An image encryption scheme based on constructing large permutation with chaotic sequence

Xuanping Zhang\textsuperscript{a,}\textsuperscript,* Liping Shao\textsuperscript{b}, Zhongmeng Zhao\textsuperscript{a}, Zhigang Liang\textsuperscript{a}

\textsuperscript{a}Department of Computer Science, Xi’an Jiaotong University, Xi’an 710049, China
\textsuperscript{b}School of Computer Science, Shaanxi Normal University, Xi’an 710062, China

\textbf{A R T I C L E I N F O}

\textbf{Article history:}
Available online 17 September 2013

\textbf{A B S T R A C T}

This paper proposes a chaos-based image encryption scheme with a permutation–diffusion structure. In the proposed scheme, the large permutation with the same size as the plainimage is used to shuffle the positions of image pixels totally. An effective method is also presented to construct the large permutation quickly and easily by combining several small permutations, where small permutations are directly generated using a chaotic map. In the diffusion stage, the pixel is enciphered by \textit{exclusive or} with the previous ciphered pixel and a random number produced by the Logistic map with different initial conditions. Test results and analysis by using several security measures have shown that the proposed scheme is efficient and reliable, and can be applied to real-time image encryption.

\textcopyright 2013 Elsevier Ltd. All rights reserved.

\textbf{1. Introduction}

The rapid growth of computer networks and the development in digital multimedia processing enable a large number of digital images to be easily transmitted in open networks\cite{1}. To protect images from unauthorized access has become a common interest in both research and application. Due to some intrinsic properties of digital image such as bulky data capacity and high correlation among pixels, traditional data encryption techniques may not be suitable for images\cite{2}. Motivated by the chaotic properties such as aperiodicity, high sensitivity to initial conditions and parameters, ergodicity and pseudo-randomness, many researchers have investigated and analyzed various chaos-based image encryption schemes\cite{3}.

Matthews\cite{4} first used Logistic map in image encryption and proposed a chaos-based image encryption scheme, then Scharinger\cite{5} presented a Baker map based image encryption algorithm. Fridrich\cite{6} suggested that a chaos-based image encryption scheme should comprise the iteration of permutation and diffusion\cite{7}, and Chen\cite{8} used the 3D cat map to encrypt image. Recently, a lot of chaos-based image encryption schemes have been proposed\cite{9–15}. Most of them are based on permutation. These schemes first produce a pseudo-random permutation from chaotic map and then use it to permute images\cite{16}.

A typical method for generation of a permutation using one-dimension chaotic map is given in\cite{17}. However, it needs huge computations to generate a large permutation by sorting. Moreover, because of the chaos periodicity caused by finite precision effect\cite{18}, chaotic map may produce duplicate elements in chaotic sequence, which increases the difficulty to generate exclusive elements for a large chaotic sequence. In order to avoid the difficulties in generating large permutation, some algorithms use small permutations to shuffle image pixels locally. But this method needs many iterations to achieve an effect equivalent to that of a large permutation.
In order to provide a reasonable security, a lot of chaos-based image encryption algorithms repeatedly shuffle image pixels by using a Permutation Array (PA) which consists of many different permutations. Since it’s quite expensive to generate a large-scale PA directly, the current research is concentrated on how to combine several small PA into a big one under certain constraints [19–22]. Based on Ding’s work [19], Yoon [16] proposed an image encryption algorithm by using large PA. However this algorithm requires a large amount of time and storage to construct a PA. Given an image with \( n \) pixels, the algorithm will produce a PA containing \( n \)-permutations of length \( n \), and requires \( O(n^2) \) storages. In practical image encryption, the number of permutations needed is far less than \( n \). Therefore most of the generated permutations are actually unused.

In this paper, a rapid and efficient method for generating large permutation is proposed by introducing the combination operation, which can produce a PA containing \( n \)-permutations of length \( n \). Therefore we present a method to generate large permutation by introducing the combination operation, which can produce a PA containing \( n \)-permutations of length \( n \). By using a Permutation Array (PA) which consists of many different permutations. Since it’s quite expensive to generate a large-scale PA directly, the current research is concentrated on how to combine several small PA into a big one under certain conditions. Based on this method, an image encryption algorithm is proposed. We also design a diffusion method based on the operations of exclusive or and shift to increase the security. Experiment results show that the proposed algorithm is efficient and reliable.

The rest of this paper is organized as follows. Section 2 introduces combination operation of permutations, proves some properties of this operation, and presents how to construct the large permutation. Section 3 describes the proposed image encryption algorithm in detail. Section 4 shows results of experiments over various security measures, and conclusion of this paper is given in Section 5.

2. Combination of permutations

Although some techniques have been developed to construct large size PA, they expend a mass of time and storage. Therefore we present a method to generate large permutation by introducing the combination operation, which can produce a single permutation each time.

