entropy

MDPI

Article

Fast and Efficient Image Encryption Algorithm Based on Modular Addition and SPD

Khushbu Khalid Butt [1], Guohui Li [2,*], Sajid Khan [1] and Sohaib Manzoor [3]

[1] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China; khushbukhalid@hust.edu.cn (K.K.B.); khansajid@hust.edu.cn (S.K.)

[2] Faculty of Computer Science Department, Huazhong University of Science and Technology, Wuhan 430074, China

[3] School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China; sohaibmanzoor@hust.edu.cn

* Correspondence: guohuili@hust.edu.cn

Received: 14 December 2019; Accepted: 10 January 2020; Published: 16 January 2020

Abstract: Bit-level and pixel-level methods are two classifications for image encryption, which describe the smallest processing elements manipulated in diffusion and permutation respectively. Most pixel-level permutation methods merely alter the positions of pixels, resulting in similar histograms for the original and permuted images. Bit-level permutation methods, however, have the ability to change the histogram of the image, but are usually not preferred due to their time-consuming nature, which is owed to bit-level computation, unlike that of other permutation techniques. In this paper, we introduce a new image encryption algorithm which uses binary bit-plane scrambling and an SPD diffusion technique for the bit-planes of a plain image, based on a card game trick. Integer values of the hexadecimal key SHA-512 are also used, along with the adaptive block-based modular addition of pixels to encrypt the images. To prove the first-rate encryption performance of our proposed algorithm, security analyses are provided in this paper. Simulations and other results confirmed the robustness of the proposed image encryption algorithm against many well-known attacks; in particular, brute-force attacks, known/chosen plain text attacks, occlusion attacks, differential attacks, and gray value difference attacks, among others.

Keywords: image encryption; modular addition; scrambling plus diffusion (SPD); SHA-512; security; entropy

1. Introduction

1.1. Background

Recent technological advancements in networks and communication technologies have significantly improved communications. As a result, multimedia security (with respect to the communication of multimedia content) became of widespread concern, especially when the spread of multimedia content became out of control due to almost global internet access. Thus, the public is at a serious risk of confidential information leakage and has become more concerned about the security of multimedia communication in the modern, globalized world [1].

Internet technology has progressed greatly and digital image processing techniques have improved along with it, resulting in the availability of various applications for digital images. Such applications are expected to increase in leaps and bounds in the near future. Recently, image security has attracted considerable attention, as trillions of images are transmitted over the internet every day—shared through mobile phones and other digital devices, and outsourced to cloud storage. This has led to the rise of encryption as the most popular technique for protecting the privacy of users and their digital images. Various image encryption algorithms have been developed, which have utilized different kinds of techniques for this purpose [2–8].

Over the last decade, a variety of chaotic systems have been developed [9,10] for the construction of image cryptosystems, including and not limited to the 3D baker map [11], the 2D logistic map [12], the 3D cat map [13], the 2D standard map [14], the 3D LCA map [15], and the 2D henon map [16]. Moreover, some chaotic systems have been combined with other approaches for this purpose, such as DNA computing [17,18], one-time keys [19], cellular automata [20], and the perceptron model [21], which have been used for the specific objective of enhancing image cryptosystem security.

There are two main stages in chaos-based image encryption systems, with respect to their composition: permutation and diffusion. The repetition of these two stages multiple times can attain a desirable security level. The first stage (called the permutation stage) tends to decrease strong relationships between pixels that are adjacently located. Permutation methods can be classified into two further categories: bit-level and pixel-level, corresponding to the smallest processing entity in each case.

In the first category, a pixel is considered the smallest scrambling element in permutation; for example, a new block image encryption algorithm that was proposed by Wang et al. [22] used a random grouping technique to scramble the image pixels with the help of the Arnold cat map. This particular method generally eliminated the drawback of periodicity of the Arnold cat map. Another method, proposed by Hua et al. [23] known as the high-speed image scrambling method, changes the positions of rows and columns of pixels at the same time, thereby efficiently reducing strongly correlated neighboring pixels.

Parvin et al. [24] came up with and implemented a new strategy for scrambling images by shifting the pixels in rows and columns through a chaotic sequence. Their strategy is very easy to implement, along with being very efficient. Usually, pixel-based permutation uses the strategy of shuffling the image by changing pixel positions without having to modify the pixel values; thus, the histograms of the permuted images and the original images tend to be identical, in most cases, making these pixel-level based permutations susceptible to histogram attacks and known or chosen plain text attacks, if these images do not go through diffusion or have bad diffusions [25,26].

