Nonlinear Dyn (2021) 104:807–825
https://doi.org/10.1007/s11071-021-06308-3

ORIGINAL PAPER

Design and application of an *S*-box using complete Latin square

Zhongyun Hua · **Jiaxin Li** · **Yongyong Chen** · **Shuang Yi**

Received: 9 September 2020 / Accepted: 11 February 2021 / Published online: 17 March 2021

Abstract Since a substitution box (*S*-box) is the nonlinearity part of a symmetric key encryption scheme, it directly determines the performance and security level of the encryption scheme. Thus, generating *S*-box with high performance and efficiency is attracting. This paper proposes a novel method to construct *S*-box using the complete Latin square and chaotic system. First, a complete Latin square is generated using the chaotic sequences produced by a chaotic system. Then an *S*-box is constructed using the complete Latin square. Performance analyses show that the *S*-box generated by our proposed method has a high performance and can achieve strong ability to resist many security attacks such as the linear attack, differential attack and so on. To show the efficiency of the constructed *S*-box, this paper further applies the *S*-box to image encryption application. Security analyses show that the developed image encryption algorithm is able to encrypt different kinds of images into cipher images with uniformly distributed histograms. Performance evaluations demonstrate that it has a high security level and can outperform several state-of-the-art encryption algorithms.

Keywords *S*-box · Latin square · Chaotic system · Image encryption · Secure communication

Z. Hua (✉) · J. Li · Y. Chen
School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen, Shenzhen 518055, China
e-mail: huazyum@gmail.com

Y. Chen
e-mail: YongyongChen.cn@gmail.com

S. Yi
Engineering Research Center of Forensic Science, Chongqing Education Committee, College of Criminal Investigation, Southwest University of Political Science and Law, Chongqing 401120, China

1 Introduction

With the fast development of information technology, more and more multimedia data, including digital image, audio and video, are generated and spread over all kinds of networks. Particularly, the digital image is one of the most widely used data formats. Because a digital image can show much potential information, serious information security incident may happen when a secret image is accessed unauthorizedly. Therefore, it is very important to protect the contents of digital image. Recently, to securely communicate the digital images, researchers have developed many technologies such as data hiding [19], image encryption [1,14,44] and watermarking [47].

Among all the image security technologies, the image encryption is one of the most straightforward and effective methods [15,52,54]. An image encryption algorithm transforms a meaningful image into an unrecognizable cipher image. Only with the correct key, one can completely recover the original image. With a wrong key, one cannot obtain any useful information. Among all the techniques for image encryption, the chaos theory is the most widely used and effective one [29,42,55]. This is because chaos the-

ory has many similar properties with the principles of image encryption [13,48]. However, when a chaotic system is implemented in a digital platform, chaos degradation happens because of the limitation of precision. This results in that the image encryption schemes only depending on chaotic systems have many security issues [8,10]. One effective way to solve this issue is to combine the chaotic systems with other techniques. For example, Zhang et al. proposed an effective image encryption algorithm using the combination of the 3-cell chaotic map and biological operations [53]. The simulation results verify the high security level of the proposed algorithm. Besides, many image encryption algorithms have been proposed using other techniques such as the compressive sensing [36] and frequency domain transformation [30].

Generally, a substitution box (S-box) is a square matrix that takes a number of bits as input and transforms them into the same number of output bits [33]. It is one of the most important modules in an symmetric key encryption algorithm and can randomly change the pixel positions of an image. Thus the security level of an encryption algorithm is highly determined by its used S-box. Without a high-performance S-box, the encryption algorithm may be easily broken by many security attacks [35]. Thus, one key point of developing image encryption is to design S-box with high performance. Until to now, researchers have developed many methods to construct S-boxes. Because a chaotic system can generate randomly distributed sequences, it is widely used to develop S-boxes [40]. For example, an S-box with high performance is constructed using a two-dimensional chaotic map [21]. When chaotic systems are used to construct S-boxes, the properties of the chaotic system highly determine the performance of the constructed S-boxes. A chaotic system with poor performance may lead to the low security level of the constructed S-box. To avoid the negative effects of chaotic systems in constructing S-boxes, many techniques are combined with chaotic systems to design S-boxes [20]. For example, the authors in [9] designed S-boxes by combining the chaotic systems with genetic algorithms. Security analysis showed that the S-boxes constructed using the combined method have better performance than that using the chaotic systems only. However, existing S-boxes still have some weaknesses in efficiency, applicability and so on. It is meaningful to construct new S-boxes with better efficiency and performance.

To construct S-boxes with better efficiency and performance, this paper proposes an S-box generation method using the complete Latin square and chaotic system. First, a chaotic system is employed to construct a complete Latin square. Then, the S-box is generated from the complete Latin square and chaotic sequence. Performance evaluations show the high performance and security level of the constructed S-box. Moreover, an image encryption algorithm is further proposed using the S-boxes and the security analyses verify its high security level. The main contributions and novelty of this paper are summarized as follows.

(1) We propose an S-box generation method that can efficiently generate S-boxes using the combination of complete Latin square and chaotic system.
(2) The performance of the constructed S-box is analyzed and experimentally compared with many S-boxes generated by other methods from the aspects of nonlinearity, strict avalanche criterion, bit independence criterion. The results show the high performance and efficiency of the proposed S-box.
(3) To show the application of the constructed S-box, we further develop an image encryption algorithm that uses the S-box to perform nonlinear transformation to the image.
(4) Simulation results and security analysis show that the developed image encryption algorithm has a high security level to resist many security attacks. Comparison results demonstrate that the algorithm has better performance and a higher security level than some state-of-the-art image encryption algorithms.

The important notations used throughout the paper are shown in Table 1.

The rest of this paper is organized as follows. Section 2 introduces the enhanced logistic chaotic map and complete Latin square as background. Section 3 presents the S-box generation method and analyzes the performance and efficiency of its generated S-box. Section 4 develops an image encryption algorithm using the constructed S-box. Sect. 5 evaluates the security performance of the encryption algorithm. Section 6 concludes this paper.

2 Preliminaries

This section introduces an existing chaotic system, i.e., the enhanced logistic map, and the complete Latin

Table 1 Descriptions of important notations

Notation	Description
$\mathbf{I}$	An index vector generated by sorting chaotic sequence
$\mathbf{A}$	A row complete Latin square
$\mathbf{B}$	A complete Latin square
$\mathbf{C}$	A two-dimensional coordinate matrix
$\mathbf{L}_1$	A complete Latin square of size 16×16
$\mathbf{L}_2$	The expansion matrix generated from $\mathbf{L}_1$
$\mathbf{S}$	An S-box
$\mathbf{U}$	A plain image
$\mathbf{D}$	A Latin square of size 16×16
$\mathbf{H_r}$	The row permutation matrix
$\mathbf{H_c}$	The column permutation matrix
$\mathbf{P}$	The permuted image
$\mathbf{T}$	The confused image
$\mathbf{W}$	The diffusion matrix
$\mathbf{C}$	The diffused image
$\mathbf{C}_b$	The 4×4 block in $\mathbf{C}$
K	The original secret key
K_i	The different key from K in the ith part
ϑ_α	The threshold of NPCR with the significance level of α
θ_α^-	The low bound of the UACI interval with the significance level of α
θ_α^+	The high bound of the UACI interval with the significance level of α
$N(f)$	The nonlinearity of boolean function f
$S_f(\cdot)$	The Walsh–Hadamard transform of boolean function f

square as a background. They will be used to generate S-box in Sect. 3.

2.1 The enhanced logistic map

Because the chaos theory has many significant properties including the initial state sensitivity, ergodicity and unpredictability, it is widely used to generate pseudo-random numbers [3,25,50]. When a chaotic system is used to generate pseudo-random numbers, the chaos performance of the system highly determines the performance of the pseudo-random numbers. Recently, an enhanced logistic map is introduced by performing a sine function to the original logistic map [11]. Since the sine function has complicated nonlinear properties and bounded orbits, the generated enhanced logistic map can achieve complex chaos performance and has a large parameter range. Mathematically, the enhanced logistic map is defined as

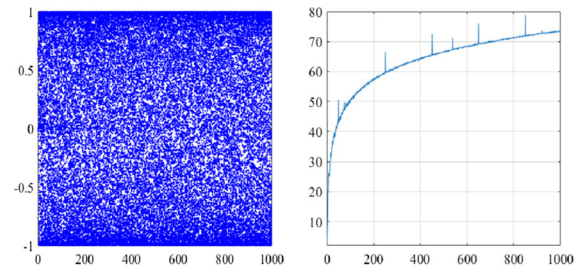

Fig. 1 Chaos performance of the enhanced logistic map. **a** Bifurcation diagram; **b** Lyapunov exponents

$$x_{i+1} = E(L(x_i)) = \sin(\pi a x_i(1 - x_i)), \tag{1}$$

where a is a control parameter and $a \in (0, +\infty)$. According to the discussion in [11], when $a \geq 2$, all the fixed points of the enhanced logistic map are unstable, indicating that the enhanced logistic map has chaotic behaviors.

The bifurcation diagram plots the visited or approached points of a dynamical system with the change of its control parameter. The Lyapunov exponent (LE) measures the separation rate of two trajectories of a dynamical system starting from extremely close initial states [11]. A positive LE indicates that the two close trajectories will exponentially diverge in each unit time and eventually evolute to be two totally different trajectories. Thus, a dynamical system with a positive LE has chaotic behaviors if its phase plane is also compacted. Figure 1 plots the bifurcation diagram and LEs of the enhanced logistic map when the parameter $a \in [2, 1000)$. As can be seen, the enhanced logistic map has a large chaotic range and its chaotic range are continuous. Besides, the outputs of the enhanced logistic map distribute very randomly on the whole phase plane. These indicate that the enhanced logistic map has complex dynamics properties and chaotic behaviors.

2.2 Complete Latin square

Here, we introduce the complete Latin square. First, the concepts of the Latin square and complete Latin square are introduced. Then, a pair of orthogonal matrices

are generated by expanding the complete Latin square. Finally, a matrix scrambling operation is used to shuffle the elements of the orthogonal matrices.

