*electronics*

MDPI

*Article*

# Adaptive Chaotic Image Encryption Algorithm Based on RNA and Pixel Depth

**Xiaoqiang Zhang [1,2,*] and Xuangang Yan [1,2]**

[1] School of Information and Control Engineering, China University of Mining and Technology, Xuzhou 221116, China; yxgcumt@163.com
[2] Xuzhou Key Laboratory of Artificial Intelligence and Big Data, Xuzhou 221116, China
* Correspondence: zhangxiaoqiang@cumt.edu.cn; Tel.: +86-183-6125-5457

**Abstract:** To prevent the leakage of image content, image encryption technology has received increasing attention. Most current algorithms are only suitable for the images of certain types and cannot update keys in a timely manner. To tackle such problems, we propose an adaptive chaotic image encryption algorithm based on RNA and pixel depth. Firstly, a novel chaotic system, two-dimensional improved Logistic-adjusted-Sine map is designed. Then, we propose a three-dimensional adaptive Arnold transform for scrambling. Secondly, keys are generated by the hash values of the plain image and current time to achieve one-image, one-key, and one-time pad simultaneously. Thirdly, we build a pre-permuted RNA cube for 3D adaptive scrambling by pixel depth, chaotic sequences, and adaptive RNA coding. Finally, selective diffusion combined with pixel depth and RNA operations is performed, in which the RNA operators are determined by the chemical structure and properties of amino acids. Pixel depth is integrated into the whole procedure of parameter generation, scrambling, and diffusion. Experiments and algorithm analyses show that our algorithm has strong security, desirable performance, and a broader scope of application.

**Keywords:** image encryption; RNA; pixel depth; chaotic system; 3D adaptive Arnold transform



**Citation:** Zhang, X.; Yan, X. Adaptive Chaotic Image Encryption Algorithm Based on RNA and Pixel Depth. *Electronics* **2021**, *10*, 1770. https://doi.org/10.3390/electronics10151770

Academic Editor: Kenji Suzuki

Received: 19 June 2021
Accepted: 22 July 2021
Published: 24 July 2021



## 1. Introduction

With the increasing use of instant messaging technology, images are widely used for communication. Meanwhile, some image content contains sensitive information, so image content security becomes an essential issue for scientists and engineers. As a standard and effective technology to protect the content security of digital multimedia information, the image encryption technology plays a significant role in many applications. To better protect the image content, besides the generation of cipher images, the applicability of the algorithm and the real-time update of the key are the crucial problems that researchers must focus on.

After decades of development, various excellent algorithms have emerged. To overcome the shortcomings of traditional algorithms, scholars have introduced chaos theory to image encryption. Classic text encryption algorithms such as Data Encryption Standard (DES), Advanced Encryption Standard (AES), and RSA cannot perform well in image encryption [1–4]. Matthews proposed that the chaotic system can be used in cryptography in 1989 [5]. As a result of its high sensitivity to initial values and parameters, pseudo randomness, ergodicity, complexity, etc., the chaotic system is extensively utilized in image encryption [6]. Recently, a series of image encryption algorithms have been proposed based on chaos [7–13]. Liu et al. proposed a multidimensional chaotic image encryption algorithm based on DNA coding. The traditional three-dimensional (3D) Lorenz system is improved to form a four-dimensional (4D) hyperchaotic Lorenz system for chaotic encryption [14]. By generating three new chaotic signals from two nearby orbits of one-dimensional (1D) chaotic maps, Zhou et al. proposed a simple color image cryptosystem with a very high level of security [15]. To improve the security of image

encryption, researchers pay more attention to the high-dimensional chaotic encryption algorithms [16–20]. Arnold is a classic map usually used to scramble the pixel positions in many image encryption algorithms [21–24]. In 2020, Wang et al. proposed a hyperchaotic image encryption algorithm based on bit-level permutation and DNA encoding, in which the authors employ a six-dimensional hyperchaotic system; the key stream generated by the hyperchaotic system is related to the plain image. DNA encoding and manipulation are used to change the pixel values [25].

In recent years, the efficiency of algorithms has also become one of the focuses of researchers [26–29]. Xian et al. proposed a novel chaotic image encryption algorithm combining chaotic sub-block scrambling based on spiral transformation and chaotic digit selection diffusion, which can improve the diffusion efficiency [30]. Considering that only part of the data in the image is relatively informative, Zhang et al. presented a multiple-image encryption algorithm based on bit planes and chaos. Since the high bit planes contain most of the content in the image, Zhang's algorithm only operates the four high bit planes. Since the high bit planes are one-half of the original image, Zhang's algorithm has excellent efficiency [31]. Liu et al. proposed an image encryption algorithm based on the region of interest. Different from traditional image encryption schemes, Liu's algorithm does not encrypt the whole image. Liu used a histogram of gradient direction, feature extraction, and support vector machine to separate the region of interest from the whole image. Then, the pixels in the region of interest are messed up by using the improved Henon sequence and Joseph sequence. Since the region of interest is only a part of the original image, Liu's encryption scheme also has high efficiency [32]. Motivated by the above discussions, this paper proposes an adaptive chaotic image encryption algorithm based on RNA and pixel depth. We employ the permutation–diffusion framework.

The main contributions of the paper can be highlighted as follows: (1) A novel chaotic system—two-dimensional improved Logistic-adjusted-Sine map (2D-ILASM) is designed on the basis of two-dimensional Logistic-adjusted-Sine map (2D-LASM); (2) We propose a 3D adaptive Arnold transform for scrambling; (3) Keys are generated by the hash values of the plain image and current time to achieve one-time pad; (4) Selective diffusion combined with the pixel depth and RNA operations is performed, in which the RNA operators are determined by the chemical structure and properties of amino acids; (5) We mix the pixel depth of the plain image in the whole encryption procedure to increase the application range of the algorithm; (6) Experimental results and algorithm analyses show that our algorithm has strong security, high efficiency, and broad scope of application.

The rest of this paper is structured as follows. Section 2 introduces the related works. The theoretical principles of the 2D-ILASM chaotic system, Chen-4D chaotic system, RNA coding rules and operations, pixel depth, and the 3D adaptive Arnold transform are described in Section 3. Section 4 provides an adaptive chaotic image encryption algorithm based on RNA and pixel depth. Experimental results are also performed in Section 4. In Section 4, the performance of the proposed algorithm is evaluated through various tests. The discussion is depicted in Section 5. The conclusions are drawn in Section 6.

## 2. Related Works

Zarebnia et al. used the hybrid chaotic system and cyclic shift to ensure their algorithm has better security. Utilizing hybrid or high-dimensional hyperchaotic systems, although the security of image encryption algorithm has been enhanced, the running cost of the algorithm is also increased due to the numerous parameters [33]. Some researchers have realized the importance of updating keys timely. Gan et al. proposed a chaotic image encryption algorithm based on 3D bit-plane permutation. They said the algorithm attains "one plain image, one key" and "one time, one key" by the hash value of the plain image. However, the hash value only changes with the plain image but cannot change with the encryption moment, so the algorithm cannot achieve "one time, one key" [16].

Compared with traditional technologies, DNA and RNA operations have the advantages of parallelism, lower power consumption, and larger storage. Moreover, after

being transcribed from DNA, RNA participates in the translation stage. RNA has the function of controlling protein synthesis through the codon table. This function is equivalent to another coding opportunity and provides more available variables for designing encryption algorithms. Mehdi et al. proposed a new image security technology based on the concept of nucleic acid. The cipher image after DNA operations is used as the input for the RNA stage, thus using the codon truth table and key for RNA for further image encryption [34]. Mahmud et al. used an RNA sequence and genetic algorithm to propose an image encryption algorithm based on an evolutionary RNA codon truth table. They first use Logistic mapping to generate a specified number of initial cipher images. Then, the codon array is updated by using cryptographic keys and cryptographic RNA tables to form the initial population of the genetic algorithm [35]. Abdellatif proposed a novel nucleotide grade color image encryption algorithm, which applied RNA to synthesize amino acids and isolated exon genes from introns [36]. Zhang et al. used the chaotic system to scramble the original RNA truth table randomly and then used the complementary rule to generate four truth tables to replace the RNA bases in the cipher image [37]. Most of the above algorithms are designed for a single image type, which makes them unable to show ideal performance when encrypting different types of images.

## 3. Materials and Methods

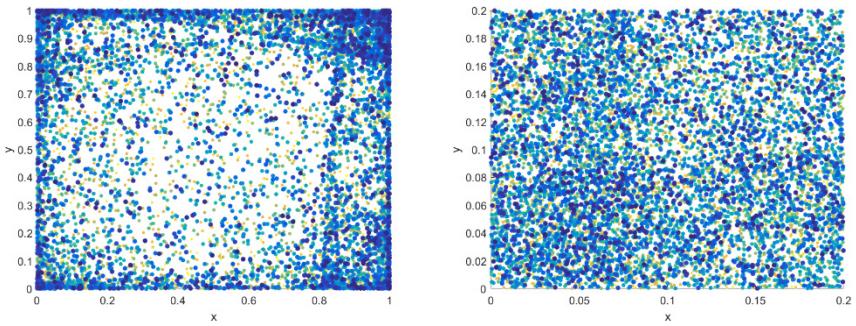
### *3.1. 2D-ILASM Chaotic System*

Recently, Zhou et al. [38] presented a novel chaotic system named 2D-LASM, which is derived from 1D Sine and Logistic maps. Compared to Sine and Logistic maps, 2D-LASM has a more complex structure, wider chaotic range, and better ergodicity. It is defined as

$$\begin{cases} x_{i+1} = \sin(\pi\mu(y_i + 3)x_i(1 - x_i)) \\ y_{i+1} = \sin(\pi\mu(x_{i+1} + 3)y_i(1 - y_i)) \end{cases} \quad (1)$$

where the system parameter $\mu \in [0, 1]$, the state variables $x, y \in (0, 1)$, and the system is chaotic. When $\mu = 0.7$, the Lyapunov exponents are 0.7884 and 0.7495, the system is hyperchaotic.

Figure 1a plots the phase diagram of the chaotic system with the initial values $x_0 = 0.1$, $y_0 = 0.1$, and$\mu = 0.7$. From Figure 1a, it is clear that the system is unable to cover the entire phase plane, so the security of the algorithm cannot be ensured. To further analyze the performance of 2D-LASM, we select 10,000 points from chaotic sequence generated by 2D-LASM. Figure 2a,b show the sequence statistical histograms of 2D-LASM. From Figure 2a,b, we find that the chaotic sequence is distributed unevenly; most of the data are concentrated on both sides, while the middle distribution is less. However, for a valid chaotic system, the numerical distribution of the chaotic sequence should be as close to uniform as possible.




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

**Figure 1.** Phase diagrams of 2D-LASM and 2D-ILASM: (**a**) 2D-LASM; (**b**) 2D-ILASM.

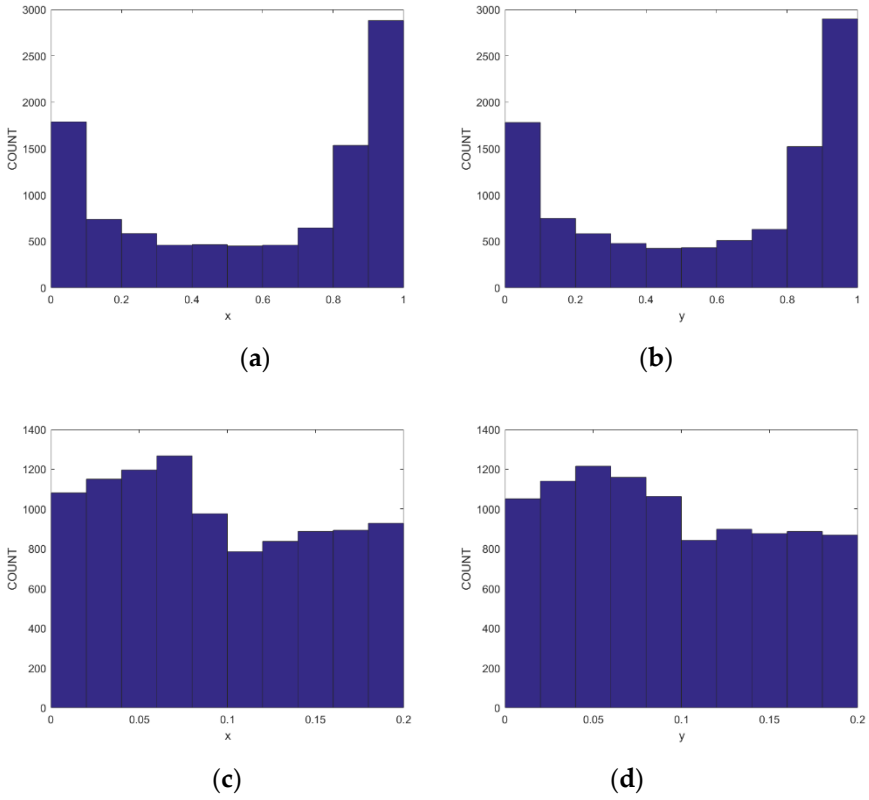



**Figure 2.** Sequence statistical histograms of 2D-LASM and 2D-ILASM: (**a**) Component $x$ of 2D-LASM; (**b**) Component $y$ of 2D-LASM; (**c**) Component $x$ of 2D-ILASM; (**d**) Component $y$ of 2D-ILASM.

The experimental analyses of 2D-LASM shows that it exhibits weak chaotic behaviors in iteration. Therefore, we improve 2D-LASM by introducing the module operations, which can weaken those shortcomings and perform a better chaotic performance. The 2D-ILASM is expressed via Equation (2) as follows:

$$\begin{cases} x_{i+1} = \mathrm{mod}(\sin(\pi\mu(y_i + 3)x_i(1 - x_i)), 0.2) \\ y_{i+1} = \mathrm{mod}(\sin(\pi\mu(x_{i+1} + 3)y_i(1 - y_i)), 0.2) \end{cases} \quad (2)$$

where the system parameter $\mu \in [0, 1]$ and the state variables $x, y \in (0,\ 0.2)$. The phase diagram of 2D-ILASM is shown in Figure 1b, which is distributed on the entire phase plane. The sequence statistical histograms of 2D-ILASM are shown in Figure 2c,d. These figures indicate that chaotic sequences scatter evenly in different intervals.