2.1. Preliminaries

Let \( Z_n = \{0, 1, 2, \ldots, n-1\} \), a permutation \( \pi \) over \( Z_n \), represented as \( \pi = (\pi_0, \pi_1, \ldots, \pi_{n-1}) \), is a one-to-one self map of set \( Z_n \), where \( \pi_i \in Z_n \). For any \( i \in Z_n \), \( \pi(i) = \pi_i \) which means that \( i \) is mapped into \( \pi_i \) by \( \pi \).

If \( x \) and \( y \) are two permutations over \( Z_n \), the Hamming distance of \( x \) and \( y \) is defined by (1), where || means the cardinality of a set.

\[
d_H(x, y) = ||\{i | x_i \neq y_i, i \in Z_n\}||
\]  

(1)

Let \( S_n \) denote the set of all \( n! \) permutations over \( Z_n \). If the Hamming distance of any two permutations in \( C \subseteq S_n \) is at least \( d \), we would call \( C \) the permutation array of \((n, d)\), denoted as \((n, d)PA\).

Suppose \( C \) is a \((n, d)PA\) over \( Z_m \) of size \( m \), \( C \) can be represented by an \( m \times n \) matrix in which every row is a permutation over \( Z_m \). We say that \( C \) is \( r \)-bounded if no element of \( Z_m \) appears more than \( r \) times in any column of \( C \); \( C \) is \( r \)-balanced if each element of \( Z_m \) appears exactly \( r \) times in each column of \( C \); and \( C \) is \( r \)-separable if it is the disjoint union of \( r \) \((n, d)PA\) of size \( n \).

2.2. Combination operation

Let \( x = (x_0, x_1, \ldots, x_{m-1}) \) and \( y = (y_0, y_1, \ldots, y_{m-1}) \) be permutations over \( Z_m \) and \( Z_n \) respectively, the combination of permutations \( x \) and \( y \), denoted by \( x \odot y \), is a sequence of \( mn \) elements, which is defined by (2), where \( \sqcup \) is concatenation operator of sequences.

\[
x \odot y = \bigcup_{i=0}^{m-1} \bigcup_{j=0}^{n-1} (nx_i + y_j)
\]  

(2)

Let \( p = x \odot y = (p_0, p_1, \ldots, p_{mn-1}) \), and for any \( i \in Z_m \), \( j \in Z_n \) and \( k = ni + j \), then \( p_k \) can be calculated by (3).

\[
p_k = nx_i + y_j = nx_{k/m} + y_{k\%m}
\]  

(3)

Lemma 1. If \( x \), \( y \), and \( z \) are permutations, then \( x \odot (y \odot z) = (x \odot y) \odot z \).

Proof. Let \( x = (x_0, x_1, \ldots, x_{m-1}) \), \( y = (y_0, y_1, \ldots, y_{n-1}) \), \( z = (z_0, z_1, \ldots, z_{m-1}) \), \( p = (x \odot y) \odot z \) and \( p' = x \odot (y \odot z) \). Denote \( u = x \odot y \) and \( v = y \odot z \), then \( p = u \odot z \) and \( p' = x \odot v \). For any given \( i \in Z_m \), \( j \in Z_n \) and \( k \in Z_l \), we have

\[
u_{ni+j} = nx_i + y_j
\]

\[
v_{lj+k} = ly_j + zk
\]

\[
p_{n(i+j)+k} = p_{(n(i+j)+k)} = lu_{ni+j} + zk = nb_i + ly_j + zk
\]

\[
p'_{n(i+j)+k} = p'(n(i+j)+k) = nb_i + y_{lj+k} = nb_i + ly_j + zk
\]
Therefore $p = p'$, that is $(x \otimes y) \otimes z = x \otimes (y \otimes z)$. □

**Theorem 1.** If $x$ and $y$ are permutations over $Z_m$ and $Z_n$ respectively, then $x \otimes y$ is a permutation over $Z_{mn}$.

**Proof.** $x \otimes y$ contains $m \times n$ elements and every element can be expressed as $nx_i + y_j$ ($i \in Z_m, j \in Z_n$). When $x_i = 0$ and $y_j = 0$, $nx_i + y_j = 0$, which is the minimum element in $x \otimes y$. When $x_i = m - 1$ and $y_j = n - 1$, $nx_i + y_j = mn(m - 1) + n - 1 = mn - 1$, which is the maximum element in $x \otimes y$. Thus elements in $x \otimes y$ must be between 0 and $mn - 1$.

For any $i, i' \in Z_m$ and $j, j' \in Z_n$, $(nx_i + y_j) - (nx_i' + y_j') = n(x_i - x_i') + (y_j - y_j')$. Since $y_j, y_j' \in Z_n$, we have $|y_j - y_j'| < n$.

If $i \neq i'$, then $x_i \neq x_i'$ and $|x_i - x_i'| > 1$. So $n(x_i - x_i') + (y_j - y_j') \neq 0$.