2.2.1 Basic concepts

A Latin square is a square matrix that an element occurs exactly once in each row and exactly once in each column. A complete Latin square is a special Latin square that all ordered pairs of elements occurs once each in row and column. When the elements in two Latin squares are arranged in pairs, the two Latin squares are orthogonal if all the element pairs are different from each other. The Latin square, complete Latin square and orthogonal Latin squares are be described as Definitions 1, 2 and 3 , respectively.

Definition 1 A quare matrix of size $N \times N$ is a N-order Latin square if each element appears only once in each row and in each column.

Definition 2 A Latin square is a complete Latin square matrix if all ordered pairs of elements occurs once each in row and column.

Definition 3 For two Latin squares $\mathbf{A}_1 = (a_{i,j}^{(1)})^{N\times N}$ and $\mathbf{A}_2 = (a_{i,j}^{(2)})^{N\times N}$, they are orthogonal if all the element pairs $(a_{i,j}^{(1)}, a_{i,j}^{(2)})$ are different from each other.

There are many methods to construct a complete Latin square. If the first row of a complete Latin square is found, the complete Latin square then can be generated. Generally, there are two methods to generate the first row of a complete Latin square of size $N \times N$ ($N = 2m$ and $m \in \mathbb{Z}$) [45]. To introduce the methods, the primitive root should be introduced as background.

Definition 4 An integer g is a primitive root of n if there is an integer z such that $k \equiv g^z \bmod n$ for every number k that is relatively prime to n.

Then the two generation methods of the first row of a complete Latin square are shown as follows.

Method 1: The first row of a complete Latin square of size $N \times N$ ($N = 2m$ and $m \in \mathbb{Z}$) is generated as

$$0, 1, 2m-1, 2, 2m-2, 3, 2m-3, \ldots, m-1, m+1, m \tag{2}$$

Method 2: If $N + 1$ is a prime number, find the primitive roots g of $N + 1$. For each primitive root g, calculate $a_z \equiv g^z \bmod (N+1)$, $z \in \{1, 2, \ldots, N\}$, and then sort a_i in an ascending order. The orders of all the indices i are the first row of the complete Latin square.

A numeral example is provided to explain the generation of the first row of the complete Latin square of size 4×4 using *Method 2*. First, $N+1 = 4+1 = 5$, and we can find that 2 is the only primitive root of 5. Then calculate $a_1 \equiv 2^1 \mod 5 = 2$, $a_2 \equiv 2^2 \mod 5 = 4$, $a_3 \equiv 2^3 \mod 5 = 3$, $a_4 \equiv 2^4 \mod 5 = 1$. After sorting a_i in an ascending order, we can get the first row of the complete Latin square as $\{a_4, a_1, a_3, a_2\} = \{4, 1, 3, 2\}$.

A complete Latin square of size $N \times N$ can be generated as follows.

Step 1: Generate the first row of the complete Latin square using one of the two methods above;

Step 2: The elements in the current row are derived from its previous row. For example, the elements in the i-th row can be calculated as row(i)=(row(i-1)+1) mod N. After obtaining the elements in all the rows, a row complete Latin square can be obtained;

Step 3: Set the elements in the 1-st column as the elements in the 1-st row, and move each row begin with its 1-st elements to the corresponding column. Then a complete Latin square is obtained.

A numeral example is provided to show the generation of the complete Latin square from the row complete Latin square in *Step 3*. Suppose a row complete Latin square is shown as the matrix **A** in Eq. (3), then a complete Latin square **B** can be generated as follows.

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 3 & 2 \\ 1 & 2 & 0 & 3 \\ 2 & 3 & 1 & 0 \\ 3 & 0 & 2 & 1 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 & 1 & 3 & 2 \\ 1 & 2 & 0 & 3 \\ 3 & 0 & 2 & 1 \\ 2 & 3 & 1 & 0 \end{pmatrix}. \tag{3}$$

(1) Set the first row and column in **B** as the first row of **A**, namely 0, 1, 3, 2. (2) The second row of **B** has a first element 1. By searching the first elements in all the rows of **A** and the second row is found. Then the second row and column of **B** are set as the second row of **A**, namely 1, 2, 0, 3. (3) The third row of **B** has a

first element 3. By searching the first elements in all the rows of **A** and the fourth row is found. Then the third row and column of **B** are set as the fourth row of **A**, namely 3, 0, 2, 1. (4) The fourth row of **B** has a first element 2. By searching the first elements in all the rows of **A** and the third row is found. Then the fourth row and column of **B** are set as the third row of **A**, namely 2, 3, 1, 0.

2.2.2 *Orthogonal matrices*

Using the complete Latin square, a pair of orthogonal matrices can be generated. For a complete Latin square **B**, a new square matrix $\mathbf{B}_2$ with the same size can be expanding from **B** as follows [51]. (1) Set the elements in i-th ($i \in \{1, 2, \cdots, N-1\}$) column of $\mathbf{B}_2$ as the elements in the $(i+1)$-th column of **B**. (2) Set the elements in the N-th column of $\mathbf{B}_2$ as the elements in the N-th column of **B**. Then the matrices **B** and $\mathbf{B}_2$ are two orthogonal matrices. For example, the matrix $\mathbf{B}_2$ shown in Eq. (4) is generated from the complete Latin square **B** shown in Eq. (3). When the elements in **B** and $\mathbf{B}_2$ are used as x coordinate and y coordinate, respectively, a two-dimensional coordinate matrix can be generated as **C** in Eq. (4).

$$\mathbf{B}_2 = \begin{pmatrix} 1 & 3 & 2 & 2 \\ 2 & 0 & 3 & 3 \\ 0 & 2 & 1 & 1 \\ 3 & 1 & 0 & 0 \end{pmatrix},$$

$$\mathbf{C} = \begin{pmatrix} (0,1) & (1,3) & (3,2) & (2,2) \\ (1,2) & (2,0) & (0,3) & (3,3) \\ (3,0) & (0,2) & (2,1) & (1,1) \\ (2,3) & (3,1) & (1,0) & (0,0) \end{pmatrix}. \quad (4)$$

As can be seen that all the coordinates in **C** are different with each other. This indicates that the matrix **B** and $\mathbf{B}_2$ are two orthogonal matrices. An *S*-box has the property that each element appears once, and this property is similar with the orthogonal matrices. Thus, using the orthogonal matrices generated from the complete Latin square, one can construct an *S*-box with high performance.

2.3 Matrix scrambling

To make the distributions of all the elements in a matrix more uniform, we randomly scramble the element positions of the matrix. The whole scrambling

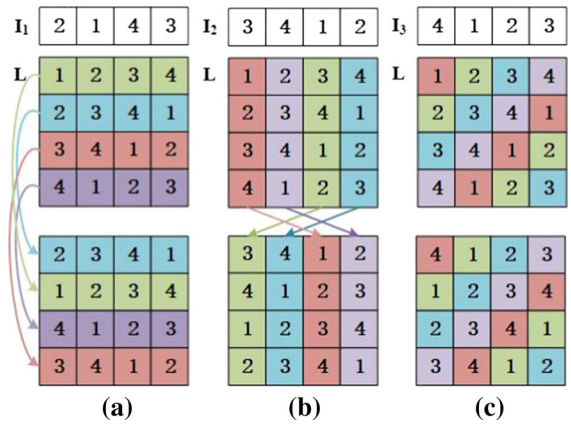

Fig. 2 The process of matrix scrambling. **a** Row scrambling; **b** Column scrambling; **c** Entity scrambling

can be divided into three components: row scrambling, column scrambling and entity scrambling. First, three index vectors are generated by sorting three random sequences. The row scrambling is performed to shuffle the row positions of each row while the column scrambling is to shuffle the column positions of each column. The entity scrambling is to replace the entities of the matrix using the elements of the index vector. To better explain the process of matrix scrambling, we provide a numeral example of size 4×4 and Fig. 2 shows the matrix scrambling process. As can be seen, the i-th row is permuted to $\mathbf{I}_1(i)$-th row, and the i-th column is permuted to $\mathbf{I}_2(i)$-th column. The entity scrambling is to replace the entity i in the matrix with $\mathbf{I}_3(i)$. After the matrix scrambling, the correlation in adjacent elements can be broken and this leads to the uniform distribution of all elements.

3 Design of *S*-box

This section presents an *S*-box generation method using the complete Latin square and enhanced logistic map.

3.1 *S*-box Generation

Figure 3 shows the block diagram of the *S*-box generation. First, chaotic sequences are generated by the enhanced logistic map using the given initial state. A part of chaotic sequences is used to generate the complete Latin square. Using the complete Latin square, two orthogonal matrices are produced. By scrambling the orthogonal matrices using the chaotic sequences,

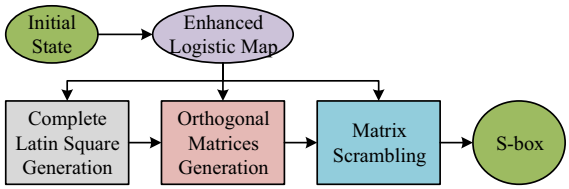

Fig. 3 Block diagram of *S*-box generation algorithm

an *S*-box can be generated. The detailed generation of a 16×16 *S*-box can be described as follows.

Step 1: Three chaotic sequences of length 16 are generated. By sorting each chaotic sequences, three index vectors $\mathbf{I}_1$, $\mathbf{I}_2$ and $\mathbf{I}_3$ can be obtained. Because the number 17 has 8 primitive roots, 8 different sequences can be used as the first row of the complete Latin square by *Method 2* in Sect. 2.2.1. Counting with the *Method 1*, there are 9 different choices to generate the first row of the complete Latin square. Thus, set $r_1 = \mathbf{I}_3(16) \mod 9$ to decide which one is selected.

Step 2: A complete Latin square $\mathbf{L}_1$ of size 16×16 is generated using the processes introduced in Sect. 2.2.1.

Step 3: An expansion matrix $\mathbf{L}_2$ is generated from $\mathbf{L}_1$ using the processes introduced in Section 2.2.2, where $\mathbf{L}_1$ and $\mathbf{L}_2$ are two orthogonal matrices.