The ideal pseudorandom sequence used in cryptosystems should possess fine statistical properties [39]. To analyze the output performance of the 2D-ILASM, the test standard of NIST SP800-22 is introduced [40,41]. The NIST SP800-22 is a statistical package consisting of 15 test items that are used to verify the randomness of the produced sequences. For each test, the $p$-value is expected to fall into [0.01, 1] to accept the sequences as random. If a $p$-value for a test is determined to be equal to 1, then the sequence appears to have perfect randomness. As shown in Table 1, the tested sequences may pass all the test items. Therefore, the outputs of the 2D-ILASM can be considered randomness and used as key streams for image encryption. According to the above analyses, 2D-ILASM is suitable for generating chaotic sequences with excellent properties. In our algorithm, the proposed chaotic system is used to produce highly random chaotic sequences.

**Table 1.** NIST SP800-22 test.

| Test Items | *p*-Value | | Results |
|---|---|---|---|
| | X | Y | |
| Frequency test | 0.898977 | 0.596600 | Pass |
| Frequency test within a block | 0.758731 | 0.291061 | Pass |
| Runs test | 0.632272 | 0.963952 | Pass |
| Test for longest run of ones in a block | 0.931298 | 0.408512 | Pass |
| Binary matrix rank test | 0.424943 | 0.434479 | Pass |
| Discrete Fourier transform test | 0.657982 | 0.349486 | Pass |
| Non-overlapping template matching test | 0.342668 | 0.962718 | Pass |
| Overlapping template matching test | 0.144245 | 0.174696 | Pass |
| Maurer's "Universal Statistical" test | 0.138558 | 0.148293 | Pass |
| Linear complexity test | 0.876999 | 0.988846 | Pass |
| Serial test * | 0.989594 | 0.263659 | Pass |
| Approximate entropy test | 0.825184 | 0.091003 | Pass |
| Cumulative sums test * | 0.976507 | 0.439422 | Pass |
| Random excursions test * | 0.641919 | 0.553448 | Pass |
| Random excursions variant test * | 0.757583 | 0.467278 | Pass |

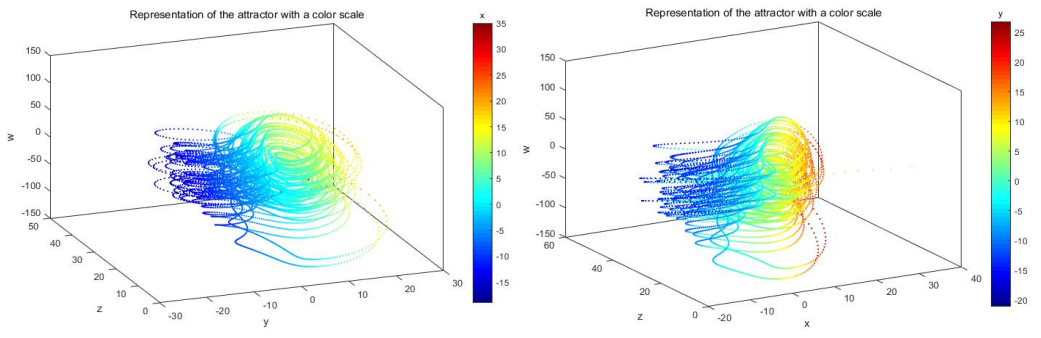
* The average values of multiple tests.

### 3.2. Chen-4D Hyperchaotic System

Based on the 3D Chen chaos, a 4D Chen hyperchaotic system has been reconstructed [42]. The 4D Chen hyperchaotic system has high dynamics complexity, a large key space, and more resistance to exhaustive attacks. The specific equation is as follows:

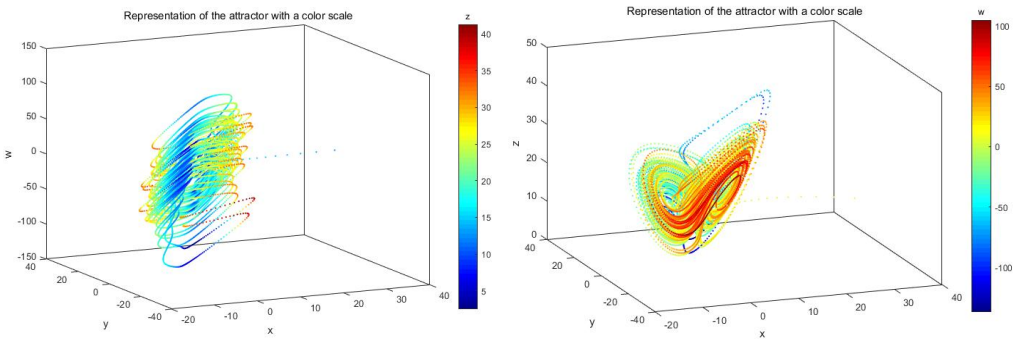
$$\begin{cases} \dot{z} = a(u - z) + w \\ \dot{u} = bz - zv + cu \\ \dot{v} = zu - dv \\ \dot{w} = uv + ew \end{cases} \tag{3}$$

where $a$, $b$, $c$, $d$, and $e$ are parameters for controlling chaotic coefficients, and, $z$, $u$, $v$, and $w$ are state variables of the chaotic system. When $a = 35$, $b = 7$, $c = 12$, and $e \in (0.085, 0.798]$, the conditions of the hyperchaotic system are met, and the system is in a hyperchaotic state. The phase diagrams of the Chen hyperchaotic system are depicted in Figure 3.




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

**Figure 3.** *Cont.*




(**c**) (**d**)

**Figure 3.** Phase diagrams of the Chen hyperchaotic attractor: (**a**) Phase diagram of the y−z−w plane; (**b**) Phase diagram of the x−z−w plane; (**c**) Phase diagram of the x−y−w plane; (**d**) Phase diagram of the x−y−z plane.

### *3.3. RNA Encoding Rules and Operations*

DNA and RNA are the core materials in the process of biological genetics. A DNA sequence is composed of four bases, namely A (adenine), C (cytosine), G (guanine), T (thymine), which are arranged in a certain order. An RNA sequence is also built of four nucleic acid bases. There is only one different base; that is, U (uracil) in RNA replaces T in DNA [43]. In the past several years, DNA coding technology is widely used in image encryption for its excellent characteristics in computing. In RNA coding, two binary numbers are mapped to one base under the designed coding rule. In biology, single-stranded RNA synthesizes its complementary strand according to the principle of complementary base pairing and then forms double-stranded RNA [43]. Based on the above discussion, we adopt RNA coding and operation in this paper, A and U are complementary, and C and G are complementary. Following the RNA complementation rules, we can only get eight coding rules shown in Table 2.

**Table 2.** Binary coding rules for RNA sequences.

| Rule | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 00 | A | A | U | U | C | C | G | G |
| 01 | C | G | C | G | A | U | A | U |
| 10 | G | C | G | C | U | A | U | A |
| 11 | U | U | A | A | G | G | C | C |

Analogous to DNA operations and inspired by the RNA complementary pairing rules, there are six operators in RNA operations: addition, subtraction, add-complement, sub-complement, XOR (exclusive OR), and XNOR (exclusive NOR). In RNA calculation, the proposed eight coding rules and six operations make 48 different choices for every operation between two bases. Tables 3–8 show the addition operation, the subtraction operation, the add-complement operation, the sub-complement operation, the XOR operation, and the XNOR operation, respectively, which are designed under the coding rule 1.

**Table 3.** RNA addition rule.

| + | A | G | C | U |
|---|---|---|---|---|
| A | A | G | C | U |
| G | G | C | U | A |
| C | C | U | A | G |
| U | U | A | G | C |

**Table 4.** RNA subtraction rule.

| − | **A** | **G** | **C** | **U** |
|---|---|---|---|---|
| A | A | U | C | G |
| G | G | A | U | C |
| C | C | G | A | U |
| U | U | C | G | A |

**Table 5.** RNA add-complement rule.

| **+′** | **A** | **G** | **C** | **U** |
|---|---|---|---|---|
| A | U | C | G | A |
| G | C | G | A | U |
| C | G | A | U | C |
| U | A | U | C | G |

**Table 6.** RNA sub-complement rule.

| **−′** | **A** | **G** | **C** | **U** |
|---|---|---|---|---|
| A | U | A | G | C |
| G | C | U | A | G |
| C | G | C | U | A |
| U | A | G | C | U |

**Table 7.** RNA XOR rule.

| **XOR⊕** | **A** | **G** | **C** | **U** |
|---|---|---|---|---|
| A | A | G | C | U |
| G | G | A | U | C |
| C | C | U | A | G |
| U | U | C | G | A |

**Table 8.** RNA XNOR rule.

| **XNOR⊙** | **A** | **G** | **C** | **U** |
|---|---|---|---|---|
| A | U | C | G | A |
| G | C | U | A | G |
| C | G | A | U | C |
| U | A | U | C | U |

The three adjacent bases on the messenger RNA chain that determine an amino acid during protein synthesis are called codons. RNA is composed of four bases, and every three bases are combined into a codon. In theory, there are $4 \times 4 \times 4 = 64$ kinds of base combinations, that is, 64 kinds of codons. Then, m-RNA can be combined with t-RNA, amino acids can be combined with different polarities, and chemical structures can be formed, as shown in Figure 4.

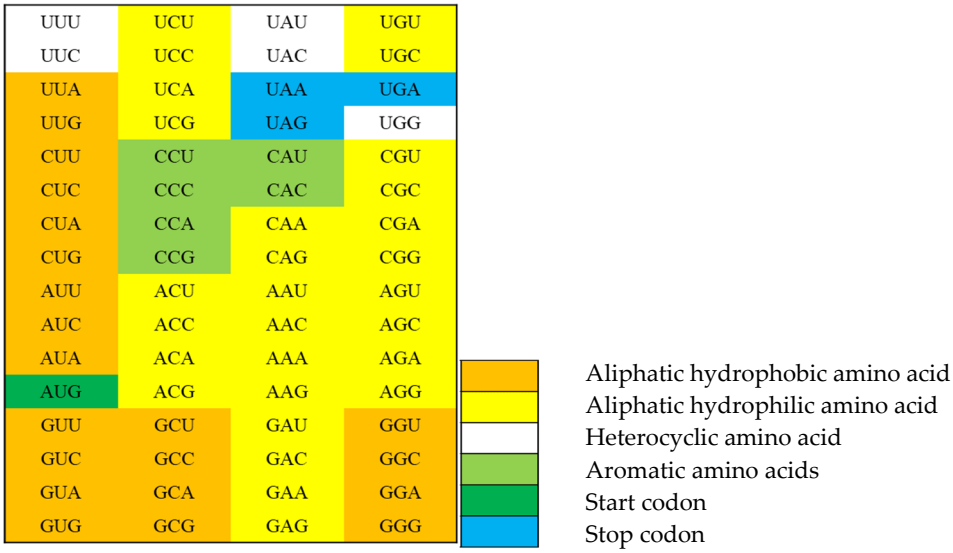



**Figure 4.** Amino acid codon table in RNA.

The difference in the structure of amino acids depends on the difference of the side chain groups. Wang et al. [43] divided amino acid types according to the polarity of amino acid side chain groups to build the RNA operator controller. In the proposed algorithm, amino acids are classified according to the chemical structure and properties of the side chain groups. By the correspondence between the first three bits of the four vertices of the RNA scrambled image and the codons, the operators used in the selective diffusion are selected. Due to the different number of codons for each amino acid, to make the RNA operators distributed in the calculation process more evenly, we consider dividing amino acids into six types. Each type represents a kind of operator. The specific correspondence is shown in Table 9.

**Table 9.** Correspondence between amino acid category and operator.

| Category of Amino Acids | Operator |
|---|---|
| Aliphatic hydrophobic amino acid | $\oplus$ |
| Aliphatic hydrophilic amino acid | $\odot$ |
| Aromatic amino acids | $+$ |
| Stop codon | $-$ |
| Heterocyclic amino acid | $+'$ |
| Start codon | $-'$ |

### *3.4. Pixel Depth*

Pixel depth refers to the number of bits used to store each pixel, which are measured in BPP (Bit Per Pixel), and it is also used to measure the resolution of an image. Pixel depth determines the number of colors each pixel of a color image may have or determines the number of gray levels each pixel of a grayscale image may have. The more the number of bits representing the pixel value in the image, the more color types it can express, the deeper its pixel depth, and the larger its BPP value.

When the BPP of an image is 8 or 16, the maximum number of colors that a single pixel can express is $2^8 = 256$ or $2^{16} = 65{,}536$, and the range of colors displayed is limited, which is called grayscale or high color image. When the BPP of an image is 24, the maximum number of colors that a single pixel can express is $2^{24} = 16{,}777{,}216$. Use 24 bits to display a pixel composed of 8 bits that are red, 8 bits that are green, and 8 bits that are blue. Each color can be displayed completely, so images with 24 bits and above are called true-color images. When the BPP of an image is 32, $2^{32} = 2^{24} + 2^8$, the common 32-bit color in the computer field does not represent $2^{32}$ colors, but an 8-bit ($2^8 = 256$ level) transparency is

added to the 24-bit color. Therefore, the total number of 32-bit colors is the same as 24-bit colors, and 32-bit colors are called a full-color image.

Pixel depth is one of the properties of the image itself, just like its size. Integrating pixel depth into the encryption process can increase the security and scope of the algorithm. In the proposed algorithm, the pixel depth of the plain image participates in the whole process of parameter generation, scrambling, and diffusion to enhance applicability and security. Lena images with different pixel depths are shown in Figure 5.

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

**Figure 5.** Lena images with different pixel depths: (**a**) 8 BPP (grayscale); (**b**) 16 BPP (grayscale); (**c**) 24 BPP (color); (**d**) 32 BPP (color).

### *3.5. 3D Adaptive Arnold Transform*

Arnold transform (AT) is a 2D map [44] as shown in Equation (4).

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \mathrm{mod} N \tag{4}$$

where $N$ is the order of the matrix, $x, y \in \{1, 2, \cdots, N\}$ are the positions of elements before applying AT, and $x', y' \in \{1, 2, \cdots, N\}$ are the positions of elements after applying AT.

The first three iterations of the 2D-AT in the cycle, shown in Figure 6a–c, failed to hide the content of the plain image ideally. This situation can be changed after three iterations. For the above reasons, some image scrambling methods based on 2D-AT usually require multiple operations to achieve better performance, so the scrambling efficiency may be low. Due to its 2D characteristics, 2D-AT is mostly applied to grayscale images. When applied to color images, 2D-AT is mainly used on the RGB components separately—the image is not treated as a whole, so the scrambling results are not satisfactory.

To overcome the shortcomings of 2D-AT and obtain better scrambling performance, in this paper, we build the 3D adaptive Arnold transform (3D-AAT) by adding two adaptive parameters $q_1$ and $q_2$ in 3D-AT. The 3D-AAT is given by

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & q_2 + 1 & 2q_2 \\ 1 & q_1 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \mathrm{mod} N \tag{5}$$

where $N$ is the order of the matrix, $x, y, z \in \{1, 2, \cdots, N\}$ are the position of elements before applying 3D-AAT, $q_1$ and $q_2$ are two adaptive parameters calculated by Equation (10), and $x', y', z' \in \{1, 2, \cdots, N\}$ are the position of elements after applying 3D-AAT.

