

S-box		c														
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
r	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	10	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	11	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	12	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	13	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	14	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	15	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Table 1 Statistical properties of randomness test

Statistical test	Freq	BlkFreq	CuSumFwd	CuSumRev	Runs	LineComp
p value	0.618385	0.056785	0.042255	0.794391	0.964295	0.492436
Pass rate (%)	99.30	98.60	99.20	99.30	98.70	99.00
Statistical test	LongRuns	Rank	DFFT	Nonp-Temp*	Overl-Temp	Univ
p value	0.837781	0.552383	0.955835	0.419417	0.987896	0.666245
Pass rate (%)	99.30	99.20	98.50	98.96	98.50	98.90
Statistical test	Apen	Rand-Exc*	Rand-Exc-V*	Serial 1	Serial 2	
p value	0.026057	0.459732	0.391315	0.035174	0.697257	
Pass rate (%)	99.00	99.00	98.79	99.00	99.20	

1,000 sequences of length 10^6 bits are tested. According to NIST documentation, a pass rate of 98.056 % is acceptable

3 The proposed encryption scheme

3.1 Permutation

Lian et al. pointed out that there exists some weak keys for ciphers employing the cat and the baker maps. Moreover, the key space of the two maps is not as large as that of the standard map. Therefore, they suggested using a standard map for permutation [3]. In our scheme, the discrete standard map is also employed to permute the image pixels.

To avoid the fixed point, namely the corner pixel ($s = 0,\ t = 0$), under the standard map, a random scan couple (r_s, r_t) is included to move this corner pixel together with other pixels. The modified standard map equations are given by Eq. (3).

$$\begin{cases} s_{k+1} = (s_k + t_k + r_s + r_t) \mod N \\ t_{k+1} = \left(t_k + r_t + K_c \sin \frac{N \cdot s_{k+1}}{2\pi}\right) \mod N \end{cases} \tag{3}$$

Here, (s_k, t_k) and (s_{k+1}, t_{k+1}) are the original and the permuted pixel position of an $N \times N$ image, respec-

tively. The standard map parameter K_c is a positive integer.

3.2 Encryption

The detailed encryption algorithm is described as follows:

Step 1 Randomly choose the secret keys x_0, a_{01}, y_0 and a_{02} as the initial values and control parameter in Eq. (1), respectively.

Step 2 Generate the r_s, r_t, K_c and $C(0)$ from x_0, a_{01}, y_0 and a_{02} using the following function, respectively:

$$r_s \leftarrow Bin2Int(b_1 b_2 \ldots b_{24});$$
$$r_t \leftarrow Bin2Int(b_1 b_2 \ldots b_{24});$$
$$K_c \leftarrow Bin2Int(b_1 b_2 \ldots b_{24});$$
$$C(0) \leftarrow Bin2Int(b_1 b_2 \ldots b_8)$$

where x_0, a_{01}, y_0 and a_{02} are represented in binary format 0. $b_1 b_2 b_3 \ldots b_{51} b_{52}$, respectively, $b_i \in \{0, 1\}$. b_i represents the ith bit after the decimal point. The IEEE 754 double precision floating-point format possesses 64-bit word length with a 52-bit fraction part, but only the 1st to 24th bits after the decimal point are chosen. The function *Bin2Int* (.) transforms a binary number to an integer.

Step 3 Permute the plain-image pixels using the modified standard map given by Eq. (3)

Step 4 Diffuse the permuted pixels using the scheme as followed:
To avoid known-plaintext and chosen-plaintext attacks, the pixel values are altered sequentially in the diffusion process so that the change made to a particular pixel depends on the accumulated effect of all the previous pixel values. Details of the diffusion operation are described below:

(i) Exchange the status values of two tent maps, if $C(0)$ is a odd number.

(ii) Generate a pseudorandom numbers $\varphi(i)$ (8 bits) as described in Sect. 2.1.

