symmetry

MDPI

Article

Cryptanalysis and Improvement on an Image Encryption Algorithm Design Using a Novel Chaos Based S-Box

Congxu Zhu [1,2,*], Guojun Wang [3] and Kehui Sun [4]

[1] School of Information Science and Engineering, Central South University, Changsha 410083, China
[2] Guangxi Colleges and Universities Key Laboratory of Complex System Optimization and Big Data Processing, Yulin Normal University, Yulin 537000, China
[3] School of Computer Science and Technology, Guangzhou University, Guangzhou 510006, China; csgjwang@163.com
[4] School of Physics and Electronics, Central South University, Changsha 410083, China; kehui@csu.edu.cn
* Correspondence: zhucx@csu.edu.cn or zhucongxu@126.com; Tel.: +86-189-7583-1476

Received: 13 August 2018; Accepted: 7 September 2018; Published: 14 September 2018

Abstract: This article performs the cryptanalysis of an image encryption algorithm using an S-box generated by chaos. The algorithm has the advantages of simple structure, high encryption efficiency, and good encryption performance. However, an attentive investigation reveals that it has some undiscovered security flaws. The image cryptosystem is totally breakable under proposed chosen-plaintext attack, and only two chosen plain-images are required. An array equivalent to the S-box is constructed by an elaborately designed chosen-plaintext image, and the cipher-image is deciphered without having to know the S-box itself. Both mathematical deduction and experimental results validate the feasibility of the attacking scheme. Furthermore, an improved encryption scheme is proposed, in which a feedback mechanism is introduced, a bidirectional diffusion scheme is designed, and values of the ciphertext are associated with more parameters in each diffusion process. Testing results and security analysis verify that the improved cryptographic system can achieve a higher security level and has a better performance than some of the latest encryption algorithms.

Keywords: image encryption; cryptanalysis; chaos; S-box; improved diffusion

1. Introduction

At present, more and more digital information needs to be transmitted in the public network. Therefore, ensuring the security of confidential or private information in transmission is particularly important. Encryption is the basic means to ensure the safe transmission of information. However, the method of image encryption can not be the same as those for text information encryption. In encrypting image information, people must consider some inherent characteristics of images, such as the large data characteristics of images, high redundancy characteristics, and the strong correlation between adjacent data [1]. Due to the many natural connections between chaos and cryptography, chaos becomes a good candidate in designing image encryption algorithms, and many image encryption algorithms using chaos have been proposed [2–7].

In recent years, many researchers are interested in constructing S-boxes using chaos, and applying S-boxes to image encryption. Wang et al. [8] designed S-boxes by chaotic Kent map and Logistic map and used the S-boxes for image encryption. Liu et al. [9] constructed a dynamic S-box to enhance the image encryption effect. Khan et al. [10] constructed a new S-box by using a random bit sequence generated by chaotic boolean functions and used the S-box for image encryption. Çavuşoğlu et al. [11] put forward a new algorithm for generating S-box, with high complexity, by using a novel chaotic

system. Belazi A et al. [12] proposed an advanced method of generating strong S-boxes by using a chaotic map in color image encryption. Liu et al. [13] utilized dynamic S-boxes and chaotic systems to encrypt image. The S-boxes were constructed by chaos-based DNA sequence. Devaraj et al. [14] employed dynamic S-boxes to design an image cryptosystem. The S-boxes were used to substitute pixel values of the image. Very recently, Çavuşoğlu et al. [15] proposed an image encryption scheme by using a new S-box generated by a hyper-chaotic system, which has the advantages of simple structure, high encryption speed, and good encryption performance.

Compared with information encryption, cryptanalysis is another research area to decipher keys or to target ciphertext [16–19]. Cryptanalysis can also reveal defects in cryptographic algorithms, and can also facilitate the development of cryptography. In addition, cryptanalysis can also prevent unsafe encryption algorithms from being applied to actual communications. Recent cryptanalysis researches show that some chaos based image encryption algorithms are not secure enough. For example, the image encryption scheme in Reference [9] was broken by Zhang et al. [20]. The image encryption scheme in Reference [21] was broken by Zhu and Sun [22]. The image encryption scheme in Reference [23] was broken by Ahmad et al. [24]. In order to promote cryptanalysis research and enhance the security of image encryption systems, this article analyzes the image encryption algorithm using chaos S-boxes in Reference [15]. For simplicity sake, the algorithm under study is abbreviated as "IESB" in the following content.

The rest of the present paper is organized as follows. Section 2 describes briefly the IESB under study. Detailed cryptanalysis on IESB is presented mathematically and experimentally in Section 3. The improved version of encryption algorithm is given in Section 4. Finally, Section 5 summarize the research results.

2. Description of the Original Encryption Algorithm

The plain-text images to be encrypted in IESB are 8-bit gray-scale images of size $M \times N$ (rows $\times$ columns), which can be expressed by a matrix $\mathbf{A} = [a(i, j)]$, $a(i, j) \in \{0, 1, ..., 255\}$, $i = 1, 2, \ldots, M$, $j = 1, 2, \ldots, N$. Before encryption, the two-dimensional matrix $\mathbf{A}$ is transformed into a 1D (one-dimensional) pixel vector $\mathbf{P} = [p(1), p(2), ..., p(L)]$, where $L = MN$. In this paper, an integer i in the bracket "(i)" denotes the subscript of an element in an array. The main steps of IESB can be described briefly below.

2.1. The Secret Keys and Flow Chart of IESB

The mathematical model describing the hyper chaotic system employed in IESB is as follows:

$$\begin{cases} \dot{x} = cy - x - bz \\ \dot{y} = axz - xy - bx \\ \dot{z} = dxy + b \end{cases} \quad (1)$$

In Equation (1), a, b, c, and d are the system parameters. When $a \in (0.55, 5.3)$, $b \in [0.5, 2.5]$, $c \in [1, 11.8]$, and $d \in [-15.5, -0.5]$, system (1) is chaotic. The initial condition (x_0, y_0, z_0) and parameters (a, b, c, d) are utilized as secret keys in IESB. The flow chart of IESB can be shown in Figure 1.