If $i = i'$ and $j \neq j'$, then $y_j \neq y_j'$. Therefore $n(x_i - x_i') + (y_j - y_j') = y_j - y_j' \neq 0$.

Thus, any two elements in $x \otimes y$ won’t be equal. Moreover, $x \otimes y$ includes $mn$ different elements between 0 and $mn - 1$, so $x \otimes y$ is a permutation over $Z_{mn}$. □

**Theorem 2.** Let $x$ and $x'$ be permutations over $Z_m$ and $y$ and $y'$ be permutations over $Z_n$. If $d_s(x, x') = d_1$ and $d_h(y, y') = d_2$, then $d_s(x \otimes y, x' \otimes y') = nd_1 + md_2 - d_1d_2$.

**Proof.** Given any $i \in Z_m$ and $j \in Z_n$, $(nx_i + y_j) - (nx_i' + y_j') = n(x_i - x_i') + (y_j - y_j')$. Since $|y_j - y_j'| < n$, $n(x_i - x_i') + (y_j - y_j') = 0$ if and only if $x_i = x_i'$ and $y_j = y_j'$.

Hamming distance $d_s(x, x') = d_1$ implies that there are $m - d_1$ pairs of $(x_i, x_i')$ such that $x_i = x_i'$. Similarly, there are $n - d_2$ pairs of $(y_j, y_j')$ such that $y_j = y_j'$. So there are $(m - d_1)(n - d_2)$ pairs of $(i, j)$ to satisfy $n(x_i - x_i') + (y_j - y_j') = 0$.

So $d_s(x \otimes y, x' \otimes y') = mn - (m - d_1)(n - d_2) = nd_1 + md_2 - d_1d_2$. □

Especially, when $d_1 = m$ or $d_2 = n$, $d_s(x \otimes y, x' \otimes y') = mn$.

**Theorem 3.** If $x$ and $y$ are permutations over $Z_m$ and $Z_n$ respectively, then $(x \otimes y)^{-1} = x^{-1} \otimes y^{-1}$, where $x^{-1}, y^{-1}$ and $(x \otimes y)^{-1}$ are the inverse-permutations of $x$, $y$ and $(x \otimes y)$ respectively.

**Proof.** For any permutation $\pi = (\pi_0, \pi_1, \ldots, \pi_{n-1})$ over $Z_m$, there exists an inverse permutation $\pi^{-1}$. If $k \in Z_n$, then $\pi(k) = \pi_k$ and $\pi^{-1}(\pi_k) = k$.

Let $p = x \otimes y$ and $q = x^{-1} \otimes y^{-1}$, for any given $i \in Z_m, j \in Z_n$, we have

\[ q(p_{mi}) = q(nx_i + y_j) = nx^{-1}(\langle nx_i + y_j \rangle / n) + y^{-1}(\langle nx_i + y_j \rangle / n) = nx^{-1}(x_i) + y^{-1}(y_j) = ni + j \]

This means $p^{-1} = q$, that is $(x \otimes y)^{-1} = x^{-1} \otimes y^{-1}$. □

### 2.3. Algorithm of large permutation construction

As discussed in Section 2, a large permutation can be constructed by using small permutations. The proposed method in this study is composed of two steps. First, let $n = n_0 \times n_1 \times \cdots \times n_{m-1}$, and generate $m$ permutations $p_i (i = 0, 1, \ldots, m - 1)$ of length $n_i$ respectively by chaotic map. Next, shift one of the permutation $p_i$ and construct a large permutation $\pi$ by computing $\pi = p_0 \otimes p_1 \ldots \otimes p_{m-1}$.

The small permutation can be generated by using one-dimension chaotic map $f$. Set the initial value $u_0$ of $f$ and produce a sequence $S = \{x_0, x_1, \ldots, x_{l-1}\}$ of length $l$ by iteration. By quick sorting on $S$, an ordered sequence $S' = \{x'_0, x'_1, \ldots, x'_{l-1}\}$ can be obtained. For every element $x_i \in S$, there exists an element $x'_i \in S'$ such that $x_i = x'_i$. So we can define a one to one self map $p$ of $Z_l$ as $p(i) = j$ if $x_i \in S$, $x'_i \in S'$ and $x_i = x'_i$. This map $p$ is a permutation.

A large permutation of length $n$ can be generated by combining $m$ small permutations. A different large permutation can be produced by shifting one of $m$ small permutations and combining them. If we shift permutation $p_i$ to the left cyclically by one single position each time, a different permutation $p'_i$ can be produced, and the Hamming distance $d_s(p_i, p'_i) = n_i$ where $n_i$ is the length of $p_i$. Let $\pi = p_0 \otimes p_1 \ldots \otimes p_{m-1}$ and $\pi' = p_0 \otimes p_1 \ldots \otimes p_{m-1}$, then $\pi$ and $\pi'$ are permutations on $Z_n$ according to Theorem 1, and $d_s(\pi, \pi') = n$ according to Theorem 2.