(iii) The cipher-pixel value is calculated from the value of the currently operated and the previously operated pixels, according to Eq. (4):

$$C(i) = \varphi(i) \oplus \{(P(i) + 2 \cdot \varphi(i) \bmod G\} \oplus C(i-1) \quad (4)$$

where $P(i)$ and $C(i)$ are the currently operated plain-image pixel and the cipher-image pixel, respectively. G is the total number of possible gray scales in the plain-image. $C(i-1)$ is the previous cipher-image pixel. $C(0)$ is a secret initial value derived from the key, as described by Step 2. The inverse form of Eq. (4) for decryption is given by:

$$P(i) = \{\varphi(i) \oplus C(i) \oplus C(i-1) + G - 2 \cdot \varphi(i)\} \bmod G \quad (5)$$

(iv) Exchange the status values of two tent maps, if $C(i)$ is an odd number, and return to (ii) until all plain-image pixels are processed.

It should be noticed that the permutation and diffusion processes are performed simultaneously in a single scan of image. The value is altered while relocating a pixel [4–6].

Step 5 Repeat Steps 3 and 4 for $R \geq 2$ rounds according to the security requirement. Notice that the cipher-image pixel of last pixel will be used as the $C(0)$ of next round. The more rounds are processed, the higher security the encryption will have, but at the expense of computational effort and time delay.

3.3 Decryption

Since the permutation and diffusion are performed simultaneously in a single scan of plain-image pixels, the decryption procedure is slightly different from the encryption one. Details are described as follows:

Step 1 From the x_0, a_{01}, y_0 and a_{02} received secretly from the sender, determine the values of the parameters r_s, r_t, K_c and $C(0)$ using the same bit assignment stated in Sect. 3.2.

Step 2 Permute the pixels of the cipher-image reversely to obtain an intermediate image.

Step 3 Perform the reverse operations in the intermediate image to remove the effect of diffusion. The detail operations are the same as those described in Sect. 3.3, except that Eq. (4) is replaced by Eq. (5).

Step 4 Repeat Steps 2 and 3 for R rounds.

4 Performance analysis

To evaluate and test the proposed algorithm, a series of experiments is conducted. In experiments, the image

Table 2 NPCR and UACI values at different encryption rounds and different pixel positions of the proposed

Image	Round = 1 (change in pixel)						Round = 2 (change in pixel)					
	(0, 0)		(123, 420)		(511,511)		(0, 0)		(123, 420)		(511, 511)	
	NPCR	UACI	NPCR	UACI	NPCR	UACI	NPCR	UACI	NPCR	UACI	NPCR	UACI
Babala	99.5956	33.4884	75.5020	25.3893	00.0004	00.0000	99.6002	33.4583	99.6048	33.4934	99.6132	33.5483
Lena	99.6193	33.3755	75.5306	25.3519	00.0004	00.0000	99.6132	33.5019	99.6101	33.4469	99.6132	33.4012

for testing is the standard 512 × 512 image with 8-bit grayscale, and the initial values and controls of two tent maps are chosen randomly as $x_0 = 0.1345645961$, $y_0 = 0.9432234875$, $a_{01} = 0.4565625849$, $a_{02} = 0.2435724359$.

4.1 Key space analysis

To resist the brute-force attack, the key space of any encryption algorithms should be sufficiently large. In the proposed image encryption algorithm, the four secret keys x_0, a_{01}, y_0 and a_{02}, which are the initial values and control parameter of two tent maps, are used to generate the pseudorandom sequences which are then employed for encryption and decryption. So, the key space is composed of x_0, a_{01}, y_0 and a_{02}. If the state value of all chaotic maps is represented by the IEEE 754 double precision floating-point standard, the key space is much larger than 2^{128}. This is enough large for the general requirement of resisting brute-force attack.

4.2 Differential attack

To resist differential attack, any tiny modification in the plain-image should cause a significant difference in the cipher-image. Two measures are usually employed to measure this capability quantitatively: *number of pixels change rate* (NPCR) and *unified average changing intensity* (UACI). They are defined as follows [3–11]:

$$\mathrm{NPCR} = \frac{\sum_{r,c} D\,(r,c)}{W \times H} \times 100\,\% \tag{6}$$

$$\mathrm{UACI} = \frac{1}{W \times H}\left[\sum_{r,c} \frac{|C_1\,(r,c) - C_2\,(r,c)|}{255}\right] \times 100\,\% \tag{7}$$

where $C_1(r,\ c)$ and $C_2(r,\ c)$ are the grayscale values of the pixels at position $(r,\ c)$ of C_1 and C_2, respectively, C_1 and C_2 are the two cipher-images whose corresponding plain-images have only one-pixel difference. W and H are the width and height of the image, respectively. The element $D(r,\ c)$ is determined by $C_1(r,\ c)$ and $C_2(r,\ c)$. Namely, if $C_1(r,\ c) = C_2(r,\ c)$, then $D(r,\ c) = 0$; otherwise, it is 1. The values of these two quantitative measures (NPCR and UACI) for our algorithm are listed in Table 2.