Step 4: Randomly scramble matrices $\mathbf{L}_1$ and $\mathbf{L}_2$ under the index vectors $\mathbf{I}_1$, $\mathbf{I}_2$ and $\mathbf{I}_3$ using the method introduced in Section 2.3. Specifically, perform the row scrambling, column scrambling and entity scrambling on $\mathbf{L}_1$ and $\mathbf{L}_2$ under the control of $\mathbf{I}_1$, $\mathbf{I}_2$ and $\mathbf{I}_3$.

Step 5: The *S*-box $\mathbf{S}$ is generated by combining the two scrambled orthogonal matrices via

$$\mathbf{S}(i, j) = 16 \times (\mathbf{L}_1(i, j) - 1) + \mathbf{L}_2(i, j), \tag{5}$$

Table 2 shows a generated *S*-box of size 16×16. Because the proposed generation process does not contain complex matrix row, column or transform operations. Its time complexity is low and thus can achieve a high generation efficiency.

3.2 Performance analysis of *S*-box

An *S*-box is expected to have complex nonlinear properties. Here, we analyze the properties of the constructed *S*-box from different aspects.

3.2.1 Nonlinearity

Since an *S*-box is the nonlinearity part of the symmetric key encryption algorithm, its nonlinearity can greatly decide the security level of the encryption algorithm. An *S*-box with a better nonlinearity is more robust to resist different kinds of linear attacks. Thus, the nonlinearity is one of the most important properties of an *S*-box. The nonlinearity of an *S*-box can be measured by the Walsh spectrum [37], which is defined as

$$N(f) = 2^{n-1}(1 - 2^{-n} \max_{\omega \epsilon \mathrm{GF}(2^n)} |S_f(\omega)|), \tag{6}$$

where)GF(2^n) denotes the Galois Field with 2^n elements, and $S_f(\omega)$ is the Walsh–Hadamard transform of boolean function f that can be calculated as

$$S_f(\omega) = \sum_{\omega \epsilon \mathrm{GF}(2^n)} -1^{f(x) \oplus x * \omega}. \tag{7}$$

A larger $N(f)$ means the better nonlinearity of the *S*-box and its ideal value for an *S*-box of size 16×16 is 112. The *S*-box generated by our proposed method can achieve the values of 104, 108, 102, 102, 106, 108, 106, 106. The minimum value is the indicator of the nonlinearity of the *S*-box, and the *S*-box generated by our proposed algorithm can achieve a minimum value of 102. This value is larger than most of the *S*-boxes generated by other algorithms, as shown in Table 7. Thus, our generated *S*-box owns a relatively good nonlinearity.

3.2.2 Strict avalanche criterion

The strict avalanche criterion (SAC) was introduced to measure the avalanche effect that reflects the intuitive idea nonlinearity [6]. It points out that when a function satisfies the strict avalanche criterion, each of its output bits should change with a probability of a half for the change of an input bit. Without strict avalanche effect, an encryption algorithm can be easily broken by building the relationship between the inputs and outputs. Thus, the SAC should be tested for every bit.

Table 2 An example of the generated S-box of size 16×16

53	75	33	114	1	230	176	255	131	90	212	109	28	152	201	183
130	17	80	172	256	47	10	147	85	237	105	126	180	203	214	56
31	231	88	211	120	132	107	169	182	49	146	208	37	14	252	77
193	12	164	32	52	119	185	136	219	102	45	79	250	82	238	149
174	223	195	87	35	160	229	74	13	242	59	140	104	22	177	121
228	58	189	249	205	21	158	210	44	71	175	8	134	112	115	81
7	99	213	232	69	202	34	29	254	156	129	84	123	191	64	166
248	117	98	43	18	221	76	190	151	196	96	233	161	51	138	15
41	141	62	150	222	178	199	108	68	24	3	251	95	122	165	240
125	198	159	142	239	241	83	48	170	5	184	50	215	73	27	100
106	153	246	61	86	11	143	225	128	163	23	181	206	36	72	220
224	244	9	186	137	168	54	91	97	127	78	19	157	236	39	194
92	192	26	4	154	67	253	197	226	46	118	167	57	209	111	139
70	94	135	207	103	60	216	116	25	187	245	145	227	173	2	42
179	40	235	101	171	89	113	6	63	144	204	218	66	247	148	30
155	162	124	65	188	110	20	55	200	217	234	38	16	133	93	243

The authors in [43] proposed an effective method to test whether an S-box can satisfy the SAC or not. First, an 8-bit vector X is input into the S-box, and the corresponding 8-bit output vector Y is obtained by substitution. Then a set of vectors $X_1, X_2, \ldots, X_8$ are generated, where X and X_j differ only on bit j. The output vectors $Y_1, Y_2, \ldots, Y_8$ can be calculated by Y_j=$\mathbf{S}(X_j)$, where $\mathbf{S}(\cdot)$ is the substitution operation with S-box. A set of 8-bit binary avalanche vectors $V_1, V_2, \ldots, V_8$ can be calculated by $V_j = Y \oplus Y_j$. An 8×8 dependence matrix is generated by adding the value of bit i in V_j to element $a_{i,j}$. Repeat the steps above n times by randomly generating the vector X and the finial dependence matrix is generated by calculate the average elements.

The ideal value of the elements in dependent matrix is 0.5. An S-box can achieve high performance in SAC when every element of the matrix is close to the ideal value. Table 3 shows the dependent matrix of the proposed S-box. As can be seen, most of the elements are close to the ideal value 0.5. This indicates that the proposed S-box can obtain relatively ideal values.

3.2.3 Bit independence criterion

The bit independence criterion (BIC) is another desirable property for a cryptographic algorithm. For an S-box: $\{0, 1\}^n \rightarrow \{0, 1\}^n$, it can satisfy the BIC when for all $i, j, k \in 1, 2, \ldots, n$ and $j \neq k$, the change of input bit i can cause the independent change of output bits j and k.

The authors in [43] introduced an effective method to test the BIC. The method points out that for two output bits f_i and f_j $(i \neq j)$ of an S-box, if $f_i \oplus f_j$ has high nonlinearity and satisfies the SAC, the S-box can show good performance in BIC. Thus the BIC-SAC and BIC-nonlinearity should be calculated for every bit. To calculate the BIC-SAC, for an input vector X, the output vector $Y, Y_1, Y_2, \ldots, Y_8$ are calculated using the method introduced in Section 3.2.2. Then two xor matrix P and Q are generated, where $P_{i,j}$ is the xor result of the i-th and j-th bits of Y, while $Q_{i,j,k}$ is the xor result of the i-th and j-ths bit of Y_k, where $i, j, k = 1, 2, \ldots, 8$. For each k, calculate $V_{i,j,k} = P_{i,j} \oplus Q_{i,j}$, and add $V_{i,j,k}$ to element $a_{i,j}$ of the 8×8 dependence matrix A. Repeat these steps n times to obtain the average values for each elements of A, and the BIC-SAC values are obtained. The BIC-nonlinearity values are calculated by Eqs. (6) and (7). The only difference is to change the $f(x)$ in Eq. (7) to $f_i \oplus f_j$, where f_i and f_j are the i-th and the j-th bits of $f(x)$. A BIC-SAC value close to 0.5 and a larger BIC-nonlinearity value means the better performance.

Tables 4 and 5 show the BIC-SAC and BIC-nonlinearity test results of the proposed S-box. As can be seen, the minimum and maximum values of the BIC-

Table 3 The dependence matrix of the proposed *S*-box

0.5156	0.5625	0.5781	0.5312	0.5625	0.5469	0.5000	0.5469
0.5156	0.5781	0.5781	0.5000	0.5469	0.5312	0.4688	0.5938
0.5312	0.5312	0.5312	0.5156	0.5312	0.4844	0.5312	0.5625
0.5625	0.5312	0.5312	0.5469	0.5625	0.5938	0.5312	0.5625
0.5000	0.5000	0.5469	0.5156	0.5781	0.5625	0.4844	0.5312
0.5312	0.4844	0.5938	0.5156	0.5156	0.5156	0.5781	0.4688
0.5000	0.5625	0.5312	0.5156	0.5156	0.5469	0.5000	0.5781
0.5625	0.5000	0.5000	0.5625	0.5781	0.5312	0.4844	0.5625

Table 4 BIC-SAC matrix of the proposed *S*-box

0	0.5039	0.5078	0.4961	0.4590	0.5215	0.5176	0.5234
0.5039	0	0.4805	0.5332	0.5020	0.4961	0.5156	0.5039
0.5078	0.4805	0	0.5078	0.5000	0.4961	0.4941	0.5215
0.4961	0.5332	0.5078	0	0.4902	0.4941	0.4688	0.5078
0.4590	0.5020	0.5000	0.4902	0	0.4941	0.5117	0.4863
0.5215	0.4961	0.4961	0.4941	0.4941	0	0.4883	0.4883
0.5176	0.5156	0.4941	0.4688	0.5117	0.4883	0	0.4902
0.5234	0.5039	0.5215	0.5078	0.4863	0.4883	0.4902	0

Table 5 BIC-nonlinearity matrix of the proposed *S*-box

0	98	100	100	104	102	104	106
98	0	102	100	100	108	108	104
100	102	0	100	102	100	106	108
100	100	100	0	102	106	108	102
104	100	102	102	0	102	104	104
102	108	100	106	102	0	106	104
104	108	106	108	104	106	0	100
106	104	108	102	104	104	100	0

SAC are 0.4590 and 0.5234, and the mean value is 0.5, which equals to the ideal value. The minimum and maximum values of the BIC-nonlinearity are 98 and 108, and the mean value is 103.21. These indicate that the *S*-box shows good performance in this criterion.

3.2.4 Differential approximation probability

The differential attack is an effective cryptanalysis technique by analyzing the connections between the cipher image with the corresponding plain image. As the important nonlinearity part of the image encryption algorithm, the *S*-box should have the ability to resist this attack. The differential approximation probability (DAP) can measure the ability of an *S*-box to resist the differential attack [4].