If we shift permutation $p_i$ cyclically to left for $n_i$ times, $n_i$ different permutations will be produced. All different combinations of $m$ permutations can construct $n_0 \times n_1 \times \cdots \times n_{m-1} = n$ different permutations, which make up a 1-balanced $(n, n)PA$.

Let’s number all $n$ large permutations from 0 to $n - 1$. Given a number $k (0 \leq k < n)$, let
\[ k_i = \begin{cases} \frac{k}{n_{m-1}}, & \text{if } i = m - 1 \\ \left[ \frac{k}{\prod_{j=1}^{m-1} n_j} \right] \% n_i, & \text{if } i < m - 1 \end{cases} \] (4)

Then we shift \( p_i \) to the left cyclically by \( k_i(i = 0, 1, \ldots, m - 1) \) positions and combine them to construct the \( k \)th permutation. Given \( m \) small permutations and a specified number \( k \), Algorithm 1 will generate the \( k \)th large permutation.

**Algorithm 1.** GenPerm: constructing \( k \)th large permutation

**Input:**
- \( m \) small permutations \( p_0, p_1, \ldots, p_{m-1} \);
- \( n_0, n_1, \ldots, n_{m-1} \), such that \( n = n_0 \times n_1 \times \cdots \times n_{m-1} \);
- \( k \), the serial number of permutation to be generated;

**Output:**
- a large permutation \( \pi \);
1. \( \pi = (0) \);
2. \textbf{for} \( i = m - 1, m - 2, \ldots, 1, 0 \) \textbf{do}
3. \quad \( j = k \% n_i \);
4. \quad \( k = \lfloor k/n_i \rfloor \);
5. \quad \( q = \text{LeftRotate}(p_i) \); \quad // cyclically shift \( p_i \) left by \( j \) positions;
6. \quad \( \pi = q \oplus \pi \);
7. \textbf{end for}
8. \textbf{Return} \( \pi \);

Now, let’s discuss the time complexity of **Algorithm 1**. According to the definition of operator “\( \oplus \)” , if the lengths of two permutations \( x \) and \( y \) are \( s \) and \( t \) respectively, the time complexity of calculating \( x \oplus y \) is \( O(st) \). Therefore the time complexity of calculating \( p_0 \oplus p_1 \cdots \oplus p_{m-1} \) is \( O \left( \sum_{i=0}^{m-2} \prod_{j=i}^{m-1} n_j \right) = O(n) \). So the time complexity of **Algorithm 1** is \( O(n) \).

3. Image encryption based on permutation and diffusion

The image encryption scheme proposed in this paper consists of multiple rounds of permutation and diffusion. In permutation process, a permutation is used to permute all the pixels and the new pixel moved to the current position is taken as a substitution of the original pixel. The diffusion process modifies the pixel values sequentially and the change made to a particular pixel depends on the accumulated effect of all the previous pixel values.

3.1. Diffusion process

The function of the diffusion is to alter the pixel value sequentially so that a small change in one pixel is spread out to many pixels, hopefully the whole image. The diffusion scheme we designed performs exclusive or operation on the current pixel, a pseudo random number and the previous ciphered pixel. The pseudo random number is simply generated by Logistic map.

Set the initial value \( n_0 \in (0,1) \) of the Logistic map \( f(x) = 3.99933x(1-x) \), a pseudo random sequence \( \{x_0, x_1, \ldots, x_{n-1}\} \) is generated by iteration. Let \( z_i = [x_i \times 2^{-48}] \% 2^8 \), the pseudo random integer sequence \( \{z_0, z_1, \ldots, z_{n-1}\} \) can be achieved, where \( z_i \in [0,255] \).

Let \( s = \{s_0, s_1, \ldots, s_{n-1}\} \) be the plain image with \( n \) pixels, and \( s' = \{s'_0, s'_1, \ldots, s'_{n-1}\} \) be the corresponding ciphered image. In order to simplify the representation of diffusion operation, set \( s'_{-1} \) as the initial value of diffusion process, which can be taken as a part of secret key. The diffusion operation is represented by (5).

\[ s'_j = [(s_j + z_j) \% 256] \oplus \text{rol}(s'_{j-1}, 3) \oplus z_h \quad j = 0, 1, \ldots, n - 1 \] (5)

where \( z_h = (j + s'_{-1}) \% n_i \) is exclusive or operator, \( \text{rol}(x, 3) \) implies cyclically shifting \( x \) to the left by 3 bits. This means that the key stream used in diffusion is different from the sequence of random numbers produced by the chaotic map, and highly depends on the previous ciphered pixels.