Experimental results show that the proposed cryptosystem only needs a minimum of two rounds to achieve a high performance such as NPCR > 0.995 and UACI > 0.333. Therefore, the proposed algorithm can resist the differential attack if Round ≥ 2.

4.3 Key sensibility analysis

An ideal cryptosystems should be sensitive to key. This means that tiny change in the key results in a completely different encrypted image when applied to the same plain-image. Key sensitivity analysis has been performed for the proposed image encryption algorithm. To evaluate the key sensitivity of our algorithm, one of secret keys x_0 is changed from 0.1345645961 to 0.1345645962, denoted as x_0', and the encryption is repeated. The two corresponding cipher-images are compared, and a 99.62 % difference in pixel values is found. The results are depicted in Fig. 6, which show that our proposed algorithm is sensitive to the key even for a difference as tiny as 10^{-10}.

4.4 Statistical analysis

According to Shannon's theory, a secure cryptographic scheme should be strong enough to resist any statistical attack. In order to prove the security of the proposed

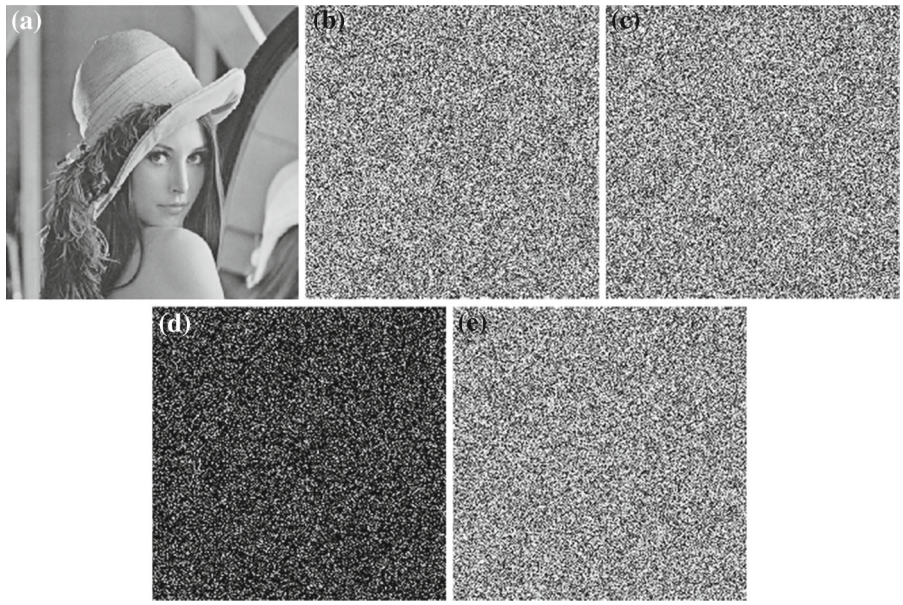

Fig. 6 Key sensitivity test: **a** plain-image, **b** cipher-image using key $x_0,\ y_0,\ a_{01},\ a_{02}$, **c** cipher-image using key $x_0',\ y_0,\ a_{01},\ a_{02}$, **d** difference image between the two cipher-images, **e** decrypted image form (**b**) using a slightly modified key $x_0',\ y_0,\ a_{01},\ a_{02}$

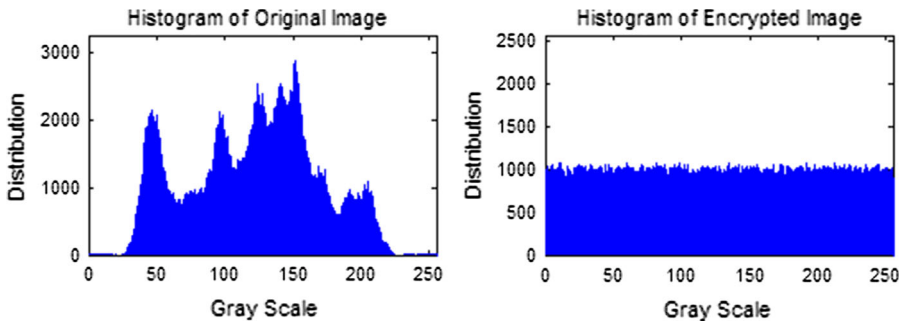

Fig. 7 Histograms of original image and encrypted image

image encryption scheme, the following statistical tests are performed.