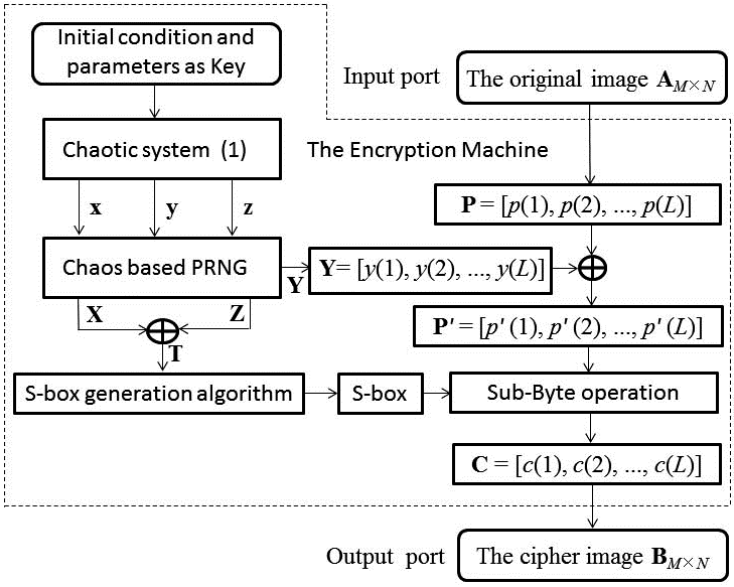


Figure 1. The flow chart of the original encryption algorithm.

2.2. Generating Chaotic Pseudo Random Number Sequences and S-Box

Firstly, based on system (1), three chaotic floating-point number sequences **x**, **y**, **z** are generated. Then, the three floating-point number sequences **x**, **y**, **z** are converted to three integer sequences **X**, **Y**, **Z** by chaos based pseudo random number generator (PRNG). Each element in **X**, **Y**, **Z** is a binary number of 8 bits and the three bit series have passed NIST tests. The S-box is created by using sequences **X**, **Z** and a novel S-box generation algorithm [15]. The S-box is a 16 × 16 sized matrix, and each number in the S-box is a unique decimal integer value between 0 and 255. The sequence $\mathbf{Y} = [y(1), y(2), \ldots, y(L)]$ and the S-box will be used for image encryption, where $L = M \times N$.

2.3. The Encryption Procedure

The encryption procedure consists of two steps which are described below.

The first step is bitwise XOR encryption with sequence **Y**:

$$p'(i) = y(i) \bigoplus p(i), i = 1, 2, \ldots, L \tag{2}$$

where, $p(i)$ and $p'(i)$ are the i-th pixel value of the plaintext image and the intermediate ciphertext image, respectively. The symbol $\oplus$ symbolizes XOR operation performed in binary bits. After the bitwise XOR operation, each byte is converted to a decimal value and the intermediate ciphertext image array $\mathbf{P'} = [p'(1), p'(2), \ldots, p'(L)]$ is generated.

The second step is sub-byte operation with the 16 × 16 sized S-box:

Suppose the 16 × 16 sized S-box is expressed as $\mathbf{SB} = [sb(j, k)]$, $j = 1, 2, \ldots, 16$, $k = 1, 2, \ldots, 16$. In fact, **SB** is a collection of all elements on the finite field GF (2^8), namely, each value of $sb(j, k)$ is a specific integer in the set {0, 1, 2, ..., 255}. The sub-byte operation process is actually mapping each pixel value $p'(i)$ of the intermediate ciphertext image into an element $sb(j, k)$ in **S**. The mapping rules from $p'(i)$ to $sb(j, k)$ can be explained in detail below. Firstly, a pixel value $p'(i)$ is represented as binary form $(b_8b_7 \ldots b_2b_1)_2$. Then, let $j = (b_8b_7b_6b_5)_2 + 1$, $k = (b_4b_3b_2b_1)_2 + 1$, and $j = 1, 2, \ldots, 16$, $k = 1, 2, \ldots, 16$. Finally, one can obtain the i-th cipher pixel value $c(i) = sb(j, k)$. After all cipher pixel values are determined, the encrypted image array $\mathbf{C} = [c(1), c(2), \ldots, c(L)]$ is obtained. In a general way, the effect of the S-box is equivalent to an abstract function fs: $p'(i) \in \mathrm{GF}(2^8) \rightarrow c(i) \in \mathrm{GF}(2^8)$, which is a

one-to-one mapping function $fs[\bullet]$. Namely, if $fs[x] = fs[y]$ holds, then $x = y$ must hold. The mapping from $p'(i)$ to $c(i)$ can be expressed as:

$$c(i) = fs[p'(i)], i = 1, 2, \ldots, L \quad (3)$$

3. The Cryptanalysis and Chosen-Plaintext Attacks

In general, we assume that the attacker knows the details of the cryptographic algorithm under analysis, and this assumption is called the Kerckhoff hypothesis. Namely, the opponent or attacker knows the algorithm of the encryption system, but does not own the decryption secret keys. With regard to the chosen-plaintext attack, the opponent or attacker can temporarily get the opportunity to use the encryption machinery, so the opponent or attacker can select some special plain images and encrypt the selected plain images to obtain their corresponding cipher images. By using the known plaintext-ciphertext pairs, the attacker can break the key or target ciphertext image encrypted by the cryptographic system.

3.1. The Algorithm of Cryptanalysis and Chosen-Plaintext Attacks

In IESB, the constituent elements of the secret key set include (x_0, y_0, z_0, a, b, c, and d). It is worth noting that the chaotic sequence **Y** and the S-box can act as an equivalent to the secret keys. The sequence **Y** and S-box are not related to the image to be encrypted. In other words, different encrypted images have the same keys **Y** and S-box. Assuming such a premise, the pixel array of the target cipher image to be recovered is $\mathbf{C} = [c(1), c(2), \ldots, c(L)]$. We use $\mathbf{P} = [p(1), p(2), \ldots, p(L)]$ to represent the pixel array of the plaintext image corresponding to **C**. **P** is unknown and is to be deciphered. Our scheme of chosen-plaintext attack can be described in detail as follows.

Step 1. Constructing the first chosen-plaintext image $\mathbf{P0} = [0, 0, \ldots, 0]$ and the output corresponds to the cipher image $\mathbf{C0} = [c0(1), c0(2), \ldots, c0(L)]$ by the encryption machinery, where $L \geq 65{,}536$. According to Equation (2), elements in the intermediate image vector $\mathbf{P0}'$ and the keystream **Y** satisfies the following relation:

$$p0'(i) = y(i), i = 1, 2, \ldots, L \quad (4)$$

According to Equations (3) and (4), we get the following relationship between **C0** and **Y**:

$$c0(i) = fs[y(i)], i = 1, 2, \ldots, L \quad (5)$$

Step 2. Consider such a fact that, pixel values of the encrypted image have only 256 levels over the set {0, 1, ..., 255}. Hence, each level has an average of $L/256$ times appearing in **C0**, which is bigger than or equal to 256 due to $L \geq 65{,}536$. Find a pixel value with at least of 256 times appearing in **C0**. Let us say that m is one of the values that satisfies the condition. That is to say, there are at least 256 elements in **C0** equal to m.

