Information Sciences 264 (2014) 317–339

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Design of image cipher using latin squares

Yue Wu [a,1], Yicong Zhou [b,*], Joseph P. Noonan [a], Sos Agaian [c]

[a] Department of Electrical and Computer Engineering, Tufts University, Medford, MA 02155, United States
[b] Department of Computer and Information Science, University of Macau, Macau 999078, China
[c] Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249, United States

ARTICLE INFO

Article history:
Received 22 February 2013
Received in revised form 10 September 2013
Accepted 19 November 2013
Available online 27 November 2013

Keywords:
Image encryption
Latin square
Substitution–permutation network
Confusion–diffusion
Error tolerance

ABSTRACT

In this paper, we introduce a symmetric-key Latin square image cipher (LSIC) for grayscale and color image encryption. Our main contributions include (1) we propose new Latin square image encryption primitives including *Latin Square Whitening, Latin Square S-box* and *Latin Square P-box*; (2) we develop probabilistic image encryption by embedding random noise into the least significant bit-plane of images; and (3) we design a new loom-like 2D substitution–permutation network maintaining good confusion and diffusion properties with extra error tolerance. Theoretical analysis and simulation results show that the proposed method has many desired properties of a secure cipher, shows robustness against different attack models, and outperforms state of the art suggested by many peer algorithms. Open-source implementation can be found on the webpage https://sites.google.com/site/tuftsyuewu/source-code.

1. Introduction

With ubiquitous digital images and digital media devices all over the world, the importance of image security has been noticed and emphasized in recent years [50]. In the real world, digital cameras capture the real scene in the format of digital images, and are widely used in many digital devices such as smart phones, IPADs, and laptops. In the virtual world, digital images, including those taken from cameras, scanned documents or pictures, and computer-aid virtual paintings and so on, are the most common elements within a webpage besides texts on the World Wide Web. Due to extensive information within a digital image, divulged image contents sometimes cause severe problems for its owner(s). In many cases, such information leakage seriously invades personal privacy, e.g. the malicious spread of photos in personal online albums or patients' medical diagnosis images, and furthermore it may cause uncountable losses for a company or a nation, e.g. a secret product design for a company or a governmental classified scanned document.

Conventionally, digital data is encrypted by bit-stream ciphers and block ciphers [1,2,6,27]. The two well-known block ciphers are the Digital Encryption Standard (DES) [1] and its successor Advanced Encryption Standard (AES) [2]. A digital image is a specific type of digital data and can be encrypted by these conventional ciphers. However, they are somewhat insuitable for digital images because of.

- Relatively small block size: digital images are normally of several kilobits (Kb) and megabits (Mb), while conventional bit-stream/block ciphers commonly has a block size less than 256 bits.

* Corresponding author. Tel.: +853 83978458; fax: +853 28838314.
E-mail addresses: ywu03@ece.tufts.edu (Y. Wu), yicongzhou@umac.mo (Y. Zhou).
[1] Tel.: +1 6176273217; fax: +1 6176273220.

0020-0255/$ - see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.ins.2013.11.027

- Neglect of the nature of digital images: digital images are of two-dimensional data, while conventional bit-stream/block ciphers encrypt an image by indirectly encrypting a pixel sequence extracted from this image.

The first defect implies the low efficiency of encrypting a digital image using a bit-stream/block cipher [9,34,50]. The second defect implies that a pixel sequence of an image is of high information redundancy with a tilted histogram and is distinctive from a common bit sequence input to a bit-stream/block cipher. Therefore, image encryption should be adaptive to image properties and natures.

In general, digital image encryption methods can be classified into two groups: perceptual-level and bit-level. The perceptual-level image encryption methods intend to transform an image into a unrecognized one using a fast algorithm, e.g. transform techniques [8,11,53] and optical encryption techniques [12,23]. In this case, images are believed to be valuable only within a certain time period, e.g. a couple of hours. To some extend, therefore, an encrypted image by using perceptual-level image encryption is insecure because it maybe cracked after a sufficiently long time. In contrast, the bit-level image encryption aims to change an image into a random-like one. In this situation, images are believed to be valuable for a quite long time period, e.g. twenty years. Nowadays, the bit-level image encryption methods are mainly based on chaotic systems [9,17,21,28,34–36,45,46,55]. Although many existing chaotic image encryption algorithms have several good cryptographical properties, they have defects in the following aspects regardless of used chaotic systems:

- A chaotic system is defined on real numbers while a cryptosystem is defined on finite numbers.
- A chaotic system may lose its chaotic nature completely and become periodic when it is discretized.
- A chaotic system's parameters and initial values can be estimated by a number of existing tools and methods.

The first defect implies difficulties of software and hardware implementation for a chaos-based image encryption method because round-off errors in real number quantizations may lead nonreversible functions for encryption and thus make the decryption process impossible [31]. The second defect implies a chaotic image encryption method could be completely nonchaotic and thus vulnerable to attacks [20]. The third defect reveals a high risk that initial values and parameters used in a chaotic system might be fully analyzed using existing tools and methods after a long-term observation [4,5,30]. For example, [24,36] are cryptanalyzed in [20,18], respectively. Besides chaotic image encryption methods, nonchaotic image encryption methods are also researched by using various random-like patterns, e.g. cellular automata [10], P-Fibonacci transform [53], wave transmission model [21], Sudoku matrices [39,40,42,47], and Gray codes [54]. Although nonchaotic image encryption methods eliminate drawbacks of chaotic encryption methods, especially round-off errors, many of them do not have good confusion and diffusion properties [29] as in chaotic encryption methods, due to the fact that the used random-like patterns are not truly random-like.

In this paper, we introduce a novel image encryption solution LSIC. In particular, to avoid chaotic image ciphers' drawbacks like round-off errors, randomness degradations, etc. [19], LSIC is designed on integers directly; to achieve good diffusion and confusion properties, it follows the design guideline of the Markov cipher and is constructed according to the SPN [32] with all Latin-square-based primitives, including *Latin Square Whitening, Latin Square Substitution* and *Latin Square Permutation*; to achieve fast computations, it adopts a large block size of 256×256 bytes; and to further improve security, it also employs the probabilistic encryption [15,16].

The rest of this paper is organized as follows: Section 2 gives a brief review on preliminary materials. Section 3 introduces the new LSIC including its key schedule, probabilistic encryption, and Latin square based encryption primitives. Section 4 discusses simulation setups with extensive results, Section 5 analyzes the cipher security theoretically and experimentally under various attacks, and finally Section 6 concludes the paper and give discussions on open questions in the proposed LSIC.

2. Latin squares

A Latin square of order N is an $N \times N$ array filled with a set of N distinctive symbol elements, where each symbol appears exactly once in each row and each column. The name *Latin Square* is motivated by the mathematician *Leonhard Euler*, who used Latin characters as symbols.

Mathematically, we define a Latin square L of the order N by an indicator function f_L on tri-tuple (r, c, i) as follows

$$f_L(r,c,i) = \begin{cases} 1, & L(r,c) = S_i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where r denotes the row index of an element in L with $r \in \mathbb{N} = \{0, 1, \ldots, N-1\}$; c denotes the column indx of an element in L with $c \in \mathbb{N}$; i denotes the index of a symbol element in L with $I \in \mathbf{N}$; and S_i is the ith symbol in the symbol set $\mathbb{S} = \{S_0, S_1, \ldots, S_{N-1}\}$.

Therefore, if L is a Latin square of the order N, then.

- for any fixed $c, i \in \mathbb{N}$, we have

$$\sum_{r=0}^{N-1} f_L(r,c,i) = 1 \tag{2}$$

- for any fixed $r, i \in \mathbb{N}$, we have

$$\sum_{c=0}^{N-1} f_L(r,c,i) = 1 \tag{3}$$

which implies that each symbol appears exactly once in each row and each column in L.

Fig. 1 shows examples of Latin squares in different orders with various symbol sets. It is worthwhile to note that the popular *Sudoku* puzzle [3,47] is a special case of the Latin square with additional block constraint as shown in Fig. 1(d).

Although Latin squares can be generated via a variety of means, we use Algorithm 1 [41] below for Latin square generation in this paper.

Algorithm 1. A Latin Square Generator $L = \mathrm{LSG}(Q_1, Q_2)$.

Input: Q_1 and Q_2 are two length-N sequences
Output: L is a Latin square of order N
$Q_{seed} = \mathrm{SortMap}(Q_1)$
$Q_{shift} = \mathrm{SortMap}(Q_2)$
for $r = 0 : 1 : N-1$ **do**
$L(r,:) = \mathrm{RowShift}(Q_{seed}, Q_{shift}(r))$
end for

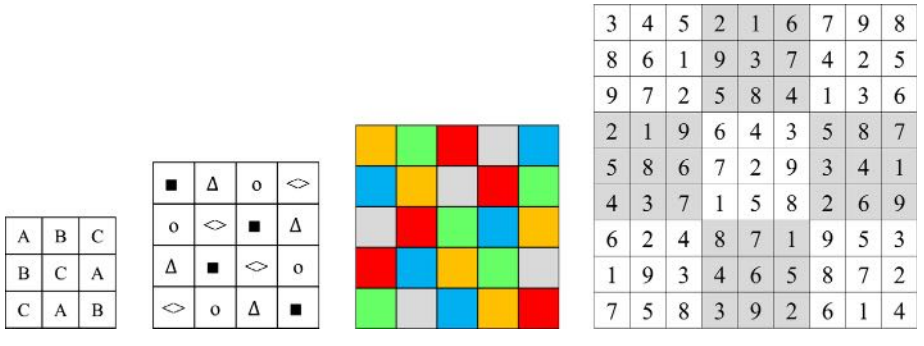
In Algorithm 1, both Q_1 and Q_2 are length-N sequences from a pseudo-random number generator (PRNG), e.g. Linear Congruential Generators (LCG) [26]; $\mathrm{SortMap}(Q)$ is a function to find the index mapping between a sequence Q and its sorted version Q^* in the ascending order; and $\mathrm{RowShift}(Q, v)$ ring shifts the sequence Q with v elements towards left. For example, when $Q_1 = Q_2 = \{0,1,2\}$ and $\mathbb{S} = \{A,B,C\}$, Algorithm 1 then generates a Latin quare as shown in Fig. 1(a).

3. Latin square image cipher

3.1. The new Latin square image cipher

To standardize the encryption/decryption processing, we choose the cipher processing block to a 256×256 byte (1 byte = 8 bits) block. In the rest of this paper, we use P to denote a 256×256 plaintext image block, C to denote a corresponding ciphertext image block of P, L to denote a keyed Latin square of the order 256 with the symbol set setting as the integers $\mathbb{S} = \{0,1,\ldots,255\}$, and K to denote a 256-bit encryption key.

The new proposed Latin square image cipher is of a substitution–permutation network (SPN) structure with eight rounds as shown in Fig. 2. It is composed of the probabilistic encryption stage *LSB noise embedding* (LSBNE) and the SPN stage containing three encryption primitives, namely *Latin Square Whitening* (LSW), *Latin Square Substitution* (LSS) and *Latin Square Permutation* (LSP). Specifically, Algorithms 3 and 4 describe the encryption (denoted as function $\mathfrak{E}(P,K)$) and decryption



(a) 3×3 (b) 4×4 (c) 5×5 (d) 9×9

Fig. 1. Latin square examples.

(denoted as function $\mathfrak{D}(C,K)$) processes of the Latin square image cipher, respectively. The detailed encryption/decryption stages in LSIC will be discussed in future sections.

Algorithm 2. Latin Square Image Cipher- Encryption $C = \mathfrak{E}(P,K)$.

Input: K is a 256-bit key; P is a 256×256 8-bit grayscale image block
Output: C is a 256×256 8-bit grayscale image block
$(\mathbf{Q_1}, \mathbf{Q_2}) = \mathrm{KDSG}(K, 8)$
for $n = 0:1:7$ **do**
  if $n == 0$ **then**
    $C_{\mathrm{LSP}} = \mathrm{LSBNE}(p)$[a]
  end if
  $L_n = \mathrm{LSG}(Q_1^n, Q_2^n)$
  $D_n = L_n(0,0)$
  $C_{\mathrm{LSW}} = \mathrm{ECR}_w(L_n, C_{\mathrm{LSP}}, D_n)$
  if $mod(n,2) \neq 0$ **then**
    $C_{\mathrm{LSS}} = \mathrm{ECR}_s^{col}(L_n, C_{\mathrm{LSW}})$
  else
    $C_{\mathrm{LSS}} = \mathrm{ECR}_s^{row}(L_n, C_{\mathrm{LSW}})$
  end if
  $C_{\mathrm{LSP}} = \mathrm{ECR}_p(L_n, C_{\mathrm{LSS}})$
end for
$L_8 = \mathrm{LSG}\left(Q_1^8, Q_2^8\right)$
$D_8 = L_8(0,0)$
$C = \mathrm{ECR}_w(L_8, C_{\mathrm{LSP}}, D_n)$

[a] LSBNE could be any function to embed the random binary noise in the least significant bit-plane of a plaintext image, e.g. a function randomly change the parity of a plaintext pixel on its LSB.