According to the experimental results shown in Figure 6d,f, we can see that images with different pixel depths all perform satisfactory results by only one round 3D-AAT. Therefore, 3D-AAT has an ideal scrambling effect and efficiency and is suitable for scrambling images with various pixel depths.

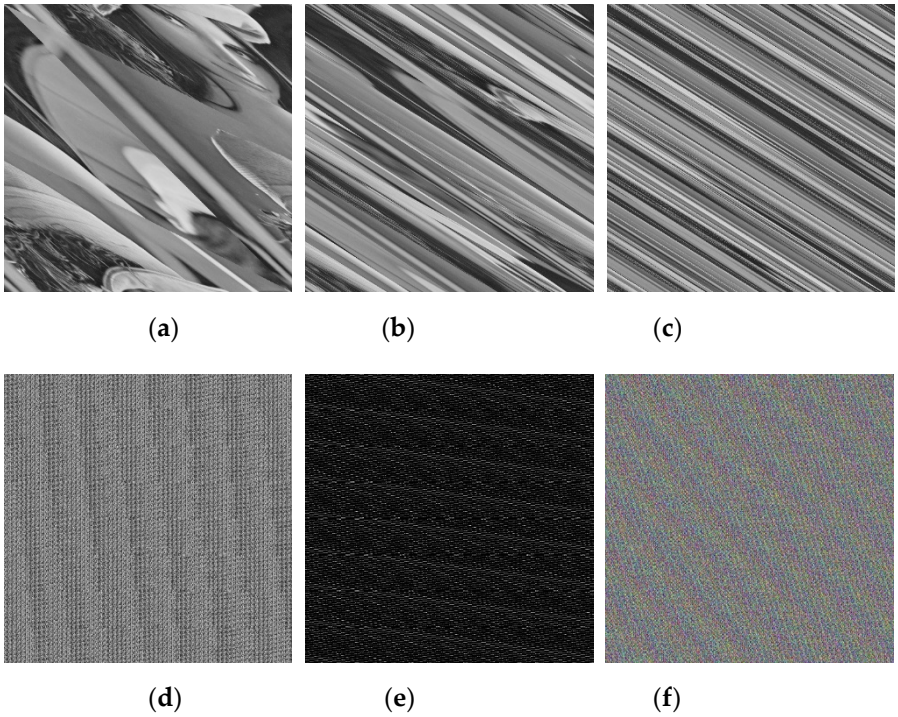


**Figure 6.** Results of 2D-AT and 3D-AAT: (**a**) One round 2D-AT on 8 BPP Lena; (**b**) Two rounds 2D-AT on 8 BPP Lena; (**c**) Three rounds 2D-AT on 8 BPP Lena; (**d**) One round 3D-AAT on 8 BPP Lena; (**e**) One round 3D-AAT on 16 BPP Lena; (**f**) One round 3D-AAT on 24 BPP Lena.

AT is a periodic map, and the period depends on the size of the image. For example, an RNA cube is obtained from a $512 \times 512$ plain image with different pixel depths, and the corresponding periods of 2D-AT and 3D-AAT are shown in Table 10. We can see from Table 10 that for the same order $N$, the period of 3D-AAT is much larger than of 2D-AT. Assuming AT's period is $T$, if an image is scrambled by AT for $t$ times, then the image can be recovered by applying $T - t$ times Arnold transform again. In practical engineering applications, the image order is generally large, and it is inefficient to use the transformation period to restore the image. Therefore, this paper uses the corresponding inverse transform to perform the corresponding operation. The 3D inverse adaptive Arnold transform (3D-IAAT) is shown as Equation (6).

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (3-2q_1)q_2+3 & 2q_2-3 & q_1-q_2-1 \\ q_1-3 & 2 & 1-q_1 \\ q_2-1 & 1-2q_2 & q_2 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \bmod N \quad (6)$$

**Table 10.** Periods of 2D-AT and 3D-AT.

| BPP | N | Period of 2D-AT | Period of 3D-AAT |
|---|---|---|---|
| 8 | 102 | 36 | 27,937 |
| 16 | 128 | 96 | 224 |
| 24 | 146 | 222 | 12,607 |
| 32 | 162 | 108 | 2457 |

## 4. Results

### *4.1. Proposed Algorithm*

#### 4.1.1. Key Generation

Secure Hash Algorithms (SHA) are a kind of hash functions released by the National Institute of Standards and Technology (NIST), which is mainly used in the integrity security

services [45]. SHA-256 is a commonly used one with an output digest length of 256 bit. To prevent hackers from inferring the original password through the rainbow table, when calculating the hash, they cannot calculate the original input only. It is necessary to add salt to make the same input get different hash values, increasing the hacking difficulty greatly. In this paper, to strengthen the plaintext sensitivity and security of the algorithm, we combine the SHA-256 hash value of the plain image and the current computer time with the external parameters to generate keys jointly.

By combining the hash value of the current computer time and the plain image, one-image, one-key, and one-time pad can be realized simultaneously. Each parameter in the proposed algorithm changes with time. Even if the key is leaked, it is secure to transmit the cipher image of the same plain image next time.

For the plain image with pixel depth $d$ and the computer's current system time, the SHA-256 is used to get two hash codes $H_1$ and $H_2$ with 256 bits. $H_1$ and $H_2$ respectively take the first 128 bits and then integrate them into a 256-bit hash code $H$ , which is divided into 8-bit blocks, i.e.,

$$\begin{cases} H_1 = j_1, j_2, \cdots, j_{256} \\ H_2 = k_1, k_2, \cdots, k_{256} \\ H = j_1, j_2, \cdots, j_{128}, k_1, k_2, \cdots, k_{128} \\ H = h_1, h_2, \cdots, h_{32} \end{cases} \tag{7}$$

where $h_i, i = 1, 2, \cdots 32$ are blocks with 8-bit length. The required key stream is calculated by Equations (8)–(10), where $\{x'_0, y'_0, z'_0, u'_0, v'_0, w'_0, \mu', e', t'\}$ are the external parameters input by the user. $\{x_0, y_0, \mu\}$ are the initial values and control parameter of a 2D-ILASM hyperchaotic system, $\{z_0, u_0, v_0, w_0, e\}$ are the initial values and control parameter of a 4D-Chen hyperchaotic system respectively, and $\{t, q_1, q_2\}$ are the iterate number and control parameters of 3D-AAT.

$$\begin{cases} x_0 = \frac{bin2dec(h_1 \odot h_2 \odot h_3 \odot x'_0)}{255} \\ y_0 = \frac{bin2dec(h_4 \odot h_5 \odot h_6 \odot y'_0)}{255} \\ \mu = \frac{bin2dec(h_7 \odot h_8 \odot h_9 \odot \mu')}{255} \end{cases} \tag{8}$$

$$\begin{cases} z_0 = \mathrm{mod}(z'_0 + \log(h_{17}h_{18}h_{19}h_{20})/10, m) \\ u_0 = \mathrm{mod}(u'_0 + \log(h_{21}h_{22}h_{23}h_{24})/10, m) \\ v_0 = \mathrm{mod}(v'_0 + \log(h_{25}h_{26}h_{27}h_{28})/10, m) \\ w_0 = \mathrm{mod}(w'_0 + \log(h_{29}h_{30}h_{31}h_{32})/10, m) \\ e = \frac{bin2dec(h_9 \odot h_{10} \odot \cdots h_{16} \odot e') + 31}{360} \end{cases} \tag{9}$$

$$\begin{cases} t = floor(bin2dec(h_4 \odot h_5 \odot h_6 \odot t')) \\ q_1 = \mathrm{mod}(floor(d + x_0 \times 10^{14}), N) \\ q_2 = \mathrm{mod}(floor(d + y_0 \times 10^{14}), N) \end{cases} \tag{10}$$

where $\mathrm{mod}(\cdot)$ denotes the modulus after division, $bin2dec(\cdot)$ denotes binary to decimal, $floor(\cdot)$ is the rounded-down function, and $\odot$ denotes the XNOR operation in the binary system.

#### 4.1.2. Encryption Process

Supposing that Alice is the sender and Bob is the recipient. The encryption flowchart of the proposed algorithm is shown in Figure 7. The encryption process is described in detail as follows.

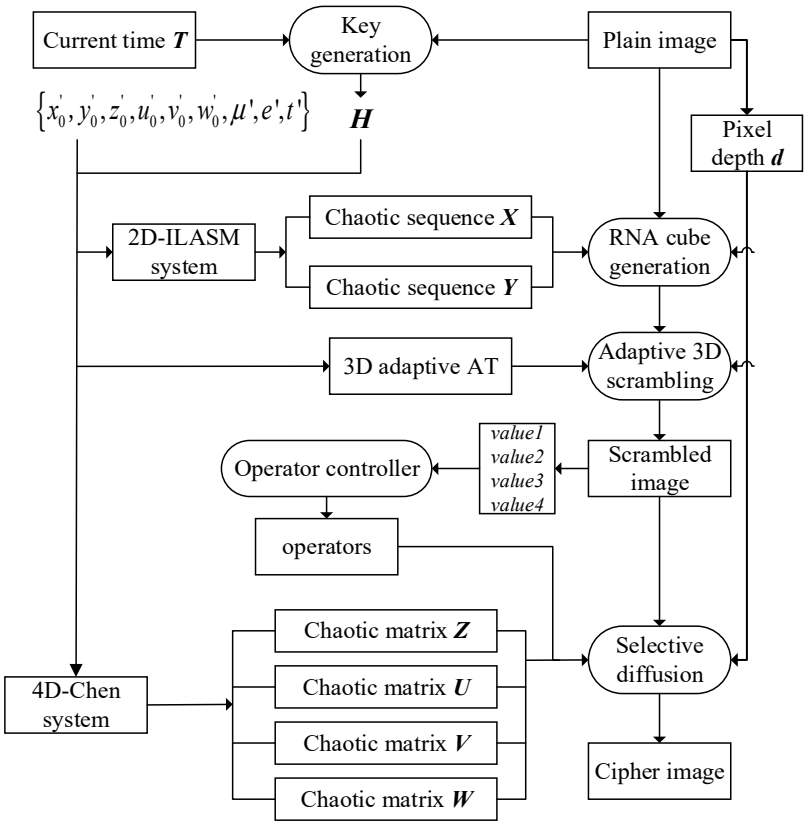



**Figure 7.** The encryption flowchart of the proposed algorithm.

Step 1: Key generation

Alice reads the current computer time and the plain image $I$ with size $m \times n$ and pixel depth $d$. The SHA-256 is used to generate the 256-bit hash value $H$ by Equation (7). Alice randomly selects external parameters $\{x_0', y_0', z_0', u_0', v_0', w_0', \mu', e', t'\}$ as the user's input keys. The parameters $\{x_0, y_0, \mu, z_0, u_0, v_0, w_0, e, t, q_1, q_2,\}$ are generated by Equations (8)–(10).

Step 2: Chaotic sequence generation

Alice iterates the 2D-ILASM system $1000 + m \times n \times d/2$ times with the initial values $x_0, y_0$ and control parameter $\mu$, discarding the first 1000 values to obtain the excellent randomness. Two chaotic sequences $X, Y$ are produced. Then, the 4D-Chen system iterates $1000 + mn$ times with the initial values $z_0, u_0, v_0, w_0$ and control parameter $e$ to generate four chaotic matrices $Z, U, V, W$ with the equal size of $m \times n$.

Step 3: Encoding rule generation

Calculate

$$r_1 = \mathrm{mod}(d + floor(x_0 \times 10^{14}), 8) + 1 \tag{11}$$

$$r_2 = \mathrm{mod}(d + floor(y_0 \times 10^{14}), 8) + 1 \tag{12}$$

$$r_3 = \mathrm{mod}(d + floor(z_0 \times 10^{14}), 8) + 1 \tag{13}$$

$$r_4 = \mathrm{mod}(d + floor(u_0 \times 10^{14}), 8) + 1 \tag{14}$$

$$r_5 = \mathrm{mod}(d + floor(v_0 \times 10^{14}), 8) + 1 \tag{15}$$

$$r_6 = \mathrm{mod}(d + floor(w_0 \times 10^{14}), 8) + 1 \tag{16}$$

where $\mathrm{mod}(\cdot)$ denotes the modulus after division and $floor(\cdot)$ is the rounded-down function. $\{r_1, r_2, r_3, r_4, r_5, r_6,\}$ are the encoding and decoding rules that need to be used in subsequent operations.

Step 4: RNA cube generation

Alice uses the RNA coding rule $r_1$ to encode the plain image $I$, and a 3D RNA matrix $I_{3D}$ of size $m \times n \times d/2$ is obtained. Using one RNA code as the layer thickness and

stratifying $I_{3D}$, we can get RNA layered images $I_j^R$, $j = 1, 2, \cdots, d/2$ with the same size $m \times n$ and corresponding vectors are $V_j^1$, $j = 1, 2, \cdots, d/2$, respectively. After that, Alice connects $V_j^1$ into a long vector $V^1$ by Equation (20) and calculates

$$[X', p_1] = sort(X(j)),\ j = 1, 2, \cdots, d/2 \tag{17}$$

$$I_j^1 = I_{p_1(j)}^R,\ j = 1, 2, \cdots, d/2 \tag{18}$$

$$[Y', p_2] = sort(Y(k)),\ k = 1, 2, \cdots, m \times n \times d/2 \tag{19}$$

$$V^1 = [V_1^1\ V_2^1\ \cdots\ V_j^1] \tag{20}$$

$$V^2(k) = V^1(p_2(k)),\ k = 1, 2, \cdots, m \times n \times d/2 \tag{21}$$

$$l = ceil(\sqrt[3]{m \times n \times d/2}) \tag{22}$$

where $sort(\cdot)$ is the sorting function, $ceil(\cdot)$ is the function that rounds toward positive infinity, $X', Y'$ are the sorting sequences, and $P_1, P_2$ are the indexes recording the elements arrangement in $X', Y'$. The scrambling result $V^2$ of $V^1$ is obtained by Equation (21). Finally, she upscales the dimension of $V^2$ to produce the scrambled RNA cube $I_{3D}^2$ with size $l \times l \times l$ after zero padding. This operation aims to initially scramble the plain image and generate a cube for the next 3D-AAT. The schematic diagram of RNA cube generation is shown in Figure 8.