(1) Histograms of the plain-image and the cipher-image.

Histograms of the plain-image and the cipher-image are shown in Fig. 7. As shown in this figure, the latter histogram is fairly uniform and significantly different from the histograms of the plain-image image. Hence, it does not reveal any statistical information of the former.

(2) Correlation of two adjacent pixels.

A secure encryption scheme should remove the correlation between adjacent image pixels in order to improve the resistance against statistical analysis. To test the correlation between two adjacent pixels in vertical, horizontal and diagonal directions of a cipher-image, respectively, the following procedures are carried out. First, randomly select 10,000 pairs of two adjacent image pixels in the corresponding direction. Then, calculate the correlation coefficient of each pair using the following formula:

$$\begin{aligned} \operatorname{cov}(x, y) &= E\left[(x - E(x))(y - E(y))\right] \\ &= \frac{1}{N}\sum_{i=1}^{N}\left[(x_i - E(x))(y_i - E(y))\right] \end{aligned} \tag{8}$$

$$r_{xy} = \frac{\operatorname{cov}(x, y)}{\sqrt{D(x)} \cdot \sqrt{D(y)}} \tag{9}$$

where x and y are grayscale values of two adjacent pixels in the image, $E(x) = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $D(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2$. In each test, $N = 10{,}000$. The correlation distributions of two adjacent pixels in the plain-image and the cipher-image are shown in Fig. 8, respectively. The measured correlation coefficients of the plain-image are close to 1, while those of the cipher-