Algorithm 3. Latin Square Image Cipher-Decryption $P = \mathfrak{D}(C,K)$.

input: K is a 256-bit key; C is a 256×256 8-bit grayscale image block
Output: P is a 256×256 8-bit grayscale image block
$(\mathbf{Q_1}, \mathbf{Q_2}) = \mathrm{KDSG}(K, 8)$
for $n = 7:-1:0$ **do**
  if $n == 7$ **then**
    $L_8 = \mathrm{LSG}\left(Q_1^8, Q_2^8\right)$
    $D_8 = L_8(0,0)$
    $P_{\mathrm{LSW}} = \mathrm{DCR}_w(L_8, C, D_8)$
  end if
  $L_n = \mathrm{LSG}(Q_1^n, Q_2^n)$
  $D_n = L_n(0,0)$
  $P_{\mathrm{LSP}} = \mathrm{DCR}_p(L_n, P_{\mathrm{LSW}})$
  if $mod(n,2) \neq 0$ **then**
    $P_{\mathrm{LSS}} = \mathrm{DCR}_s^{col}(L_n, P_{\mathrm{LSP}})$
  else
    $P_{\mathrm{LSS}} = \mathrm{DCR}_s^{row}(L_n, P_{\mathrm{LSP}})$
  end if
  $P_{\mathrm{LSW}} = \mathrm{DCR}_w(L_n, P_{\mathrm{LSS}}, D_n)$
end for
$P = P_{\mathrm{LSW}}$

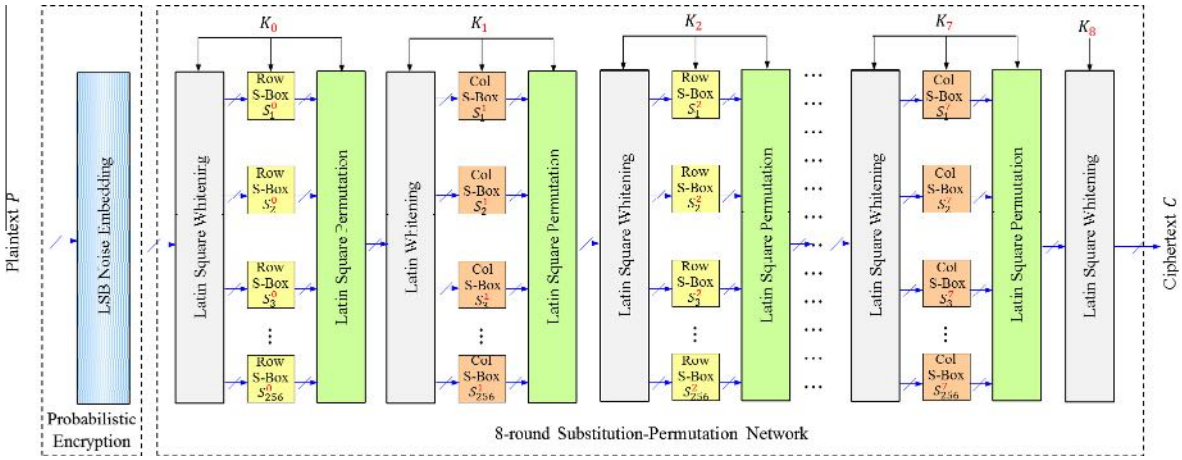


Fig. 2. The new Latin square image cipher.

3.2. LSB noise embedding

Probabilistic encryption [15,16] intends to use randomness in a cipher so that the cipher is able to encrypt one plaintext with the exact same encryption key into distinctive ciphertexts. It is well known that such randomness is crucial to achieve semantic security [7].

In this paper, we introduce such randomness by embedding noise in the least significant bit-plane of an image. More specifically, we XOR a randomly generated 256×256 bit-plane with the least significant bit-plane of the plaintext image, where the generation of this random bit-plane is completely independent on the encryption key.

Fig. 3 shows an example of LSB noise embedding. These introduced noise in LSB does not affect any image visual quality from the point view of human visual perceptibility. However, any slight change in plaintext here will lead to significant changes in ciphertext after it is encrypted by the SPN.

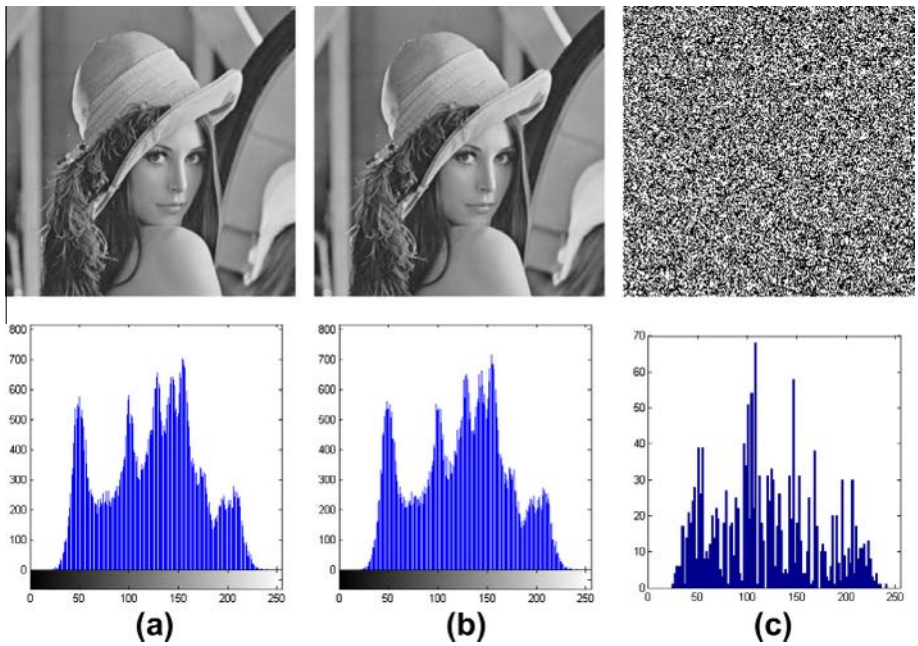


Fig. 3. Noise embedding in LSB. (a) plaintext *Lenna P* with histogram (b) noise embedded plaintext P' with histogram, and (c) $|P - P'|$ with the difference of histograms.

3.3. Key translation

In conventional block ciphers, keys are directly used without translation, e.g. in key whitening process, but the proposed LSIC uses a 256-bit encryption key K with key translation to eight key-dependent Latin squares of the order 256 before using in the LSIC. Specifically speaking, for a given 256-bit encryption key K, we.

1. Divide the encryption key K using function SKD into eight 32-bit subkeys, i.e.
$$K = [k_0, k_1, \ldots, k_7]$$
2. Generate pairs of pseudo-random sequences $\left(Q_1^0, Q_2^0\right), \left(Q_1^1, Q_2^1\right), \ldots, \left(Q_1^8, Q_2^8\right)$, each pair with 2×256 elements by using PRNGs with 32-bit states and these subkeys as seeds.
3. Generate key-dependent Latin squares, i.e. $L_0, L_1, \ldots, L_8$ with the order of 256 by feeding these pseudo-random sequences in Algorithm 1. Namely, $\forall n \in \{0, 1, \ldots, 8\}$, we have
$$L_n = \mathrm{LSG}\left(Q_1^n, Q_2^n\right)$$

The first two steps can be realized via Algorithm 2 with encryption key K and $M = 8$.

Algorithm 4. Key Dependent Sequence Generator $(\mathbf{Q_1}, \mathbf{Q_2}) = \mathrm{KDSG}(Key, M)$.

Input: K is a 256-bit key; n is a nonnegative integer
Output: $\mathbf{Q_1} = \left\{Q_1^0, \ldots, Q_1^M\right\}$ and $\mathbf{Q_2} = \left\{Q_2^0, \ldots, Q_2^M\right\}$ are n-element set of random sequences, each of a length 256.

```
K_0 = K
for n = 0 : 1 : M do
  [k_0, k_1, ..., k_7] = SKD(K_n)
  for i = 0 : 1 : 8 do
    q^i(0) = PRNG(k_i)
    for j = 1 : 1 : 63 do
      q^i(j) = PRNG(q^i(j − 1))
    end for
  end for
  Q_1^n = [q^0(0 : 31), q^1(0 : 31), ..., q^7(0 : 31)]
  Q_2^n = [q^0(32 : 63), q^1(32 : 63), ..., q^7(32 : 63)]
  K_{n+1} = [q^0(63), q^1(63), ..., q^7(63)]
end for
```

3.4. Latin square whitening

In the conventional SPN for block ciphers, the *Whitening* stage normally mixes a plaintext message P with a round key, e.g. XOR operation in [1,2], such that.

- The statistics of the plaintext message P is redistributed after mixing.
- The relationship between the ciphertext and encryption key is quite complicated and involved.

In image encryption, a plaintext message is an image block, P, composed of a number of pixels. Each pixel is represented by several binary bits (a byte). Therefore, XOR whitening scheme become inefficient for image data. It requires an encryption key with the same size as a plaintext image. It also needs to impose bitwise XOR to byte pixels. This type of image encryption is called a naive algorithm in [50]. Since the objective of key whitening is to mix plaintext data with encryption keys, we therefore define key whitening as a transposition cipher [32] in the finite field $GF(2^8)$ for image data, as shown in Eq. (4)

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

where x is a byte in plaintext, l is a corresponding byte in the keyed Latin square, y is the whitening result and $[.]_{2^8}$ denotes the computations over $GF(2^8)$. The whitening process above can be easily reversed by applying Eq. (5).

$$x = [y + l]_{2^8} \tag{5}$$

In image encryption, the plaintext byte x is a pixel, say it is located at the intersection of rth row and cth column *i.e.* $x = P(r,c)$. Let $l = L(r,c)$ be an element located at the corresponding position in the keyed Latin square L, and y be the ciphertext byte with $y = C(r,c)$, then we have the pixel-level equation

$$\begin{cases} C(r,c) = [\mathrm{SR}(P(r,c),[D]_3) + L(r,c)]_{2^8} \\ P(r,c) = \mathrm{SR}([C(r,c) + L(r,c)]_{2^8},[D]_3) \end{cases} \tag{6}$$

where symbol n denotes the current round number ($n \in [0,7]$), $D = L(0,0)$ is the rotating parameter, and SR denotes that the *spatial rotating* function (X,d) rotates an image X according to different values of the direction d as defined in Eq. (7)

$$Y = \mathrm{SR}(X,d) = \begin{cases} X & \text{if } d = 0 \\ \text{Flip } X \text{ up} \rightarrow \text{down} & \text{if } d = 1 \\ \text{Flip } X \text{ left} \rightarrow \text{right} & \text{if } d = 2 \end{cases} \tag{7}$$

Notice that if $Y = \mathrm{SR}(X,d)$, then the following identity always holds

$$X = \mathrm{SR}(Y,d) \tag{8}$$

Applying the key whitening for all pixels using the pixel-level Eq. (6), the *Latin Square Whitening* (LSW) in the image-level then can be denoted as

$$\mathrm{LSW}: \begin{cases} C = \mathrm{ECR}_w(L,P,D) \\ P = \mathrm{DCR}_w(L,C,D) \end{cases} \tag{9}$$

Therefore, we can restore the plaintext image block P from the ciphertext image block C using Eq. (9).

Fig. 4 shows an example of *Latin Square Whitening*, where the first row shows images and the second row shows corresponding histograms of these images. From this example, it is easy to verify that the ciphertext image after the *Latin Square Whitening* is unrecognizable and its pixels are redistributed to uniform-like.

3.5. Latin square row and column bijections

Since Eqs. (2) and (3) hold, each row and each column in a Latin square L of the order N is a permutation of the integer number sequence $[0,1,\ldots,N-1]$. We can define bijections (one-to-one and onto mapping) by mapping this integer number sequence to either a row or a column in a Latin square. In other words, we are able to construct forward and inverse row mapping functions (FRM and IRM) with respect to the rth row in L as shown in Eq. (10), and also forward and inverse column mapping functions (FCM and ICM) with respect to the cth column in L as shown in Eq. (11), where x and y denote the input and output of the mapping functions, respectively.