Step 3. Constructing the second chosen-plaintext image vector $\mathbf{P1} = [p1(1), p1(2), \ldots, p1(L)]$, such that

$$\begin{cases} p1(i) = 0, \text{ if } c0(i) \neq m \\ p1(i) = j, \text{ if } c0(i) = m \end{cases}, i = 1, 2, \ldots, L, j = 0, 1, \ldots, 255 \quad (6)$$

The Matlab code of constructing **P1** is shown in Algorithm 1:

Algorithm 1: The Matlab code of constructing **P1**

```
1. p1 = zeros(1, L);
2. j = 0;
3. for i = 1: L
4.       if c0(i) = = m
5.          p1(i) = j;
6.          j = j + 1;
7.       end
8.       if j = = 256
9.          break;
10.      end
11. end
```

Then obtain the corresponding ciphertext image **C1** = [$c1(1)$, $c1(2)$, . . . , $c1(L)$] by using the encryption machinery.

Step 4. Constructing an array **S** = [$s(1)$, $s(2)$, . . . , $s(256)$] that is equivalent to the S-box. The Matlab code of constructing **S** is shown in Algorithm 2:

Algorithm 2: The Matlab code of constructing **S**

```
1. s = zeros(1, 256);
2. j = 1;
3. for i = 1: L
4.       if c0(i) = = m
5.          s(j) = c1(i);
6.          j = j + 1;
7.       end
8.       if j > 256
9.          break;
10.      end
11. end
```

It is worth noting that $y(i)$ are of the same value, say y_0, in all places, i satisfies condition $c0(i) = m$, $i = i1, i2, \ldots, i256$. According to Equations (2) and (3), we have the following relations: $s(1) = c1(i1) = fs[y_0 \oplus 0]$, $s(2) = c1(i2) = fs[y_0 \oplus 1]$, ..., $s(256) = c1(i256) = fs[y_0 \oplus 255]$. Then, we obtain the following general expression:

$$s(i) = fs[y_0 \bigoplus (i - 1)],\ i = 1, 2, \ldots, 256. \tag{7}$$

Obviously, $(y_0 \oplus 0)$, $(y_0 \oplus 1)$, $(y_0 \oplus 2)$, . . . , $(y_0 \oplus 255)$ are 256 numbers different from each other. It leads to $s(1)$, $s(2)$, . . . , and $s(256)$ are different from each other.

Step 5. For each element $c0(i)$, find out where it is in **S** and obtain a position index array **u**.

Initially, set an array **u** = **[**$u(i)$**]** and $u(i) = 0$, $i = 1, 2, \ldots, L$. For each $i = 1, 2, \ldots, L$, find out where $c0(i)$ in **S**. If $c0(i) = s(j)$, then record the number j corresponding to $c0(i)$. Therefore, let $u(i) = j$. Due to the following map relations: $s(j) = fs[y_0 \oplus (j - 1)]$, and $c0(i) = fs[y(i)]$, hence, the following results are obtained by $c0(i) = s(j)$ and $j = u(i)$:

$$y(i) = y_0 \bigoplus [u(i) - 1], \tag{8}$$

Step 6. For each element $c(i)$, find out where it is in **S** and obtain a position index array **v.**

Initially, set an array **v** = **[**$v(i)$**]** and $v(i) = 0$, $i = 1, 2, \ldots, L$. For each $i = 1, 2, \ldots, L$, find out where $c(i)$ in **S**. If $c(i) = s(k)$, then record the number k corresponding to $c(i)$. Therefore, let $v(i) = k$. Because

of the following map relations: $s(k) = fs[y_0 \bigoplus (k-1)]$, and $c(i) = fs[y(i) \bigoplus p(i)]$, hence, the following results are obtained by $c(i) = s(k)$ and $k = v(i)$:

$$y(i) \bigoplus p(i) = y_0 \bigoplus [v(i) - 1], \tag{9}$$

Step 7. Utilize **u** and **v** to recover the plaintext image.

According to Equations (8) and (9), we derive the following relation as show in Equation (10):

$$p(i) = [u(i) - 1] \bigoplus [v(i) - 1]. \tag{10}$$

Step 8. Execute Equation (10) repeatedly for each $i = 1, 2, \ldots, L$, then we can recover the plaintext image vector $\mathbf{P} = [p(1), p(2), \ldots, p(L)]^{\mathrm{T}}$. Transform **P** into $M \times N$ matrix **A**, the plaintext image is finally obtained.

3.2. Examples of Chosen-Plaintext Attacks

To verify the effectiveness of our chosen-plaintext attacks, two specific experimental examples are provided in this section. The encryption and decryption algorithms are implemented in the programming language of Matlab R2016b, and the computer has the 3.30 GHz Intel(R) Core (TM) i5-4590 CPU with 4 GB memory and Microsoft Windows 7 64 bit operation system.

3.2.1. The Secret Keys and S-Box

The original keys we select are $a = 1.0$, $b = 1.0$, $c = 2.0$, $d = -3.0$, $x_0 = 1.0$, $y_0 = -1.0$, and $z_0 = 0.01$. Chaotic system (1) is solved by the ode45 solver of Matlab, and the time step is set to 0.01.

Chaotic system (1) is iterated 65,536 times and three chaotic sequences with size of 65,536 are generated (initial point is not included). The three original chaotic sequences are $\mathbf{x} = [x(1), x(2), \ldots, x(L)]$, $\mathbf{y} = [y(1), y(2), \ldots, y(L)]$, $\mathbf{z} = [z(1), z(2), \ldots, z(L)]$, and $L = 65{,}536$. Without loss of generality, we convert each float value of the original chaotic sequences to an 8-bits binary number by using the chaos based PRNG, which is expressed by Equations (11)–(13):

$$X(i) = mod(floor((|x(i)| \times 10^6 - floor(|x(i)| \times 10^6)) \times 10^3), 256); i = 1, 2, \ldots, L \tag{11}$$

$$Y(i) = mod(floor((|y(i)| \times 10^6 - floor(|y(i)| \times 10^6)) \times 10^3), 256); i = 1, 2, \ldots, L \tag{12}$$

$$Z(i) = mod(floor((|z(i)| \times 10^6 - floor(|z(i)| \times 10^6)) \times 10^3), 256); i = 1, 2, \ldots, L \tag{13}$$

where, $|x| = x$ (if $x \geq 0$) or $|x| = -x$ (if $x < 0$), the value of $floor(x)$ is a maximal integer that satisfies $floor(x) \leq x$, and the value of $mod(x, y)$ is the remainder when x is divided by y. Then, we get a new sequence **T** by using the sequence **X** and **Z**, namely, $T(i) = X(i) \bigoplus Z(i)$, $i = 1, 2, \ldots, L$. For simplicity, we select 256 different values from the sequence **T** and use these values to form the S-box matrix, which is shown in Table 1. The first 16 numbers in the sequence **Y** are also given here for reference, which are shown in Equation (14).