To calculate the DAP value, a differential distribution table with the size of 256×256 should be generated in advance. For each integer x, Δx, $\Delta y \in [1, 256]$, the element $a_{\Delta x, \Delta y}$ in the differential distribution table adds 1 if $\mathbf{S}(x) \oplus \mathbf{S}(x \oplus \Delta x) = \Delta y$, where $\mathbf{S}(\cdot)$ is the substitution operation with *S*-box. Then for each $\Delta x \in [1, 256]$, find the maximum $a_{\Delta x, \Delta y}$ and put it into a new vector M of length 256. Finally, the DAP matrix is obtained by rearranging M into a matrix of size 16×16. A lower DAP value shows a stronger *S*-

Table 6 The DAP matrix of the proposed *S*-box

6	6	6	8	6	6	6	6	6	8	8	6	10	6	6	6
6	6	6	8	6	8	6	6	6	6	8	6	8	6	6	8
6	6	8	6	8	10	6	6	6	8	8	8	8	6	6	6
10	8	6	6	6	6	6	6	8	6	8	8	8	4	6	6
6	6	6	8	8	8	6	6	10	6	10	6	6	6	6	6
6	6	6	6	6	8	8	8	6	6	6	6	8	8	8	6
6	6	12	8	10	6	6	6	8	8	8	8	8	6	8	8
6	8	6	6	8	8	6	6	6	6	6	6	6	10	6	8
8	6	8	6	6	6	6	8	6	6	6	10	8	6	6	6
8	6	6	8	8	8	6	6	8	6	10	8	6	6	8	6
6	6	6	4	6	6	6	8	8	8	6	6	10	6	8	8
6	6	6	10	6	8	6	6	6	12	6	6	6	8	10	6
6	10	6	8	8	6	10	8	6	6	6	6	8	6	6	6
6	6	6	6	8	6	8	8	6	6	8	6	6	6	6	6
6	8	8	8	6	8	6	6	10	6	6	8	6	8	6	6
6	6	6	6	8	6	14	6	6	8	6	8	8	6	8	0

Table 7 The comparison results of *S*-boxes generated by different methods

S-box	Nonlinearity			BIC-SAC	BIC-nonlinearity	SAC			DAP
	Min.	Avg.	Max.			Min.	Max.	Offset.	
Proposed	102	105.25	108	0.5000	103.21	0.4688	0.5938	0.0390	14
Ref. [38]	101	103.88	108	0.5018	102.68	0.3906	0.5781	0.0307	14
Ref. [39]	103	104.88	109	0.5039	102.96	0.3984	0.5703	0.0319	10
Ref. [7]	100	103.00	106	0.4993	103.14	0.4219	0.6094	0.0327	10
Ref. [16]	96	104.00	110	0.4980	103.00	0.3900	0.5930	0.0323	32
Ref. [17]	100	104.75	108	0.4965	105.07	0.3906	0.5938	0.0317	32
Ref. [2]	100	103.00	106	0.4983	102.93	0.3906	0.5938	0.0356	10
Ref. [22]	96	103.00	106	0.5031	100.36	0.3906	0.6250	0.0583	12
Ref. [18]	98	102.25	108	0.4992	101.57	0.3281	0.6250	0.0493	16
Ref. [32]	100	104.70	108	0.4942	103.10	0.4218	0.5781	0.0307	10
Ref. [21]	96	103.25	106	0.4864	103.07	0.3906	0.6719	0.0500	44
Ref. [27]	102	104.25	108	0.5045	103.07	0.4219	0.6016	0.0337	12
Ref. [26]	100	105.00	108	0.5038	103.00	0.3906	0.6250	0.0293	12
Ref. [31]	102	104.75	106	0.5017	103.00	0.3906	0.5938	0.0356	12

box of resisting the differential attack. Table 6 shows the DAP matrix of the proposed *S*-box. It can be seen that the minimum value is 4 and the maximum value is 14. This indicates that the proposed *S*-box has strong ability to against the differential attack.

3.2.5 Performance comparison

To show the superiority of our proposed *S*-box, we compare it with other *S*-boxes generated by some latest algorithms. Table 7 shows the comparison results in different aspects. The offset in the SAC is the average

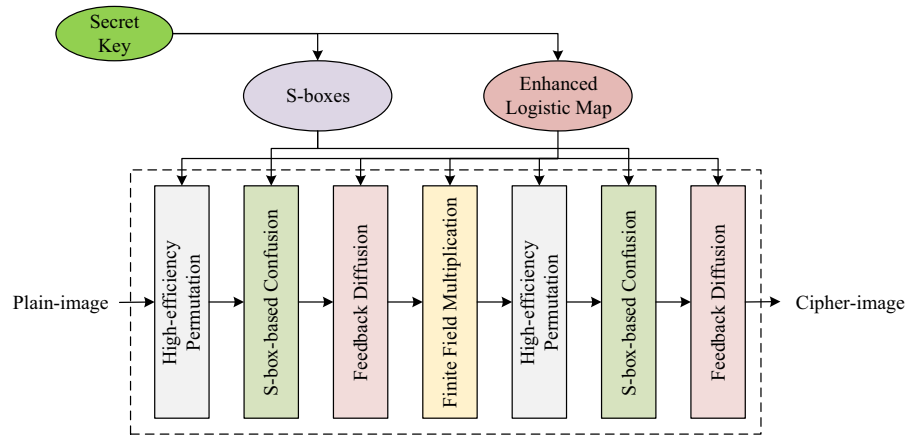

Fig. 4 Block diagram of image encryption algorithm

value of the differences between each element with the ideal value 0.5. As can be seen, the proposed *S*-box can achieve the ideal value of 0.5 in the BIC-SAC and the largest average value in the nonlinearity, compared with the other *S*-boxes. Besides, it can also outperform most *S*-boxes in the other criteria. These comparison results indicate that the *S*-box generated by the proposed algorithm has better performance than most *S*-boxes generated by many other algorithms and thus has high performance in design encryption algorithm.

4 Application of *S*-box in image encryption

To show the high efficiency of the proposed *S*-box in encryption algorithm, in this section, we apply it to an image encryption algorithm, and Fig. 4 shows the structure of the developed image encryption algorithm. The algorithm is mainly constructed by high-efficiency permutation, *S*-box-based confusion, feedback diffusion and finite field multiplication. The secret key is the initial states of the enhanced logistic map and the *S*-box generation. The chaotic sequence generated by the enhanced logistic map is used to generate the Latin square and random matrix for high-efficiency permutation and feedback diffusion. The *S*-boxes generated are used in the *S*-box-based confusion, which can effectively shuffle pixel positions. Meanwhile, feedback diffusion operation can completely change the pixel values and spread few changes of plain image to whole cipher image. The finite field multiplication can improve the security of the encryption algorithm. Two rounds encryption are used to obtain a cipher image with a high security level. Next, we will introduce the encryption steps in detail.

4.1 High-efficiency Permutation

The high-efficiency permutation is a process to simply change the pixel positions. In our algorithm, we use a Latin square generated from chaotic sequences to permute the pixels in rows and columns, respectively. The Latin square $\mathbf{D}$ is generated as

$$\mathbf{D}(i, j) = (a \times \mathbf{R}(j) + a^2 \times \mathbf{R}(i)) \mod N, \quad (8)$$

where $\mathbf{R}$ is an index vector of length N, and it is generated by sorting a chaotic sequence, and $a = \mathbf{R}(1)$. Using the Latin square $\mathbf{D}$, the high-efficiency permutation to a plain image $\mathbf{U}$ can be represented as:

Step 1: Generate the row permutation matrix $\mathbf{H_r}$ and column permutation matrix $\mathbf{H_c}$ with

$$\mathbf{H_r}(i, j) = (\mathbf{U}(i, 1) + \mathbf{D}(i, j)) \mod N,\ i \in [2, N] \quad (9)$$

$$\mathbf{H_c}(i, j) = (\mathbf{U}(1, j) + \mathbf{D}(i, j)) \mod N,\ j \in [2, N] \quad (10)$$

For each row of $\mathbf{H_r}$, if $\mathbf{H_r}(i, j) = 1$, replace $\mathbf{H}_r(i, j)$ with s - $sum(i, :)$, where $sum(i, :)$ is the sum of the ith row of $\mathbf{H_r}$, and $s = \frac{N(N+1)}{2} - 1$. For each column, change $\mathbf{H_c}$ with the same operation.

Step 2: For each row of $\mathbf{U}$, exchange the pixels on $\mathbf{U}(i, j)$ and $\mathbf{U}(i, \mathbf{H_r}(i, j))$, where $i, j \in [2, N]$.

Step 3: For each column of matrix $\mathbf{U}$, exchange the pixels on $\mathbf{U}(i, j)$ and $\mathbf{U}(\mathbf{H_c}(i, j), j)$, where $i, j \in [2, N]$.

Step 4: Initial the index vector a and b as the first row and first column of $\mathbf{D}$. Exchange the pixels on $\mathbf{U}(1, i)$ and $\mathbf{U}(1, a(i))$. Exchange the pixels on $\mathbf{U}(j, 1)$ and $\mathbf{U}(b(j), 1)$, where $i \in [1, N]$, $j \in [2, N]$.

4.2 S-box-based Confusion

To increase the security level, the confusion is designed to randomly change the positions of pixel. Here, the permuted image is confused using two S-boxes. The detailed steps of confusing an image of size 256×256 are shown as follows.

Step 1: Traverse the permutated image $\mathbf{P}$ and transform the coordinate of each pixel into 8-bit binary, marked as (u, v).

Step 2: Set the first 4 bits of u as x_1, and the remain 4 bits as y_1. Find the pixel x in the first S-box with the coordinate (x_1, y_1). Similarly, set the first 4 bits of v as x_2, and the remain 4 bits as y_2. Find the pixel y in the second S-box with the coordinate (x_2, y_2).

Step 3: Exchange the positions of the pixel on coordinates (u, v) and (x, y) in image $\mathbf{P}$.

Step 4: Repeat the above steps until all the pixels have been processed and obtain the confused image $\mathbf{T}$.