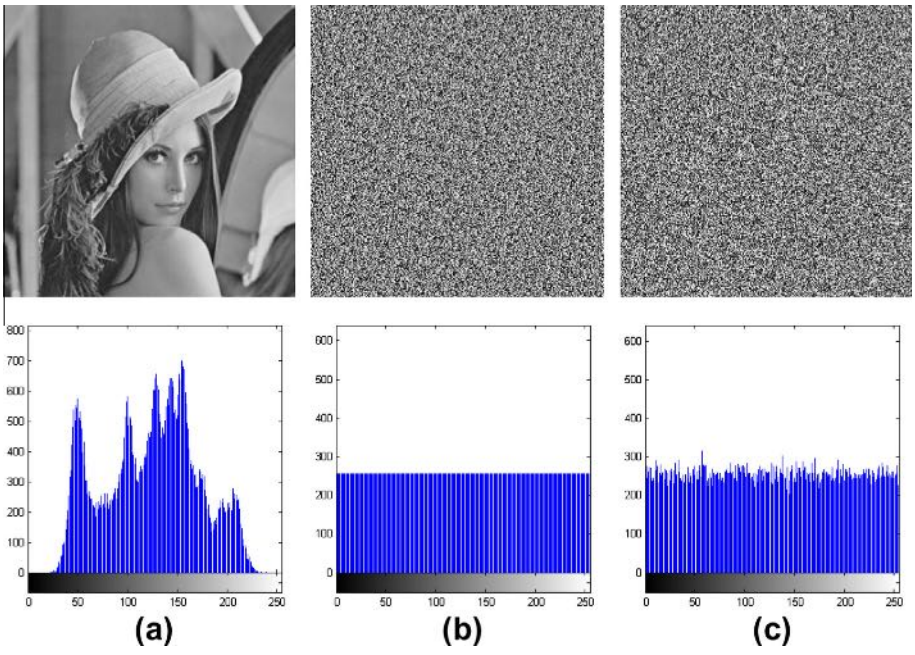


Fig. 4. A *Latin Square Whitening* example. (a) plaintext *Lenna P*, (b) reference Latin square L, and (c) ciphertext $C = \mathrm{ECR}_w(L,P,0)$.

$$\begin{cases} y = \text{FRM}\ (L, r, x) = L(r, x) \\ x = \text{IRM}\ (L, r, y) = \arg\max_{z \in \mathbb{N}} (f_L(r, z, y)) \end{cases} \tag{10}$$

$$\begin{cases} y = \text{FCM}\ (L, x, c) = L(x, c) \\ x = \text{ICM}\ (L, y, c) = \arg\max_{z \in \mathbb{N}} (f_L(z, c, y)) \end{cases} \tag{11}$$

where f_L is the tri-tuple function defined in Eq. (1). Its maximum is equal to 1, i.e. $f_L(r, x, y) = 1$, only for the column number x satisfying the constraint, $L(r, x) = y$. Furthermore, we have row mapping identities held for arbitrary x and y within a Latin square L:

$$\begin{cases} \text{IRM}\ (L, r, \text{FRM}\ (L, r, x)) = x \\ \text{FRM}\ (L, r, \text{IRM}\ (L, r, y)) = y \end{cases} \tag{12}$$

Similarly, we also have column mapping identities as follows:

$$\begin{cases} \text{ICM}\ (L, \text{FCM}\ (L, x, c), c) = x \\ \text{FCM}\ (L, \text{ICM}\ (L, y, c), c) = y \end{cases} \tag{13}$$

Fig. 5 shows the FRM and FCM functions defined by a Latin square. As can be seen, given a row number r, the effect of the forward row mapping is to use this Latin square as a look-up table and find the corresponding element in row r. Similarly, given a column number c, the effect of the forward column mapping to find the corresponding element in column c in this Latin square. Such nice property is directly from the constructional constraint in a Latin square.

Since an Nth order Latin square has N rows and N columns, there are N bijections in the N row directions and another N bijections in the N column directions in the Latin square. A bijection can be directly used as a P-Box [32] and can also serve as a S-Box [32]. We therefore are able to construct S-Boxes and P-Boxes from a Latin square.

3.6. Latin square substitution

In a cryptographic system, an S-Box is a basic component performing the byte substitution. Each S-Box can be defined as a bijection, also known as a one-to-one and onto mapping. In image encryption, an image pixel is commonly represented as a byte, i.e. a sequence of bits. For example, 8-bit grayscale image has 256 gray intensity scales with each intensity scale represented in an 8-bit sequence.

Because of the existence of FRM/IRM and FCM/ICM bijections in a Latin square, we are able to perform byte substitution in an image cipher using bijections in the row and column directions in a Latin square. The substitution with respect to a row in a Latin square is called *Latin Square Row S-box* (LSRS) in this paper:

$$LSRS: \begin{cases} C = \text{ECR}_s^{row}(L, P) \\ P = \text{DCR}_s^{row}(L, C) \end{cases} \tag{14}$$

For the pixel-level function of the LSRS, each ciphertext byte is determined by the FRM function (see Eq. (10)) using the keyed Latin square L with function parameters given by plaintext bytes and ciphertext bytes as follows

$$\text{ECR}_s^{row}: C(r, c) = \begin{cases} \text{FRM}\ (L, C(r-1, c), P(r, c)) & \text{if } r \neq 0 \\ \text{FRM}\ (L, 0, P(r, c)) & \text{if } r = 0 \end{cases} \tag{15}$$

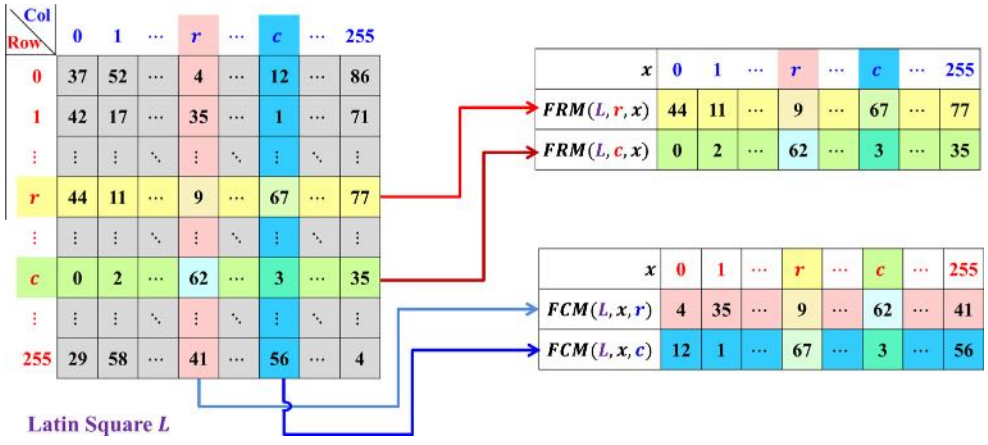


Fig. 5. Examples of building forward row mappings and forward column mappings using a Latin square.

Clearly, the plaintext bytes then can be perfectly restored from the ciphertext bytes, if we use IRM instead of FRM as follows

$$\mathrm{DCR}_s^{row} : P(r,c) = \begin{cases} \mathrm{IRM}(L, C(r-1,c), C(r,c)) & \text{if } r \neq 0 \\ \mathrm{IRM}(L, 0, C(r,c)) & \text{if } r = 0 \end{cases} \tag{16}$$

Similarly, we use bijections from columns in a Latin square to perform byte substitutions. It is called *Latin Square Column S-box* (LSCS), i.e.

$$\mathit{LSCS} : \begin{cases} C = \mathrm{ECR}_s^{col}(L,P) \\ P = \mathrm{DCR}_s^{col}(L,C) \end{cases} \tag{17}$$

and the corresponding LSCS encryption and decryption processes can also be defined as:

$$\mathrm{ECR}_s^{col} : C(r,c) = \begin{cases} \mathrm{FCM}(L, P(r,c), C(r,c-1)) & \text{if } c \neq 0 \\ \mathrm{FCM}(L, P(r,c), 0) & \text{if } c = 0 \end{cases} \tag{18}$$

$$\mathrm{DCR}_s^{col} P(r,c) = \begin{cases} \mathrm{ICM}(L, C(r,c), C(r,c-1)) & \text{if } c \neq 0 \\ \mathrm{ICM}(L, C(r,c), 0) & \text{if } c = 0 \end{cases} \tag{19}$$

Fig. 6 shows the encryption results of *Latin Square Row S-box* and *Latin Square Column S-box*. As can be seen, the plaintext image block P becomes unrecognizable after applying either the LSRS or LSCS. Histogram analysis also shows that the statistics of the image pixel intensity changes dramatically after substitution.

The Latin square row/column substitution above has excellent diffusion properties. One pixel change in the plaintext P will diffuse to a column/row of pixels after a round of the LSRS or LSCS. This diffusion quickly spreads to the entire ciphertext image in several iterations.

3.7. Latin square permutation

Unlike a S-Box performing byte substitution, a P-Box performs byte shuffling or scrambling. Each P-Box can also be defined as a bijection [32].

If we consider both input x and output y in FRM and IRM as indices (see Eq. (10)), then FRM defines a mapping $\{0,1,\ldots,255\} \to \{0,1,\ldots,255\}$ and IRM defines the corresponding inverse mapping. We therefore are able to define the *Latin square row p-box* (LSRP) with respect to rows in a Latin square L as follows,

$$\mathrm{LSRP} : \begin{cases} C(r,c_y) = P(r, \mathrm{FRM}\ (L,r,c_x)) \\ P(r,c_x) = C(r, \mathrm{IRM}\ (L,r,c_y)) \end{cases} \tag{20}$$

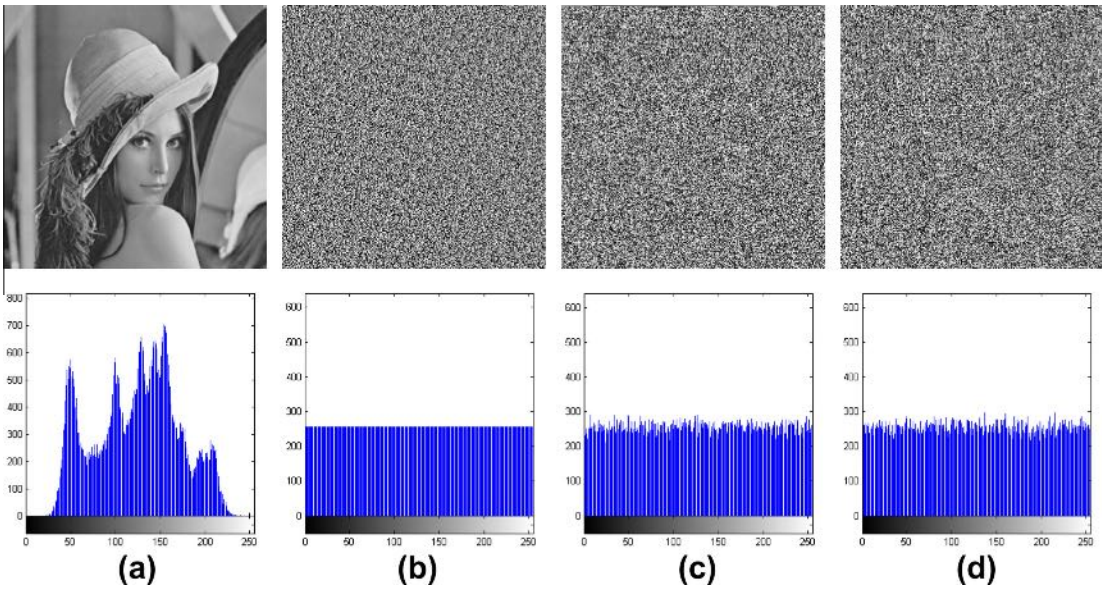


Fig. 6. A *Latin Square Substitution* example. (a) plaintext P: *Lenna Image*, (b) Latin square L, (c) ciphertext $C_r = \mathrm{ECR}_s^{row}(L,P)$, and (d) Ciphertext $C_c = \mathrm{ECR}_s^{col}(L,P)$.

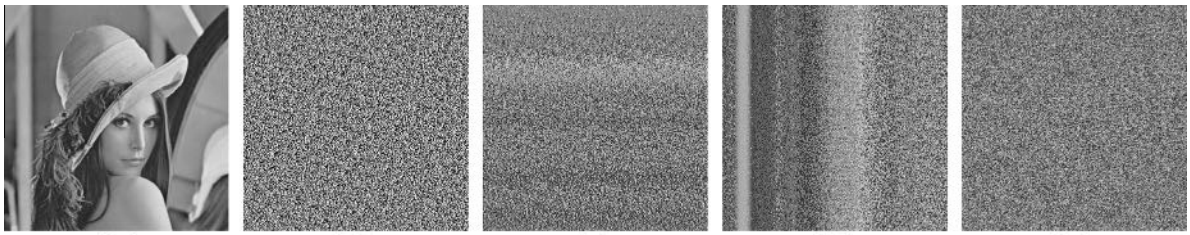


Fig. 7. A *Latin Square Permutation* example. (a) plaintext *Lenna P*, (b) Latin square *L*, (c) ciphertext with only LSRP (d) ciphertext with only LSCP, and (e) ciphertext with LSP $C = \mathrm{ECR}_p(L, P)$.

where c_x and c_y denotes the column indices before and after mapping. Consequently, for any pixel in P and its corresponding pixel in C are in the same row r after LSRP; and the column indices change before and after mapping with the relationship of $c_y = \mathrm{FRM}\ (L, r, c_x)$.

Similarly, we can also construct *Latin Square Column P-box* (LSCP) with respect to the columns in a Latin square as

$$\mathrm{LSCP}: \begin{cases} C(r_y, c) = P(\mathrm{FCM}\ (L, r_x, c), c) \\ P(r_x, c) = C(\mathrm{ICM}\ (L, r_y, c), c) \end{cases} \tag{21}$$

In order to achieve better performance, we construct our *Latin Square Permutation* by cascading LSRPs and LSCPs as follows

$$C(r, c) = C^*(\mathrm{FCM}\ (L, r, c), c) \tag{22}$$

$$C^*(r, c) = P(r, \mathrm{FRM}\ (L, r, c)) \tag{23}$$

In general, we write this *Latin Square Permutation* function as

$$\mathrm{LSP}: \begin{cases} C = \mathrm{ECR}_p(L, P) \\ P = \mathrm{DCR}_p(L, C) \end{cases} \tag{24}$$

Fig. 7 shows the permutation results of using the LSRP, LSCP and LSP. It is clear that cascading LSRP and LSCP in the LSP helps to achieve a better pixel permutation performance. Pixels in its ciphertext image become more random-like and make the ciphertext image content unintelligible.