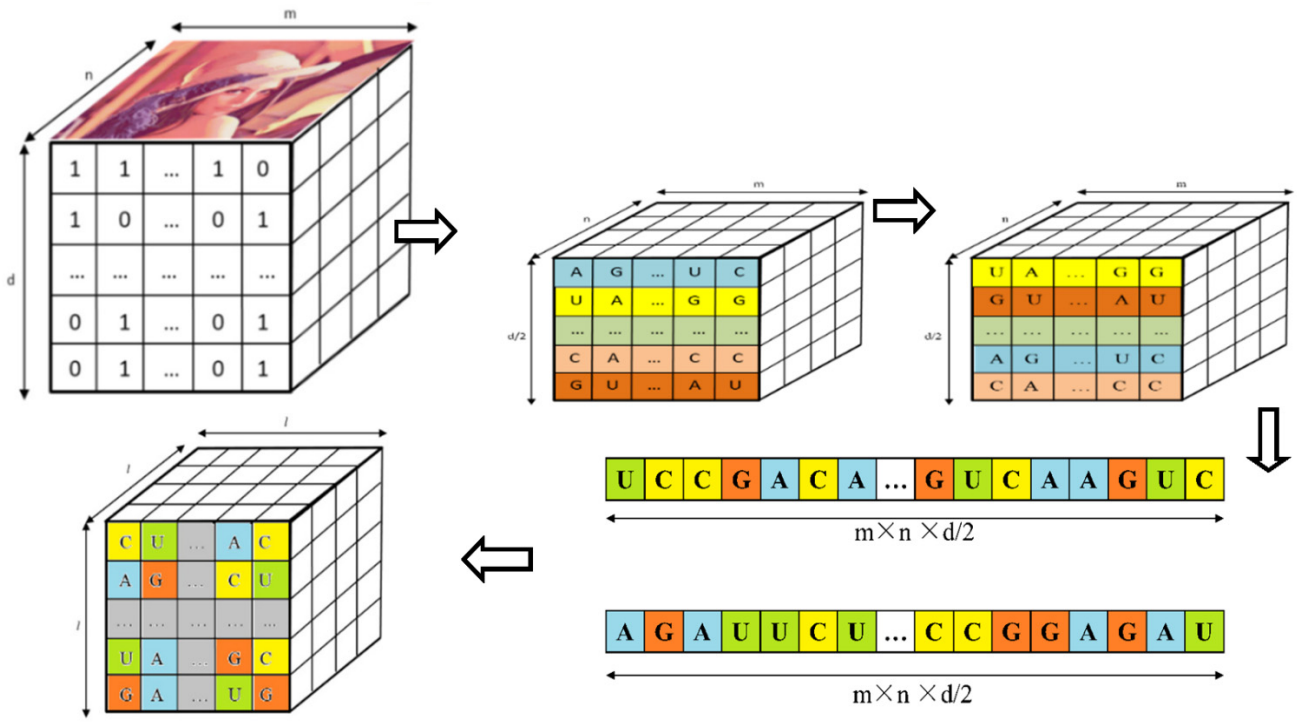



**Figure 8.** The schematic diagram of RNA cube generation.

Step 5: Adaptive 3D scrambling operation

Alice utilizes $t$ times 3D-AAT into RNA cube $I_{3D}^2$ obtained in Step 4. Let $I_{3D}^3$ be the corresponding result. After converting $I_{3D}^3$ into a 1D RNA sequence $V^3$ and deleting the previously added zero codes, Alice uses rule $r_2$ to decode the code-deleted 1D RNA sequence $V^4$. Then, she performs a dimension upscaling operation on $V^4$ to produce the scrambled image $I^4$ with size $m \times n$. At this point, the scrambling stage is completed. The parameters of 3D-AT are also used as keys.

Step 6: Operator controller generation

Six RNA operation rules: addition, subtraction, add-complement, sub-complement, XOR, and XNOR are defined in Section 3.3. Alice applied the RNA theory in biology to construct the RNA amino acid sequence generation table, as shown in Figure 4. Moreover, the corresponding relationship between the RNA amino acid sequence generation table and the RNA operator is designed, i.e., the operator controller, as shown in Table 9.

Step 7: Operator selection

The four vertices of the scrambled image $I^4$ are chosen respectively, the first six bits of their pixel values are used as control parameters, and the RNA coding rule $r_3$ is exploited to encode them. Utilizing the distribution of the parameter's RNA values in the amino acid sequence generation table shown in Figure 4, Alice checks the operator controller Table 9 to select four out of six operators, namely *operator* 1, *operator* 2, *operator* 3, and *operator* 4. The selected operators participate at the subsequent diffusion stage.

Step 8: Selective diffusion

Alice segments scrambled image $I^4$ every 8 bits into images $I_i^4$, $i = 1, 2, \cdots, d/8$ and then uses RNA coding rule $r_4$ on $I_i^4$ to obtain $I_i^5$. $Z^1, U^1, V^1, W^1$ are the results of encoding the chaotic matrices $Z, U, V, W$ by rule $r_5$. She selects operators under the corresponding relationship in Step 7. According to the operation rule in Step 8, the chaotic RNA matrices $Z^1, U^1, V^1, W^1$ and the scrambled RNA image $I_i^5$ are diffused as follows:

$$\begin{cases} Z^1\ (operator1)\ U^1\ (operator2)\ I_1^5\ (operator3)\ V^1 = I_1^e \\ U^1\ (operator2)\ I_2^5\ (operator3)\ I_1^e = I_2^e \\ V^1\ (operator3)\ I_3^5\ (operator4)\ I_2^e = I_3^e \\ W^1\ (operator4)\ I_4^5\ (operator1)\ I_3^e = I_4^e \end{cases} \tag{23}$$

where $I_i^e$, $i = 1, 2, \cdots, d/8$ are the diffused RNA images. Then, $I_i^e$ are merged into the cipher RNA image $I^e$. Finally, Alice decodes $I^e$ by rule $r_6$ into the cipher image $I^c$.

#### 4.1.3. Decryption Process

The decryption is the inverse procedure of encryption. When the SHA-256 value $H$, the pixel values of the four vertices of the scrambled image, and external parameters $\{x_0', y_0', z_0', u_0', v_0', w_0', \mu', e', t'\}$ are sent by Alice, Bob can decrypt the cipher image $I^c$ with pixel depth $d$. The decryption flowchart is shown in Figure 9.

### *4.2. Experiments*

Plain images of size 512 × 512, including grayscale image Lena with pixel depth 8BPP, grayscale image Lena with pixel depth 16BPP, and color image Baboon with pixel depth 24BPP as shown in Figure 10, are tested by the proposed encryption algorithm. The computer configuration used in the experiments is shown as follows: Intel(R) core (TM) i5-8265U CPU, 1.80 GHz processor, 8 GB RAM. We utilize standard test images as the experimental images. At 15:31 on 10 December 2020, we used 8BPP Lena as an example, whose corresponding hash value $H$ is listed as $H$= 9390454675d1f5e55b046d30f05a8ea1532c6b269a 303ac19d3d0a906bb5e250. The external parameters are shown in Table 11. The corresponding cipher images are shown in Figure 11. It can be seen from the results that the plain images are encrypted into noise-like images by the proposed algorithm. Experimental results illustrate that the cipher images appear to be noisy so that people can hardly get any meaningful information visually. Therefore, the proposed algorithm has an excellent encryption effect. The decrypted images are identical to the plain images in Figure 10, respectively.

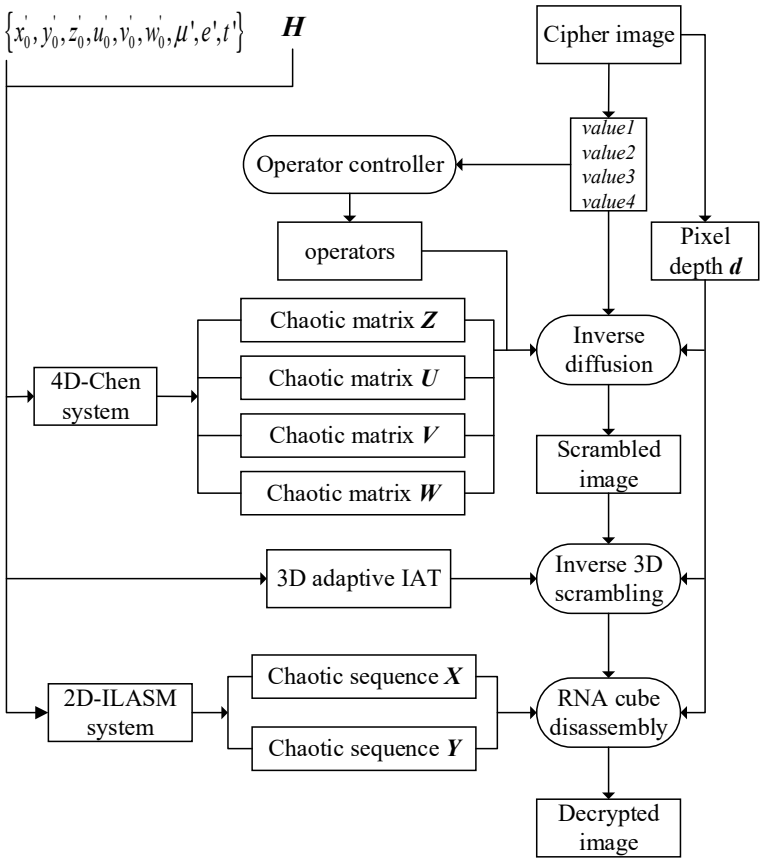


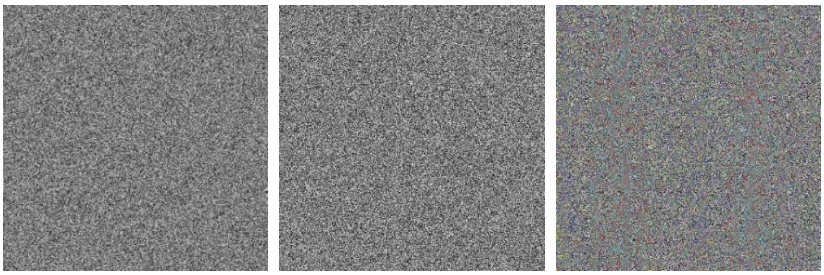

**Figure 9.** The decryption flowchart of the proposed algorithm.

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

**Figure 10.** Plain images: (**a**) 8BPP Lena; (**b**) 16BPP Lena; (**c**) 24BPP Baboon.

**Table 11.** The external parameters.

| Component | Values |
|---|---|
| External parameters | $x_0' = 0.9865$, $y_0' = 1.4335$, $z_0' = 1.4977$, $u_0' = 0.5501$, $v_0' = 2.5159$, $w_0' = 1.3714$, $\mu' = 1.6686$, $e' = 0.2759$, $t' = 2.5568$ |



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

**Figure 11.** Cipher images: (**a**) 8BPP Lena; (**b**) 16BPP Lena; (**c**) 24BPP Baboon.

### *4.3. Algorithm Analyses*

An excellent image encryption algorithm can resist several commonly used attacks, such as the brute-force attack and differential attack. This section gives detailed figures, tables, and descriptions to measure the performance of the proposed algorithm. The decryption results with the wrong key, 3D histograms of plain images and cipher images, adjacent pixel correlations of cipher images, and resistance to cropping and noise attacks are presented in figures. The key space, chi-squared test results, correlations between plain images and cipher images, information entropy, time complexity, and resistance to differential attacks are presented in numbers and tables as follows. We also compare it with some similar algorithms.

The average encryption time of the 512 × 512 grayscale images was 0.9274 s. The comparison results of the 512 × 512 8BPP Lena are shown in Table 12.

**Table 12.** Encryption time and comparisons (512 × 512 8BPP Lena).

| Algorithms | Time (Unit: Seconds) |
|---|---|
| Proposed algorithm | 0.9274 |
| Ref. [46] | 0.6840 |
| Ref. [47] | 2.7113 |
| Ref. [48] | 16.2561 |

#### 4.3.1. Key Space Analysis

The key space is equivalent to the number of all available keys in an algorithm. For an excellent encryption algorithm, its key space should be so large that it cannot be cracked by the brute-force attack. The required keys are composed of 256-bit hash value $H$ and external parameters $\{x'_0, y'_0, z'_0, u'_0, v'_0, w'_0, \mu', e', t'\}$. If the calculation accuracy is $10^{-14}$, then the key space for the proposed algorithm is about $10^{14\times 9} \times 2^{256} \approx 1.1579 \times 10^{203} \approx 2^{674}$, which is greatly larger than the required value $2^{100}$ in the cryptosystem.

$$years = \frac{Key\ Combinations \times 1000}{FLOPS} \times 31536000 \tag{24}$$

Considering the actual technology, the fastest supercomputer today (Summit) is capable of 200 PFLOPS ($10^{15}$ floating-point operation per second) or 200,000 trillion calculations per second. According to Equation (24) [45], even with five times the most powerful computing power at present, it will take $1.2359 \times 10^{196}$ years to crack the encryption system with the key space of $2^{674}$. Therefore, the key space of the proposed algorithm is huge enough to resist the brute-force attack.

#### 4.3.2. Key Sensitivity Analysis

An effective and robust encryption algorithm should be sensitive enough to even the slightest changes in its keys. The incorrect outcome will be produced when we use the key after any slightest changes to decrypt the cipher image. By subtracting the error images,

the attacker cannot extract any clues related to the cipher image. As shown in Figure 12, (a) is Lena's cipher image encrypted with the keys ($x'_0$ = 0.9865, $y'_0$ = 1.4335, $z'_0$ = 1.4977, $u'_0$ = 0.5501, $v'_0$ = 2.5159, $w'_0$ = 1.3714, $\mu'$ = 1.6686, $e'$ = 0.2759, $t'$ = 2.5568), while (b) is the decrypted image of decrypting (a) with correct keys, which is exactly the same as the plain image. Figure 12c–f show the decryption results with the error keys respectively, where the only minimal difference is between the wrong key and the correct key. Assuming the attackers subtract the error decrypted image pairwise to analyze information to help crack the cipher image, any information of the plain image will not be disclosed to them. The corresponding results are shown in Figure 12g–l. Therefore, the proposed algorithm is highly sensitive to the keys.