To better explain the process of S-box-based confusion, a numeral example of size 4×4 is provided and Fig. 5 details the process. Since the first pixel on the coordinate $(0, 0)$ is 14, transform the coordinate (0,0) into 2-bit binary, marked as (00,00). Combine the first bit and the second bit of the x-coordinate to get the coordinate (0,0). Similarly, combine the first bit and second bit of the y-coordinate to another coordinate (0,0). Then find the pixel in the first S-box S_1 with the coordinate (0,0) to get the value 3. Find the pixel in the second S-box S_2 with the coordinate (0,0), and we get the pixel 2. Find the pixel with coordinate (3,2) in $\mathbf{P}$, and we get the pixel 13. Finally exchange the positions of pixel 14 on coordinate (0,0) and pixel 13 on coordinate (3,2). Altering and performing these steps to all the pixels of $\mathbf{P}$, the S-box-based confusion operation is finished.

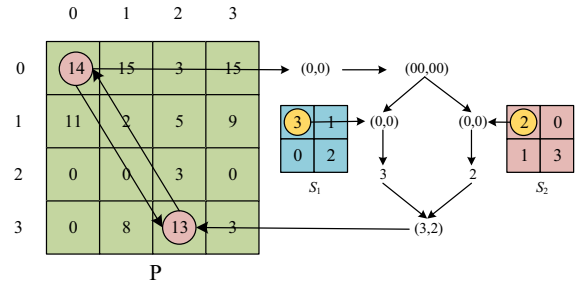

Fig. 5 An example of S-box-based confusion operation

4.3 Feedback diffusion

The diffusion is to diffuse the image by spreading a slight change in the plain image all over the whole cipher image. Many existing diffusion processes use the *xor* operation, which may spend much time in bit calculation. Our scheme diffuses the row pixels and column pixels separately. Using a matrix $\mathbf{W}$ of size $N \times N$ constructed by reshaping the chaotic sequence of the enhanced logistic map, the feedback diffusion operation for each row or column can be represented as

$$\mathbf{C}(x) = \begin{cases} (\mathbf{T}(x) + \mathbf{W}(j) + \mathbf{T}(N) + \mathbf{T}(N-1)) \bmod N \\ \quad \text{for } x = 1 \\ (\mathbf{T}(x) + \mathbf{W}(j) + \mathbf{C}(1) + \mathbf{T}(N)) \bmod N \\ \quad \text{for } x = 2 \\ (\mathbf{T}(x) + \mathbf{W}(j) + \mathbf{C}(x-1) + \mathbf{C}(x-2)) \bmod N \\ \quad \text{for } x \in [3, N-1] \\ (\mathbf{T}(x) + \mathbf{W}(k) + \mathbf{C}(x-1) + \mathbf{C}(x-2)) \bmod N \\ \quad \text{for } x = N \end{cases} \tag{11}$$

where $j = (x + T(x+1)) \bmod N$, $k = (N + r_1 + r_2) \bmod N$, $r_1 = \mathbf{W}(128, 127)$, $r_2 = \mathbf{W}(127, 128)$, and $\mathbf{T}$ is the image to be diffused.

4.4 Finite field multiplication

To achieve good diffusion and security properties, we apply the finite field multiplication for every 4×4 image block $\mathbf{C_b}$ within the plain image $\mathbf{C}$ as shown in

$$\mathbf{J_b} = (\mathbf{L} \cdot \mathbf{C_b} \cdot \mathbf{L})_{2^8}, \tag{12}$$

where $\mathbf{L}$ is a 4×4 Latin matrix of GF(2^8) generated by Eq. (8). This operation is performed within two encryp-

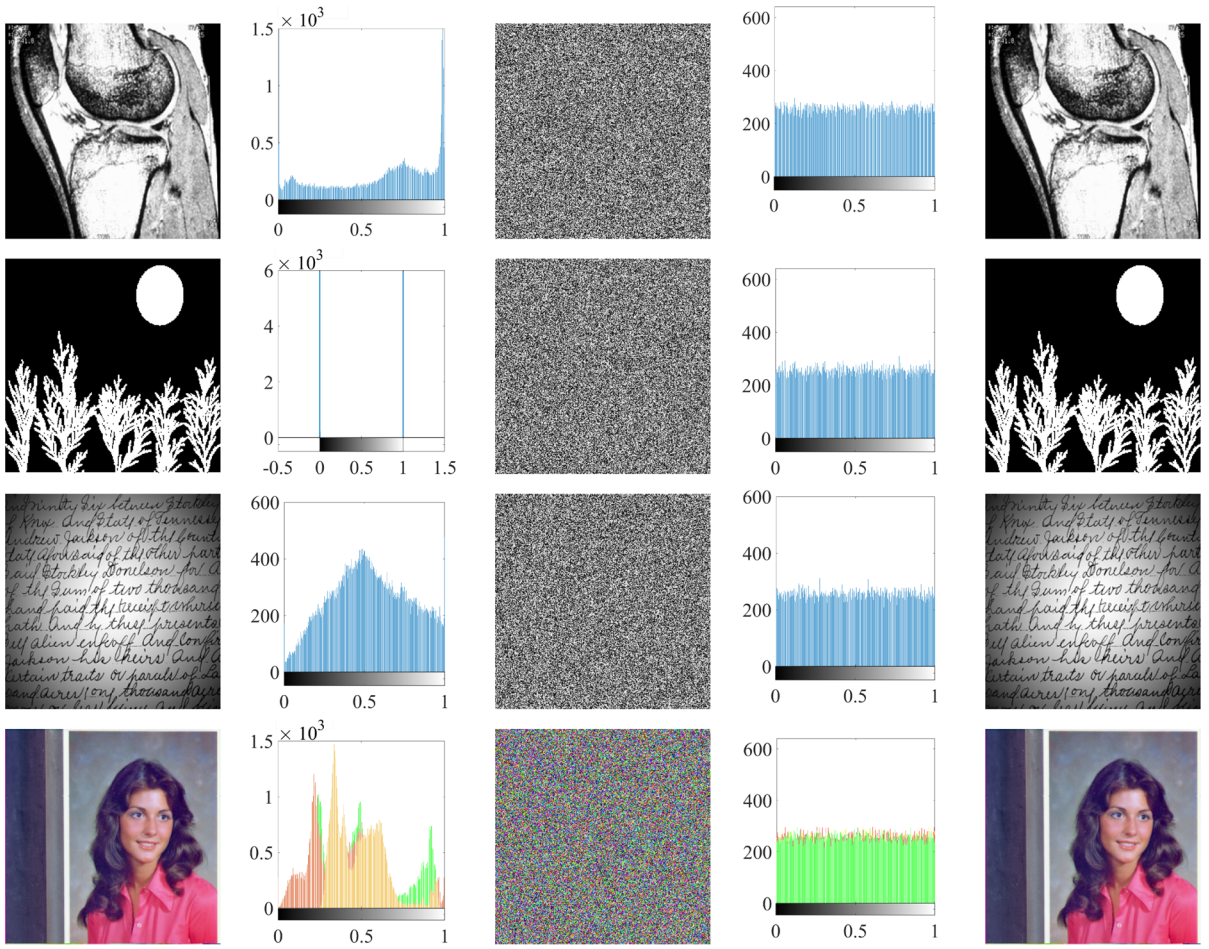

Fig. 6 Simulation results of the proposed image encryption algorithm: **a** plain images include binary, grayscale, and color images; **b** histograms of **a**; **c** encrypted results of **a**; **d** histograms of **c**; **e** decrypted results of **c**

tion rounds. The inverse process of the multiplication is shown in

$$\mathbf{C_b} = (\mathbf{L}^{-1} \cdot \mathbf{J_b} \cdot \mathbf{L}^{-1})_{2^8}, \tag{13}$$

where $\mathbf{L}^{-1}$ is the inverse matrix of $\mathbf{L}$ in GF(2^8).

4.5 Encryption algorithm

After introducing each process of the encryption algorithm, we can present the whole image encryption algorithm. The encryption process for a plain image of size 256×256 is described as follows.

Step 1: Generate four S-boxes $\mathbf{S}_1, \mathbf{S}_2, \mathbf{S}_3, \mathbf{S}_4$ of size 16×16, two index vectors $\mathbf{R}_1, \mathbf{R}_2$ of length 256, one index vector $\mathbf{Q}$ of length 4 and two index vectors $\mathbf{O}_1, \mathbf{O}_2$ of length 65536 using the enhanced logistic map with the given secret key. The $\mathbf{S}_1$ and $\mathbf{S}_2$ are used for S-box-based confusion, the $\mathbf{R}_1$ is used to generate the Latin square for high-efficiency permutation, and the $\mathbf{O}_1$ is used to generate the random matrix for feedback diffusion in the first round. The $\mathbf{Q}$ is used to generate the Latin square for finite field multiplication. The $\mathbf{S}_3$, $\mathbf{S}_4$, $\mathbf{R}_2$, and $\mathbf{O}_2$ are used for the second encryption round.

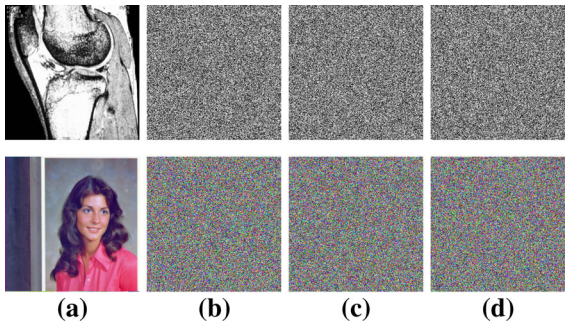

Fig. 7 Key sensitive analysis results: **a** plain images; **b** encryption result of **a**; **c** decryption results of **a** using a changed key on $a_1 = a_1 + 1$; **d** decryption results of **a** using a changed key on $x_2 = x_2 + 0.000000000000001$;

Step 2: First, generate a Latin square **D** of size 256×256 using $\mathbf{R}_1$. Then, the high-efficiency permutation matrices $\mathbf{H_c}$ and $\mathbf{H_r}$ are constructed using the Latin Square **D**. Finally, change the pixel positions of the plain image **U** using $\mathbf{H_c}$ and $\mathbf{H_r}$ to obtain the permuted image **P**.

Step 3: Perform S-box-based confusion operation to the permuted image **P** using two S-boxes $\mathbf{S}_1, \mathbf{S}_2$ to obtain the confused image **T**.