4. Simulation results

4.1. Experiment settings and dataset

We perform the following simulations using MATLAB *R2010a* in a computer with a Windows 7 Operating System, 6 Gb RAM and Intel Core2 2.66 GHz CPU. Test images are from the *Miscellaneous* dataset in the USC-SIPI image database (detailed database and image descriptions can be found at[2]). The open-source LSIC implementation can be found on the webpage https://sites.google.com/site/tuftsyuewu/source-code.

Although we believe that the performance of an image cipher should be tested over a fairly large dataset, e.g. the USC-SIPI *Miscellaneous* dataset, the analysis of many peer image encryption methods are merely about a small number of test images, e.g. the *Lenna* image. Therefore, we perform fair comparisons between the proposed LSIC and peer methods [9,13,14,17,21,22,24,25,33,37,38,49,51,55,55]. in three following aspects:

- If peer methods have performance reports on the *Lenna* image, we compare the performance of the LSIC with those methods on the identical *Lenna* image.
- If peer methods have the accessible source code [9] or the executive file (e.g. bmpPacker[3]) online, we compare the performance of these methods with the LSIC over the entire *Miscellaneous* dataset.
- If peer methods have performance reports also on the entire *Miscellaneous* dataset [24], we directly cite these reports for our comparisons.

Furthermore, we turn off LSBNE function in all simulations because we want to:

[2] The USC-SIPI image database is a public image database held by the University of South California, available at http://sipi.usc.edu/database/ as the date of 08/18/2013.

[3] bmpPacker is an encryption software containing many classic block ciphers written by Jens Gödeke. Only its AES cipher is used in our simulations. This freeware is available on: http://www.jens-goedeke.eu/tools/bmppacker/ as the date of 08/18/2013.

- prevent the possible unfair comparisons, because other compared algorithms do not have this stage
- emphasize the contributions of the Latin square encryption primitives rather than those from LSBNE

However, the probabilistic encryption stage does enhance the LSIC's security. Without this stage, encrypting a plaintext image P twice using the same security key K will always obtain two identical ciphertext images C_1 and C_2, i.e. $C_1 = C_2$. This ciphertext information provides an adversary plaintext information that unauthorized users should never know. Therefore, the LSIC with LSBNE is able to encrypt a plaintext image into distinctive ciphertext images because of the embedded noise from time to time are not necessarily the same and thus completely prevent the risk of information leakage in the deterministic encryption.

4.2. Simulation results

Image encryption results using the proposed LSIC for color and grayscale images are shown in Figs. 8 and 9, respectively. As can be seen, the proposed LSIC is able to.

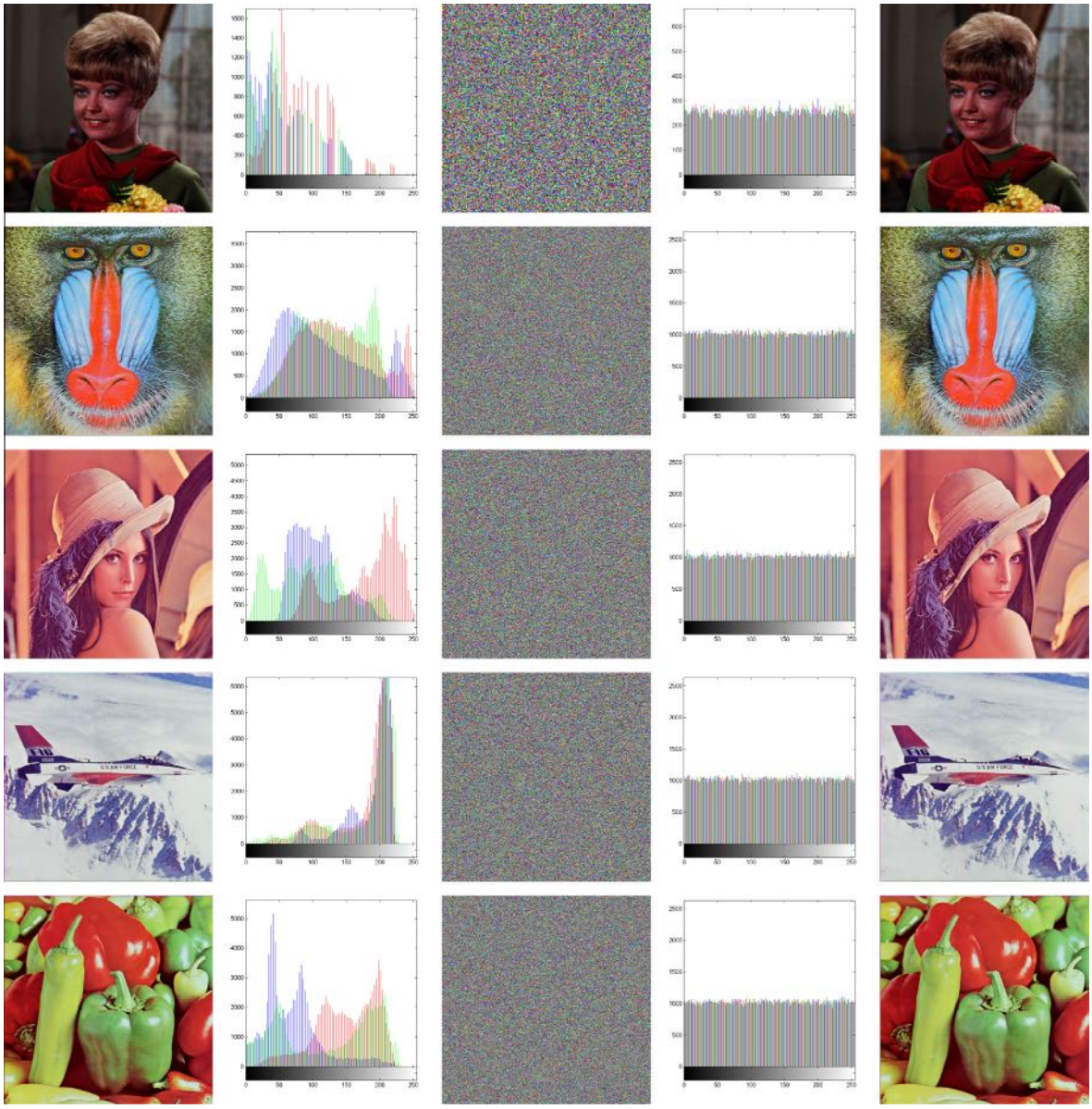


Fig. 8. Sample results of encrypting and decrypting color images using the Latin square image cipher. The 1st column: plaintext images (filenames from top to bottom are 4.1.01, 4.2.03, 4.2.04, 4.2.05 and 4.2.07, respectively); 2nd column: histograms of plaintext images; 3rd column: ciphertext images; 4th column: histograms of ciphertext images; and 5th column: deciperted text images.

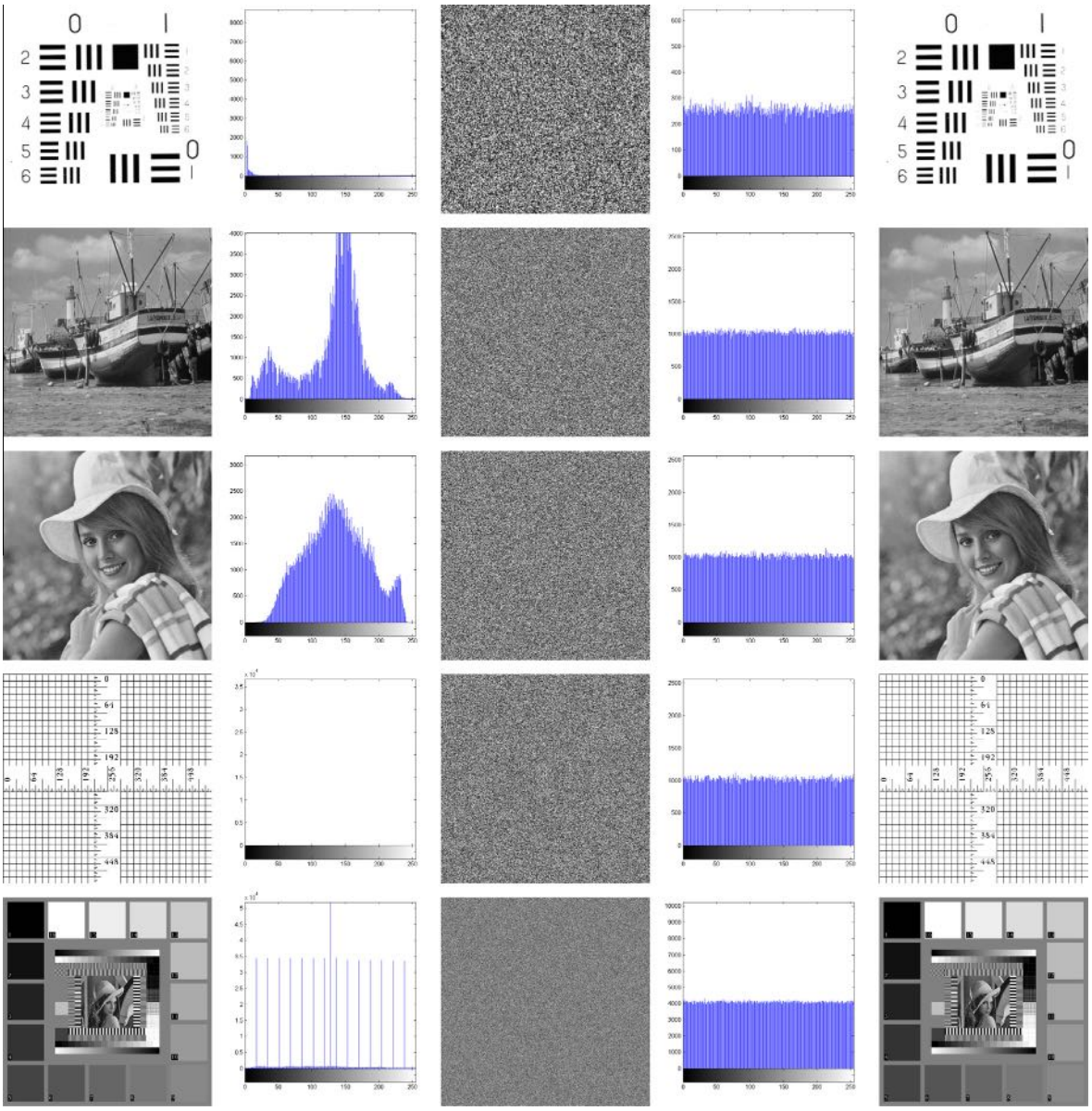


Fig. 9. Sample results of encrypting and decrypting gray images using the Latin square image cipher. The 1st: plaintext images (filenames from top to bottom are 5.1.13, boat.512, elaine.512, ruler.512, and testpad.1k, respectively); 2nd column: histograms of plaintext images; 3rd column: ciphertext images; 4th column: histograms of ciphertext images; and 5th column: deciperted text images.

- generate unintelligible ciphertext images no matter what contents or types of the plaintext images;
- make the statistics of ciphertext images to be uniform-like;
- perfectly reconstruct the plaintext images from the corresponding ciphertext images.

It is also worthwhile to note that the test images *5.1.13, ruler.512* and *testpat.1k* are extremely difficult cases for image encryption because they have large size homogeneous regions, i.e. they have extremely tilted histograms. However, the proposed LSIC encrypts those images successfully.

With regards to the LSIC complexity, for each cipher round, we need 2 operations/pixel (o/p) to generate a 256×256 keyed Latin square, 2 o/p to apply *Latin square whitening*, 2 o/p to implement *Latin square permutation*, and 2 o/p to perform *Latin square substitution* using this keyed Latin square. Thus, the proposed LSIC requires 8 o/p for each cipher round.

In simulations, the actual time complexity of the proposed LSIC is 1.2331 ± 0.10089 seconds for a 256×256-byte image block computed from all test images in the USC-SIPI *Miscellaneous* dataset with the MATLAB implementation. Equivalently, the LSIC's encryption/decryption speed is about 3.32 Mb/s per round under MATLAB. In regard to the decryption speed, the

proposed LSIC is able to decrypt each 256×256 image block (pixel in 8-bit byte) with 0.7137 ± 0.06222 s (equivalently 5.74 Mb/s per round) for all test encrypted images from the USC-SIPI *Miscellaneous* dataset. The decryption speed is faster than encryption because *Latin Square S-boxes* defined in Eqs. (14) and (17) are asymmetric for encryption and decryption (see Fig. 14).