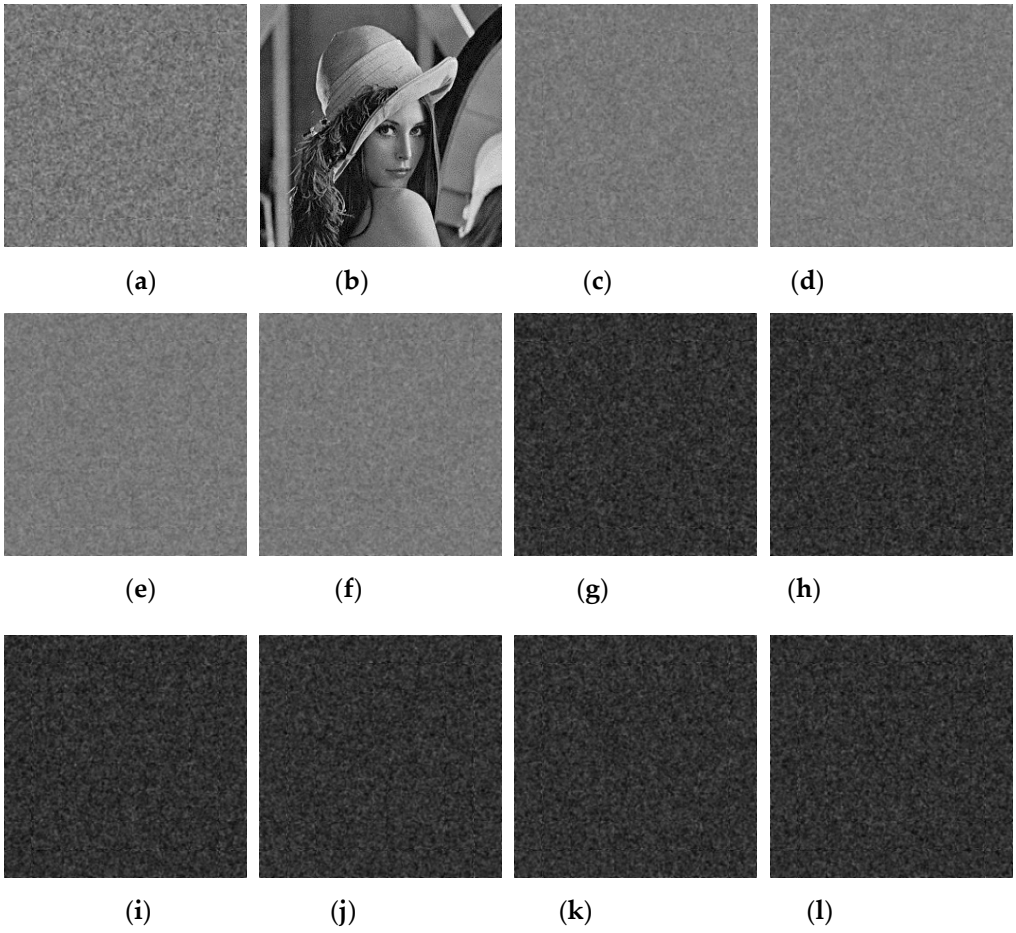



**Figure 12.** Key sensitivity test: (**a**) Initial encryption; (**b**) Decryption with correct keys; (**c**) Decryption with $x'^0_0$ = 0.986500000000001; (**d**) Decryption with $y'^0_0$ = 1.433500000000001; (**e**) Decryption with $z'^0_0$ = 1.497700000000001; (**f**) Decryption with $u'^0_0$ = 0.550100000000001; (**g**) Image of subtraction between (**c**,**d**); (**h**) Image of subtraction between (**c**,**e**); (i) Image of subtraction between (**c**,**f**); (**j**) Image of subtraction between (**d**,**e**); (k) Image of subtraction between (**d**,**f**); (**l**) Image of subtraction between (**e**,**f**).

#### 4.3.3. Histogram Analysis

The histogram can reflect the statistical characteristics of the distribution of pixel values. For an ideal encryption algorithm, the histogram of the cipher image should always be uniform [45]. Figure 13 shows the 3D histograms of the plain images and the cipher images with various pixel depth. The experimental results show that the histograms of the cipher images are evenly distributed, which are completely different from the plain images and no longer present any statistical characteristics of plain images.

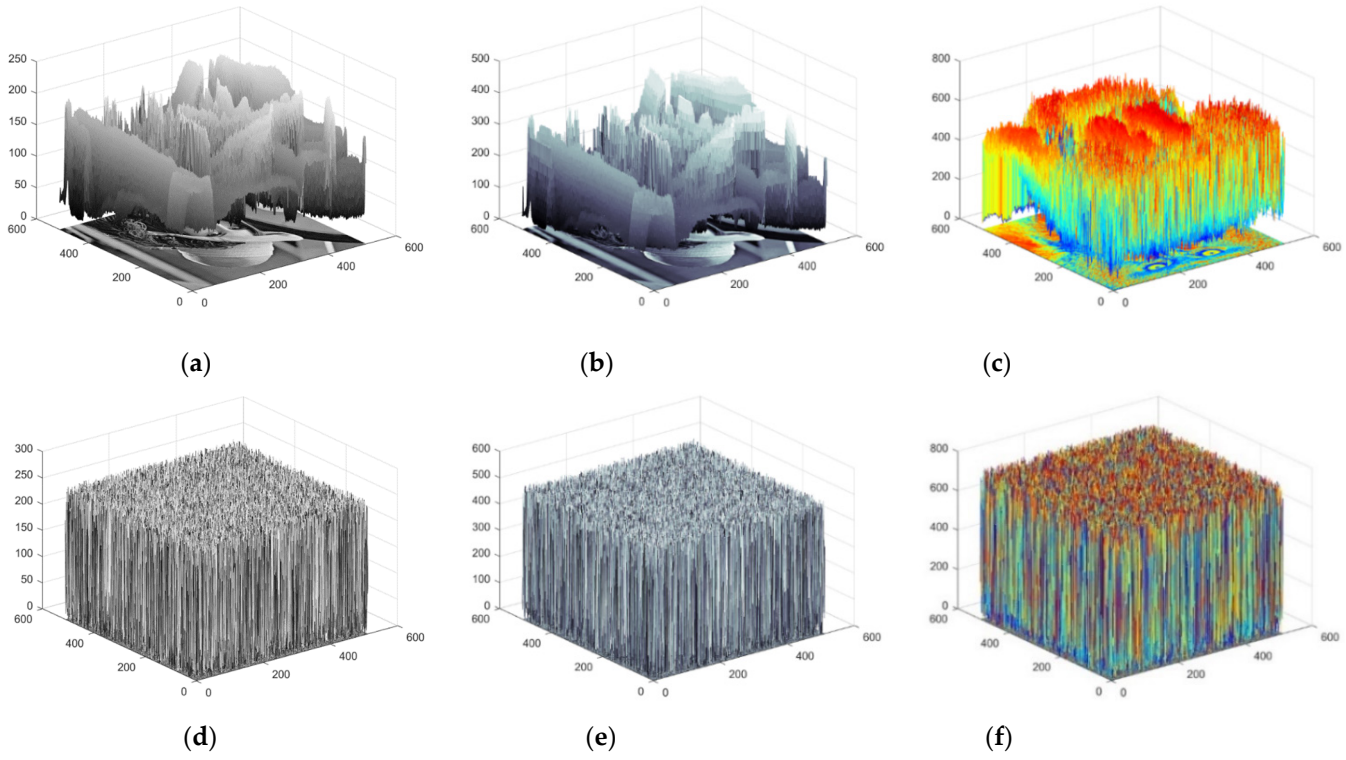



**Figure 13.** 3D histograms of the plain images and the cipher images: (**a**) Histogram of 8BPP Lena; (**b**) Histogram of 16BPP Lena; (**c**) Histogram of 24 BPP Baboon; (**d**) Histogram of 8BPP Lena's cipher image; (**e**) Histogram of 16BPP Lena's cipher image; (**f**) Histogram of 24 BPP Baboon's cipher image.

#### 4.3.4. Chi-Squared Test

The chi-square test can produce quantitative numerical results to analyze the distribution of cipher images more accurately while avoiding visual deception. We use the formula in an article [49] for comparative analysis as follows:

$$\chi^2 = \sum_{i=0}^{255} \frac{(q^i - q)^2}{q} \tag{25}$$

where $q^i$ represents the times the pixel value $i$ appears in the image. $q$ is the theoretical value, which is defined as:

$$q = \frac{m \times n}{256} \tag{26}$$

where $m$ and $n$ are the size of the image. The larger $\chi^2$ is, the more the pixel values deviate from the average level, and the more uneven the pixel distribution is. Table 13 provides $\chi^2$ values of the corresponding test results, we can see that the $\chi^2$ value of the cipher image is much smaller than that of the plain image.

**Table 13.** Test about the pixel histogram.

| Images | Component | $\chi^2_{cipher}$ | Average | $\chi^2_{255}$ | Results |
|---|---|---|---|---|---|
| Lena (8BPP) | - | 236.56 | 236.56 | | Pass |
| Lena (16BPP) | - | 258.36 | 258.36 | | Pass |
| Baboon (24BPP) | R | 278.47 | 249.16 | 293.25 | Pass |
| | G | 232.33 | | | Pass |
| | B | 236.69 | | | Pass |

4.3.5. Information Entropy Analysis

Information entropy is a measure of the indeterminacy of image information. The information entropy of an ideal cipher image should be close to 8 [50]. The information entropy of the gray image is described as

$$H(m) = -\sum_{i=0}^{255} P(m_i) \log_2 P(m_i) \tag{27}$$

where $m_i$ is the $i$ th gray level for the digital image $I$ with 256 gray levels, and $P(m_i)$ is the emergence probability of $m_i$. Table 14 shows the entropy of plain images and cipher images with various pixel depth and also includes comparison with other algorithms. Therefore, the proposed algorithm can effectively resist the statistical attack.

**Table 14.** Shannon entropy.

| Images | Plain Image | Proposed Algorithm | Ref. [51] | Ref. [46] | Ref. [47] |
|---|---|---|---|---|---|
| Lena (8BPP) | 7.44557 | 7.99935 | 7.9979 | 7.9979 | 7.99934 |
| Lena (16BPP) | 7.46533 | 7.99932 | - | - | - |
| Baboon (24BPP) | 7.36642 | 7.99934 | 7.9976 | 7.99911 | 7.99932 |

4.3.6. Differential Attack Analysis

Differential attack is often used to test the plaintext sensitivity of image encryption algorithms [49]. In the encryption procedure, the ideal encryption algorithm should respond powerfully to the slight changes in the plain image. The Number of Pixels Change Rate (NPCR) reflects the number of changed pixels in the cipher image after the plain image is changed. The Unified Average Changing Intensity (UACI) measures the average difference intensity of pixel values between two cipher images, which correspond to the original image and the changed original image. To evaluate the ability to resist the differential attack, we employed the NPCR and UACI as the indicators for evaluating the differential attack. NPCR and UACI are defined by

$$NPCR = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} f(i,j)}{m \times n} \times 100\% \tag{28}$$

$$UACI = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} |I'(i,j) - I''(i,j)|}{255 \times m \times n} \times 100\% \tag{29}$$

where $I'(i,j)$ is the cipher image of the plain image, $I''(i,j)$ is the cipher image of the modified plain image, and $f(i,j)$ is defined by

$$f(i,j) = \begin{cases} 0 & I'(i,j) = I''(i,j) \\ 1 & I'(i,j) \neq I''(i,j) \end{cases} . \tag{30}$$

Even if two images are very similar, such as only one-bit difference, their hash values of SHA-256 are completely different [51]. Since the keys are related to the hash value, their values are very sensitive to the plain image. In the experiment, a pixel $I(68, 189)$ of the plain image is chosen. To test the ability to resist the differential attack, the gray value of this pixel is changed to 200. The corresponding experimental results are shown in Figure 14 and Table 15. For the proposed algorithm, compared with standard values NPCR = 99.6094% and UACI = 33.4635%, our results are closer to theoretical values than similar algorithms, as shown in Table 15, and they demonstrate that a slight change to the plain image will result in a great change in the cipher image. Therefore, the proposed algorithm has an excellent ability to resist the differential attack.

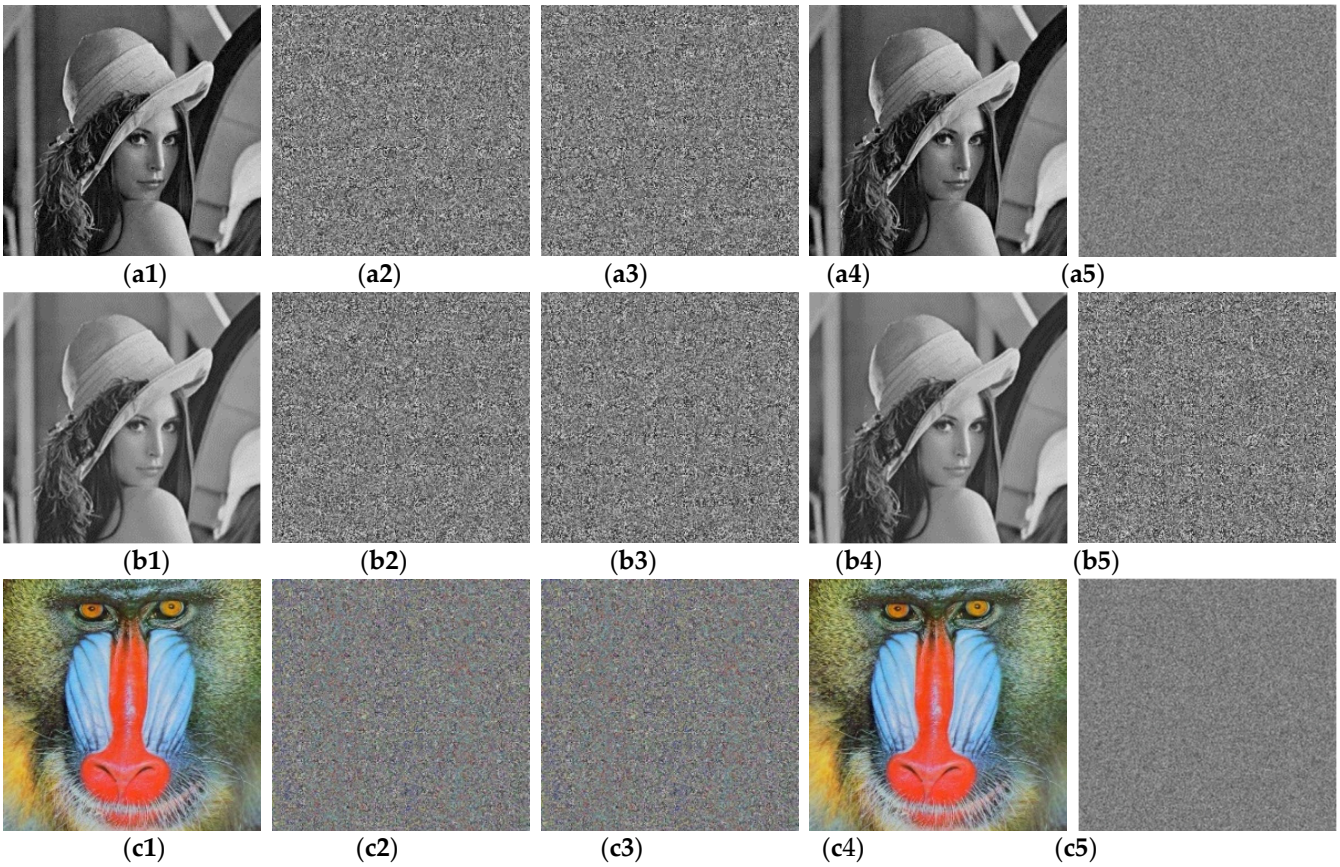



**Figure 14.** Cipher images and decrypted images of plain images with one-pixel difference: (**a1**) Lena (8BPP); (**a2**) Cipher image of (**a1**); (**a3**) Cipher image of (**a1**) changed by one pixel; (**a4**) Decryption of (**a2**); (**a5**) Decryption of (**a3**); (**b1**) Lena (16BPP); (**b2**) Cipher image of (**b1**); (**b3**) Cipher image of (**b1**) changed by one pixel; (**b4**) Decryption of (**b2**); (**b5**) Decryption of (**b3**); (**c1**) Baboon; (**c2**) Cipher image of (**c1**); (**c3**) Cipher image of (**c1**) changed by one pixel; (**c4**) Decryption of (**c2**); (**c5**) Decryption of (**c3**).