A bit is regarded as the building block and basic operating element in the second category of permutation. For this purpose, the plain image's pixel matrix is usually transformed into a binary matrix. For example, a discriminatory image encryption system based on bit-level permutation was proposed by Tao et al. [27], as each bit tends to contribute different information to an image under consideration of encrypting the first four most significant bits of each of pixel and leaving the remaining (i.e., least significant) four bits unaffected. Some other researchers [28,29] have proposed other improved schemes, in which they applied two distinct methods for permuting the most significant four bits and least significant four bits.

Liu and Wang [9] pointed out some weaknesses in these improved schemes [30], such as the schemes requiring a square original image for encryption and that they are able to permute these bits merely inside each bit-plane. High dimensional chaotic maps and bit-level permutation has also been proposed by Liu et al. for encrypting color images, but these methods tend to take more computational time than pixel-based scrambling techniques.

The fundamental characteristics of bit distribution have been analyzed by Zhang et al. [31], who proposed a multiplication and shrink scheme in the permutation phase to reduce the high correlation and dependency among significant bits and to make the distribution for each bit-plane smooth. However, such basic image characteristics at bit-level, as proposed by Zhang, have been regarded to be inapplicable to medical-related images by Chen et al. [32]. Uniform bit distribution, along with certain different image diffusion performance, could also be achieved at the same time in the permutation stage by use of a non-linear inter-pixel computing and substitution methodology.

Apart from chaos theory, various other techniques have been used to design image ciphers [33–35]. Some examples of such techniques include an image encryption scheme developed by the authors that utilizes Latin sequences for pixel permutation and substitution [36], and an innovative image encryption technique that uses a gray code for pixel permutation and a plain pixel diffusion structure for achieving the diffusion property [37].

A new hybrid encryption scheme based on FSM and cellular automata, which accompanied a DNA sequence, has also been introduced by Sajid et al. [38]. Their particular scheme provided good results, along with the concept of creating and using a local rule regarding algorithm efficiency; however, their particular technique had limitations, as its usage was confined to gray-scale images.

1.2. Contributions

After analysis of the above-mentioned research, we found that the complexity of the particular encryption technique and number of confusion–diffusion rounds are the main reasons for the high encryption times of some of the earlier image encryption algorithms. Thus, to address the problems of high complexity and encryption time in the above-mentioned algorithms, we propose a new simple, fast, and efficient image encryption algorithm, based on segmentation of the image into $m \times n$ blocks, modular addition between the pixels, and a novel method for pixel diffusion. Together, these steps increase the security level of the ciphered images against well-known cryptanalysis attacks. We have made three major contributions with this model:

First, adaptive block-based modular pixel addition is performed. In every 4×4 block, the pixels are diffused by modular addition in such a way that, for every forthcoming block (n), the pixels of the former block $(n-1)$ are added with integer values of a hexadecimal key entity to form a direct relationship between the previous block and the upcoming block.

Second, a card trick-based scrambling plus diffusion (SPD) method is proposed. For every channel, the key bit-plane can be the same or different, which can be chosen. The SPD method efficiently diffuses the pixels and reduces the correlation among the pixels, thereby enhancing the security of the key.

Third, the SHA-512 hexadecimal key (K) value is used for both modular addition and generating a seed value for the random matrix. Depending on the image size, the integer values of hexadecimal key pairs can be used in repetition or only once by taking a huge hexadecimal key (i.e., the key can be "K $\geq M \times N$" the size of an image). This also provides us a choice of whether to enlarge the key length. The rest of the paper is organized as follows: in Section 2, the preliminary work is presented. In Section 3, the proposed image encryption scheme is detailed. Our simulations and discussion are presented in Section 4, and our security analysis is in Section 5. Finally, our conclusions are given in Section 6.

2. Preliminary Work

Every decimal number can be denoted by a binary number. As an image matrix is in decimal matrix form, a binary sequence $\{ bin_{(0)} + bin_{(1)} +, \ldots, + bin_{(l-2)} + bin_{(l-1)}\}$ can be used to represent a particular image in the following way:

$$DN = \sum_{p=0}^{l-1} bin_{(p)} 2^p = bin_{(0)} 2^0 + bin_{(1)} 2^1 + \ldots + bin_{(l-1)} 2^{l-1}. \quad (1)$$

As an RGB image is comprised of three channels, each having decimal values between 0–255, its pixels can also be denoted by a binary sequence. By doing so, the RGB image encompasses 8 bit-planes for each channel, for a total of 24 bit-planes for the image [39]. In bit-plane representation form, the *i*th bit-plane contains all the *i*th binary values of every pixel in the particular image.