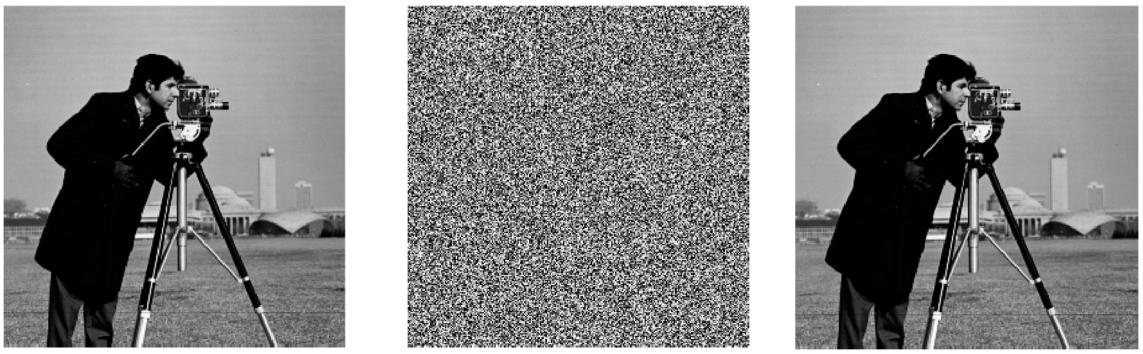
$$\mathrm{Y} = [126, 44, 130, 138, 46, 233, 44, 101, 178, 188, 137, 29, 94, 207, 9, 14, \ldots]. \tag{14}$$

Table 1. The proposed S-box come from chaos.

110	108	239	99	42	160	187	36	157	222	152	50	92	199	30	249
161	198	138	16	208	106	130	212	189	181	64	248	34	191	240	224
4	62	111	103	126	53	128	205	251	172	39	132	183	3	94	185
247	158	237	41	244	216	52	154	250	223	8	168	25	93	221	238
26	202	21	136	2	67	15	195	6	121	51	1	69	63	148	167
209	135	107	137	97	231	71	176	233	47	14	76	56	230	213	24
232	57	80	40	95	175	5	100	104	22	206	169	124	49	165	170
19	112	147	193	139	82	245	27	225	214	101	174	59	43	227	142
156	68	171	72	252	105	17	120	9	0	48	31	178	23	96	91
54	140	87	116	7	242	153	85	173	89	229	226	179	143	151	188
145	66	115	246	190	113	35	194	228	114	29	33	79	196	84	123
155	150	220	81	75	90	164	215	55	73	129	88	200	18	146	44
162	197	217	207	184	163	159	133	203	236	11	61	98	235	186	58
134	32	38	102	28	255	10	254	177	12	182	46	218	243	77	45
144	192	70	234	119	13	83	125	122	109	37	180	211	166	127	118
117	60	141	253	149	86	74	131	219	201	210	78	241	65	204	20

3.2.2. Breaking the Encrypted Test Image

The plaintext image used in the experiment is the Cameraman with a size of 256 × 256, which come from the test image database of the Computer Vision Group at University of Granada. The plaintext image Cameraman is shown in Figure 2a and the encrypted image is shown in Figure 2b. The recovered image by using our chosen-plaintext attacks is the image in Figure 2c, which is exactly the same as the original image shown in Figure 2a. Through the image Cameraman as an example, our attack attains demonstration.



(**a**) (**b**) (**c**)

Figure 2. The attack result: (**a**) The plain image Cameraman; (**b**) The cipher image; and (**c**) The cracked image.

3.2.3. A Simple Numerical Example

Suppose the plaintext image is a 2 × 2 sized gray image, whose one-dimensional pixel vector is

$$\mathbf{P} = [10, 0, 255, 128]. \tag{15}$$

The one-dimensional pixel vector of cipher image encrypted by IESB is

$$\mathbf{C} = [195, 217, 254, 145]. \tag{16}$$

Suppose **P** is unknown, we use **PP** = [$pp(1)$, $pp(2)$, $pp(3)$, $pp(4)$] to represent the plaintext version recovered from **C**.

The first chosen-plaintext image is **P0** = [0, 0, ..., 0], whose size is 256 × 256. Then the corresponding cipher image **C0** is obtained by using the IESB, whose first four pixel values are

$$\mathbf{C0} = [125, 217, 251, 228, ...]. \tag{17}$$

We find the pixel value 1 repeats more than 256 times in **C0**, and let $m = 1$. Then the second chosen-plaintext image **P1** is constructed according to the method mentioned in Section 3.1, and the first four non-zero elements of **P1** are listed for the readers' reference: $p1(752) = 1$, $p1(1342) = 2$, $p1(1796) = 3$, $p1(2117) = 4$. Then, the corresponding cipher image **C1** is obtained, and the first four elements of the image **C1** are [125, 217, 251, 228, ...]. Then, the array **S** is obtained by using **C0** and **C1** and the method mentioned in Section 3.1, and all the 256 elements of the array **S** are listed below, among them, seven important figures worthy of attention are underlined. That is:

S = [1, 76, 169, 174, 50, 248, 132, 168, 61, 46, 180, 78, 31, 226, 33, 88, 51, 14, 206, 101, 152, 64, 39, 8,

11, 182, 37, 210, 48, 229, 29, 129, 121, 47, 22, 214, 222, 181, 172, 223, 236, 12, 109, 201, 0, 89, 114, 73,

6, 233, 104, 225, 157, 189, **251**, 250, 203, 177, 122, 219, 9, 173, **228**, 55, 167, 24, 170, 142, 249, 224,

185, 238, 58, 45, 118, 20, 91, 188, 123, 44, 148, 213, 165, 227, 30, 240, 94, 221, 186, 77, 127, 204, 96,

151, 84, 146, 63, 230, 49, 43, 199, 191, 3, 93, 235, 243, 166, 65, 23, 143, 196, 18, 69, 56, 124, 59, 92, 34,

183, 25, 98, 218, 211, 241, 178, 179, 79, 200, 136, 137, 40, 193, 99, 16, 103, 41, 207, 102, 234, 253, 72,

116, 246, 81, 21, 107, 80, 147, 239, 138, 111, 237, **217**, 38, 70, 141, 171, 87, 115, 220, 202, 135, 57, 112,

108, 198, 62, 158, 197, 32, 192, 60, 68, 140, 66, 150, 26, 209, 232, 19, 110, 161, 4, 247, 162, 134, 144,

117, 156, 54, **145**, 155, **195**, 176, 100, 27, 36, 212, 205, 154, 133, **254**, **125**, 131, 120, 85, 194, 215, 15,

71, 5, 245, 187, 130, 128, 52, 159, 10, 83, 74, 17, 153, 35, 164, 67, 231, 175, 82, 160, 106, 53, 216, 163,

255, 13, 86, 105, 242, 113, 90, 2, 97, 95, 139, 42, 208, 126, 244, 184, 28, 119, 149, 252, 7, 190, 75].