**Table 15.** NPCR and UACI values of the cipher images (%).

| Images | NPCR | | | | UACI | | | |
|---|---|---|---|---|---|---|---|---|
| | **Proposed** | **Ref. [48]** | **Ref. [52]** | **Ref. [53]** | **Proposed** | **Ref. [48]** | **Ref. [52]** | **Ref. [53]** |
| Lena (8BPP) | 99.6132 | 99.57 | 99.5636 | 99.6101 | 33.4236 | 33.33 | 33.4417 | 33.4745 |
| Lena (16BPP) | 99.6145 | - | - | - | 33.4572 | - | - | - |
| Baboon (24BPP) | 99.6021 | 99.57 | 99.6293 | 99.6113 | 33.4768 | 33.47 | 33.3796 | 33.4928 |

#### 4.3.7. Encryption Quality Analysis

The encryption quality of cipher images is often quantitatively verified by mean square error (MSE), peak signal-to-noise ratio (PSNR), and structural similarity index (SSIM). The specific formulas of them are defined as follows [45]:

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} [P(i,j) - E(i,j)]^2 \tag{31}$$

$$PSNR = 20 \log_{10}(\frac{255}{\sqrt{MSE}}) \tag{32}$$

$$SSIM = \frac{(2\overline{PE} + C_1)(2\sigma_{PE} + C_2)}{(\overline{P}^2 + \overline{E}^2 + C_1)(\sigma^2{}_P + \sigma^2{}_E + C_2)} \tag{33}$$

where

$$C_1 = (K_1 L)^2 \tag{34}$$

$$C_2 = (K_2 L)^2 \tag{35}$$

$$\overline{P} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} P(i,j) \tag{36}$$

$$\overline{E} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} E(i,j) \tag{37}$$

$$\sigma_P = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} [P(i,j) - \overline{P}]^2 \tag{38}$$

$$\sigma_E = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} [E(i,j) - \overline{E}]^2 \tag{39}$$

$$\sigma_{PE} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} [P(i,j) - \overline{P}][E(i,j) - \overline{E}] \tag{40}$$

where $M \times N$ is the size of the image, $P$ is the plain image, $E$ is the cipher image, $\overline{P}$ is the mean of the plain image, $\overline{E}$ is the mean of the cipher image, $\sigma_P$ is the standard deviation of the plain image, $\sigma_E$ is the standard deviation of cipher image, and $\sigma_{PE}$ is the cross-correlation of the plain image and cipher image. $L$ is the dynamic range of the pixel values (for grayscale images, $L = 255$) with $K_1 = 0.01$ and $K_2 = 0.03$.

Currently, the cipher image with a large MSE value, a PSNR lower than 10 dB, and an SSIM close to 0 means an efficient pseudorandom ciphertext and has a structural difference with its plain image. As shown in Table 16, the indicators of the proposed algorithm are relatively close to ideal values.

**Table 16.** Encryption quality analysis.

| Encrypted Image | MSE | PSNR (dB) | SSIM |
|---|---|---|---|
| Lena (8BPP) | 7757 | 9.2338 | 0.0106 |
| Lena (16BPP) | 65,535 | 7.9416 | 0.0103 |
| Baboon (24BPP) | 8562 | 8.8047 | 0.0089 |

#### 4.3.8. Correlation Analysis

The strong correlation of adjacent pixels is an important feature of digital images [34]. Therefore, the correlation of adjacent pixels is one of the important criteria for evaluating the performance of an image encryption algorithm. To analyze the performance of the algorithm in the pixel correlation test, we conducted corresponding experiments. The correlation coefficient of each pair is defined by

$$rx, y = E((x - E(x))(y - E(y)))D(x)D(y) \tag{41}$$

where $E(x)$ and $D(x)$ are the mathematical expectation and variance of the data $x$, respectively. They are defined by

$$E(x) = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{42}$$

$$D(x) = \frac{1}{N} \sum_{i=1}^{N} [x_i - E(x)]^2 \tag{43}$$

For the proposed algorithm, 40,000 pairs of adjacent pixels are randomly selected from the plain images and the cipher images. For all plain images and cipher images with

various pixel depth, Figures 15–20 reflect their horizontal, vertical, and diagonal relevance for adjacent pixels, respectively. For the proposed algorithm, Table 17 lists the adjacent pixels of the plain image and the correlation coefficients of their corresponding cipher images. Experimental results show that the correlation coefficients of the plain images are close to 1, whereas the correlation coefficients of the cipher images are close to 0 in all directions. Therefore, the proposed algorithm can destroy the correlation between adjacent pixels well and protect the content of the plain image.

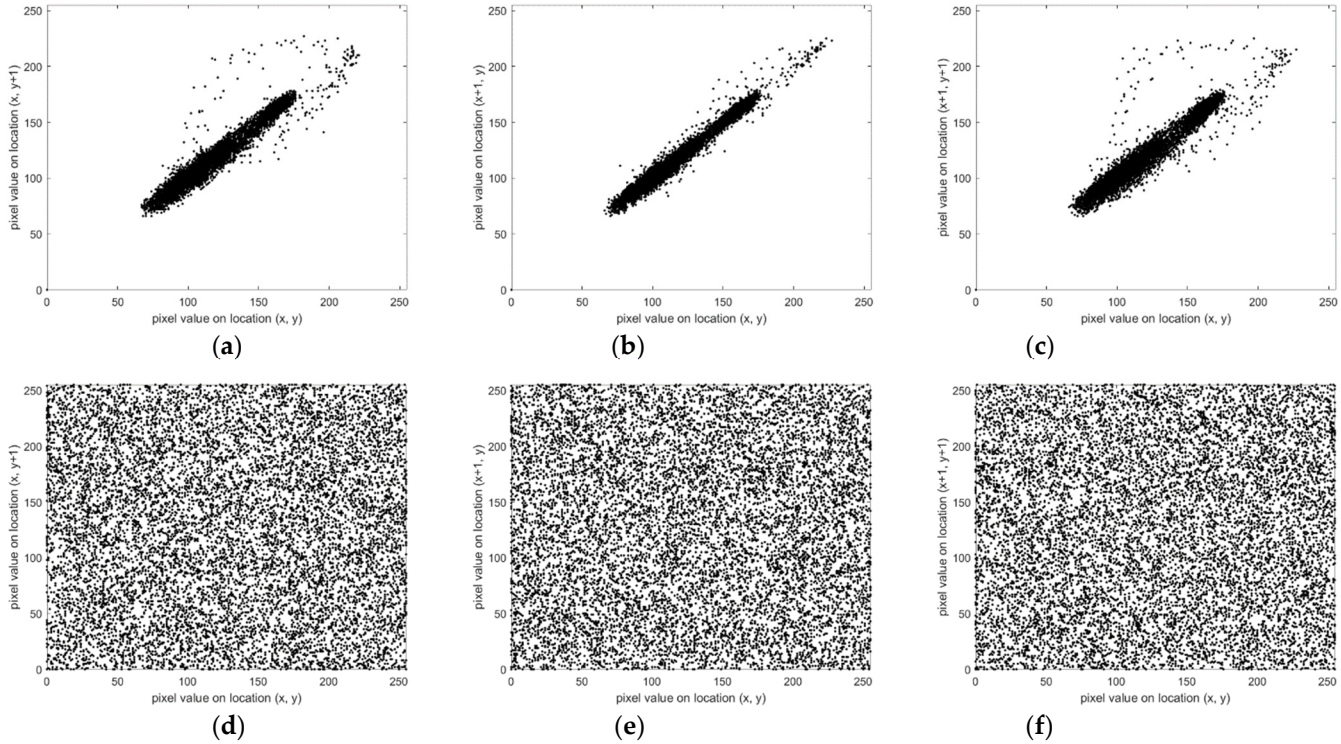



**Figure 15.** Adjacent pixel correlation of Lena (8BPP) and its corresponding cipher image: (**a**) Horizontal direction; (**b**) Vertical direction; (**c**) Diagonal direction; (**d**) Horizontal direction; (**e**) Vertical direction; (**f**) Diagonal direction.

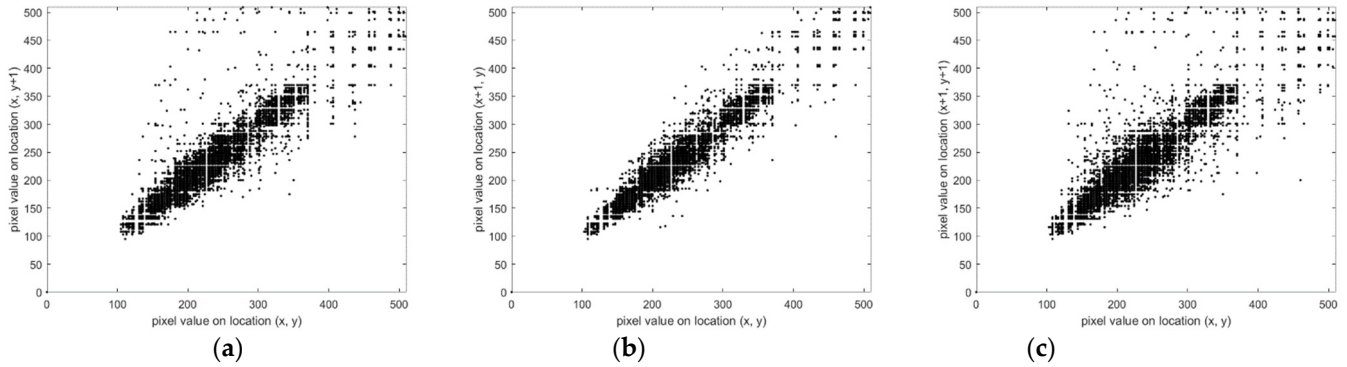



**Figure 16.** *Cont.*

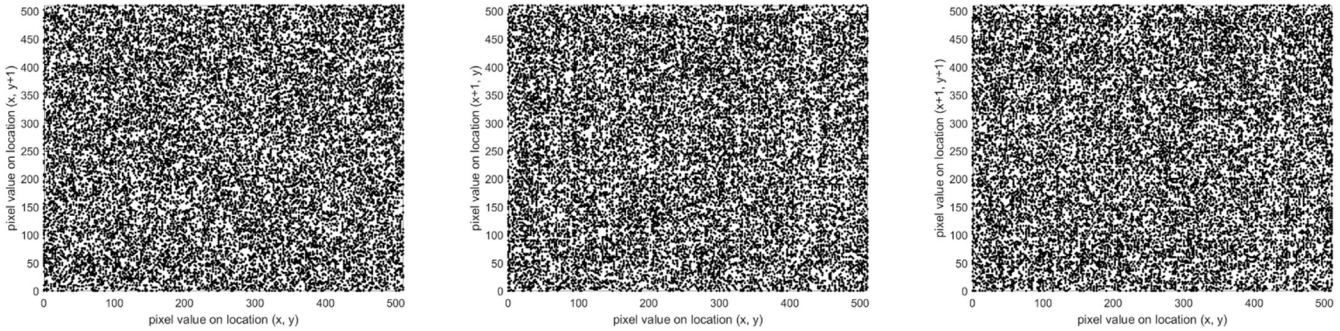



**Figure 16.** Adjacent pixel correlation of Lena (16BPP) and its corresponding cipher image: (**a**) Horizontal direction; (**b**) Vertical direction; (**c**) Diagonal direction; (**d**) Horizontal direction; (**e**) Vertical direction; (**f**) Diagonal direction.

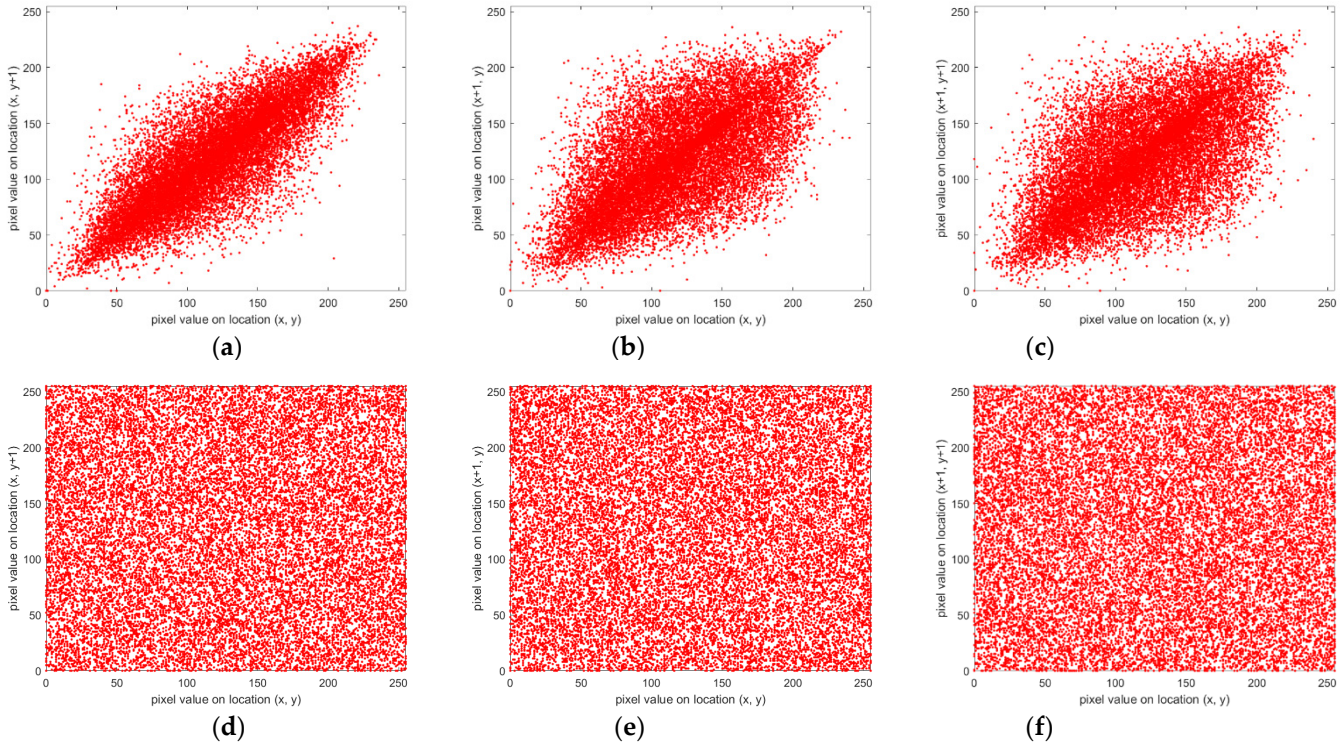



**Figure 17.** Red-component Adjacent pixel correlation of Baboon and corresponding cipher image: (**a**) Horizontal direction; (**b**) Vertical direction; (**c**) Diagonal direction; (**d**) Horizontal direction; (**e**) Vertical direction; (**f**) Diagonal direction.