From (5), we can deduce the inverse diffusion operation as

\[ s_j = [(s'_j \oplus \text{rol}(s'_{j-1}, 3) \oplus z_h) + 256 - z_j] \% 256 \] (6)

3.2. The proposed encryption algorithm

The encryption algorithm initially generates \( m \) small permutations \( \{p_0, p_1, \ldots, p_{m-1}\} \), then repeatedly performs permutation–diffusion on image. Let \( s^{(0)} = \{s_0, s_1, \ldots, s_{n-1}\} \) be the plain image of \( n \) pixels, and \( r \) be the rounds of permutation and diff-
fusion, the process of image encryption is shown in Fig. 1, where \( \sigma \) is diffusion operation and \( \pi_i \) is the permutation used in \( i \)th round of encryption.

In each round of encryption, first a random number \( k \) is generated by iterating the chaotic map with an initial condition \( w_0 \) and the \( k \)th permutation is constructed by Algorithm 1 in Section 2. Then the pixels of plain image are rearranged by this permutation to get a shuffled image. Next, the diffusion sequence is produced by iterating Logistic map with different initial conditions, and each pixel of the shuffled image is enciphered with the help of (5). The encryption algorithm is described in detail as Algorithm 2.

In the process of image encryption, the secret key is comprised of:

(a) initial value \( u_0 \in (0, 1) \) used to generate \( m \) permutations;
(b) initial value \( v_0 \in (0, 1) \) used to generate diffusion sequence;
(c) initial value \( w_0 \in (0, 1) \) used to construct a large permutation;
(d) rounds of permutation–diffusion \( r, r \geq 2 \);
(e) \( m \) and \( n_0, n_1, \ldots, n_{m-1} \), such that \( n = n_0 \times n_1 \times \cdots \times n_{m-1} \); and
(f) initial value \( s_0 \) for diffusion.

Algorithm 2. The proposed algorithm of image encryption

| Input: | (1) plaintext image \( s = \{s_0, s_1, \ldots, s_{n-1}\} \);
| Output: | (2) the secret key includes \( u_0, v_0, w_0, r, m, n_0, n_1, \ldots, n_{m-1}, s'_{n-1} \);
| 1: \( n = n_0 \times n_1 \times \cdots \times n_{m-1} \); | 2: Generate \( m \) small permutations \( P = \{p_0, p_1, \ldots, p_{m-1}\} \);
| 3: \( s' = s \); | 4: for \( i = 0, 1, \ldots, r - 1 \) do
| 5: Generate a random number \( k \); | 6: \( \pi = \text{Genperm}(P, \{n_0, n_1, \ldots, n_{m-1}\}, k) \);
| 7: \( x = \{s'_{0(0)}, s'_{0(1)}, \ldots, s'_{n(n-1)}\} \); | 8: Generate diffusion sequence \( Z = \{z_0, z_1, \ldots, z_{n-1}\} \);
| 9: for \( j = 0, 1, \ldots, n - 1 \) do
| 10: \( h = (j + s'_{j-1}) \mod n \); | 11: \( s'_j = [\lfloor x_j + z_j \rfloor \mod 256] \oplus \text{rol}(s'_{j-1}, 3) \oplus z_h \);
| 12: end for | 13: end for
| 14: Return \( s' \) |

3.3. Decryption algorithm

Decryption algorithm is similar to encryption algorithm. The initialization in decryption is the same as in encryption. The only difference comes out in the iterations. In each iteration of decryption, the inverse diffusion is performed on cipher image first, and then the inverse-permutation is constructed for permuting all pixels of the image. The execution order of all inverse-permutations is the inverse one in encryption, namely from \( \pi_{r-1}^{-1} \) to \( \pi_0^{-1} \). The inverse diffusion is calculated by (6). The inverse permutation can be constructed according to Thoerem 3.
4. Simulation results analysis

Experimental results are given in this section to demonstrate the efficiency of the proposed image encryption method. In the following experiments, the algorithm is performed on gray scale images in condition of $u_0 = 0.35669$, $v_0 = 0.31376$, $w_0 = 0.79637$, $s_1 = 0 \times 36$. The different security measures are used to evaluate the performance such as visual effect, histogram, correlation coefficients, anti-differential attack, and encryption speed. Additionally, three chaos-based image encryption algorithms, Fridrich’s [6], Chen’s [8] and Zhang’s [11], are also tested for comparison.

4.1. Visual effect of encrypted images

We performed the proposed algorithm on the gray scale image Lena of size $512 \times 512$. The encryption results are shown in Fig. 2, where Fig. 2(a) is the original image, Fig. 2(b) and (c) are the encrypted images generated after 1 and 2 rounds of encryption respectively. From Fig. 2, we can observe that the images encrypted by the proposed algorithm are similar to random noise images.