Step 4: Generate a random matrix **W** of size 256×256 using $\mathbf{O}_1$, and then perform the feedback diffusion to each row and column of the confused image **T** using **W**.

Step 5: Generate a Latin square of size 4×4 in GF(2^8) using **Q** and multiply the Latin square with image block $\mathbf{C_b}$ of the confused image **C**.

Step 6: Repeat *Steps 2* to *Steps 4* using the rest parameters to obtain the cipher image **B**.

Since the proposed encryption algorithm is a symmetric key encryption algorithm, the decryption processes are the inverse operation of the encryption processes.

5 Simulation results and security analysis

This section simulates the designed image encryption scheme using S-boxes and analyzes its security level.

5.1 Simulation results

An effective image encryption algorithm should have the ability to encrypt different kinds of digital images into unrecognizable cipher images. Only with the correct key, the decryption process can correctly recover the original image. Without correct key, one cannot obtain any useful information. Figure 6 shows the simulation results of the designed image encryption scheme for binary, grayscale and color images. One can observe from Fig. 6a, b that all the plain images have very obvious patterns, as shown in their histograms. Figure 6c shows that the cipher images encrypted by our encryption scheme are random-like and don't contain any meaningful information. Figure 6d indicates that the cipher images are uniformly distributed. One cannot get any useful information from their histograms. With the correct secret key, one can totally recover the original image, as shown in Fig. 6e. Thus, the proposed encryption algorithm has strong ability to encrypt both grayscale and color images into unrecognizable cipher images with high security level.

5.2 Key sensitivity

An encryption algorithm should have very sensitive secret key. Otherwise, it will exist equivalent keys. The secret key in our proposed encryption algorithm is consisted as $K = \{a_1, x_1, a_2, x_2\}$. Several experiments are designed to test the key sensitivity of the proposed image encryption scheme. First, randomly generate a secret key K_1 and then separately change a tiny to the components a_1 and x_2 to generate other two secret keys K_2 and K_3. These secret keys are

$$\begin{aligned} K_1 &= \{20, 15, -0.339800693932357, \\ &\quad - 0.092813258691572\} \\ K_2 &= \{21, 15, -0.339800693932357, \\ &\quad - 0.092813258691572\} \\ K_3 &= \{20, 15, -0.339800693932357, \\ &\quad - 0.092813258691571\} \end{aligned} \tag{14}$$

Fig. 7 shows the key sensitivity analysis results. One can see that a tiny change to the secret key can result in totally different decrypted results. This straightforwardly indicates that the proposed encryption scheme is sensitive to its secret key.

The key sensitivity of our proposed image encryption scheme can be quantitatively tested by the number of bit change rate (NBCR). For two data sequences Z_1

Table 8 The NBCRs of cipher images and NBCRs of decrypted images with a tiny change in the secret keys

		Origin key	Changed component	NBCRs	
				Encryption process	Decryption process
a	a_1	46	47	0.5004	0.5002
	a_2	27	26	0.5005	0.5006
x	x_1	−0.101325026440898	−0.101325026440897	0.4998	0.5000
	x_2	−0.761517651598448	−0.761517651598447	0.5003	0.5001
Mean	–	–	–	0.5003	0.5002

and Z_2 with the same size, their NBCR can be calculated by

$$\text{NBCR} = \frac{H(Z_1, Z_2)}{L_b} \times 100\%, \tag{15}$$

where L_b is the length of Z_1 or Z_2, and $H(Z_1, Z_2)$ calculates the Hamming distance of Z_1 and Z_2. The ideal value of NBCR is 50%, and it can be achieved when the two data sequences are completely independent.

For each component of the secret key, we test its key sensitivity using the following steps. (1) Randomly generate a secret key as an original key and separately cause a tiny change to each component of the original key to generate a group of new secret keys. (2) Encrypt the same plain image using these secret keys and calculate the NBCRs between the cipher image by the original key and the cipher images by each new key. (3) Decrypt the same cipher image using these secret keys and calculate the NBCRs between the decrypted result by the original key and the decrypted results by each new key. Table 8 lists the calculated results. As can be seen, all the NBCRs are close to the ideal value 0.5. This indicates that a tiny change in each component of the secret key can cause totally different cipher images in encryption process and cause totally different decrypted images in decryption process. Then the image encryption algorithm is quite sensitive to its secret key in both the encryption and decryption processes.

5.3 Ability to resist differential attack

The differential attack is one of the most used and effective security attack methods. An image encryption algorithm with high security level should have the ability to resist this attack. If it has strong ability to resist the differential attack, it must be sensitive to the change of the plain image. Specifically, the slight change in the plain image can change it to the totally different cipher image. The number of pixel change rate (NPCR) and unified average changing intensity (UACI) are two criteria to test the ability of an encryption algorithm in resisting the differential attack [41]. For two images E_1 and E_2 encrypted from two plain images with only one bit difference, their NPCR and UACI are calculated as

$$\text{NPCR} = \frac{\Sigma_{j,k} A(j,k)}{L \times M} \times 100\% \tag{16}$$

and

$$\text{UACI} = \frac{1}{L \times M} \Sigma_{j,k} \left(\frac{E_1(j,k) - E_2(j,k)}{255}\right) \times 100\% \tag{17}$$

respectively. The $A(j,k) = 1$ if $E_1(j,k) \neq E_2(j,k)$; otherwise $A(j,k) = 0$. According to the discussions in [46], an image can pass the NPCR test if the obtained NPCR is greater than the threshold value ϑ_α, which is calculated as

$$\vartheta_\alpha = \frac{L - \phi^{-1}(\alpha)\sqrt{\frac{L}{MN}}}{L+1} \tag{18}$$

where α is a significance level and L is the largest allowed value of the image, $L = 255$ in an 8-bit grayscale image, M and N are the height and width of an image, respectively. A greater NPCR result means better performance in resisting differential attack. An image can pass the UACI test if the obtained value falls

Table 9 NPCR results of different image encryption algorithms on special positions ($\vartheta^*_{0.05} = 99.5693$)

Position	Encryption schemes					
	Proposed	Ref. [5]	Ref. [28]	Ref. [34]	Ref. [12]	Ref. [49]
(2,2)	99.6201	99.6262	99.6231	99.5911	99.5575	99.6109
(255,255)	99.6094	99.5621	99.6338	99.6216	99.6521	99.6124
(255,2)	99.6399	99.5895	99.6155	99.5926	99.5850	99.6124
(2,255)	99.6460	99.6307	99.6201	99.5743	99.6002	99.5667
(127,128)	99.5850	99.6506	99.5759	99.5636	99.5819	99.6124
Mean	99.6201	99.6118	99.6137	99.5886	99.5953	99.6030

Table 10 UACI results of different image encryption algorithms on special positions ($\theta^{*-}_{0.05}, \theta^{*+}_{0.05}$) = (33.2824,33.6447)

Position	Encryption schemes					
	Proposed	Ref. [5]	Ref. [28]	Ref. [34]	Ref. [12]	Ref. [49]
(2,2)	33.6123	33.5461	33.2999	33.5461	33.5528	33.6355
(255,255)	33.4635	33.5422	33.3869	33.5422	33.5502	33.2864
(255,2)	33.4320	33.3666	33.5216	33.3666	33.5560	33.4453
(2,255)	33.4381	33.4738	33.5175	33.4738	33.4678	33.3733
(127,128)	33.4633	33.5909	33.3038	33.5306	33.3957	33.4891
Mean	33.4818	33.5032	33.4059	33.4919	33.5045	33.4459
Offset	0.0413	0.0791	0.1024	0.0671	0.0681	0.1144

into the interval of $(\theta^{*-}_{\alpha}, \theta^{*+}_{\alpha})$, which are calculated as

$$\begin{cases} \theta^{*-}_{\alpha} = \frac{L+2}{3L+3} - \phi^{-1}(\frac{\alpha}{2})\frac{(L+2)(L^2+2L+3)}{18(l+1)^2LMN} \\ \theta^{*+}_{\alpha} = \frac{L+2}{3L+3} + \phi^{-1}(\frac{\alpha}{2})\frac{(L+2)(L^2+2L+3)}{18(l+1)^2LMN} \end{cases}$$

If the UACI result is close to the median value of the interval, the relative algorithm will have a better performance in resisting differential attack. According the recommendations in [46], our experiments set the significance level $\alpha = 0.05$. Thus, the threshold for the NPCR with image size 256×256 is $\vartheta_{0.05} = 99.5693$ and the interval for the UACI with image size 256×256 is $(\theta^{*-}_{0.05}, \theta^{*+}_{0.05}) = (33.2824, 33.6447)$.

First, we choose five peripheral pixels in a plain image of size 256×256 and separately change only one bit to obtain five changed images. Then calculate the NPCR and UACI values of the cipher images encrypted from the original and changed images. Tables 9 and 10 show the calculation results for different image encryption schemes. As can be seen, the proposed image encryption algorithm can almost achieve the largest NPCR values and its UACI values are the closest to the median value of the interval, namely 33.46355.

Second, we randomly change one bit in the plain image and calculate the NPCR and UACI values of the cipher images encrypted from the original and changed images. Table 11 lists the best NPCR and UACI results of 100 times of experiments. One can see from the table that the proposed image encryption algorithm can achieve relatively higher NPCR values, and its UACI values are most close to the median value of the interval. Thus, the proposed algorithm shows strong ability to resist differential attack from this aspect.