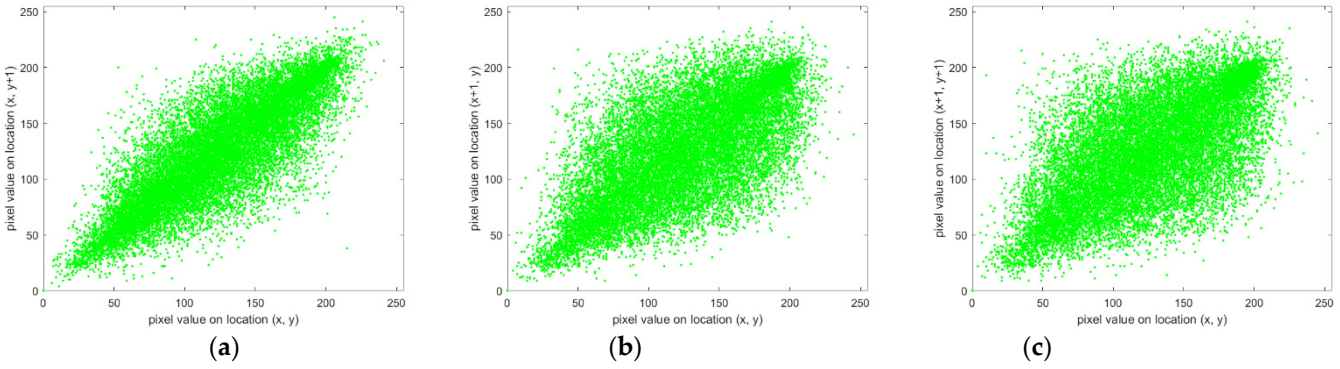
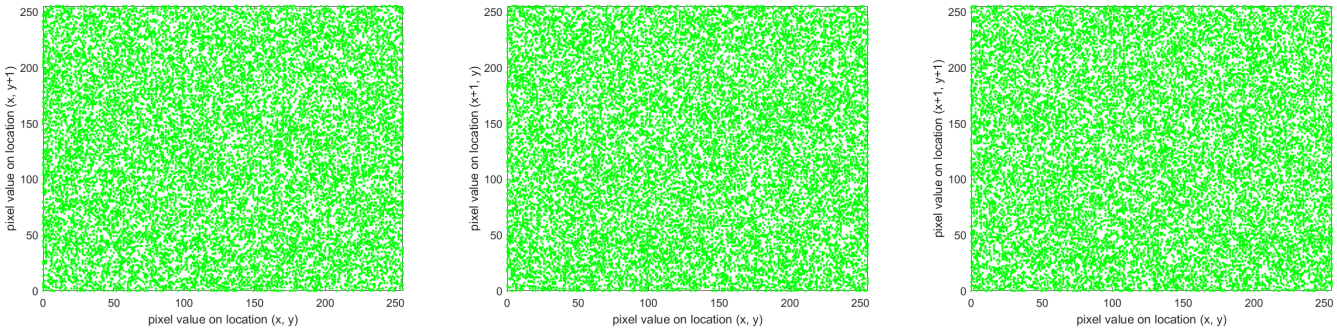
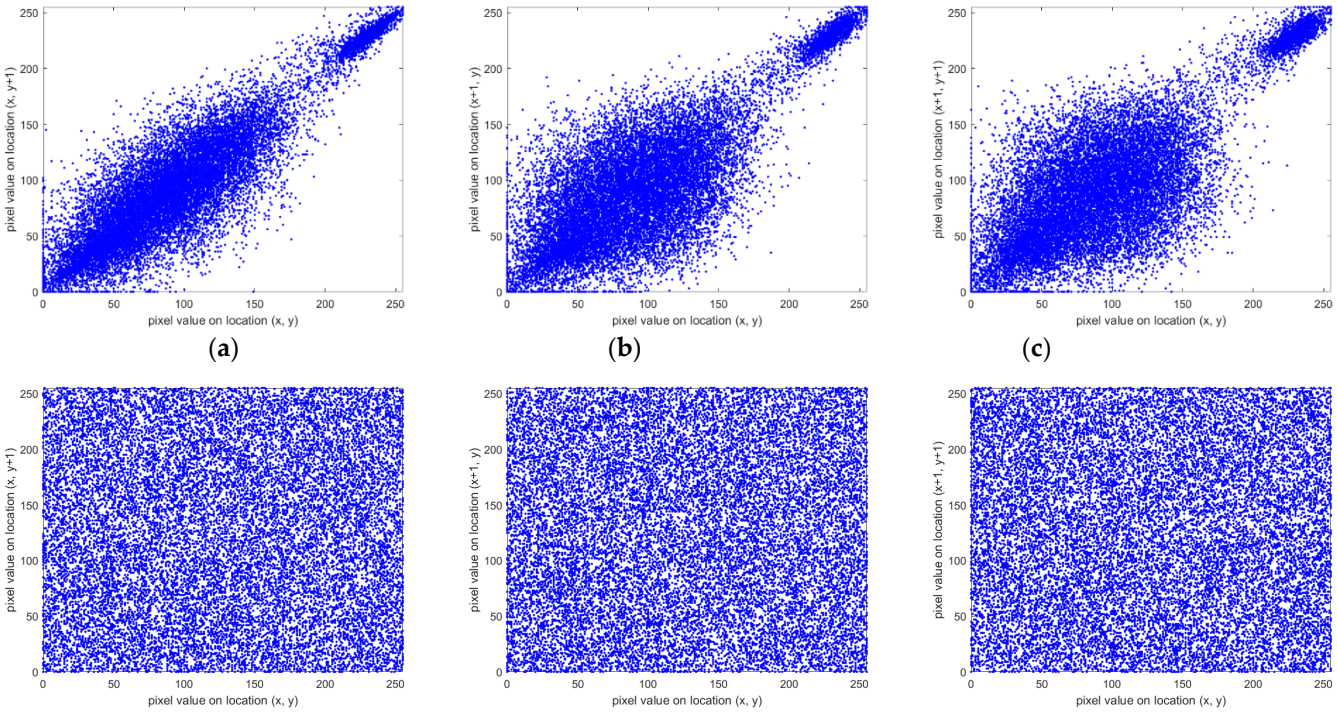



**Figure 18.** *Cont.*




(**d**) (**e**) (**f**)

**Figure 18.** Green-component Adjacent pixel correlation of Baboon and corresponding cipher image: (**a**) Horizontal direction; (**b**) Vertical direction; (**c**) Diagonal direction; (**d**) Horizontal direction; (**e**) Vertical direction; (**f**) Diagonal direction.




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

(**d**) (**e**) (**f**)

**Figure 19.** Blue-component Adjacent pixel correlation of Baboon and corresponding cipher image: (**a**) Horizontal direction; (**b**) Vertical direction; (**c**) Diagonal direction; (**d**) Horizontal direction; (**e**) Vertical direction; (**f**) Diagonal direction.

**Table 17.** Correlation coefficient values of adjacent pixels.

| Algorithms | Images | Components | Plain Images | | | Cipher Images | | |
|---|---|---|---|---|---|---|---|---|
| | | | H | V | D | H | V | D |
| Proposed | Lena (8BPP) | - | 0.9719 | 0.9849 | 0.9591 | 0.0024 | −0.0017 | 0.0011 |
| | Lena (16BPP) | - | 0.9726 | 0.9839 | 0.9604 | 0.0056 | 0.0012 | 0.0054 |
| | Baboon (24BPP) | R | 0.9427 | 0.8758 | 0.8503 | 0.0012 | −0.0030 | −0.0012 |
| | | G | 0.9139 | 0.8170 | 0.7770 | 0.0017 | 0.0011 | 0.0008 |
| | | B | 0.9489 | 0.8936 | 0.8691 | 0.0043 | −0.0006 | −0.0018 |
| Ref. [51] | Lena (8BPP) | - | 0.9276 | 0.9574 | 0.9231 | 0.0009 | −0.0028 | −0.0027 |
| Ref. [46] | Lena (8BPP) | - | 0.9329 | 0.9650 | 0.9066 | 0.0017 | 0.0019 | 0.0008 |

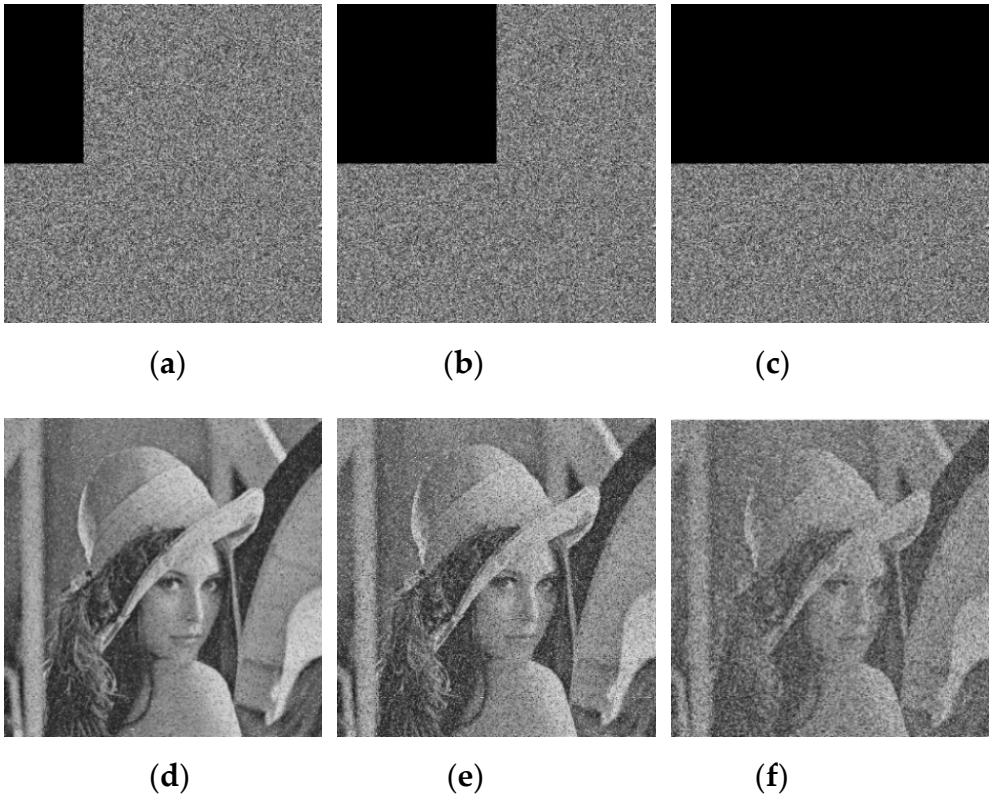



**Figure 20.** The results of cropping attack: (**a**) 12.5% cropping; (**b**) 25% cropping; (**c**) 50% cropping; (**d**) Decryption of (**a**); (**e**) Decryption of (**b**); (**f**) Decryption of (**c**).

#### 4.3.9. Robustness Analysis

Destroying the integrity of ciphertext is a common method in hacker attacks, which can prevent the recipient from obtaining plaintext information successfully. Cropping and noise attacks are the two most common methods [30]. In this section, cipher images that have been damaged in different ways and degrees are used as samples to participate in the test, and the results are satisfactory. In the cropping attacks test, we set 12.5%, 25%, and 50% of cropping in a cipher image. As shown in Figure 20, the attacked cipher images still contain considerable plaintext information after being decrypted. In the noise attacks test, salt and pepper noises with strengths of 2%, 10%, and 20% were mixed into the cipher image respectively, and the decryption results are shown in Figure 21.

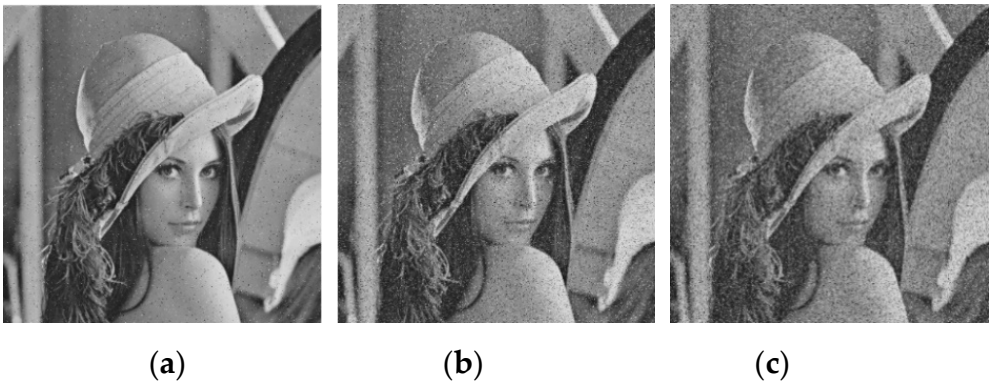



**Figure 21.** The results of noise attack analysis: (**a**) Noise with 2% of intensity; (**b**) Noise with 10% of intensity; (**c**) Noise with 20% of intensity.