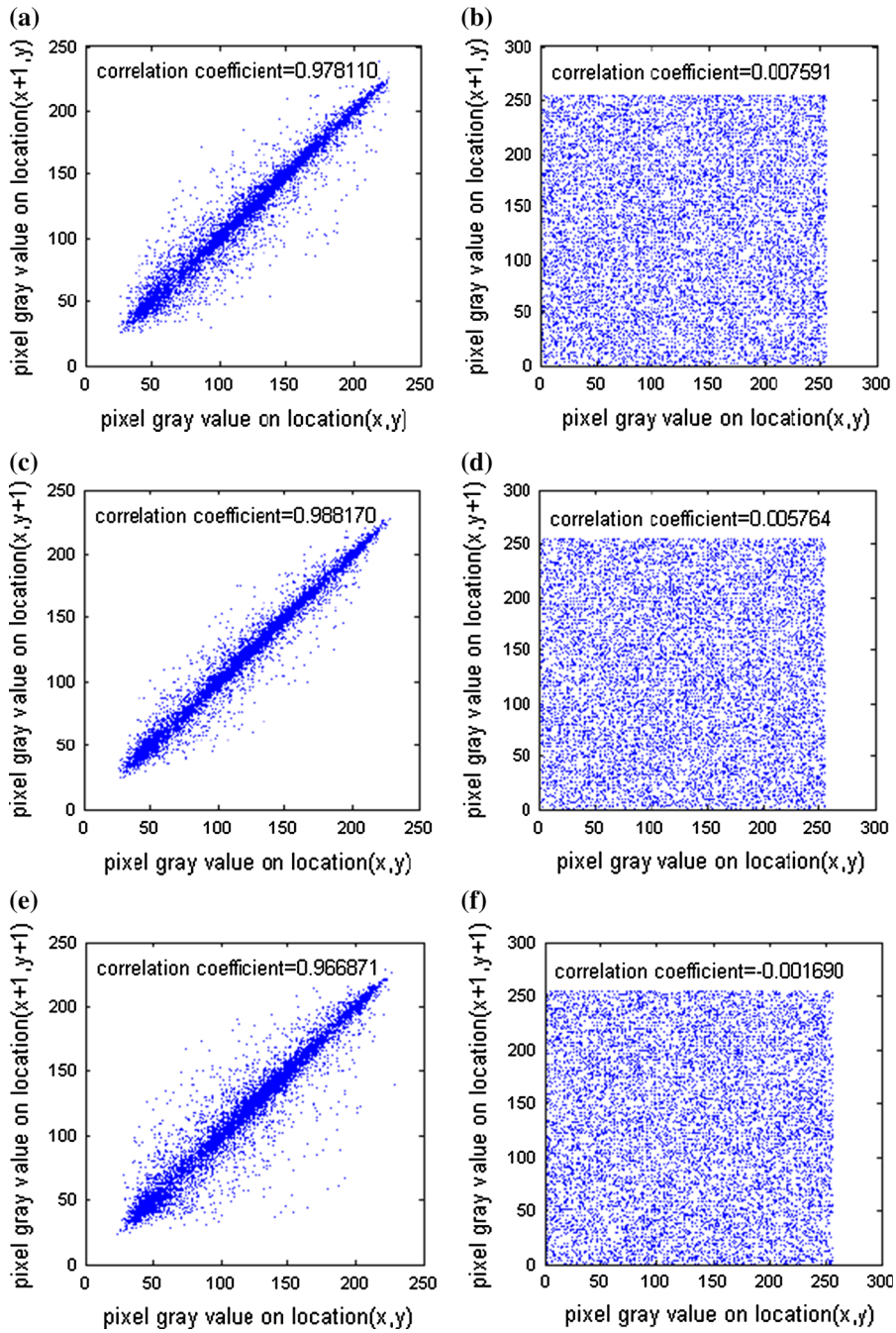

Fig. 8 Correlations of two adjacent pixels in **a** horizontal direction of the plain-image, **b** horizontal direction of the cipher-image, **c** vertical direction of the plain-image, **d** vertical direction of the cipher-image, **e** diagonal direction of the plain-image, **f** diagonal direction of the cipher-image

image are nearly 0. This indicates that the proposed algorithm has successfully removed the correlation of adjacent pixels in the plain-image so that neighbor pixels in the cipher-image virtually have no correlation. Therefore, the proposed algorithm possesses high security against statistical attacks.

4.5 Information entropy analysis

Information entropy, such as K–S, is the most outstanding feature of the randomness. It is well known that the entropy $H(s)$ of a message source s can be measured by

Table 3 Results of information entropy of plain-image, cipher-image and cipher-image with one-pixel change in plain-image

Image	Plain-image	Cipher-images (plain pixel changed in pixel)			
		(no change)	(0, 0)	(123, 420)	(511, 511)
Babala	7.6321	7.9992	7.9992	7.9993	7.9993
Lena	7.4295	7.9994	7.9992	7.9993	7.9993

Table 4 The round number of scanning-image, permutation and diffusion and extracting times of per round to achieve NPCR $>$ 0.996 and UACI $>$ 0.334

	Round number of scanning-image	Round number of permutation	Round number of diffusion	Conversion times floating-points to integers of per round
Proposed	2	2	2	2
Lian et al. [6]	18	18	6	N^2
Wong et al. [7]	4	4	2	N^2
Wang et al. [xx]	2	2	2	$N^2/8$

$$H(s) = -\sum_{i=0}^{M-1} P(s_i) \log_2 P(s_i) \tag{10}$$

where M is the total number of symbols $s_i \in s$, $P(s_i)$ represents the probability of occurrence of symbol s_i. For a truly random source emitting 256 symbols, the ideal entropy is $H(s) = 8$. If the output of a cipher emits symbols with the entropy value of less than 8, there is a certain degree of predictability which threatens its security. $H(s)$ has been tested on the encrypted images. The results are shown in Table 3. Experiments results show that the cipher-images are close to a true random source and the proposed algorithm is secure against the entropy attack.

4.6 Resistance to known-plaintext and chosen-plaintext attacks

To resist the known-plaintext and chosen-plaintext attacks, two different plain-images should have different keystreams even if they are encrypted with identical keys. In the proposed algorithm, the status values of two tent maps are exchanged according to the previous pixel cipher value. Consequently, the next state value of maps is related to the plain-image. Since the pseudorandom number $\varphi(i)$, which is the keystream of the proposed algorithm, is related to the state values of maps, different images will have different $\varphi(i)$. It is difficult to decrypt a particular cipher-image using the keystream $\varphi(i)$ obtained from other images. Therefore, the proposed algorithm can well resist the known-plaintext and chosen-plaintext attacks.