5. Security analysis

A good image encryption algorithm/cipher should be able to resist all known types of attacks and cryptanalysis. Its performance should be independent on a used encryption key or a plaintext image. First of all, this cipher should be theoretically secure against known attacks. Secondly, a good image encryption algorithm/cipher should have both *confusion property* and *diffusion property*, which are proposed in [29] as criteria for secure ciphers. The *confusion property* implies the cipher's ability to encrypt an arbitrary plaintext image into random-like ciphertext, so that an adversary cannot extract any information of the plaintext image from ciphertext images. The *diffusion property* implies the cipher has capability to diffuse even a slight change in a plaintext image over the entire ciphertext image. Furthermore, a good image encryption algorithm/cipher should also be secure with respect to attacks against encryption keys [32] and robust under noisy environment [53]. In this section, we discuss the security of the proposed LSIC under all above analysis and deeply study the LSIC's performance with extensive simulation results and comparisons.

5.1. Theoretical analysis

5.1.1. Brute-force attack

In cryptography, a brute-force attack, or exhaustive key searching, is a strategy that is able to crack any encrypted data by searching all possible keys in the key space until the right key is found. It is well known that the key length, i.e. the size of the key space used in an encryption system, determines the practical feasibility of performing a brute-force attack: longer keys mean more difficulties than shorter ones in the terms of the time complexity.

The proposed LSIC has a key space of 256 bits, which is larger than or equal to the mainstream cipher standards such as DES [1] and AES [2], where the later one with 192-bit key is known to be large enough to resist the brute-force attacks. Therefore, the proposed LSIC also has a sufficient large key space against the brute-force attack.

It is worthwhile to note the theoretical key space of the proposed LSIC has an even larger key space than 256-bit. As stated in Section 3, the proposed LSIC depends on eight 256×256 Latin squares. Regardless of all other methods for generating these Latin squares, the Latin square generator in Algorithm 1, where a Latin square is uniquely determined by two length 256 permutation sequence, can generate $256! \times 256! \approx 2^{3368}$ number of Latin squares. Consequently, the theoretical key size is about $(2^{3368})^8 = 2^{26944}$, i.e. about 26,944 bits. This implies that the adopted key length in the proposed LSIC can be easily increased to withstand the brute-force attacks when the current 256-bit key length is not long enough in the future. Therefore, the proposed LSIC has an extremely large key space against the brute-force attacks.

5.1.2. Ciphertext and plaintext attacks

In cryptography, the ciphertext and plaintext are commonly used in attack models to analyze the security of an encryption system. Specifically speaking, we consider a ciphertext-only attack assuming that an adversary is only able to access a set of ciphertext, a known plaintext attack assuming that an adversary is able to access a set of plaintexts and corresponding ciphertexts, and a chose-plaintext attack assuming that an adversary is able to access arbitrary plaintexts to be encrypted and obtain the corresponding ciphertexts. As can be seen from attack assumptions, the chosen-plaintext attack provides an adversary the most information about plaintexts and ciphertexts among three attack models. Therefore, if a cipher is able to resist chosen-plaintext attacks, it is also immune to ciphertext-only and known-plaintext attacks.

Many components of LSIC is designed with considerations to resist chosen-plaintext attacks:

- Nonlinear Key Translation: Instead of directly applying the key for encryption in a conventional block cipher, e.g. key whitening, an encryption key used in the proposed LSIC is first translated to eight 256×256 Latin squares, and then used for constructing key dependent encryption primitives for image encryption. Therefore, the relationship between the key and ciphertext is more involved and complicated than the conventional key usage.
- Probabilistic Encryption: The proposed LSIC has a pre-process using probabilistic encryption by introducing a small amount of noise to the LSB of a plaintext image. As a result, the LSIC is able to generate significantly different ciphertext images even when the key and the plaintext image are unchanged. It also ensures the deciphered images visually unchanged. This helps the LSIC to achieve semantic security [7].
- Dynamic S-boxes and P-boxes: All S-boxes and P-boxes in the proposed LSIC are dependent on Latin squares in each round. They are dynamically changed instead of fixed from key to key. This implies that, even if an adversary is able to crack an encryption key, all other keys are still safe because P-boxes and S-boxes associated with other keys have not been unbroken yet. They ensure the security of ciphertext images by other keys.

Consequently, the proposed LSIC has good resistance to ciphertext and pliantext attacks.

5.2. Confusion property analysis

The confusion property [29] of a secure cipher emphasizes the ability of making the relationship between the key and the ciphertext as complex and involved as possible. If a cipher has good confusion property, it is very difficult to find the encryption key, even if a large number of plaintext–ciphertext pairs produced by the same encryption key are available for an adversary. In other words, a cipher with good confusion property should be able to withstand the known-plaintext attacks [32] and ciphertext-only attacks [32]. To demonstrate the excellent confusion property of the proposed Latin square image cipher, we perform two types of statistical analysis: entropy and correlation.

5.2.1. Information entropy analysis

Information entropy is a quantitative measure of the randomness of a signal. Because a digital image is a type of digital signal, the entropy analysis can be used to measure the randomness of a test image [43,48]. The information entropy of an image can be defined as Eq. (25), where X denotes the test image, x_i denotes the ith possible value in X, and $\Pr(x_i)$ is the probability of $X = x_i$, i.e. the probability of pulling a random pixel in X and its value is x_i. The maximum of $H(X)$ is achieved when X is uniformly distributed as shown in Eq. (26), i.e. X has a complete flat histogram and the symbol F denotes the number of allowed intensity scales associated with the image format. For example, $F = 256$ is the number of intensity scales of a gray-scale image or a RGB color image. (see Table 1)

$$H(X) = -\sum_{i=1}^{n} \Pr(x_i) \log_2 \Pr(x_i) \tag{25}$$

$$\Pr(X = x_i) = 1/F \tag{26}$$

For the test images in simulation, $F = 256$ and therefore the upper-bound of information entropy is 8. The comparison results of information entropy between the proposed LSIC and peer algorithms are listed in Table 2, where the method with the highest entropy score is shaded for each test image. These results show that the LSIC outperforms listed peer image encryption algorithms/methods.

The results of information entropy analysis for the complete set of test images are listed in Table 3. These results illustrate that the proposed LSIC (1) obtains ciphertext images whose information entropy scores much closer to the entropy upper-bound than those encrypted by other methods; (2) is effective and robust to different image contents and pixel intensity statistics; and (3) outperforms the conventional AES cipher implemented by bmpPacker and the method proposed in [9] in most cases.

5.2.2. Adjacent pixel correlation analysis

Adjacent pixel correlation (APC) is a quantitative measure of the image randomness. It measures the correlations between adjacent pixels within an image. The adjacent pixel correlation can be defined as an autocorrelation function as shown in Eq.

Table 1
Comparison results of information entropy of the *Lenna.* image.

Method	Entropy
Chen et al. [9]	7.9938
Xiang et al. [49]	7.9950
Wong et al. [38] (reported in [33])	7.9690
Zhang et al. [51]	7.9980
Zhu et al. [55]	7.9901
Sun et al. [33]	7.9965
Wu et al. [46]	7.9970
Zhou et al. [52]	7.9969
LSIC	7.997161

Table 2
Comparison results of information entropy of color images.

P	Channel\Method	Entropy			
		Wang et al. [37]	Liu et al. [22]	Patidar et al. [25]	LSIC
$512 \times 512 \times 3$	Red	7.999324	7.9871	7.9957	7.999351
4.2.04	Green	7.999371	7.9802	7.9963	7.999416
(*ColorLenna*)	Blue	7.999292	7.9878	7.9951	7.999314
$512 \times 512 \times 3$	Red	N/a	7.9877	7.9952	7.999309
4.2.07	Green	N/a	7.9881	7.9959	7.999285
(*ColorPepper*)	Blue	N/a	7.9877	7.9954	7.999258

Table 3
Comparison results of information entropy analysis.

File	Plaintext	Ciphertext		
		bmpPacker[a]	Chen [9]	LSIC
4.1.01	6.898139	7.988561	7.998430	7.998963
4.1.02	6.294498	7.979298	7.999061	7.999027
4.1.03	5.970916	7.990187	7.999113	7.999122
4.1.04	7.426957	7.990089	7.999032	7.998948
4.1.05	7.068625	7.981035	7.998891	7.999140
4.1.06	7.537089	7.989820	7.998902	7.998936
4.1.07	6.583485	7.990487	7.999033	7.999241
4.1.08	6.852716	7.997651	7.999182	7.999199
4.2.01	7.242831	7.998747	7.999760	7.999808
4.2.02	6.416494	7.997469	7.999763	7.999756
4.2.03	7.762436	7.997581	7.999752	7.999765
4.2.04	7.750197	7.997733	7.999772	7.999799
4.2.05	6.663908	7.997463	7.999746	7.999756
4.2.06	7.762170	7.997550	7.999747	7.999789
4.2.07	7.669826	7.908851	7.999781	7.999766
5.1.09	6.709312	7.906739	7.997011	7.997266
5.1.10	7.311807	7.941872	7.997144	7.997667
5.1.11	6.452275	7.929405	7.996520	7.996775
5.1.12	6.705667	7.363568	7.996729	7.996730
5.1.13	1.548314	7.903557	7.997143	7.997299
5.1.14	7.342433	7.992544	7.997271	7.997022
5.2.08	7.201008	7.987446	7.999257	7.999359
5.2.09	6.993994	7.984631	7.999199	7.999240
5.2.10	5.705560	7.998720	7.999303	7.999250
5.3.01	7.523737	7.998599	7.999801	7.999820
5.3.02	6.830330	7.990318	7.999831	7.999819
7.1.01	6.027415	7.989542	7.999321	7.999353
7.1.02	4.004499	7.983057	7.999302	7.999284
7.1.03	5.495740	7.993621	7.999192	7.999325
7.1.04	6.107418	7.983570	7.999313	7.999315
7.1.05	6.563196	7.985406	7.999296	7.999302
7.1.06	6.695283	7.985253	7.999268	7.999279
7.1.07	5.991599	7.990304	7.999221	7.999437
7.1.08	5.053448	7.983746	7.999236	7.999371
7.1.09	6.189814	7.985088	7.999271	7.999412
7.1.10	5.908790	7.998478	7.999235	7.999391
7.2.01	5.641454	7.985051	7.999829	7.999811
boat.512	7.191370	7.989566	7.999401	7.999353
elaine.512	7.505984	6.492821	7.999231	7.999252
gray21.512	4.392295	7.979908	7.999772	7.999364
house	7.485787	7.997553	7.999781	7.999786
numbers.512	7.729247	6.893247	7.999201	7.999263
ruler.512	0.500033	7.903134	7.999298	7.999258
testpat.1k	4.407726	7.903132	7.999800	7.999809
Statistics	# Best	0	12	32
	Mean	7.905055	7.999049	7.999105
	Std	0.290957	0.000903	0.000856

[a] $q^i(j_1 : j_2)$ denotes a vector of pseudo-random numbers with elements $[q^i(j_1), q^i(j_1+1), \ldots, q^i(j_2-1), q^i(j_2)]$.

(27), where X_t denotes an extracted pixel sequence and X_{t+1} denotes a pixel sequence where each pixel is the adjacent pixel of the corresponding pixel in X_t, μ is the mean value defined by Eq. (28) and σ is the standard deviation defined by Eq. (29), the definition of mathematical expectation is given in Eq. (30).

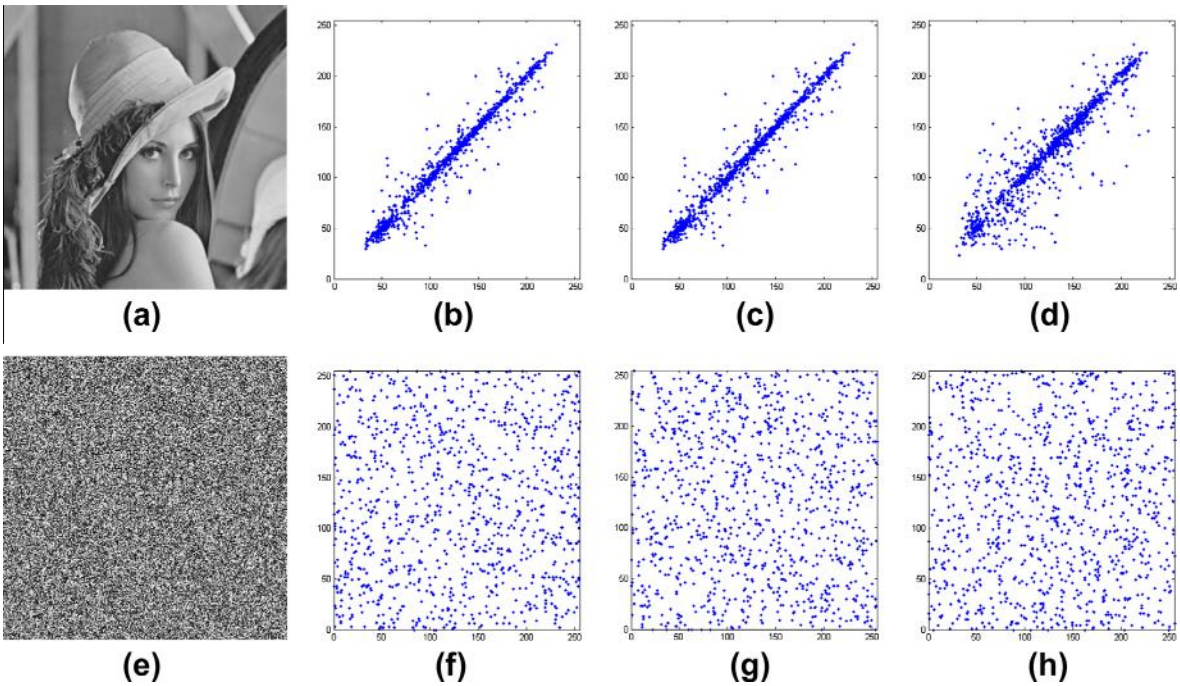


Fig. 10. Adjacent pixel correlations before and after encryption. (a) plaintext *Lenna P*, (b) horizontal adjacent pixels in *P*, (c) vertical adjacent pixels in *P*, (d) diagonal adjacent pixels in *P*, (e) ciphertext $C = \mathfrak{E}(P, K)$, (f) horizontal adjacent pixels in *C*, (g) vertical adjacent pixels in *C*, and (h) diagonal adjacent pixels in *C*.