Seek $c0(i)$ and $c(i)$ in **S**, we get the following results:

$c0(1) = 125 = s(203)$, $c(1) = 195 = s(193)$. Hence, $u(1) = 203$, $v(1) = 193$.

$c0(2) = 217 = s(153)$, $c(2) = 217 = s(153)$. Hence, $u(2) = 153$, $v(2) = 153$.

$c0(3) = 251 = s(55)$, $c(3) = 254 = s(202)$. Hence, $u(3) = 55$, $v(3) = 202$.

$c0(4) = 228 = s(63)$, $c(4) = 145 = s(191)$. Hence, $u(4) = 63$, $v(4) = 191$.

Therefore,

$$pp(1) = [u(1) - 1] \bigoplus [v(1) - 1] = 202 \bigoplus 192 = 10.$$

$$pp(2) = [u(2) - 1] \bigoplus [v(2) - 1] = 152 \bigoplus 152 = 0.$$

$$pp(3) = [u(3) - 1] \bigoplus [v(3) - 1] = 54 \bigoplus 201 = 255.$$

$$pp(4) = [u(4) - 1] \bigoplus [v(4) - 1] = 62 \bigoplus 190 = 128.$$

Consequently, **PP** = [10, 0, 255, 128], which means **PP** = **P**. Through the above examples, our chosen-plaintext attack scheme has been verified.

4. The Improved Image Encryption Scheme

The original algorithm IESB has the advantages of simple structure and high encryption efficiency. However, there is a security flaw in the cryptosystem, which results in that the IESB can not resist a chosen-plaintext attack. To overcome the security flaws, we redesign a new and improved image chaotic encryption scheme in this section.

4.1. Algorithm Description of the Improved Scheme

The improved encryption scheme consists of a three stage of process: Generate chaotic key sequences {X, Y, Z} and S-box, the first round diffusion, and the second round diffusion. The algorithm steps in our improved image encryption scheme are as follows:

Step 1. Set the initial conditions x_0, y_0, z_0 system parameters a, b, c, d of system (1). Read the plain image **I** of size $M \times N$ and transform its 2D matrix into a 1D pixel sequence $\mathbf{P} = [p(1), p(2), ..., p(L)]$, where $L = MN$.

Step 2. Iterate the chaotic system (1) with initial state values and system parameters for L_0 times and obtain three chaotic pseudorandom sequences $\mathbf{x} = [x(i)]$, $\mathbf{y} = [y(i)]$, $\mathbf{z} = [z(i)]$, where $i = 1, 2, ..., L_0$, $L_0 \geq 65{,}536$.

Step 3. Convert the original chaotic sequences to 8-bit integer sequences {**X**, **Y**, **Z**} by using Equations (11)–(13).

Step 4. Get sequence **T** by using **X** and **Z**: $\mathbf{T} = \mathbf{X} \oplus \mathbf{Z}$.

Step 5. From the elements of **T**, select 256 different numbers to generate the S-box with size of 16×16, and transform the 2D matrix of S-box into an equivalent 1D sequence $\mathbf{S} = [s(1), s(2), ..., s(256)]$.

Step 6. Perform the first round diffusion encryption with chaotic key sequence **Y**, and obtain the intermediate ciphertext image pixel sequence $\mathbf{P}' = [p'(i)]$. The operations are represented by the following formulas:

$$p'(1) = \mathrm{mod}(p(1) + y(1) + seed, 256); \tag{18}$$

$$p'(i) = \mathrm{mod}(p(i) + y(i) + p'(i-1), 256);\ i = 2, 3, ..., L. \tag{19}$$

where, $\mathrm{mod}(x, y)$ returns the remainder after x is divided by y. $p(i)$ is the i-th pixel value of the plaintext image and $p'(i)$ is the i-th pixel value of the intermediate ciphertext image. *seed* is a new parameter used as a key. Here, the tilde notations represent variables.

Step 7. Perform the second round diffusion encryption with the equivalent of chaotic S-box, and obtain the final ciphertext image pixel sequence $\mathbf{C} = [c(i)]$. The operations are represented by the following formulas:

$$j = \mathrm{double}(p'(L)) + 1;\ c(L) = \mathrm{mod}(p'(L) + s(j) + seed, 256); \tag{20}$$

$$J = \mathrm{double}(p(i)) + 1;\ c(i) = \mathrm{mod}(p'(i) + s(j) + c(i+1), 256);\ i = L - 1, L - 2, ...,1. \tag{21}$$

Step 8. Transform the 1D ciphertext image pixel sequence **C** into a 2D matrix **CI**, then the cipher image is obtained.

The operation steps of the decryption are the inverse process of the above encryption operation.

In our improved encryption process, the core algorithm is the diffusion operation, which repeats two rounds. The first one is performed in the forward order ($i = 1 \sim L$), the second round is performed in the reverse order ($i = L \sim 1$), and a feedback mechanism is introduced in the diffusion encryption process so that the former encrypted pixel value affects the ciphertext value of the latter pixel. Thus, the pixel at any position changes slightly and almost all ciphertext will change, making the algorithm highly sensitive to plaintext. In addition, values of the ciphertext are associated with more parameters in the two rounds of the diffusion encryption formula. Even if the attacker obtains the corresponding ciphertext with special selected plaintext, i.e., known $p(i)$ and $c(i)$, the secret keys $\{s(i), y(i), seed\}$ can not

be solved by the encryption relation, and the target ciphertext can not be decrypted directly. Therefore, our improved scheme can effectively resist chosen-plaintext attacks.

4.2. Performance Test and Analysis of the Improved Scheme

In order to confirm the security performance of the improved algorithm and to compare it with Reference [15] and other literature, we use the gray-scale image Lena of size 256 × 256 for simulation. The software and hardware environment for the simulation are the same as those in the Section 3.2. The initial secret key parameters used in the simulation are as: $a = 1.0$, $b = 1.0$, $c = 2.0$, $d = -3.0$, $x_0 = 1.0$, $y_0 = -1.0$, $z_0 = 0.01$, and $seed = 35$. The encryption results of the gray-scale Lena image is exhibited in Figure 3. From the results, one can see that the cipher image has nothing to do with the corresponding original plaintext image, and it becomes unrecognizable.