4.7 Speed analysis

With the exception of security consideration, other issues of an image cryptosystem such as the operation speed are also significant, especially for real-time applications. The actual execution time of an algorithm is determined by many factors such as algorithm, programming skill, programming language and execution environment. Therefore, we discuss mainly the performance of the proposed scheme from the computational complexity perspective. The running speed of an algorithm based on chaotic maps is mainly determined by the following three factors:

(1) Architecture of encryption/decryption.
(2) Encryption rounds of algorithm.
(3) Generating means of pseudorandom sequences.

To architecture of encryption/decryption, the permutation and diffusion processes are combined in the proposed algorithm, so only one time image-scanning step is required in each encryption round. This leads to a speed advantage compared with algorithms separating permutation and diffusion operations.

As shown in Table 4, the proposed cryptosystem and the Wang's [xx] only need a minimum of two

overall rounds to achieve a high performance such as NPCR > 0.996 and UACI > 0.333 for a tiny change at any position of the plain-image. The results show that the round number of encryption required by the proposed scheme is fewer than that by Wong's and Lian's. Thus, the proposed algorithm indeed leads to a faster encryption speed.

To generating pseudorandom sequences, the algorithms of Wong's and Lian's extract bits to generate pseudorandom numbers directly from the each iteration values of the logistic map and mask the image pixels one by one. In Wang's algorithm, 8 times extracting operation are required per 8×8 block. Because the state value of a chaotic map is a floating-point number, and a pseudorandom number is usually an integer, the conversion from floating points to integers cannot be avoided in practical applications. Computer simulation results show that such a conversion is time-consuming [5]. Thus multiplication and conversion from floating points to integers should be avoided in order to have high efficiency of generating pseudorandom numbers. In our algorithm, only two times conversion is required per round as in Sect. 2.2 and Table 4, so the conversion from floating points to integers is avoided. Therefore, compared with these algorithms, our algorithm has faster running speed.

5 Conclusion

A fast and secure image encryption is proposed and analyzed. This employs two technologies to improve the encryption/decryption speed. One is to combine the permutation and diffusion stages. As a result, the image needs to be scanned only once in each encryption round. Another is an effective generation of pseudorandom numbers by S-Box lookup, XOR, Modular and cyclic shift operations and so on. It avoids some time-consuming operations such as bit extraction form floating-points and conversion from floating-points to integers, so a higher encryption speed is obtained. Then, both theoretical analyses and experimental tests have been carried out. The results show that satisfactory security performance is achieved in only two overall encryption rounds and so the speed efficiency is improved. Moreover, the security of the proposed scheme is verified by the analyses on its size of key space, key sensitivity, statistical and differential properties and so on. In conclusion, the new cipher indeed has excellent potential for practical image encryption applications.

Acknowledgments Our sincere thanks go to the anonymous reviewers for their valuable comments. The work described in this paper was supported by the grants from the National Natural Science Foundation of China (No. 61003256), the Postdoctoral Science Foundation of China (2011M501391, 20110490082), the Natural Science Foundation of CQ CSTC (No. 2010BB2279) and the Program for excellent talents in Chongqing.