Table 4
Comparison results of APC of the *Lenna* image.

Encryption method	Horizontal	Vertical	Diagonal
Original *Lenna*	0.9400	0.9709	0.9710
Chen et al. [9]	−0.00024	−0.24251	0.23644
Liao et al. [21]	0.0127	−0.0190	−0.0012
Fu et al. [14]	0.0368	0.0392	0.0068
Wu et al. [46]	−0.0002150	0.0014913	0.0040264
Zhou et al. [52]	−0.0054	0.0045	0.0031
LSIC	0.0053365	−0.0027616	0.0016621

$$APC = \mathrm{E}[(X_t - \mu)(X_{t+1} - \mu)]/\sigma^2 \tag{27}$$

$$\mu = \mathrm{E}[X] \tag{28}$$

$$\sigma = \sqrt{\mathrm{E}[(X - \mu)^2]} \tag{29}$$

$$\mathrm{E}[x] = \sum_{i=1}^{N} x_i / N \tag{30}$$

The closer to zero the correlation coefficient is, the weaker relationship between the pixel sequence and its adjacent pixel sequence. Because the adjacent pixel sequence can be extracted from the horizontal, vertical, or diagonal direction, the adjacent pixel correlation will be analyzed with respect to these three directions.

Fig. 10 shows the APC plots of the randomly selected 1024 pairs of adjacent pixels along the horizontal, vertical and diagonal directions. In the APC plots, the horizontal axis denotes the intensity of one set of randomly selected pixels while the vertical axis denotes the intensity of its corresponding set of adjacent pixels. The APC results for the *Lenna* image are listed in Table 4. The proposed LSIC decorrelates adjacent pixels in the plaintext image after the encryption process. It outperforms the listed encryption algorithms [9,14,17,21] due to its smaller correlation coefficients.

Table 5 shows the APCs for test plaintext images and their corresponding ciphertext images using the LSIC and peer algorithms, where the best APC score is shaded for each test image. These results further demonstrate that the proposed LSIC outperforms peer algorithms in most trails with the smallest mean and standard deviation of APCs.

Table 5
Comparison results of adjacent pixel correlation analysis.

File	Plaintext	Ciphertext				
		bmp-Packer[a]	Mao [13]	Chen [9]	Pareek [24]	LSIC
4.1.01	955.730	11.397	1.780	1.240	3.961	1.382
4.1.02	926.227	13.327	3.000	1.973	8.119	1.384
4.1.03	922.433	24.657	1.760	2.170	5.314	1.956
4.1.04	959.193	9.840	1.443	0.883	11.863	1.097
4.1.05	953.143	11.130	0.640	1.437	9.243	0.432
4.1.06	932.417	26.303	1.487	0.910	3.994	0.208
4.1.07	979.317	10.723	2.050	1.680	1.905	2.572
4.1.08	972.013	11.790	1.393	2.187	2.922	0.722
4.2.01	988.877	6.757	0.790	0.427	7.686	0.533
4.2.02	945.423	3.790	1.137	1.050	8.184	1.402
4.2.03	857.587	6.603	1.083	0.897	7.076	1.036
4.2.04	978.600	6.940	1.523	0.743	2.325	0.557
4.2.05	943.307	7.493	1.067	0.587	2.833	2.199
4.2.06	959.510	7.117	1.263	0.977	8.145	1.950
4.2.07	974.480	5.347	1.290	0.990	0.815	0.146
5.1.09	911.973	59.040	3.053	5.233	0.779	4.829
5.1.10	853.567	61.290	7.663	7.397	7.672	0.188
5.1.11	890.580	33.700	2.580	4.567	4.110	1.766
5.1.12	954.440	45.793	4.097	4.943	11.780	3.443
5.1.13	831.833	162.693	4.160	3.697	17.896	2.424
5.1.14	892.687	67.513	4.340	1.407	8.989	4.538
5.2.08	884.630	10.943	1.600	2.137	6.210	0.615
5.2.09	850.460	18.137	1.817	1.253	6.024	0.168
5.2.10	917.130	21.857	1.403	1.070	1.512	2.442
5.3.01	974.543	3.887	1.400	1.203	0.297	0.752
5.3.02	890.127	3.523	0.863	0.730	1.944	0.158
7.1.01	926.903	10.853	1.890	2.217	6.857	0.670
7.1.02	928.663	13.270	3.263	1.747	6.561	1.017
7.1.03	925.900	22.620	0.957	1.877	11.244	0.352
7.1.04	958.447	8.640	1.297	0.780	2.139	2.735
7.1.05	914.000	22.010	1.857	1.397	6.582	1.407
7.1.06	908.887	22.580	1.373	1.170	1.338	0.331
7.1.07	866.260	21.397	0.483	1.577	2.992	0.432
7.1.08	934.933	12.813	1.257	2.833	6.279	0.045
7.1.09	935.977	24.920	0.633	1.213	10.011	0.025
7.1.10	946.177	20.797	2.547	1.707	6.344	2.182
7.2.01	951.530	5.330	1.180	0.613	4.317	0.273
boat.512	942.427	19.453	1.207	0.887	9.823	2.120
elaine.512	969.757	14.950	1.607	2.777	7.839	4.381
house	993.220	116.010	1.213	2.507	10.950	0.757
gray21.512	941.383	5.990	1.590	1.280	12.766	1.359
numbers.512	692.123	26.230	2.257	1.733	11.103	1.948
ruler.512	313.253	51.003	1.947	0.890	3.126	1.270
testpat.1k	752.093	45.560	1.557	1.067	0.720	0.703
Statistics	#Best	0	2	17	2	23
	Mean	25.364	1.882	1.820	6.195	1.384
	Std	30.331	1.275	1.406	3.968	1.222

[a] $q^i(j_1 : j_2)$ denotes a vector of pseudo-random numbers with elements $[q^i(j_1), q^i(j_1+1), \ldots, q^i(j_2-1), q^i(j_2)]$.

5.3. Diffusion property analysis

The diffusion property describes the cipher's ability of diffusing a change in a plaintext image over its corresponding ciphertext image. If a cipher is weak in diffusion, it might be vulnerable against the differential attacks [32]. The number of changing pixel rate (NPCR) and the unified averaged changed intensity (UACI) are two most common quantitative methods to evaluate an image encryption method/algorithm/cipher for resist differential attacks [9,21,44,55].

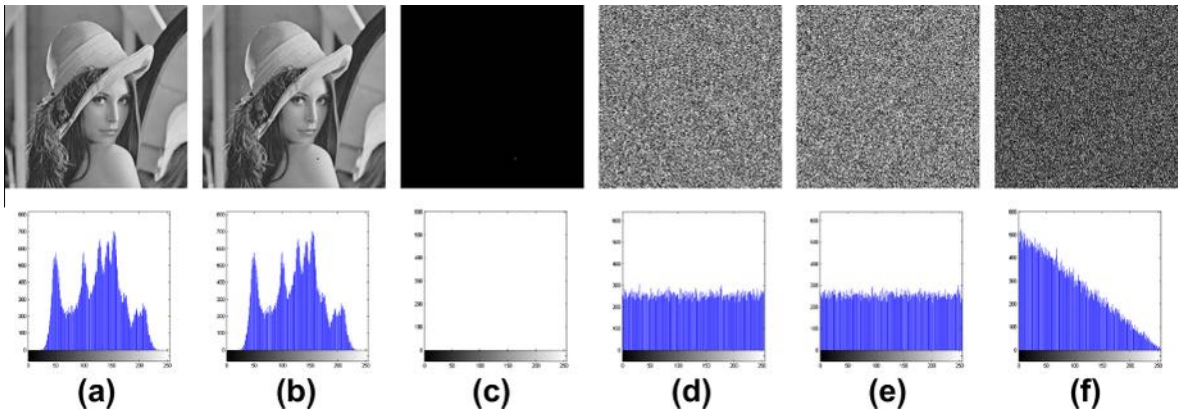


Fig. 11. Sample results of the LSIC's diffusion property. (a) original plaintext *Lenna* P^1, (b) modified plaintext *Lenna* P^2, (c) plaintext differences $|P^1 - P^2|$, (d) ciphertext $C^1 = \mathfrak{E}(P^1, K)$, (e) ciphertext $C^2 = \mathfrak{E}(P^2, K)$, and (f) ciphertext difference $|C^1 - C^2|$.

Table 6
Comparison results of the NPCR and UACI scores of the *Lenna* image.

Encryption method	UACI%	*P*-value	Result	NPCR%	*P*-value	Result
Chen et al. [9]	33.14	0.0005	Fail	99.25	0.0000	Fail
Liao et al. [21]	33.48	0.8587	Success	99.65	0.9523	Success
Zhu et al. [55]	33.48	0.8587	Success	99.63	0.8014	Success
Wu et al. [46]	33.39	0.4263	Success	99.58	0.1140	Success
Zhou et al. [52]	33.24	0.0156	Success	99.60	0.3502	Success
LSIC	33.4936	0.7450	Success	99.6689	0.9927	Success

Fig. 11 shows the sample results of the LSIC's diffusion property. The plaintext image P^1 differs from P^2 only one pixel on *Lenna*'s shoulder. C^1 and C^2 are corresponding ciphertext images using the same encryption key. As can be seen, a single pixel difference between P^1 and P^2 diffuses to the entire cipher image and leads to a significant difference between C^1 and C^2.

Mathematically, the NPCR, $\mathcal{N}(C^1, C^2)$, and UACI, $\mathcal{U}(C^1, C^2)$, can be defined as Eqs. (31) and (32), respectively. They measure two ciphertext images, C^1 and C^2, whose plaintext images are slightly different. Their difference Diff (i,j) defined in Eq. (33) denotes whether two pixels located at the image grid (i,j) of C^1 and C^2 are equal. The symbols T and S denote the number of pixels in the ciphertext image and the number of allowed pixel intensity scales, respectively.

$$\mathcal{N}(C^1, C^2) = \sum_{i=1}^{M}\sum_{j=1}^{N} \frac{\text{Diff }(i,j)}{T} \times 100\% \tag{31}$$

$$\mathcal{U}(C^1, C^2) = \sum_{i=1}^{M}\sum_{j=1}^{N} \frac{|C^1(i,j) - C^2(i,j)|}{S \cdot T} \times 100\% \tag{32}$$

$$\text{Diff }(i,j) = \begin{cases} 0, & \text{if } C^1(i,j) = C^2(i,j) \\ 1, & \text{if } C^1(i,j) \neq C^2(i,j) \end{cases} \tag{33}$$

It is noticeable that NPCR concentrates on the absolute number of pixels which changes values in differential attacks, while the UACI focuses on the averaged difference between the paired ciphertext images. Moreover, both NPCR and UACI scores can also be interpreted qualitatively, namely we can use these scores to test whether one can distinguish these ciphertext images from truly random ones. In particular, [44] demonstrated that NPCR and UACI follow normal distributions shown in Eqs. (34) and (35), respectively.

$$\text{NPCR} \sim \text{N}\left(\mu_{\text{npcr}}, \sigma^2_{\text{npcr}}\right), \text{ with } \mu_{\text{npcr}} = \frac{S-1}{S} \text{ and } \sigma^2_{\text{npcr}} = \frac{S-1}{TS^2} \tag{34}$$

$$\text{UACI} \sim \text{N}\left(\mu_{\text{uaci}}, \sigma^2_{\text{uaci}}\right), \text{ with } \mu_{\text{uaci}} = \frac{S+1}{3S} \text{ and } \sigma^2_{\text{uaci}} = \frac{(S+1)(S^2+2)}{18T(S-1)S^2} \tag{35}$$

Table 7
Comparison results of the LSIC's diffusion properties.