5.4 Correlation analysis

High correlations exist in adjacent pixels of a natural image. An effective encryption algorithm should have the ability to break these correlations. The correlation of adjacent pixels can be calculated by the correlation coefficient. For two data sequences X and Y with the same length, their correlation coefficient is defined as

$$\mathrm{Corr}(X, Y) = \left| \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \right| \tag{19}$$

Table 11 The UACI and NCPR results for different image encryption schemes

	Encryption schemes	Images			
		Lena	Word	Finger	Tree
NPCR	Proposed	99.6353	99.6521	99.6307	99.6536
	Ref. [5]	99.6292	99.6384	99.6307	99.6277
	Ref. [28]	99.6231	99.6246	99.6445	99.6521
	Ref. [34]	99.6277	99.6216	99.6460	99.5972
	Ref. [12]	99.6506	99.6414	99.6033	99.6277
	Ref. [49]	99.6140	99.6368	99.6353	99.6201
UACI	Proposed	33.4693	33.5088	33.4620	33.4594
	Ref. [5]	33.4769	33.5319	33.5800	33.5723
	Ref. [28]	33.4854	33.3132	33.3537	33.5390
	Ref. [34]	33.3175	33.3507	33.4749	33.3901
	Ref. [12]	33.5187	33.5104	33.5810	33.6061
	Ref. [49]	33.4683	33.4595	33.4550	33.3803

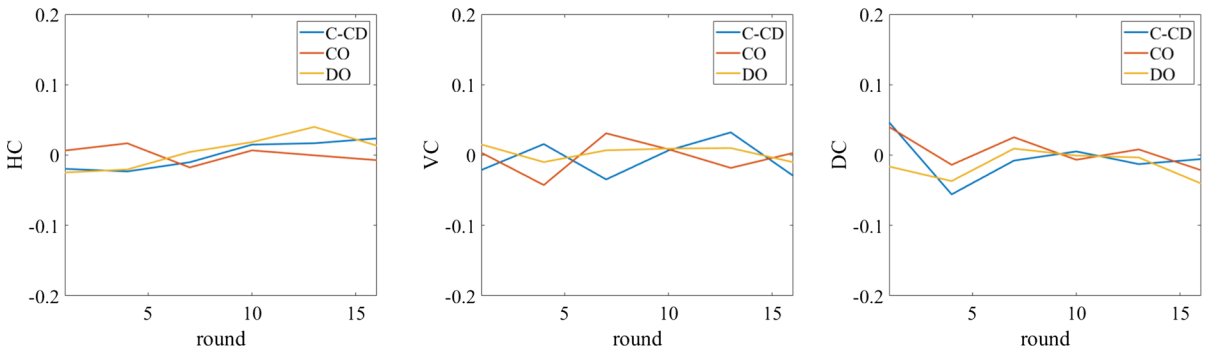

Fig. 8 Correlation coefficients under the confusion-only (CO), diffusion-only (DO) and combination of confusion and diffusion (C-CD) along the **a** horizontal, **b** vertical and **c** diagonal directions

where $E(\cdot)$, μ and σ represent the mathematical expectation, mean and standard deviation, respectively. When using the correlation coefficient to test the adjacent correlation of an image, X represents a sequence of pixels and Y represents another sequence of pixels, which are the horizontal, vertical or diagonal pixels adjacent to that in X. The Corr value closes to 1 if the X and Y have strong correlations. Otherwise, the value close to 0. Thus, a lower Corr value indicates a weaker correlation of two sequences of adjacent pixels.

First, we investigate the ability of confusion and diffusion operations in our proposed scheme to break the strong correlations of adjacent pixels. Figure 8 shows experiment results of the confusion-only, diffusion-only and combination of confusion and diffusion. As

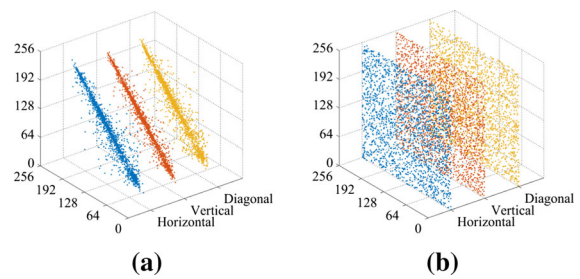

Fig. 9 Distributions of adjacent pixel pairs in **a** the plain image and **b** its cipher image encrypted by the proposed image encrypting scheme

can be seen, the performance in the confusion-only and diffusion-only processes are close to that in the combination of confusion and diffusion processes. Thus,

Table 12 The correlation coefficients of the plain images and cipher images encrypted by different encryption schemes

Images	Direction	Plain images	Cipher images encrypted by different encryption schemes					
			Proposed	Ref. [5]	Ref. [28]	Ref. [34]	Ref. [12]	Ref. [49]
Lena	Horizontal	0.9655	− 0.00048481	− 0.01180000	− 0.01060000	0.00960000	− 0.02080000	− 0.00440000
	Vertical	0.9335	0.00340000	− 0.01230000	− 0.01110000	0.00850000	− 0.02060000	− 0.00410000
	Diagonal	0.8978	− 0.00750000	− 0.01090000	− 0.01040000	0.00880000	− 0.03340000	− 0.00450000
Word	Horizontal	0.8429	0.00540000	− 0.00760000	0.00550000	0.00610000	0.03480000	− 0.00550000
	Vertical	0.7703	0.00095000	− 0.00620000	0.00420000	0.00630000	− 0.03930000	− 0.00510000
	Diagonal	0.6481	− 0.00170000	− 0.00940000	0.00780000	0.00580000	0.00420000	− 0.00600000
Finger	Horizontal	0.8829	− 0.00170000	− 0.00580000	0.00450000	0.00510000	0.03830000	− 0.00790000
	Vertical	0.6972	0.00086888	− 0.00710000	− 0.00290000	0.00029000	0.00630000	− 0.00800000
	Diagonal	0.7757	0.00180000	− 0.00390000	− 0.00350000	0.00540000	0.02300000	− 0.00800000
Tree	Horizontal	0.8774	0.00150000	− 0.00360000	0.00790000	− 0.00530000	− 0.00420000	− 0.00350000
	Vertical	0.8035	− 0.00250000	− 0.00160000	0.00760000	− 0.00520000	− 0.00340000	− 0.00350000
	Diagonal	0.7866	− 0.00340000	0.00002549	0.00810000	− 0.00540000	− 0.00430000	− 0.00350000

the confusion and diffusion processes in our proposed scheme have strong ability to break the correlations of adjacent pixels.

Figure 9 plots the adjacent pixel pairs of the plain image and its cipher image encrypted by the proposed algorithm. It shows that the pixel pairs in the plain image mostly distribute on the diagonal line, which indicates strong correlations between adjacent pixels. However, the pixel pairs in the cipher image are randomly distributed on the whole plane, indicating the weak correlations between adjacent pixels. It indicates that the proposed image encryption algorithm can effectively break the strong correlations of the adjacent pixels in a plain image.

Table 12 lists the Corr values of cipher images encrypted by different image encryption schemes. It shows that the cipher images encrypted by our proposed image encryption algorithm have the smallest Corr values, compared with other encryption schemes.

5.5 Ability to resist chosen-plaintext attack

The chosen-plaintext attack is an effective and commonly used security attack. By choosing some special images such as all-zero images to encrypt and analyzing the obtained cipher images, the attackers can constructed the equivalent keys for the encryption processes. Using these equivalent keys, the attackers can successfully attack a cipher image without the secret key. For example, in [23], the attacker first constructs plain images with zeros and ones and then build equivalent keys for the permutation process. In [24], the attack can also build the equivalent keys for the diffusion process by selecting plain images to encrypt. In these attacks, the permutation matrices and diffusion matrices are only related to the secret keys and are fixed for different plain images. Using chosen-plaintext attack, the equivalent keys for the permutation and diffusion processed can be constructed.

However, in the proposed image encryption scheme, the permutation and diffusion matrices are related to both the plain image and secret key. In the permutation process, the row permutation matrix and column permutation matrix are related to the first value of each row and column of the plain image. In the diffusion process, the i-th value of the diffusion matrix is related to the $(i + 1)$-th pixel of the image. Thus, when encrypting different plain images, the permutation and diffusion matrices are different. Then it is impossible to construct the equivalent keys in the chosen-plaintext attack. Thus, the proposed encryption algorithm has strong ability to resist the chosen-plaintext attack. In addition, since the chosen-plaintext attack is more powerful than the ciphertext-only and known-plaintext attacks, our encryption scheme can also well defense these security attacks.

6 Conclusion

In this paper, we proposed a new S-box construction method using the complete Latin square and enhanced

logistic map. First, a complete Latin square is generated using the chaotic sequences produced by the enhanced logistic map. By expanding the complete Latin square, a pair of orthogonal matrices can be generated. The *S*-box is generated after scrambling and combining the orthogonal matrices. The performance of the proposed *S*-box is analyzed using the nonlinearity, strict avalanche criterion, bit independence criterion and differential approximation probability. The analysis results show that the proposed *S*-box has strong ability to resist different kinds of security attacks, and it has better performance than several newly proposed *S*-boxes. To show the application of the proposed *S*-box, we design an image encryption algorithm. The main components of the algorithm are high-efficiency permutation, *S*-box-based confusion, feedback diffusion and finite field multiplication. The first two operations can randomly shuffle the pixel positions with high security. The feedback diffusion operation can change the pixel values and spread the change of a pixel to the whole cipher image. Simulation results show that the image encryption algorithm proposed has strong ability to encrypt different kinds of digital images into unrecognizable cipher images. The security analyses demonstrate that the image encryption algorithm proposed has better performance than some advanced image encryption algorithms. Our future work will investigate the construction of 3D *S*-box and its application in color image.