References

1. Pisarchik, A.N., Zanin, M.: Image encryption with chaotically coupled chaotic maps. Phys. D **237**, 2638–2648 (2008)
2. Fridrich, J.: Symmetric ciphers based on two-dimensional chaotic maps. Int. J. Bifurc. Chaos **8**(6), 1259–1284 (1998)
3. Lian, S., Sun, J., Wang, Z.: A block cipher based on a suitable use of the chaotic standard map. Chaos Solitons Fractals **26**, 117–129 (2005)
4. Wong, K.W., Kwok, B.S., Law, W.S.: A fast image encryption scheme based on chaotic standard map. Phys. Lett. A **372**, 2645–2652 (2008)
5. Wang, Y., Wong, K.-W., Liao, X., Chen, G.: A new chaos-based fast image encryption algorithm. Appl. Soft Comput. **11**, 514–522 (2011)
6. Yang, H., Wong, K.-W., Liao, X., Zhang, W., Wei, P.: A fast image encryption and authentication scheme based on chaotic maps. Commun. Nonlinear Sci. Numer. Simul. **15**, 3507–3517 (2010)
7. Wang, Y., Liao, X., Xiao, D., Wong, K.-W.: One-way hash function construction based on 2D coupled map lattices. Inf. Sci. **178**, 1391–1406 (2008)
8. Shannon, C.E.: Communication theory of secrecy system. Bell Syst. Tech. J. **28**, 656–715 (1949)
9. Li, D., Hu, G.: A keyed hash function based on the modified coupled chaotic map lattice. Commun. Nonlinear Sci. Numer. Simul. **17**, 2579–2587 (2012)
10. Yuen, C.-H., Wong, K.-W.: A chaos-based joint image compression and encryption scheme using DCT and SHA-1. Appl. Soft Comput. **11**, 5092–5098 (2011)
11. Mohammad Seyedzadeh, S., Mirzakuchaki, S.: A fast color image encryption algorithm based on coupled two-dimensional piecewise chaotic map. Signal Process. **92**, 1202–1215 (2012)
12. Pisarchik, A.N., Zanin, M.: Image encryption with chaotically coupled chaoticmaps. Phys. D **237**, 2638–2648 (2008)
13. Xiang, T., Wong, K.W., Liao, X.: Selective image encryption using a spatiotemporal chaotic system. Chaos **17**, 0231151–02311512 (2007)
14. Wang, X., Qin, X.: A new pseudo-random number generator based on CML and chaotic iteration. Nonlinear Dyn. **70**, 1589–1592 (2012)
15. Liu, N., Guo, D., Parr, G.: Complexity of chaotic binary sequence and precision of its numerical simulation. Nonlinear Dyn. **67**, 549–556 (2012)
16. Wei, J., Liao, X., Wong, K.W., Zhou, T.: Cryptanalysis of a cryptosystem using multiple one-dimensional chaotic maps. Commun. Nonlinear Sci. Numer. Simul. **12**, 814–822 (2007)

17. Wang, K., Pei, W., Zou, L., Song, A., He, Z.: On the security of 3D cat map based symmetric image encryption scheme. Phys. Lett. A **343**, 432–439 (2005)
18. Deng, S., Li, Y., Xiao, D.: Analysis and improvement of a chaos-based hash function construction. Commun. Nonlinear Sci. Numer. Simul. **15**(5), 1338–1347 (2010)
19. Wang, S., Shan, P.: Security analysis of a one-way hash function based on spatiotemporal chaos. Chin. Phys. B **20**, 090504–090507 (2011)
20. Kanso, A., Smaoui, N.: Irregularly decimated chaotic map(s) for binary digits generations. Int. J. Bifurcat. Chaos **19**(4), 1169–1183 (2009)
21. Zhang, Y., Xiao, D., Wen, W., Nan, H.: Cryptanalysis of image scrambling based on chaotic sequences and Vigenère cipher. Nonlinear Dyn. (2014). doi:10.1007/s11071-014-1435-9
22. Zhang, Y., Xiao, D.: Cryptanalysis of S-box-only chaotic image ciphers against chosen plaintext attack. Nonlinear Dyn. **72**(4), 751–756 (2013)
23. Yi, X., Tan, C.H., Siew, C.K.: A new block cipher based on chaotic tent maps. IEEE Trans. Circuits Syst. I **49**(12), 1826–1829 (2002)
24. Jakimoski, G., Kocarev, L.: Chaos and cryptography: block encryption ciphers based on chaotic maps. IEEE Trans. Circuits Syst. I **48**(2), 163–169 (2001)
25. Stojanovski, T., Kocarev, L.: Chaos-based random number generators-part I: analysis. IEEE Trans. Circuits Syst. I **48**(3), 281–288 (2001)
26. Stojanovski, T., Kocarev, L.: Chaos-based random number generators-part II: practical realization. IEEE Trans. Circuits Syst. I **48**(3), 382–385 (2001)
27. NIST Special Publication 800–22rev1a. http://csrc.nist.gov/groups/ST/toolkit/rng/index.html