File	UACI%	*P*-value	Result	NPCR%	*P*-value	Result
4.1.01	33.4402	0.6618	Success	99.5885	0.0689	Success
4.1.02	33.3715	0.0846	Success	99.6272	0.8974	Success
4.1.03	33.513	0.3540	Success	99.6297	0.9257	Success
4.1.04	33.5303	0.2110	Success	99.6023	0.3075	Success
4.1.05	33.4018	0.2473	Success	99.6028	0.3201	Success
4.1.06	33.5235	0.2612	Success	99.5956	0.1637	Success
4.1.07	33.5855	0.0223	Success	99.6302	0.9306	Success
4.1.08	33.4874	0.6548	Success	99.6089	0.4865	Success
4.2.01	33.4763	0.6325	Success	99.6058	0.3056	Success
4.2.02	33.4431	0.4436	Success	99.6179	0.8872	Success
4.2.03	33.5031	0.1382	Success	99.612	0.6455	Success
4.2.04	33.5113	0.0735	Success	99.6146	0.7712	Success
4.2.05	33.4475	0.5477	Success	99.6231	0.9745	Success
4.2.06	33.4145	0.0661	Success	99.6113	0.6078	Success
4.2.07	33.4708	0.7856	Success	99.6031	0.1862	Success
5.1.09	33.6739	0.0229	Success	99.617	0.6228	Success
5.1.10	33.5039	0.6624	Success	99.617	0.6228	Success
5.1.11	33.5079	0.6313	Success	99.5529	0.0102	Success
5.1.12	33.4536	0.9143	Success	99.6033	0.4016	Success
5.1.13	33.3681	0.3018	Success	99.6033	0.4016	Success
5.1.14	33.6094	0.1146	Success	99.5636	0.0301	Success
5.2.08	33.5379	0.1076	Success	99.6067	0.4131	Success
5.2.09	33.3704	0.0439	Success	99.6151	0.6808	Success
5.2.10	33.5169	0.2483	Success	99.5987	0.1905	Success
5.3.01	33.5039	0.0807	Success	99.6115	0.6364	Success
5.3.02	33.4856	0.3398	Success	99.6077	0.3917	Success
7.1.01	33.5084	0.3317	Success	99.5918	0.0746	Success
7.1.02	33.4485	0.7448	Success	99.6124	0.5980	Success
7.1.03	33.5236	0.1938	Success	99.6346	0.9808	Success
7.1.04	33.5454	0.0765	Success	99.6029	0.2975	Success
7.1.05	33.3958	0.1427	Success	99.6063	0.4004	Success
7.1.06	33.4756	0.7942	Success	99.6262	0.9164	Success
7.1.07	33.4469	0.7188	Success	99.5953	0.1240	Success
7.1.08	33.5569	0.0434	Success	99.6128	0.6107	Success
7.1.09	33.3964	0.1463	Success	99.5953	0.1240	Success
7.1.10	33.3942	0.1335	Success	99.588	0.0397	Success
7.2.01	33.4813	0.4422	Success	99.6142	0.7858	Success
boat.512	33.3792	0.0680	Success	99.5941	0.1050	Success
elaine.512	33.443	0.6567	Success	99.6185	0.7731	Success
gray21.512	33.4651	0.9534	Success	99.6174	0.8730	Success
house	33.457	0.8874	Success	99.6063	0.4004	Success
numbers.512	33.4484	0.7432	Success	99.6258	0.9112	Success
ruler.512	33.4471	0.7220	Success	99.6002	0.2257	Success
testpat.1k	33.4748	0.6261	Success	99.6018	0.1068	Success

Table 6 shows corresponding NPCR and UACI scores, randomness test P-values and test results of C^1 and C^2 using the proposed LSIC and other encryption algorithms.[4] Table 7 shows the detailed NPCR and UACI scores, test *P*-values and results for the entire test image set. As one can see, all tested images passed both the NPCR and UACI randomness tests, indicating that the encrypted images are undistinguishable from random images under these two tests.

5.4. Key sensitivity analysis

A secure cipher should have high key sensitivities in both encryption and decryption processes. Simulation results with respect to encryption and decryption stages are shown in Figs. 12 and 13. The encryption keys in our simulations are listed below in the HEX format:

$$K^1 = \text{B9B5ED7585C8B15D7454ED271AA3A3A3A07B00321C11759D0FDE340234384BC9}$$

$$K^2 = \text{B9B5ED7585C8B15D7454ED271AA3A3A3A07B00321C11759D0FDE340234384BC8}$$

$$K^3 = \text{39B5ED7585C8B15D7454ED271AA3A3A3A07B00321C11759D0FDE340234384BC8}$$

[4] NPCR and UACI randomness tests are introduced in [44] and its implementation is available from. The test significance levels for both tests are set to 0.01, i.e. if a test *P*-value is smaller than 0.01, this score then fails to pass the test.

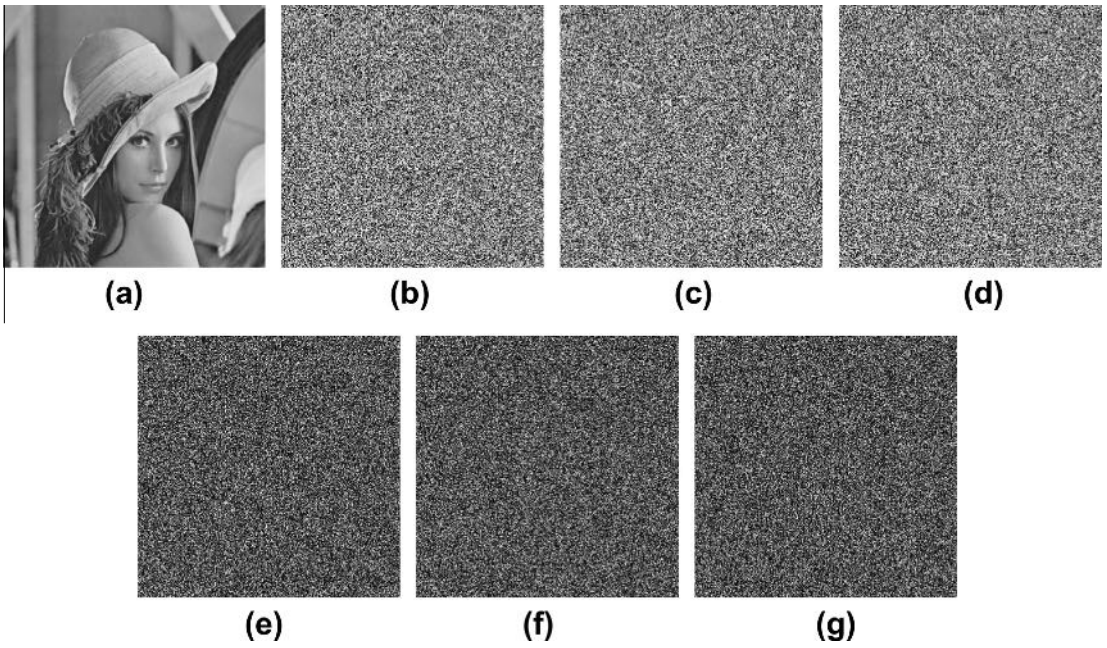


Fig. 12. Key sensitivity analysis at the encryption stage. (a) plaintext P *Lenna*, (b) ciphertext $C^1 = \mathfrak{E}(P, K^1)$, (c) ciphertext $C^2 = \mathfrak{E}(P, K^2)$, (d) ciphertext $C^3 = \mathfrak{E}(P, K^3)$, (e) ciphertext difference $|C^1 - C^2|$, (f) ciphertext difference $|C^2 - C^3|$, and (g) ciphertext difference $|C^3 - C^1|$.

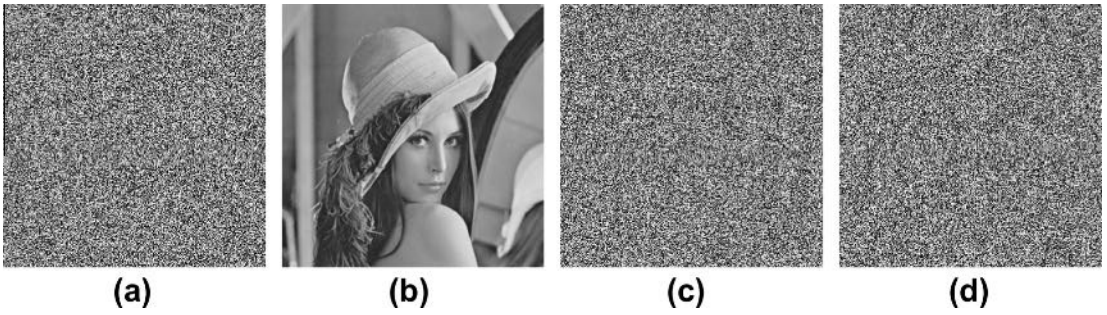


Fig. 13. Key sensitivity analysis at the decryption stage. (a) ciphertext C^1, (b) deciphered text $D^1 = \mathfrak{D}(C^1, K^1)$, (c) deciphered text $D^2 = \mathfrak{D}(C^1, K^2)$, and (d) deciphered text $D^3 = \mathfrak{D}(C^1, K^3)$.

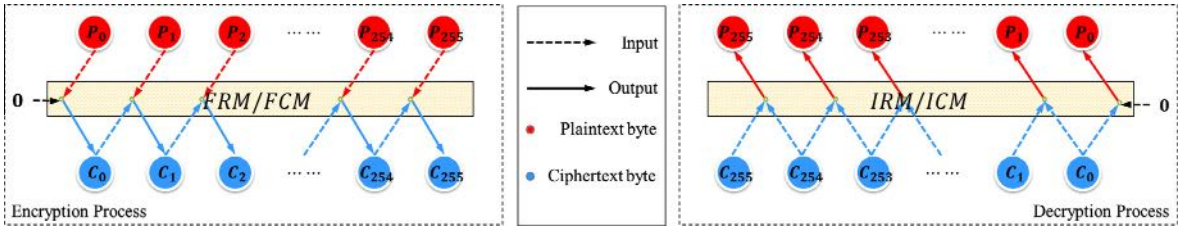


Fig. 14. *Latin Square Substitution* has an asymmetric structure for encryption and decryption.

As can be seen, K^1 differs K^2 only for the last bit; K^2 differs K^3 only for the first bit; and K^1 differs K^3 only for the first and the last bit. Although the hamming distances between K^1, K^2 and K^3 are very small, i.e. they are very similar to each other, their corresponding ciphertext images C^1, C^2 and C^3 have significant differences. These can be verified by the fact that their differences, i.e. $|C^1 - C^2|$, $|C^2 - C^3|$ and $|C^3 - C^1|$ are random-like images as shown in Fig. 12. The example shows the proposed LSIC is sensitive to the encryption keys in the encryption stage.

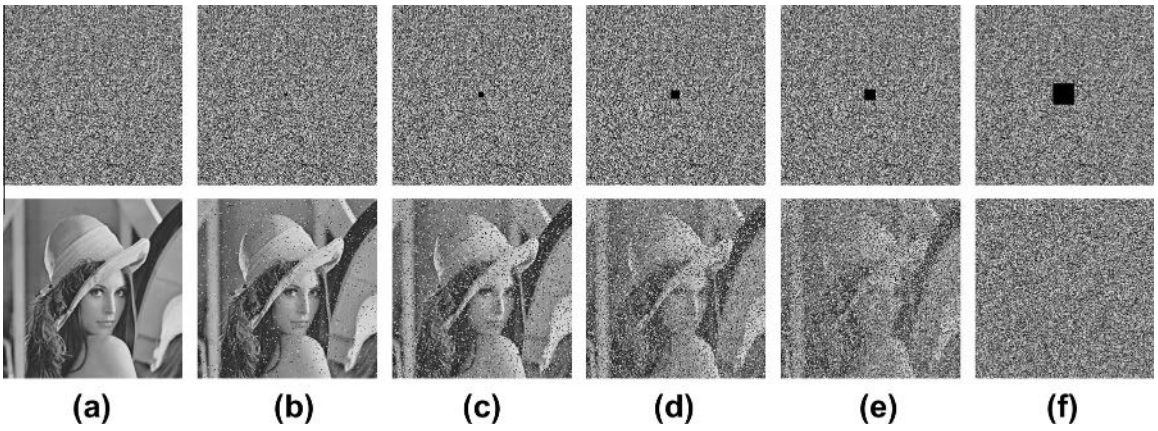


Fig. 15. Sample results of the LSIC's noise robustness in the decryption process. (a) ciphertext C and its decrypted text $D = \mathfrak{D}(C, K)$, (b) C^1 with 0.025% noise and $D^1 = \mathfrak{D}(C^1, K)$, (c) C^2 with 0.1% noise and $D^2 = \mathfrak{D}(C^2, K)$, (d) C^3 with 0.22% noise and $D^3 = \mathfrak{D}(C^3, K)$, (e) C^4 with 0.39% noise and $D^4 = \mathfrak{D}(C^4, K)$, and (f) C^5 with 1.56% noise and $D^5 = \mathfrak{D}(C^5, K)$.

Similar results can also be obtained in the decryption stage as shown in Fig. 13. We decrypt the same ciphertext image C^1 using the encryption keys K^1, K^2 and K^3, respectively. As can be seen, using the correct encryption key K^1, decrypted image D^1 perfectly reconstructs the original plaintext image. However, decrypted images D^2 and D^3 using key K^2 and K^3 are random-like ones which do not contain any information related to the original plaintext image.

5.5. Error tolerance analysis

Due to possible noise in channels or coding errors, one cannot guarantee that an encrypted image sent on one side is identical to the image received on the other side. Unless an image cipher is able to tolerate errors a certain level of errors, it is impossible for a receiver to correct decrypt the received encrypted image with small errors even with a right key. Therefore, a good cipher is favored to tolerate errors.