In order to evaluate the randomness of the encrypted image, we encrypt a gray image of size $6000 \times 6000$ for several rounds, and then apply NIST test suite [23] to test randomness of encrypted image. In NIST test, set the number of subsequences $m = 100$, the length of subsequences $l = 10^6$ bits, and lowest passing rate $P_a \geq 0.96$. When the number of encryption rounds is $k = 1, 2$ and 4, the test results compared with Chen’s [8] are given in Table 1.

From Table 1, it can be seen that the more rounds of encryption, the more tests passed, and the better randomness of ciphered image. For Chen’s method, image ciphered by 1 round of encryption can pass 7 tests, 2 rounds of encryption can pass 13 tests, and 4 rounds of encryption can pass all tests. For the proposed algorithm, only 1 round of encryption can pass all tests. Therefore, image encrypted by the proposed algorithm has good randomness.

4.2. Statistical analysis

An effective encryption algorithm should be robust against any statistical attack. To prove the robustness of the proposed image encryption algorithm, statistical analyses have been performed on ciphered image to demonstrate its superior confu-

![Fig. 2. Visual effects of encrypted image.](image)

<table>
<thead>
<tr>
<th>Tests</th>
<th>Chen’s</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$k = 1$</td>
<td>$k = 2$</td>
</tr>
<tr>
<td>Frequency</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Block frequency</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Runs test</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>Longest run</td>
<td>0.59</td>
<td>0.98</td>
</tr>
<tr>
<td>Matrix rank</td>
<td>0.99</td>
<td>1.00</td>
</tr>
<tr>
<td>DFT test</td>
<td>0.72</td>
<td>0.99</td>
</tr>
<tr>
<td>Non-overlapping</td>
<td>0.85</td>
<td>1.00</td>
</tr>
<tr>
<td>Overlapping</td>
<td>0.71</td>
<td>0.99</td>
</tr>
<tr>
<td>Universal statistical</td>
<td>0.99</td>
<td>0.99</td>
</tr>
<tr>
<td>Linear complexity</td>
<td>0.98</td>
<td>0.99</td>
</tr>
<tr>
<td>Serial test</td>
<td>0.00</td>
<td>0.99</td>
</tr>
<tr>
<td>Approximate entropy</td>
<td>0.00</td>
<td>0.96</td>
</tr>
<tr>
<td>Cumulative sums</td>
<td>1.00</td>
<td>0.98</td>
</tr>
<tr>
<td>Random excursions</td>
<td>0.91</td>
<td>0.84</td>
</tr>
<tr>
<td>Excursions variant</td>
<td>0.91</td>
<td>0.85</td>
</tr>
</tbody>
</table>
sion and diffusion properties which strongly resist statistical attacks. This is shown by calculating the histogram, the information entropy and the correlation of two adjacent pixels in the ciphered image.

4.2.1. Histograms of encrypted images
The ideal goal of image encryption is that encrypted images have histograms with random behavior. Fig. 3 shows the histograms of the plain image and the images encrypted for 1 round by using algorithms of Chen’s [8], Zhang’s [11] and ours. Obviously, the pixel values of these encrypted images are distributed among 0~255 close to the ideal uniform distribution.

The $\chi^2$ test is used to further analyze the uniformity of the pixel value distribution. The $\chi^2$ value of a image with 256 gray levels is calculated by (7).

$$\chi^2 = \sum_{i=1}^{256} \frac{(n_i - n_i/256)^2}{n_i/256}$$

where $n_i$ is the occurrence frequency of gray level $i$, $n_i/256$ is the expected occurrence frequency of each gray level. The $\chi^2$ values of plain and encrypted images are listed in Table 2. Assuming a significant level of 0.05, $\chi^2(0.05,255) = 293.25$. The $\chi^2$ value of encrypted image is less than $\chi^2(0.05,255)$. This means that the histogram distribution of encrypted image is uniform. Compared with other three methods, our scheme has the same superior performance in the histogram distribution.

4.2.2. Information entropy analysis
Information entropy is the most significant feature of disorder, or more precisely unpredictability. It can be used to measure the distribution of gray value in image. The entropy $H(m)$ of an $m$ can be calculated as:

$$H(m) = \sum_{i=0}^{2^k-1} p(m_i) \log_2 \frac{1}{p(m_i)}$$

Table 2
\begin{tabular}{|c|c|c|c|c|}
\hline
 & Plain-image & Fridrich’s & Chen’s & Zhang’s & Proposed \\
\hline
$\chi^2$ & 158063 & 286.152 & 279.887 & 245.203 & 256.128 \\
\hline
\end{tabular}

Fig. 3. Histograms of the original image and the encrypted images, (a) original image, (b) Chen’s scheme, (c) Zhang’s scheme, and (d) the proposed scheme.
where $2^N$ is the total number of symbols, $m_i \in m$ and $p(m_i)$ represents the probability of symbol $m_i$. For a random image with 256 gray levels, the entropy should ideally be $H(m) = 8$. Therefore, an effective encryption algorithm should produce an encrypted image of entropy close to 8.