2.1. Scrambling Plus Diffusion (SPD)

An effective practice to diminish the correlation between neighboring pixels is scrambling. In this paper, we introduce a novel method for bit-plane scrambling plus diffusion (SPD). In the proposed method, bit-planes are scrambled, but at the same time, the pixels values are diffused by altering the binary sequence of every pixel; hence, the technique is called SPD. The proposed novel method of SPD is based on a card trick. A video of this trick can be seen in the following link: (https://www.youtube.com/watch?v=2E-FNqICDgg). The scrambling is comprised of two phases.

First, before performing the SPD and modular addition, simple scrambling is performed over the plain image to shuffle the plain pixel values. The code for ordinary scrambling can be found in the Supplementary Materials. Second, the proposed SPD method is basically designed for 8 bit-planes, but, because in a color image there are three channels and each channel has 8 bit-planes for a total of 24 bit-planes, we have a choice to either use the same key bit-plane for all three channels or different keys to enlarge the key size and make it stronger. In the procedure below, instead of elaborating a lengthy sequence combination for all 24 bit-planes, we explain the process for just one channel. First of all, we map the 8 bit-planes of a particular channel of the image with the eight variables given below:

8	7	6	5	4	3	2	1
1 1 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0 0 0	1 1 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 1	1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1 0 1 0	1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1	1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 0 1	1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 1 1 1 0 0 0	0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1 1 0 0	0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 0
rKL	rKH	rKP	rKD	rQL	rQH	rQP	rQD

If the key bit-plane is **KL**, the arrangement will be as follows.

8	4	2	6	3	1	7	5
1 1 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0 0 0	1 1 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 1	1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1 0 1 0	1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1	1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 0 1	1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 1 1 1 0 0 0	0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1 1 0 0	0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 0
rKL	rQL	rQP	rKP	rQH	rQD	rKH	rKD

If the key bit-plane is **KH**, the arrangement will be as follows.

7	3	5	1	2	4	8	6
1 1 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0 0 0	1 1 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 1	1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1 0 1 0	1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1	1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 0 1	1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 1 1 1 0 0 0	0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1 1 0 0	0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 0
rKH	rQH	rKD	rQD	rQP	rQL	rKL	rKP

If the key bit-plane is **KP**, the arrangement will be as follows.

6	2	8	4	1	3	5	7
1 1 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 1 1 1 0 0 0	1 1 0 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 0 1 1 0 0 1	1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1 0 1 0	1 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 1 1 1 1 0 0 1	1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 0 0 1	1 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 1 1 1 0 0 0	0 1 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1 1 0 0	0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1 0 1 0 1 0
rKP	rQP	rKL	rQL	rQD	rQH	rKD	rKH

If the key bit-plane is **KD**, the arrangement will be as follows.

5	1	7	3	2	4	6	8
1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	0 1 0 0	0 1 0 1
0 0 0 1	0 0 0 1	1 0 0 1	1 0 0 1	0 0 1 1	0 0 0 1	0 0 0 1	0 0 0 0
0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 0	0 1 0 1	0 1 0 1	0 1 0 1
0 1 1 1	0 1 1 1	0 1 0 1	0 1 0 1	0 0 1 1	0 1 0 1	0 1 0 1	0 1 1 1
1 1 1 1	1 1 0 1	1 1 0 1	1 1 1 1	0 1 1 1	1 0 1 1	1 1 0 1	1 1 1 0
1 0 0 0	1 0 0 1	1 0 1 0	1 0 0 1	1 0 0 1	1 0 0 0	1 1 0 0	1 0 1 0
rKD	rQD	rKH	rQH	rQP	rQL	rKP	rKL

If the key bit-plane is **QL**, the arrangement will be as follows.

4	8	2	6	7	5	3	1
1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	0 1 0 0	0 1 0 1
0 0 0 1	0 0 0 1	1 0 0 1	1 0 0 1	0 0 1 1	0 0 0 1	0 0 0 1	0 0 0 0
0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 0	0 1 0 1	0 1 0 1	0 1 0 1
0 1 1 1	0 1 1 1	0 1 0 1	0 1 0 1	0 0 1 1	0 1 0 1	0 1 0 1	0 1 1 1
1 1 1 1	1 1 0 1	1 1 0 1	1 1 1 1	0 1 1 1	1 0 1 1	1 1 0 1	1 1 1 0
1 0 0 0	1 0 0 1	1 0 1 0	1 0 0 1	1 0 0 1	1 0 0 0	1 1 0 0	1 0 1 0
rQL	rKL	rQP	rKP	rKH	rKD	rQH	rQD

If the key bit-plane is **QH**, the arrangement will be as follows.