4.3.10. Time Complexity Analysis

The size of the plain image is $m \times n$. For our algorithm, two stages mainly consume time. One stage is the generation of random sequences. A Chen-4D hyperchaotic system and 2D-ILASM chaotic systems iterates multiple rounds to generate six sequences $\{X, Y, Z, U, V, W\}$ whose longest length is $m \times n \times d/2$. Therefore, the complexity of this stage is $O(m \times n \times d/2)$. The other stage is the adaptive RNA coding and operation, including six RNA coding and four RNA operation steps. The complexity of this stage is still $O(m \times n)$. Finally, the complexity for the proposed algorithm is $O(m \times n \times d/2)$, which means that its complexity is linear. In summary, the proposed algorithm's time complexity is determined by the size and the pixel depth of the plain image.

## 5. Discussion

As we envisioned, integrating the current time and pixel depth into the encryption algorithm can update keys timely while making the algorithm self-adaptive to solve the problems of some algorithms. Since the proposed algorithm operates at the RNA base level throughout, it is time-consuming to a certain extent. Though the encryption speed of the proposed algorithm meets the basic requirements of image encryption algorithms and is faster than some existing ones, it is still not enough for real-time encryption. The proposed algorithm is only for one image. We may extend it to multiple-image encryption in subsequent research. In theory, the proposed algorithm can be utilized for images with greater pixel depth (such as remote sensing images). However, this has already involved the research field of hyperspectral image processing and has not been carried out in this paper. The experimental results have been able to prove the performance of the proposed algorithm. The authors will focus on the aforementioned problems to conduct deeper research.

## 6. Conclusions

This paper proposes an adaptive, chaotic image encryption algorithm based on RNA and pixel depth to enhance encryption performance and applicability. This paper designs a novel chaotic system called "2D-ILASM" and the 3D-AAT for scrambling, and keys are generated from the hash values of the plain image and current time; then, a pre-permuted RNA cube is constructed for 3D adaptive scrambling finally. RNA operation is introduced to realize the selective diffusion effect of RNA bases. Through the experimental results and algorithm analyses, the proposed algorithm is suitable for images with different pixel depths, updates the keys in real time, and has sufficient security to resist various attacks, such as the brute-force attack, differential attack, and statistical analysis attack, etc. Furthermore, this algorithm can extend to multiple images. Therefore, when people need to transfer different types of valuable images at the same time, the proposed algorithm is the better choice to ensure applicability, security, and the transmission privacy.

**Author Contributions:** Conceptualization, X.Z.; methodology, X.Z.; software, X.Y.; validation, X.Z.; investigation, X.Y.; resources, X.Z.; data curation, X.Y.; writing—original draft preparation, X.Z. and X.Y.; writing—review and editing, X.Y.; visualization, X.Y.; supervision, X.Z.; funding acquisition, X.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by "the Future Outstanding Talent Assistance Program of China University of Mining and Technology, grant number: 2020WLJCRCZL023" and "the Postgraduate Research & Practice Innovation Program of Jiangsu Province, grant number: KYCX20_1883".

**Acknowledgments:** Authors would like to express their sincerely thanks to the anonymous reviewers for their constructive comments and suggestions.

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

## References


1. Zhang, Y.; Tang, Y. A plaintext-related image encryption algorithm based on chaos. *Multimed. Tools Appl.* **2018**, *77*, 6647–6669. [CrossRef]
2. Zhu, S.; Zhu, C.; Wang, W. A New Image Encryption Algorithm Based on Chaos and Secure Hash SHA-256. *Entropy* **2018**, *20*, 716. [CrossRef] [PubMed]
3. Xu, L.; Li, Z.; Li, J.; Hua, W. A novel bit-level image encryption algorithm based on chaotic maps. *Opt. Lasers Eng.* **2016**, *78*, 17–25. [CrossRef]
4. Wang, X.; Feng, L.; Zhao, H. Fast image encryption algorithm based on parallel computing system. *Inform. Sci.* **2019**, *486*, 340–358. [CrossRef]
5. Matthews, R. On the derivation of a chaotic encryption algorithm. *Cryptologia* **1989**, *13*, 29–42. [CrossRef]
6. Herbadji, D.; Belmeguenai, A.; Derouiche, N.; Liu, H. Colour image encryption scheme based on enhanced quadratic chaotic map. *IET Image Process.* **2020**, *14*, 40–52. [CrossRef]
7. Wu, J.; Shi, J.; Li, T. A Novel Image Encryption Approach Based on a Hyperchaotic System, Pixel-Level Filtering with Variable Kernels, and DNA-Level Diffusion. *Entropy* **2020**, *22*, 5. [CrossRef]
8. Zhang, Q.; Han, J.; Ye, Y. Image encryption algorithm based on image hashing, improved chaotic mapping and DNA coding. *IET Image Process.* **2019**, *13*, 2905–2915. [CrossRef]
9. Zhang, X.; Wang, X. Multiple-Image Encryption Algorithm Based on the 3D Permutation Model and Chaotic System. *Symmetry* **2018**, *10*, 660. [CrossRef]
10. Anwar, S.; Meghana, S. A pixel permutation based image encryption technique using chaotic map. *Multimed. Tools Appl.* **2019**, *78*, 27569–27590. [CrossRef]
11. El-Latif, A.A.A.; Abd-El-Atty, B.; Belazi, A.; Iliyasu, A.M. Efficient Chaos-Based Substitution-Box and Its Application to Image Encryption. *Electronics* **2021**, *10*, 1392. [CrossRef]
12. Musanna, F.; Dangwal, D.; Kumar, S. Novel image encryption algorithm using fractional chaos and cellular neural network. *J. Ambient Intell. Humaniz. Comput.* **2021**, 1–22, (prepublish). [CrossRef]
13. Pourasad, Y.; Ranjbarzadeh, R.; Mardani, A. A New Algorithm for Digital Image Encryption Based on Chaos Theory. *Entropy* **2021**, *23*, 341. [CrossRef] [PubMed]
14. Liu, Y.; Zhang, J. A Multidimensional Chaotic Image Encryption Algorithm based on DNA Coding. *Multimed. Tools Appl.* **2020**, *79*, 21579–21601. [CrossRef]
15. Zhou, S.; Wang, X.; Wang, M.; Zhang, Y. Simple colour image cryptosystem with very high level of security. *Chaos Solitons Fractals* **2020**, *141*, 110225. [CrossRef]
16. Gan, Z.; Chai, X.; Han, D.; Chen, Y. A chaotic image encryption algorithm based on 3-D bit-plane permutation. *Neural Comput. Appl.* **2019**, *31*, 7111–7130. [CrossRef]
17. Arab, A.; Rostami, M.J.; Ghavami, B. An image encryption method based on chaos system and AES algorithm. *J. Supercomput.* **2019**, *75*, 6663–6682. [CrossRef]
18. Xu, Q.; Sun, K.; Cao, C.; Zhu, C. A fast image encryption algorithm based on compressive sensing and hyperchaotic map. *Opt. Lasers Eng.* **2019**, *121*, 203–214. [CrossRef]
19. Wang, X.; Ren, Q.; Jiang, D. An adjustable visual image cryptosystem based on 6D hyperchaotic system and compressive sensing. *Nonlinear Dynam.* **2021**, *104*, 4543–4567. [CrossRef]
20. Zhang, Q.; Han, J. A novel color image encryption algorithm based on image hashing, 6D hyperchaotic and DNA coding. *Multimed. Tools Appl.* **2021**, *80*, 13841–13864. [CrossRef]
21. Batool, S.I.; Waseem, H.M. A novel image encryption scheme based on Arnold scrambling and Lucas series. *Multimed. Tools Appl.* **2019**, *78*, 27611–27637. [CrossRef]
22. Li, X.; Li, T.; Wu, J.; Xie, Z.; Shi, J. Joint image compression and encryption based on sparse Bayesian learning and bit-level 3D Arnold cat maps. *PLoS ONE* **2019**, *14*, e224382. [CrossRef]
23. Jithin, K.C.; Sankar, S. Colour image encryption algorithm combining Arnold map, DNA sequence operation, and a Mandelbrot set. *J. Inf. Secur. Appl.* **2020**, *50*, 102428. [CrossRef]
24. Jiao, K.; Ye, G.; Dong, Y.; Huang, X.; He, J. Image Encryption Scheme Based on a Generalized Arnold Map and RSA Algorithm. *Secur. Commun. Netw.* **2020**, *2020*, 1–14. [CrossRef]
25. Wang, T.; Wang, M. Hyperchaotic image encryption algorithm based on bit-level permutation and DNA encoding. *Opt. Laser Technol.* **2020**, *132*, 106355–106367. [CrossRef]
26. Kumar, A.; Raghava, N.S. An efficient image encryption scheme using elementary cellular automata with novel permutation box. *Multimed. Tools Appl.* **2021**, *80*, 21727–21750. [CrossRef]
27. Patro, K.A.K.; Acharya, B. An efficient dual-layer cross-coupled chaotic map security-based multi-image encryption system. *Nonlinear Dynam.* **2021**, *104*, 2759–2805. [CrossRef]
28. Li, Z.; Peng, C.; Tan, W.; Li, L. An Efficient Plaintext-Related Chaotic Image Encryption Scheme Based on Compressive Sensing. *Sensors* **2021**, *21*, 758. [CrossRef]
29. El-Khamy, S.E.; Mohamed, A.G. An efficient DNA-inspired image encryption algorithm based on hyper-chaotic maps and wavelet fusion. *Multimed. Tools Appl.* **2021**, *80*, 23319–23335. [CrossRef]

30. Xian, Y.; Wang, X.; Yan, X.; Li, Q.; Wang, X. Image Encryption Based on Chaotic Sub-Block Scrambling and Chaotic Digit Selection Diffusion. *Opt. Laser. Eng.* **2020**, *134*, 106202–106223. [CrossRef]
31. Zhang, L.; Zhang, X. Multiple-image encryption algorithm based on bit planes and chaos. *Multimed. Tools Appl.* **2020**, *79*, 20753–20771. [CrossRef]
32. Liu, Y.; Zhang, J.; Han, D.; Wu, P.; Sun, Y.; Moon, Y.S. A multidimensional chaotic image encryption algorithm based on the region of interest. *Multimed. Tools Appl.* **2020**, *79*, 17669–17705. [CrossRef]
33. Zarebnia, M.; Pakmanesh, H.; Parvaz, R. A fast multiple-image encryption algorithm based on hybrid chaotic systems for gray scale images. *Optik* **2019**, *179*, 761–773. [CrossRef]
34. Yadollahi, M.; Enayatifar, R.; Nematzadeh, H.; Lee, M.; Choi, J. A novel image security technique based on nucleic acid concepts. *J. Inf. Secur. Appl.* **2020**, *53*, 102505–102515. [CrossRef]
35. Abbasi, A.A.; Mazinani, M.; Hosseini, R. Chaotic evolutionary-based image encryption using RNA codons and amino acid truth table. *Opt. Laser Technol.* **2020**, *132*, 106465–106477. [CrossRef]
36. JarJar, A. Two Feistel rounds in image cryptography acting at the nucleotide level exploiting dna and rna property. *SN Appl. Sci.* **2019**, *1*, 1411–1427. [CrossRef]
37. Zhang, D.; Chen, L.; Li, T. Hyper-Chaotic Color Image Encryption Based on Transformed Zigzag Diffusion and RNA Operation. *Entropy* **2021**, *23*, 361. [CrossRef]
38. Hua, Z.; Zhou, Y. Image encryption using 2D Logistic-adjusted-Sine map. *Inform. Sci.* **2016**, *339*, 237–253. [CrossRef]
39. Zhao, H.; Xie, S.; Zhang, J.; Wu, T. Efficient image encryption using two-dimensional enhanced hyperchaotic Henon map. *J. Electron. Imaging* **2021**, *29*, 23007–23033. [CrossRef]
40. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. *NIST Spec. Publ.* **2010**. 22–800 Revision 1a.
41. Chai, X.; Fu, X.; Gan, Z.; Lu, Y.; Chen, Y. A color image cryptosystem based on dynamic DNA encryption and chaos. *Signal Process.* **2019**, *155*, 44–62. [CrossRef]
42. Zhang, J.; Hou, D.; Ren, H. Image Encryption Algorithm Based on Dynamic DNA Coding and Chen's Hyperchaotic System. *Math. Probl. Eng.* **2016**, *2016*, 1–11. [CrossRef]
43. Wang, X.; Guan, N. A novel chaotic image encryption algorithm based on extended Zigzag confusion and RNA operation. *Opt. Laser Technol.* **2020**, *131*, 106366–106382. [CrossRef]
44. Pan, T.G.; Li, D.Y. A New Algorithm of Image Encryption Based on 3D Arnold Cat. *Adv. Eng. Forum.* **2011**, *1*, 183–187. [CrossRef]
45. Murillo-Escobar, M.A.; Meranza-Castillón, M.O.; López-Gutiérrez, R.M.; Cruz-Hernández, C. Suggested Integral Analysis for Chaos-Based Image Cryptosystems. *Entropy* **2019**, *21*, 815. [CrossRef] [PubMed]
46. Wang, X.; Liu, L. Application of chaotic Josephus scrambling and RNA computing in image encryption. *Multimed. Tools Appl.* **2021**, *80*, 1–22. [CrossRef]
47. Zhu, H.; Zhao, Y.; Song, Y. 2D Logistic-Modulated-Sine-Coupling-Logistic Chaotic Map for Image Encryption. *IEEE Access* **2019**, *7*, 14081–14098. [CrossRef]
48. Amina, S.; Mohamed, F.K. An efficient and secure chaotic cipher algorithm for image content preservation. *Commun. Nonlinear Sci.* **2018**, *60*, 12–32. [CrossRef]
49. Chai, X.; Chen, Y.; Broyde, L. A novel chaos-based image encryption algorithm using DNA sequence operations. *Opt. Lasers Eng.* **2017**, *88*, 197–213. [CrossRef]
50. Rushdi, M.H.A.A.; Nehary, E.A. Image encryption via discrete fractional Fourier-type transforms generated by random matrices. *Signal Process. Image Commun.* **2016**, *49*, 25–46. [CrossRef]
51. Liao, X.; Kulsoom, A.; Ullah, S. A modified (Dual) fusion technique for image encryption using SHA-256 hash and multiple chaotic maps. *Multimed. Tools Appl.* **2016**, *75*, 11241–11266. [CrossRef]
52. Zhang, X.; Wang, L.; Zhou, Z.; Niu, Y. A Chaos-Based Image Encryption Technique Utilizing Hilbert Curves and H-Fractals. *IEEE Access* **2019**, *7*, 74734–74746. [CrossRef]
53. Song, Y.; Zhu, Z.; Zhang, W.; Yu, H.; Zhao, Y. Efficient and Secure Image Encryption Algorithm Using a Novel Key-Substitution Architecture. *IEEE Access* **2019**, *7*, 84386–84400. [CrossRef]