The entropies of plain image and ciphered images using various schemes are calculated and listed in Table 3. As can be seen from Table 3, the entropy of the ciphered image produced by the proposed scheme is very close to the theoretical value of 8 and is competitive with that of other three schemes. This means that the ciphered images are close to a random source and the proposed algorithm is secure against the entropy attack.

### 4.2.3. Correlation coefficients

We analyze the correlations between two horizontally, vertically and diagonally adjacent pixels. Fig. 4 shows the horizontal, vertical and diagonal correlations of plain image and encrypted image produced by the proposed algorithm. Fig. 4(a) and (c) display the three correlations in the plain image, where the representative points concentrate near the diagonal, so the correlation coefficient is rather close to 1, which implies a strong correlation between the adjacent pixels. Fig. 4(d)–(f) show the three correlations in the encrypted image, where the adjacent pixels in horizontal, vertical and diagonal are almost irrelevant.

Table 4 gives the correlation coefficients of plain and encrypted images produced by four algorithms. It is clear from Table 4 that all correlation coefficients of encrypted images close to zero. Our algorithm exhibits the same effect as other three algorithms, and can effectively decrease the correlations among the adjacent pixels of the plain image.

### 4.3. Sensitivity analysis

To resist differential analysis, cipher text should be sensitive to both plain text and secret key, that is, cipher text should change dramatically in response to any small changes in plain text or secret key. The NPCR(Number of Pixels Change Rate) and UACI(Unified Average Changing Intensity) are commonly used to evaluate sensitivity to plain text and secret key. Given two

### Table 3

Information entropy of plain and encrypted images.

<table>
<thead>
<tr>
<th></th>
<th>Plain-image</th>
<th>Fridrich’s</th>
<th>Chen’s</th>
<th>Zhang’s</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entropy</td>
<td>0.7445507</td>
<td>7.999262</td>
<td>7.999229</td>
<td>7.999296</td>
<td>7.999322</td>
</tr>
</tbody>
</table>

**Fig. 4.** Horizontal, vertical and diagonal correlations of plain image and encrypted images, (a)–(c) correlations of plain image, (d)–(f) correlations of encrypted image.
images \(x = \{x_0, x_1, \ldots, x_{n-1}\}\) and \(y = \{y_0, y_1, \ldots, y_{n-1}\}\), the \(\text{NPCR}\) and \(\text{UACI}\) are defined as (9) and (10) [24]. For two random images, the average \(\text{NPCR}\) is about 0.9961, and the average \(\text{UACI}\) is about 0.3346 [25].

\[
\text{NPCR} = \frac{1}{n} \sum_{i=0}^{n-1} |x_i \neq y_i|, i = 0, 1, \ldots, n - 1
\]

\[
\text{UACI} = \frac{1}{n} \sum_{i=0}^{n-1} \frac{|x_i - y_i|}{255}
\]

4.3.1. Sensitivity to plaintext

Two plain images are used in the test of plaintext sensitivity. One of them is the gray scale image of size 512 \times 512 used as the original plain image, and the other is generated by modifying the value of the center pixel in original image. We encrypt these two images with the same secret key for 10 rounds, and calculate \(\text{NPCR}\) and \(\text{UACI}\) of the encrypted images after each round. Table 5 gives the \(\text{NPCR}\) and \(\text{UACI}\) values of the images encrypted for 10 rounds by different algorithms.

Table 5: Plaintext sensitivity of different algorithms.

<table>
<thead>
<tr>
<th>Round</th>
<th>Fridrich's</th>
<th>Chen's</th>
<th>Zhang's</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NPCR</td>
<td>UACI</td>
<td>NPCR</td>
<td>UACI</td>
</tr>
<tr>
<td>1</td>
<td>0.4375</td>
<td>0.2196</td>
<td>0.5276</td>
<td>0.2656</td>
</tr>
<tr>
<td>2</td>
<td>0.7615</td>
<td>0.3136</td>
<td>0.9921</td>
<td>0.3338</td>
</tr>
<tr>
<td>4</td>
<td>0.9147</td>
<td>0.3324</td>
<td>0.9927</td>
<td>0.3342</td>
</tr>
<tr>
<td>6</td>
<td>0.9142</td>
<td>0.3320</td>
<td>0.9936</td>
<td>0.3351</td>
</tr>
<tr>
<td>8</td>
<td>0.9381</td>
<td>0.3330</td>
<td>0.9948</td>
<td>0.3343</td>
</tr>
<tr>
<td>10</td>
<td>0.9381</td>
<td>0.3336</td>
<td>0.9959</td>
<td>0.3347</td>
</tr>
</tbody>
</table>

Table 6: Sensitivity to secret key.