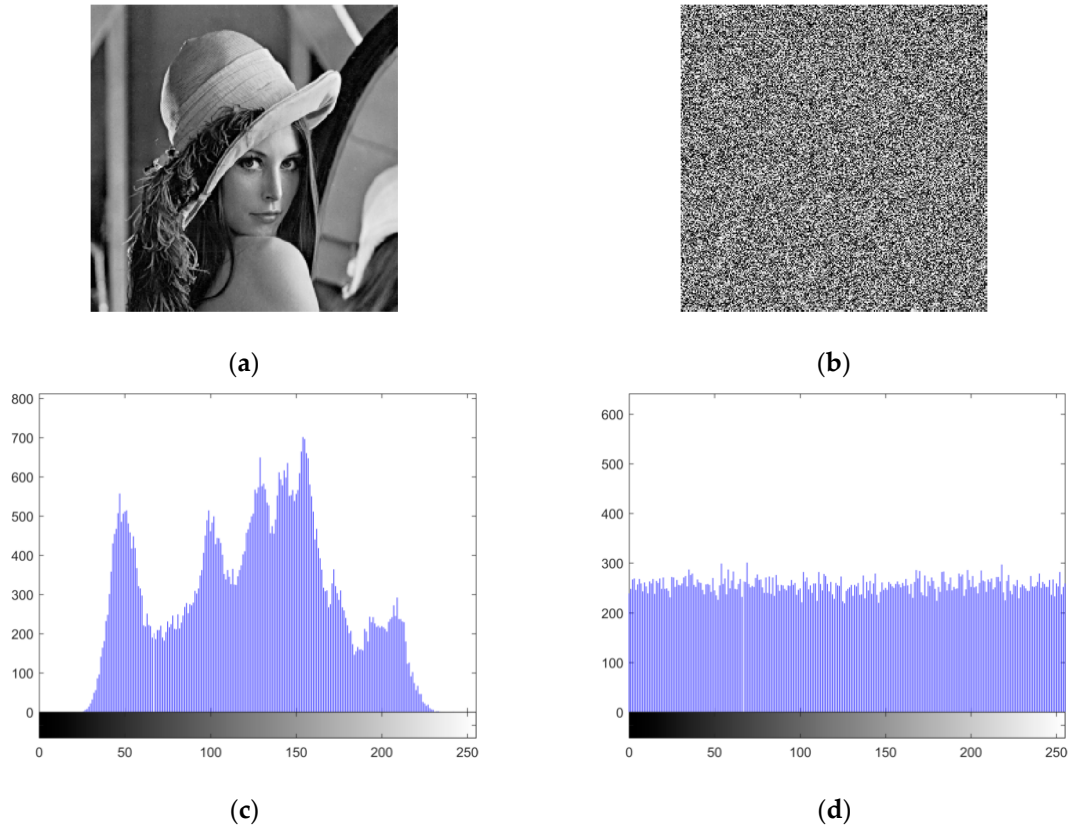


Figure 3. Encryption result for Lena gray-scale image: (**a**) Plain-image; (**b**) encrypted image; (**c**) histogram of plain image in (**a**); and (**d**) histogram of encrypted image in (**b**).

4.2.1. Histogram Analysis

A histogram can reveal the pixel values distribution situation in an image. Usually, the histogram of a meaningful plain image has a non-uniform distribution. For an image encryption algorithm with high security, the encrypted image must have the histogram with an uniform distribution. For the image Lena, the histograms of plain, and its cipher image, are available in Figure 3c,d. We can see that the histogram of the plain image is non-uniform, and the histogram of the cipher image is almost flat and uniform like the distribution of random data. Hence, the improved encryption scheme completely conceals the pixel distribution information of the original image, and can resist statistical attacks.

4.2.2. Pixels Correlation Analysis

Usually, adjacent pixels in a meaningful image have a very high correlation. A high security encryption algorithm must destroy the relevance between adjacent pixels in an image. The correlation

coefficient is the index to measure the correlation between adjacent pixels. The smaller the absolute value of correlation coefficient, the lower the correlation between adjacent pixels. The correlation coefficient γ_{xy} of adjacent pixels is computed as:

$$E(x) = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{22}$$

$$D(x) = \frac{1}{n}\sum_{i=1}^{n} [x_i - E(x)]^2, \tag{23}$$

$$\mathrm{Conv}(x,y) = \frac{1}{n}\sum_{i=1}^{n} [x_i - E(x)][y_i - E(y)], \tag{24}$$

$$\gamma_{xy} = \frac{\mathrm{Conv}(x,y)}{\sqrt{D(x)}\sqrt{D(y)}}. \tag{25}$$

where x and y are pixel values of two adjacent pixels in the image, γ_{xy} is the correlation coefficient of two adjacent pixels x and y. To compute correlation coefficients of the original plain image and its corresponding cipher image, we sampled all horizontally adjacent pairs of pixels. The results for the Lena image is listed in Table 2. Evidently, the encrypted image using our improved encryption scheme has smaller absolute values of correlation coefficient than the schemes in [2,15,23,24].

Table 2. Correlation Coefficients of Two Adjacent Pixels in Horizontal Direction.

Plain-Image	Cipher-Image	Cipher-Image Ref. [2]	Cipher-Image Ref. [15]	Cipher-Image Ref. [23]	Cipher-Image Ref. [24]
0.924879	0.000249	0.005497	0.5310	−0.00114	0.000329

4.2.3. Information Entropy Analysis

Information entropy is a common index to judge the randomness of an information source. Let s be an information source, and its information entropy is computed as:

$$H(s) = -\sum_{i=0}^{2^n-1} P(s_i)\log_2[P(s_i)], \tag{26}$$

where $P(s_i)$ denotes the occurrence probability of symbol s_i, 2^n is the total states of the information source. If $P(s_i) = 1/2^n$, then the information source is completely random. For an image with 256 gray-scale, the pixel values have 2^8 levels, so the ideal value of information entropy is 8. The information entropy of an encrypted image should be as close as possible to 8. The information entropy results of encrypted images by our improved scheme and schemes in References [9,15,23] are listed in Table 3. The results show that all entropy values are significantly closer to the ideal value eight, so the randomness is satisfactory. Besides, our improved scheme has better results than References [9,15,23]. Hence, our improved encryption scheme is more capable of resisting entropy-based attacks.

Table 3. Entropies of Encrypted Lena Image by Three Encryption Schemes.