Acknowledgements This work was supported in part by the Natural Scientific Research Innovation Foundation in Harbin Institute of Technology under Grant HIT.NSRIF.2020077, in part by the National Natural Science Foundation of China under Grants 62071142 and 62002301, in part by the Natural Science Foundation of Chongqing under Grant cstc2019jcyj-msxmX0393, and in part by the Education Committee foundation of Chongqing under Grant KJQN201900305.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Alawida, M., Samsudin, A., Teh, J.S., Alkhawaldeh, R.S.: A new hybrid digital chaotic system with applications in image encryption. Signal Process. **160**, 45–58 (2019)
2. Anees, A., Ahmed, Z.: A technique for designing substitution box based on Van der Pol oscillator. Wirel. Pers. Commun. **82**(3), 1497–1503 (2015)
3. Bao, H., Hua, Z., Wang, N., Zhu, L., Chen, M., Bao, B.C.: Initials-boosted coexisting chaos in a 2D sine map and its hardware implementation. IEEE Trans. Ind. Inform. (2020)
4. Biham, E., Shamir, A.: Differential cryptanalysis of DES-like cryptosystems. J. Cryptol. **4**(1), 3–72 (1991)
5. Cao, C., Sun, K., Liu, W.: A novel bit-level image encryption algorithm based on 2D-LICM hyperchaotic map. Signal Process. **143**, 122–133 (2018)
6. Castro, J.C.H., Sierra, J.M., Seznec, A., Izquierdo, A., Ribagorda, A.: The strict avalanche criterion randomness test. Math. Comput. Simul. **68**(1), 1–7 (2005)
7. Chen, G., Chen, Y., Liao, X.: An extended method for obtaining S-boxes based on three-dimensional chaotic Baker maps. Chaos Solitons Fractals **31**(3), 571–579 (2007)
8. Chen, J., Han, F., Qian, W., Yao, Y.D., Zhu, Z.I.: Cryptanalysis and improvement in an image encryption scheme using combination of the 1D chaotic map. Nonlinear Dyn. **93**(4), 2399–2413 (2018)
9. Guesmi, R., Farah, M.A.B., Kachouri, A., Samet, M.: A novel design of chaos based S-boxes using genetic algorithm techniques. In: 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA). IEEE, pp 678–684 (2014)
10. Hua, Z., Zhang, Y., Zhou, Y.: Two-dimensional modular chaotification system for improving chaos complexity. IEEE Trans. Signal Process. **68**, 1937–1949 (2020)
11. Hua, Z., Zhou, B., Zhou, Y.: Sine chaotification model for enhancing chaos and its hardware implementation. IEEE Trans. Ind. Electron. **66**(2), 1273–1284 (2018)
12. Hua, Z., Zhou, Y.: Design of image cipher using block-based scrambling and image filtering. Inf. Sci. **396**, 97–113 (2017)
13. Hua, Z., Zhou, Y., Pun, C.M., Chen, C.P.: Image encryption using 2D Logistic-Sine chaotic map. In: 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 3229–3234. IEEE (2014)
14. Hua, Z., Zhu, Z., Yi, S., Zhang, Z., Huang, H.: Cross-plane colour image encryption using a two-dimensional logistic tent modular map. Inf. Sci. **546**, 1063–1083 (2021)
15. Huang, X., Dong, Y., Jiao, K., Ye, G.: Asymmetric pixel confusion algorithm for images based on RSA and Arnold transform. Front. Inf. Technol. Electron. Eng. **21**(12), 1783–1794 (2020). https://doi.org/10.1631/FITEE.2000241
16. Hussain, I., Shah, T., Gondal, M.A., Khan, W.A.: Construction of cryptographically strong 8×8 S-boxes. World Appl. Sci. J. **13**(11), 2389–2395 (2011)
17. Hussain, I., Shah, T., Gondal, M.A., Khan, W.A., Mahmood, H.: A group theoretic approach to construct cryptographically strong substitution boxes. Neural Comput. Appl. **23**(1), 97–104 (2013)
18. Jamal, S.S., Khan, M.U., Shah, T.: A watermarking technique with chaotic fractional S-box transformation. Wirel. Pers. Commun. **90**(4), 2033–2049 (2016)
19. Jia, Y., Yin, Z., Zhang, X., Luo, Y.: Reversible data hiding based on reducing invalid shifting of pixels in histogram shifting. Signal Process. **163**, 238–246 (2019)
20. Khairullin, I., Bobrov, V.: On cryptographic properties of some lightweight algorithms and its application to the construction of S-boxes. In: 2019 IEEE Conference of

Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), pp. 1807–1810. IEEE (2019)
21. Khan, M., Asghar, Z.: A novel construction of substitution box for image encryption applications with gingerbreadman chaotic map and S_8 permutation. Neural Comput. Appl. **29**(4), 993–999 (2018)
22. Khan, M., Shah, T., Batool, S.I.: Construction of S-box based on chaotic Boolean functions and its application in image encryption. Neural Comput. Appl. **27**(3), 677–685 (2016)
23. Li, C., Lin, D., Lü, J.: Cryptanalyzing an image-scrambling encryption algorithm of pixel bits. IEEE Multimed. **24**(3), 64–71 (2017)
24. Li, C., Zhang, Y., Xie, E.Y.: When an attacker meets a cipher-image in 2018: a year in review. J. Inf. Sec. Appl. **48**, 102361 (2019)
25. Li, H., Hua, Z., Bao, H., Zhu, L., Chen, M., Bao, B.: Two-dimensional memristive hyperchaotic maps and application in secure communication. IEEE Trans. Ind, Electron (2020)
26. Liu, H., Kadir, A., Xu, C.: Cryptanalysis and constructing S-box based on chaotic map and backtracking. Appl. Math. Comput. **376**, 125 (2020)
27. Liu, L., Lei, Z.: An approach for constructing the S-box using the CML system. J. Phys. Conf. Ser. **1303**, 012090 (2019)
28. Liu, W., Sun, K., Zhu, C.: A fast image encryption algorithm based on chaotic map. Opt. Lasers Eng. **84**, 26–36 (2016)
29. Liu, Z., Wang, Y., Zhao, Y., Zhang, L.Y.: A stream cipher algorithm based on 2D coupled map lattice and partitioned cellular automata. Nonlinear Dyn. **101**, 1383–1396 (2020)
30. Luo, Y., Du, M., Liu, J.: A symmetrical image encryption scheme in wavelet and time domain. Commun. Nonlinear Sci. Numer. Simul. **20**(2), 447–460 (2015)
31. Naseer, A., Siddiqui, N.: A novel approach for construction of S-box using modified Pascal's triangle. Int. J. Comput. Sci. Inf. Sec. **18**(1) (2020)
32. Özkaynak, F., Çelik, V., Özer, A.B.: A new S-box construction method based on the fractional-order chaotic Chen system. Signal Image Video Process. **11**(4), 659–664 (2017)
33. Özkaynak, F., Yavuz, S.: Designing chaotic S-boxes based on time-delay chaotic system. Nonlinear Dyn. **74**(3), 551–557 (2013)
34. Ping, P., Xu, F., Mao, Y., Wang, Z.: Designing permutation-substitution image encryption networks with Henon map. Neurocomputing **283**, 53–63 (2018)
35. Qin, F., Wang, C., Li, Z., Kim, H.s., Zhou, Y., Wu, Y.: Lift: a low-overhead practical information flow tracking system for detecting security attacks. In: 2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06), pp. 135–148. IEEE (2006)
36. Song, Y., Zhu, Z., Zhang, W., Guo, L., Yang, X., Yu, H.: Joint image compression-encryption scheme using entropy coding and compressive sensing. Nonlinear Dyn. **95**(3), 2235–2261 (2019)
37. Sosa, P.M.: Calculating nonlinearity of Boolean functions with Walsh–Hadamard Transform, pp. 1–4. UCSB, Santa Barbara (2016)
38. Tang, G., Liao, X.: A method for designing dynamical S-boxes based on discretized chaotic map. Chaos Solitons Fractals **23**(5), 1901–1909 (2005)
39. Tang, G., Liao, X., Chen, Y.: A novel method for designing S-boxes based on chaotic maps. Chaos Solitons Fractals **23**(2), 413–419 (2005)
40. Tanyildizi, E., Özkaynak, F.: A new chaotic S-box generation method using parameter optimization of one dimensional chaotic maps. IEEE Access **7**, 117829–117838 (2019)
41. Ullah, A., Jamal, S.S., Shah, T.: A novel scheme for image encryption using substitution box and chaotic system. Nonlinear Dyn. **91**(1), 359–370 (2018)
42. Wang, S., Wang, C., Xu, C.: An image encryption algorithm based on a hidden attractor chaos system and the Knuth-Durstenfeld algorithm. Opt. Lasers Eng. **128**, 105995 (2020)
43. Webster, A., Tavares, S.E.: On the design of S-boxes. In: Conference on the Theory and Application of Cryptographic Techniques, pp. 523–534. Springer (1985)
44. Wen, W., Wei, K., Zhang, Y., Fang, Y., Li, M.: Colour light field image encryption based on DNA sequences and chaotic systems. Nonlinear Dyn. **99**(2), 1587–1600 (2020)
45. Williams, E.: Experimental designs balanced for the estimation of residual effects of treatments. Aust. J. Chem. **2**(2), 149–168 (1949)
46. Wu, Y., Noonan, J.P., Agaian, S., et al.: NPCR and UACI randomness tests for image encryption. Cyber journals: multidisciplinary journals in science and technology. J. Sel. Areas Telecommun. (JSAT) **1**(2), 31–38 (2011)
47. Xia, Z., Wang, X., Zhou, W., Li, R., Wang, C., Zhang, C.: Color medical image lossless watermarking using chaotic system and accurate quaternion polar harmonic transforms. Signal Process. **157**, 108–118 (2019)
48. Xu, C., Sun, J., Wang, C.: An image encryption algorithm based on random walk and hyperchaotic systems. Int. J. Bifurc. Chaos **30**(4), 2050060 (2020)
49. Xu, L., Li, Z., Li, J., Hua, W.: A novel bit-level image encryption algorithm based on chaotic maps. Opt. Lasers Eng. **78**, 17–25 (2016)
50. Xu, Y.M., Yao, Z., Hobiny, A., Ma, J.: Differential coupling contributes to synchronization via a capacitor connection between chaotic circuits. Front. Inf. Technol. Electron. Eng. **20**(4), 571–583 (2019)
51. Yang, L., Lv, X.: Image encryptionalgorithm based on complete Latin square. Appl. Res. Comput. **32**(11), 3435–3438 (2015)
52. Ye, G., Pan, C., Huang, X., Mei, Q.: An efficient pixel-level chaotic image encryption algorithm. Nonlinear Dyn. **94**(1), 745–756 (2018)
53. Zhang, Y., Li, Y., Wen, W., Wu, Y., Chen, J.X.: Deciphering an image cipher based on 3-cell chaotic map and biological operations. Nonlinear Dyn. **82**(4), 1831–1837 (2015)
54. Zhang, Y., Ren, G., Hobiny, A., Ahmad, B., Ma, J.: Mode transition in a memristive dynamical system and its application in image encryption. Int. J. Mod. Phys. B **34**(27), 2050244 (2020)
55. Zhao, C.F., Ren, H.P.: Image encryption based on hyperchaotic multi-attractors. Nonlinear Dyn. **100**, 679–698 (2020)