<table>
<thead>
<tr>
<th>Round</th>
<th>NPCR</th>
<th>UACI</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.9963</td>
<td>0.9961</td>
</tr>
<tr>
<td>2</td>
<td>0.9961</td>
<td>0.9960</td>
</tr>
<tr>
<td>4</td>
<td>0.9960</td>
<td>0.9961</td>
</tr>
<tr>
<td>6</td>
<td>0.9960</td>
<td>0.9961</td>
</tr>
<tr>
<td>8</td>
<td>0.9960</td>
<td>0.9961</td>
</tr>
<tr>
<td>10</td>
<td>0.9961</td>
<td>0.9961</td>
</tr>
</tbody>
</table>

4.3.2. Sensitivity to key

Because of the different constitutions of key in different algorithms, it is difficult to compare their key sensitivities. We only test key sensitivity of the proposed algorithm. Given two keys with only a tiny difference of 10^{-10} in \(u_0\), we encrypt the plain image for 10 rounds. Table 6 shows \(\text{NPCR}\) and \(\text{UACI}\) of the encrypted images. Both are close to the mean values of a random image. Therefore, the proposed algorithm is rather sensitive to the change of secret key.

Table 7: Encryption time of different algorithms with 8 rounds (in s).

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>256 \times 256</th>
<th>512 \times 512</th>
<th>1024 \times 1024</th>
<th>2048 \times 2048</th>
<th>4096 \times 4096</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fridrich [6]</td>
<td>0.01292</td>
<td>0.05040</td>
<td>0.19911</td>
<td>0.79405</td>
<td>3.97146</td>
</tr>
<tr>
<td>Chen [8]</td>
<td>0.02304</td>
<td>0.09083</td>
<td>0.36626</td>
<td>1.46370</td>
<td>5.85070</td>
</tr>
<tr>
<td>Zhang [11]</td>
<td>0.09814</td>
<td>0.43386</td>
<td>1.90271</td>
<td>8.38100</td>
<td>37.3067</td>
</tr>
<tr>
<td>Proposed</td>
<td>0.00887</td>
<td>0.03191</td>
<td>0.13774</td>
<td>0.53187</td>
<td>2.11078</td>
</tr>
</tbody>
</table>
4.4. Speed analysis

The speed of the proposed image encryption technique is tested by using the Microsoft VC++ programming on a personal computer with 3.10 GHz Intel(R) Core(TM) i5-2400 CPU and 4 G memory running on Microsoft Windows 7. Several grayscale images of different sizes have been encrypted for 8 rounds to test the encryption time. The test results are listed in Table 7. Obviously, the proposed algorithm is much faster than the others. This is because the proposed algorithm uses large permutation to shuffle image, which directly moves each pixel to the destined location without coordinate transformation, and furthermore the large permutation is easily and quickly generated by combining small permutations.

5. Conclusion

A chaotic image encryption scheme based on total shuffling approach has been proposed, which comprises the iteration of permutation and diffusion. An effective method based on the combination operation has also been presented to construct the large permutation for totally shuffling the positions of image pixels, which can reduce the time complexity of generating large permutation to $O(n)$. In the permutation step, the permutation with the same size of the plain-image is chosen to disrupt the high correlation among the image pixels to increase the security level of the encrypted images. In the diffusion step, the key stream depending on both secret key and the image pixels is generated by the chaotic map to resist chosen-plaintext attack. Several security measures, such as visual effect, statistical analysis, sensitivity analysis and speed analysis, have been used to evaluate the performance of this image encryption scheme. Test results and analysis have shown the proposed method can efficiently resist against statistical attack, differential attack and plaintext attack, and has a significant advantage in encryption speed. Accordingly, the proposed image encryption method is very well suitable for the real-time secure image transmission over public networks.

Acknowledgments

The authors acknowledge the National Natural Science Foundation of China (Grant Nos: 61100239 and 60803088), the Ph.D. Programs Foundation of Ministry of Education of China (Grant No: 2010020110063).

References


Xuanping Zhang is an associate professor at Department of Computer Science and Technology of Xi’an Jiaotong University, Xi’an, China. He received MSc degree in 1987 from University of Science and Technology of China, and received PhD degree in 2006 from Xi’an Jiaotong University. His research interests include information security, digital encryption, digital watermarking.

Liping Shao received PhD degree from Xi’an Jiaotong University, Xi’an, China in 2010. He is currently an associate professor at School of Computer Science of Shaanxi Normal University, Xi’an, China. His research interests include image processing, image encryption, information hiding.
Zhongmeng Zhao received BS degree, MSc degree and PhD degree from Xi’an Jiaotong University, Xi’an, China in 1982, 1985 and 2010 respectively. He is currently a professor at Department of Computer Science and Technology of Xi’an Jiaotong University. Computer networks, new information technique and data mining are among his professional interests.

Zhigang Liang received MSc degree in Computer Science and Technology from Xi’an Jiaotong University, Xi’an, China in 2011. His research interests include image encryption, chaotic system.