Proposed	Ref. [9]	Ref. [15]	Ref. [23]
7.9977	7.993459	7.95667	7.9964

4.2.4. Sensitivity Analysis

In order to resist differential attacks, a fine encryption scheme should be very sensitive to minor alterations in plain images and any key components. When the key remains the same, if the plaintext

to be encrypted changes slightly, causing a huge change in the ciphertext, the encryption algorithm is said to be highly sensitive to plaintext. When the encrypted plaintext is kept the same, if the encryption key changes slightly, the ciphertext will change dramatically. The encryption algorithm is said to be highly sensitive to the key. The number of the pixel change rate (NPCR) and the unified average changing intensity (UACI) are two metrics to measure sensitivity. NPCR and UACI are defined as:

$$\text{NPCR} = \frac{\sum_{i,j} D(i,j)}{M_1 \times M_2} \times 100\%, \tag{27}$$

$$\text{UACI} = \frac{1}{M_1 \times M_2} \sum_{i,j} \frac{|c_1(i,j) - c_2(i,j)|}{255} \times 100\%. \tag{28}$$

where, $c_1(i, j)$ and $c_2(i, j)$ represent pixel values of two cipher images at the same position (i, j). $D(i, j)$ represents the difference between $c_1(i, j)$ and $c_2(i, j)$. If $c_1(i, j) = c_2(i, j)$ then $D(i, j) = 0$, otherwise $D(i, j) = 1$. For an 8-bit gray image, the optimal value of NPCR is $\text{NPCR}_\text{E} = 99.61\%$, the optimal value of UACI is $\text{UACI}_\text{E} = 33.46\%$.

When analyzing the sensitivity of the algorithm to the content of the plaintext, we encrypt two images, one is Lena, the other is Lena with one pixel $p(4000)$ and has a minor alteration value of + 1. Then, NPCR and UACI values are calculated from two ciphertext images. Table 4 exhibits the values of NPCR and UACI. From Table 4, one can see that the NPCR and UACI scores are quite near to the respective optimal values. Among these four algorithms, our improved scheme has the best performance. In fact, the IESB algorithm is not sensitive to plaintext, so both the PCR and UACI values, with regard to plaintext sensitivity, are close to zero. It is worth pointing out that the results of NPCR = 99.62987 and UACI = 31.83459 are also given in Reference [15], but the meaning is completely different. That is, the results are based on the difference between a plaintext image and the corresponding ciphertext image, not on the difference between the two ciphertext images.

Table 4. Results of NPCR and UACI for Plain Image Sensitivity.

	Proposed	**Ref. [15]**	**Ref. [24]**	**Ref. [2]**
NPCR (%)	99.63226	nearly 0	99.627	99.6002
UACI (%)	34.59600	nearly 0	33.452	33.463

To evaluate the key sensitivity, at first, we encrypt Lena image with keys ($x_0 = 1.0$, $y_0 = -1.0$, $z_0 = 0.01$, $a = 1.0$, $b = 1.0$, $c = 2.0$, $d = -3.0$, and $seed = 35$). Then, we added 10^{-14} to one of the floating-point key values, and +1 to the integer *seed* of the key, while all others stayed unchanged, and we encrypt the same plain image Lena again. Then, NPCR and UACI values are calculated from two ciphertext images. Table 5 shows the experimental results. From Table 5, one can see that our improved encryption scheme is sensitive to all secret keys.

Table 5. Results of NPCR and UACI for Key Sensitivity.

	$x_0 + 10^{-14}$	$y_0 + 10^{-14}$	$z_0 + 10^{-14}$	$a + 10^{-14}$	$b + 10^{-14}$	$c + 10^{-14}$	$d + 10^{-14}$	$seed + 1$
NPCR (%)	99.61	99.65	99.63	99.63	99.62	99.60	99.58	99.60
UACI (%)	34.20	33.63	32.93	32.39	33.93	34.49	33.11	33.56

4.2.5. Key Space Analysis

The secret key parameters used in our proposed improved scheme includes the three initial state values (x_0, y_0, z_0), four parameters (a, b, c, d), and one integer *seed*. The first seven parameters are all floating-point numbers, and the $seed \in [0, 255]$. Hence, for the working precision of 10^{14} with a floating-point number, our key space is found to be more than $256 \times 10^{14 \times 7} \approx 2^{334}$. Key space in our improved encryption scheme is larger than the key space of 2^{199} in Reference [2], 2^{226} in Reference [15],

10^{45} in Reference [23]. Because the key space is large enough, our improved algorithm can resist brute-force attack.

A general problem concerning the use of chaotic systems in encryption is given by References [25–28] when chaotic systems are implemented on finite precision machines (e.g., computers). The impact of this problem on the proposed encryption scheme is mainly to narrow for the key space. In addition, the randomness of the key sequence is reduced, but this factor has little effect on the security of the encryption scheme.

4.2.6. Computation Efficiency

To demonstrate speed performance in the proposed improved scheme, the encryption time cost by our improved scheme, and the scheme in Reference [15], are measured, respectively, under the same computing environment [15]. The time cost by our improved scheme is 2.325 s, while it takes 2.465 s by the scheme in Reference [15]. It can be seen that our improved scheme has a slightly faster encryption speed than the scheme in Reference [15]. The low time cost in our improved scheme is due to the discarding binary XOR operations.

5. Conclusions

This paper analyzed a chaotic S-box based image encryption algorithm (IESB) in detail. We found that the S-box and the secret key stream **Y** of the system are the equivalent keys of the cryptosystem. The equivalent keys are not related to the image to be encrypted. For the above reasons, the original algorithm (IESB) can not resist chosen-plaintext attacks. We ascertained that the encrypted image can be deciphered with only two chosen plain-images. An ingenious method of constructing explicit chosen-plaintext is found, and an equivalent array of S-box is constructed. We just need the equivalent sequence of S-box without knowing the S-box and the secret key stream **Y** itself to decipher the target ciphertext. The attacking scheme has been proved by theoretical analysis and supported by experimental results. As an optimization method, a new and improved image encryption scheme is developed to conquer these flaws of the original algorithm. In the improved scheme, a feedback mechanism is introduced, a bidirectional diffusion scheme is designed, and values of the ciphertext are associated with more parameters in each diffusion process. Experimental results and security analysis certify that the improved encryption scheme can achieve a higher security level and has a better performance than some recently proposed encryption algorithms.

As for image encryption technology, some future studies is worth considering, such as efficient image encryption technology in resource-constrained mobile social network [29], sensor network communication environment [30]. And searchable encryption [31], which is a very promising direction in the field of cloud computing.

Author Contributions: This paper is the result of collaboration among all the authors in all aspects.

Funding: This research was funded by [the Open Project of Guangxi Colleges and Universities Key Laboratory of Complex System Optimization and Big Data Processing] grant number [No. 2016CSOBDP0103]; [the National Natural Science Foundation of China] grant number [No. 61472451].