In practice, the *Latin Square Substitution* is of an asymmetric structure as shown in Fig. 14. Encrypting one plaintext byte requires one plaintext byte and one previous ciphertext byte, while decrypting one ciphertext byte requires two adjacent ciphertext byte but no plaintext byte. This process is similar to weave a thread of ciphertext bytes on the ciphertext image, which has to be done by intersecting the longitudinal threads of the plaintext bytes.

As a result, changing one pixel in a plaintext image influence all pixels after it in its row in the first encryption round, further all pixels after these influenced pixels in the corresponding columns in the second encryption round, and then more pixels in the third encryption round, so on and so forth until the last round. As a result, a small change in the plaintext image will lead to a completely different ciphertext image. In contrast, changing one pixel in a ciphertext image only leads a two-pixel-change in the first decryption round, and at most $2^8 = 256$ pixel changes in the deciphered image. Hence the *Latin Square Substitution* achieves both above-mentioned goals, and gains encryption diffusion properties against the plaintext change and decryption robustness against the ciphertext noise simultaneously. More results and examples will be provided in the future sections.

Fig. 15 shows the results of the LSIC's decryption robustness against various noise levels in ciphertext images. After the decryption process, noise concentrating in the center square of a ciphertext image distributed almost evenly over the deciphered image. Due to the psychovisual redundancy within an image, human vision system is still able to recognize the deciphered image contents as long as it is not fully unintelligible.

6. Conclusion

In this paper, we have introduced a symmetric-key Latin square image cipher with probabilistic encryption. This new LSIC has distinctive characteristics: (1) the LSIC is purely defined in integers, and thus it can be easily implemented in software and hardware without causing finite precision or descritization problems; (2) the LSIC constructs all encryption primitives based on one keyed Latin square, including whitening, substitution and permutation, and thus the LSIC attains high sensitivities to any key change; (3) the LSIC encrypts image pixels in the unit of byte instead of bit and processes a 256×256 plaintext image at one time, and thus it is efficient for image data; (4) the LSIC arranges all these encryption primitives in the framework of substitution–permutation network, and thus it attains good confusion and diffusion properties [29]; (5) the LSIC also integrates probabilistic encryption allowing to encrypt one plaintext image into different ciphertext images with one encryption key; and (6) the LSIC's decryption stage is robust against a certain level of errors.

The LSIC's effectiveness and robustness have been demonstrated by extensive simulation results using the complete USC-SIPI *Miscellaneous* image dataset. Theoretical security analysis has shown that the LSIC is able to withstand the brute-force attacks, ciphertext-only attacks, known-plaintext attacks and chosen-plaintext attacks. Experimental security analysis with comparisons to peer algorithms have indicated that the LSIC outperforms or reaches state of the art. All these analysis and results have demonstrated that the LSIC is very suitable for digital image encryption.

The essence of the LSIC is to use a short key to first trigger large size Latin squares, then to construct randomized encryption primitives according to these keyed Latin squares for future image encryption. Following this main pipeline, it is not difficult to see that we can easily adjust the used key length and Latin square sizes for a particular application. One open question is whether '*weak*' Latin squares exist, though we failed to find any so far. If such '*weak*' Latin squares exist, then the next questions is how to recognize them in an early stage. Eventually, we need some means to evaluate the suitability of a Latin square for image encryption. However, none of these question is answered in this paper.

Acknowledgement

This work was supported in part by the Macau Science and Technology Development Fund under Grant 017/2012/A1 and by the Research Committee at University of Macau under Grants MYRG113(Y1-L3)-FST12-ZYC and MRG001/ZYC/2013/FST.

References

[1] Data Encryption Standard, Federal Information Processing Standards Publication 46 (1977).
[2] Advanced Encryption Standard, Federal Information Processing Standards Publication 197 (2001).
[3] L. Aaronson, Sudoku science, IEEE Spect. 43 (2006) 16–17.
[4] G. Àlvarez, S. Li, L. Hernandez, Analysis of security problems in a medical image encryption system, Comput. Biol. Med. 37 (2007) 424–427.
[5] G. Àlvarez, F. Montoya, M. Romera, G. Pastor, Cryptanalysis of a discrete chaotic cryptosystem using external key, Phys. Lett. A 319 (2003) 334–339.
[6] R. Anderson, B. Schneier, Description of a new variable-length key, 64-bit block cipher (blowfish), in: Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish), Springer, Berlin/Heidelberg, 1994, pp. 191–204.
[7] M. Bellare, A. Desai, E. Jokipii, P. Rogaway, A concrete security treatment of symmetric encryption, in: The 38th Annual Symposium on Foundations of Computer Science, pp. 394–403.
[8] G. Bhatnagar, Q.J. Wu, B. Raman, Discrete fractional wavelet transform and its application to multiple encryption, Inform. Sci. 223 (2013) 297–316.
[9] G. Chen, Y. Mao, C.K. Chui, A symmetric image encryption scheme based on 3D chaotic cat maps, Chaos, Solitons Fract. 21 (2004) 749–761.
[10] R.J. Chen, J.L. Lai, Image security system using recursive cellular automata substitution, Pattern Recog. 40 (2007) 1621–1631.
[11] T.H. Chen, C.S. Wu, Compression-unimpaired batch-image encryption combining vector quantization and index compression, Inform. Sci. 180 (2010) 1690–1701.
[12] W. Chen, X. Chen, Optical image encryption using multilevel Arnold transform and noninterferometric imaging, Opt. Eng. 50 (2011) 117001.
[13] E.B. Corrochano, Y. Mao, G. Chen, Chaos-based image encryption, in: Handbook of Geometric Computing, Springer, Berlin Heidelberg, 2005, pp. 231–265.
[14] C. Fu, B. bin Lin, Y. sheng Miao, X. Liu, J. jie Chen, A novel chaos-based bit-level permutation scheme for digital image encryption, Optics Commun. 284 (2011) 5415–5423.
[15] E. Fujisaki, T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, in: Advances in Cryptology CRYPTO' 99, vol. 1666, Springer, Berlin/Heidelberg, 1999, pp. 79–79.
[16] Z. Galil, S. Haber, M. Yung, Symmetric public-key encryption, in: Advances in Cryptology CRYPTO 85 Proceedings, LNCS 218, Springer Berlin/Heidelberg, 1986, pp. 128–137.
[17] A. Kumar, M.K. Ghose, Extended substitution–diffusion based image cipher using chaotic standard map, Commun. Nonlin. Sci. Numer. Simul. 16 (2011) 372–382.
[18] C. Li, L.Y. Zhang, R. Ou, K.W. Wong, S. Shu, Breaking a novel colour image encryption algorithm based on chaos, Nonlin. Dynam. 70 (2012) 2383–2388.
[19] S. Li, G. Chen, X. Mou, On the dynamical degradation of digital piecewise linear chaotic maps, Int. J. Bifurcation Chaos 15 (2005) 3119–3151.
[20] S. Li, X. Mou, Y. Cai, Z. Ji, J. Zhang, On the security of a chaotic encryption scheme: problems with computerized chaos in finite computing precision, Comput. Phys. Commun. 153 (2003) 52–58.
[21] X. Liao, S. Lai, Q. Zhou, A novel image encryption algorithm based on self-adaptive wave transmission, Signal Process. 90 (2010) 2714–2722.
[22] H. Liu, X. Wang, Color image encryption using spatial bit-level permutation and high-dimension chaotic system, Optics Commun. 284 (2011) 3895–3903.
[23] O. Matoba, T. Nomura, E. Perez-Cabre, M. Millan, B. Javidi, Optical techniques for information security, Proc. IEEE 97 (2009) 1128–1148.
[24] N.K. Pareek, V. Patidar, K.K. Sud, Image encryption using chaotic logistic map, Image Vision Comput. 24 (2006) 926–934.
[25] V. Patidar, N.K. Pareek, G. Purohit, K.K. Sud, A robust and secure chaotic standard map based pseudorandom permutation–substitution scheme for image encryption, Optics Commun. 284 (2011) 4331–4339.
[26] W. Press, Numerical Recipes: The Art of Scientific Computing, Cambridge University Press, 2007.
[27] B. Schneier, The Twofish Encryption Algorithm: A 128-bit Block Cipher, John Wiley, 1999.
[28] S.M. Seyedzadeh, S. Mirzakuchaki, A fast color image encryption algorithm based on coupled two-dimensional piecewise chaotic map, Signal Process. 92 (2012) 1202–1215.
[29] C.E. Shannon, Communication theory of secrecy systems, Bell Syst. Tech. J. 28 (1949) 656–715.
[30] E. Solak, C. Çkal, Algebraic break of image ciphers based on discretized chaotic map lattices, Inform. Sci. 181 (2011) 227–233.
[31] E. Solak, C. Çokal, Comment on Encryption and decryption of images with chaotic map lattices, Chaos 18 (3) (2008) 038101 (31?).
[32] D. Stinson, Cryptography: Theory and Practice, Chapman and Hall CRC, 2006.
[33] F. Sun, Z. L, S. Liu, A new cryptosystem based on spatial chaotic system, Optics Commun. 283 (2010) 2066–2073.
[34] X. Tong, M. Cui, Image encryption scheme based on 3D baker with dynamical compound chaotic sequence cipher generator, Signal Process. 89 (2009) 480–491.
[35] C.K. Volos, I. Kyprianidis, I. Stouboulos, Image encryption process based on chaotic synchronization phenomena, Signal Process. 93 (2013) 1328–1340.
[36] X. Wang, L. Teng, X. Qin, A novel colour image encryption algorithm based on chaos, Signal Process. 92 (2012) 1101–1108.
[37] X. Wang, J. Zhao, H. Liu, A new image encryption algorithm based on chaos, Optics Commun. 285 (2012) 562–566.
[38] K. Wong, S. Ho, C. Yung, A chaotic cryptography scheme for generating short ciphertext, Phys. Lett. A 310 (2003) 67–73.
[39] Y. Wu, S.S. Agaian, J.P. Noonan, Sudoku Associated Two Dimensional Bijections for Image Scrambling, 2012, arXiv:1207.5856.
[40] Y. Wu, J.P. Noonan, S. Agaian, Binary data encryption using the sudoku block cipher, in: IEEE International Conference on Systems Man and Cybernetics, IEEE, pp. 3915–3921.

[41] Y. Wu, J.P. Noonan, S. Agaian, Dynamic and implicit latin square doubly stochastic s-boxes with reversibility, in: International Conference on Systems, Man, and Cybernetics, IEEE, pp. 3358–3364.
[42] Y. Wu, J.P. Noonan, S. Agaian, Image encryption using the rectangular sudoku cipher, in: International Conference on System Science and Engineering, IEEE, pp. 704–709.
[43] Y. Wu, J.P. Noonan, S. Agaian, A novel information entropy based randomness test for image encryption, in: International Conference on Systems, Man, and Cybernetics, IEEE, pp. 2676–2680.
[44] Y. Wu, J.P. Noonan, S. Agaian, NPCR and UACI randomness tests for image encryption, cyber journals: multidisciplinary journals in science and technology, Journal of Selected Areas in Telecommunications (JSAT) (2011) 31–38.
[45] Y. Wu, J.P. Noonan, S. Agaian, A wheel-switch chaotic system for image encryption, in: International Conference on System Science and Engineering, IEEE, pp. 23–27.
[46] Y. Wu, G. Yang, H. Jin, J.P. Noonan, Image encryption using the two-dimensional logistic chaotic map, J. Electron. Imag. 21 (2012). 013014-1.
[47] Y. Wu, Y. Zhou, J.P. Noonan, K. Panetta, S. Agaian, Image encryption using the Sudoku matrix, in: Mobile Multimedia/Image Processing, Security, and Applications 2010, vol. 7708, SPIE, 2010, p. 77080P.
[48] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J.P. Noonan, P. Natarajan, Local shannon entropy measure with statistical tests for image randomness, Inform. Sci. 222 (2013) 323–342.
[49] T. Xiang, X. Liao, G. Tang, Y. Chen, K. Wong, A novel block cryptosystem based on iterating a chaotic map, Phys. Lett. A 349 (2006) 109–115.
[50] M. Yang, N. Bourbakis, S. Li, Data-image-video encryption, IEEE Potentials 23 (2004) 28–34.
[51] Q. Zhang, L. Guo, X. Wei, Image encryption using DNA addition combining with chaotic maps, Math. Comput. Modell. 52 (2010) 2028–2035.
[52] Y. Zhou, L. Bao, C. Philip Chen, Image encryption using a new parametric switching chaotic system, Signal Process. 93 (11) (2013) 3039–3052.
[53] Y. Zhou, K. Panetta, S. Agaian, C.L.P. Chen, Image encryption using P-Fibonacci transform and decomposition, Optics Commun. 285 (5) (2012) 594–608.
[54] Y. Zhou, K. Panetta, S. Agaian, C.L.P. Chen, (n, k, p)-gray code for image systems, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet. 43 (2) (2013) 515–529.
[55] Z. liang Zhu, W. Zhang, K. wo Wong, H. Yu, A chaos-based symmetric image encryption scheme using a bit-level permutation, Inform. Sci. 181 (2011) 1171–1186.