Acknowledgments: The authors are thankful to the reviewers for their comments and suggestions to improve the quality of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, Y.; Xiao, D.; Shu, Y.; Li, J. A novel image encryption scheme based on a linear hyperbolic chaotic system of partial differential equations. *Signal Process. Image Commun.* **2013**, *28*, 292–300. [CrossRef]
2. Bashir, Z.; Watrobski, J.; Rashid, T.; Zafar, S.; Salabun, W. Chaotic dynamical state variables selection procedure based image encryption scheme. *Symmetry* **2017**, *9*, 312. [CrossRef]

3. Wang, X.; Liu, C.; Zhang, H. An effective and fast image encryption algorithm based on chaos and interweaving of ranks. *Nonlinear Dyn.* **2016**, *84*, 1595–1607. [CrossRef]
4. Ye, G.; Zhao, H.; Chai, H. Chaotic image encryption algorithm using wave-line permutation and block diffusion. *Nonlinear Dyn.* **2016**, *83*, 2067–2077. [CrossRef]
5. Zhang, Y.; Xiao, D. Double optical image encryption using discrete chirikov standard map and chaos-based fractional random transform. *Opt. Lasers Eng.* **2013**, *51*, 472–480. [CrossRef]
6. Liu, H.; Kadir, A.; Sun, X.; Li, Y. Chaos based adaptive double-image encryption scheme using hash function and s-boxes. *Multimed. Tools Appl.* **2018**, *77*, 1391–1407. [CrossRef]
7. Zhu, C. A novel image encryption scheme based on improved hyperchaotic sequences. *Opt. Commun.* **2012**, *285*, 29–37. [CrossRef]
8. Wang, X.; Wang, Q. A novel image encryption algorithm based on dynamic s-boxes constructed by chaos. *Nonlinear Dyn.* **2014**, *75*, 567–576. [CrossRef]
9. Liu, Y.; Tong, X.; Ma, J. Image encryption algorithm based on hyper-chaotic system and dynamic s-box. *Multimed. Tools Appl.* **2016**, *75*, 7739–7759. [CrossRef]
10. 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.* **2016**, *27*, 677–685. [CrossRef]
11. Cavusoglu, U.; Zengin, A.; Pehlivan, I.; Kacar, S. A novel approach for strong s-box generation algorithm design based on chaotic scaled zhongtang system. *Nonlinear Dyn.* **2017**, *87*, 1081–1094. [CrossRef]
12. Belazi, A.; Khan, M.; Abd El-Latif, A.A.; Belghith, S. Efficient cryptosystem approaches: S-boxes and permutation-substitution-based encryption. *Nonlinear Dyn.* **2017**, *87*, 337–361. [CrossRef]
13. Liu, Y.; Wang, J.; Fan, J.; Gong, L. Image encryption algorithm based on chaotic system and dynamic s-boxes composed of DNA sequences. *Multimed. Tools Appl.* **2016**, *75*, 4363–4382. [CrossRef]
14. Devaraj, P.; Kavitha, C. An image encryption scheme using dynamic s-boxes. *Nonlinear Dyn.* **2016**, *86*, 927–940. [CrossRef]
15. Cavusoglu, U.; Kacar, S.; Pehlivan, I.; Zengin, A. Secure image encryption algorithm design using a novel chaos based s-box. *Chaos Solitons Fractals* **2017**, *95*, 92–101. [CrossRef]
16. Li, C.; Lin, D.; Lu, J. Cryptanalyzing an image-scrambling encryption algorithm of pixel bits. *IEEE Multimed.* **2017**, *24*, 64–71. [CrossRef]
17. Li, C.; Liu, Y.; Xie, T.; Chen, M.Z.Q. Breaking a novel image encryption scheme based on improved hyperchaotic sequences. *Nonlinear Dyn.* **2013**, *73*, 2083–2089. [CrossRef]
18. Yap, W.-S.; Phan, R.C.W.; Yau, W.C.; Heng, S.H. Cryptanalysis of a new image alternate encryption algorithm based on chaotic map. *Nonlinear Dyn.* **2015**, *80*, 1483–1491. [CrossRef]
19. Zhu, C.X.; Sun, K.H. Cryptanalysis and improvement of a class of hyperchaos based image encryption algorithms. *Acta Phys. Sin.* **2012**, *61*, 120503.
20. Zhang, X.; Nie, W.; Ma, Y.; Tian, Q. Cryptanalysis and improvement of an image encryption algorithm based on hyper-chaotic system and dynamic s-box. *Multimed. Tools Appl.* **2017**, *76*, 15641–15659. [CrossRef]
21. Wu, X.; Zhu, B.; Hu, Y.; Ran, Y. A novel color image encryption scheme using rectangular transform-enhanced chaotic tent maps. *IEEE Access* **2017**, *5*, 6429–6436.
22. Zhu, C.; Sun, K. Cryptanalyzing and improving a novel color image encryption algorithm using rt-enhanced chaotic tent maps. *IEEE Access* **2018**, *6*, 18759–18770. [CrossRef]
23. Wang, W.; Si, M.; Pang, Y.; Ran, P.; Wang, H.; Jiang, X.; Liu, Y.; Wu, J.; Wu, W.; Chilamkurti, N.; et al. An encryption algorithm based on combined chaos in body area networks. *Comput. Electr. Eng.* **2018**, *65*, 282–291. [CrossRef]
24. Ahmad, M.; Al Solami, E.; Wang, X.Y.; Doja, M.; Beg, M.; Alzaidi, A. Cryptanalysis of an Image Encryption Algorithm Based on Combined Chaos for a Ban System, and Improved Scheme using SHA-512 and Hyperchaos. *Symmetry* **2018**, *10*, 266. [CrossRef]
25. Li, S.; Chen, G.; Mou, X. On the dynamical degradation of digital piecewise linear chaotic maps. *Int. J. Bifurc. Chaos* **2005**, *15*, 3119–3151. [CrossRef]
26. Curiac, D.I.; Volosencu, C. Chaotic trajectory design for monitoring an arbitrary number of specified locations using points of interest. *Math. Probl. Eng.* **2012**, *2012*, 940276. [CrossRef]
27. Li, S.; Chen, G.; Wong, K.-W.; Mou, X.; Cai, Y. Baptista-type chaotic cryptosystems: Problems and countermeasures. *Phys. Lett. A* **2004**, *332*, 368–375. [CrossRef]

28. Curiac, D.I.; Iercan, D.; Dranga, O.; Dragan, F.; Banias, O. Chaos-Based Cryptography: End of the Road? In Proceedings of the International Conference on Emerging Security Information, System and Technologies, Valencia, Spain, 14–20 October 2007; pp. 71–76.
29. Zhang, S.; Wang, G.; Liu, Q.; Abawajy, J.H. A trajectory privacy-preserving scheme based on query exchange in mobile social networks. *Soft Comput.* **2018**, *22*, 6121–6133. [CrossRef]
30. Bhuiyan, M.Z.A.; Wang, G.; Wu, J.; Cao, J.; Liu, X.; Wang, T. Dependable structural health monitoring using wireless sensor networks. *IEEE Trans. Depend. Secur. Comput.* **2017**, *14*, 363–376. [CrossRef]
31. Zhang, Q.; Liu, Q.; Wang, G. PRMS: A personalized mobile search over encrypted outsourced data. *IEEE Access* **2018**, *6*, 31541–31552. [CrossRef]