3	7	1	5	8	6	4	2
1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	0 1 0 0	0 1 0 1
0 0 0 1	0 0 0 1	1 0 0 1	1 0 0 1	0 0 1 1	0 0 0 1	0 0 0 1	0 0 0 0
0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 0	0 1 0 1	0 1 0 1	0 1 0 1
0 1 1 1	0 1 1 1	0 1 0 1	0 1 0 1	0 0 1 1	0 1 0 1	0 1 0 1	0 1 1 1
1 1 1 1	1 1 0 1	1 1 0 1	1 1 1 1	0 1 1 1	1 0 1 1	1 1 0 1	1 1 1 0
1 0 0 0	1 0 0 1	1 0 1 0	1 0 0 1	1 0 0 1	1 0 0 0	1 1 0 0	1 0 1 0
rQH	rKH	rQD	rKD	rKL	rKP	rQL	rQP

If the key bit-plane is **QP**, the arrangement will be as follows.

2	6	4	8	5	7	1	3
1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	0 1 0 0	0 1 0 1
0 0 0 1	0 0 0 1	1 0 0 1	1 0 0 1	0 0 1 1	0 0 0 1	0 0 0 1	0 0 0 0
0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 0	0 1 0 1	0 1 0 1	0 1 0 1
0 1 1 1	0 1 1 1	0 1 0 1	0 1 0 1	0 0 1 1	0 1 0 1	0 1 0 1	0 1 1 1
1 1 1 1	1 1 0 1	1 1 0 1	1 1 1 1	0 1 1 1	1 0 1 1	1 1 0 1	1 1 1 0
1 0 0 0	1 0 0 1	1 0 1 0	1 0 0 1	1 0 0 1	1 0 0 0	1 1 0 0	1 0 1 0
rQP	rKP	rQL	rKL	rKD	rKH	rQD	rQH

If the key bit-plane is **QD**, the arrangement will be as follows.

1	5	3	7	6	8	2	4
1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	1 1 0 0	0 1 0 0	0 1 0 1
0 0 0 1	0 0 0 1	1 0 0 1	1 0 0 1	0 0 1 1	0 0 0 1	0 0 0 1	0 0 0 0
0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 1	0 1 0 0	0 1 0 1	0 1 0 1	0 1 0 1
0 1 1 1	0 1 1 1	0 1 0 1	0 1 0 1	0 0 1 1	0 1 0 1	0 1 0 1	0 1 1 1
1 1 1 1	1 1 0 1	1 1 0 1	1 1 1 1	0 1 1 1	1 0 1 1	1 1 0 1	1 1 1 0
1 0 0 0	1 0 0 1	1 0 1 0	1 0 0 1	1 0 0 1	1 0 0 0	1 1 0 0	1 0 1 0
rQD	rKD	rQH	rKH	rKP	rKL	rQP	rQL

3. Image Encryption

The general structural diagram of our proposed algorithm is given in Figure 1. The proposed image encryption is comprised of two stages: the first is modular addition and the second is SPD over the binary bit-planes for each of the RGB channels. The detailed encryption steps are discussed below.

Step 1: Obtain an image of size $H \times W$ and take the SHA-512 to get its 128 bit hexadecimal key; arrange the hexadecimal key, and get the decimal value of each key entity.

Step 2: Take the seed value from the hash key value to generate a random matrix equal in size to the input image ($H \times W$). Perform the scrambling operation.

Step 3: Divide the scrambled image into RGB channels, as shown below:

$$Image_{RGB} \leftrightarrow \begin{cases} RedChannel. \\ GreenChannel. \\ BlueChannel. \end{cases}$$

Step 4: Subdivide each channel, and the random matrix, into 4 × 4 blocks as follows:

$$Red = B_{r1}, B_{r2}, B_{r3}, \ldots, B_{r(n-1)}, B_{rn} \tag{2}$$

$$Green = B_{g1}, B_{g2}, B_{g3}, \ldots, B_{g(n-1)}, B_{gn} \tag{3}$$

$$Blue = B_{b1}, B_{b2}, B_{b3}, \ldots, B_{b(n-1)}, B_{bn} \tag{4}$$

$$Random_{matrix} = B_{RM1}, B_{RM2}, B_{RM3}, \ldots, B_{RM(n-1)}, B_{RMn}, \tag{5}$$

where $B_i = (1, 2, 3, \ldots, n)$ represents the particular block; r, g, and b represent the red, green, and blue channels, respectively; $B_{RM(i)} = (1, 2, 3, \ldots, n)$ represents the particular random matrix block; and n is the total number of blocks in the particular image.

Step 5: Take the red channel and perform modular addition over the pixels in every block to encrypt the blocks, as described below:

$$Red = \begin{cases} E_{r1} = [B_{RM1} + Br_1 + IN_{k1}]mod256 \\ E_{r2} = [B_{RM2} + Br_2 + E_{r1} + IN_{k2}]mod256 \\ E_{r3} = [B_{RM3} + Br_3 + E_{r2} + IN_{k3}]mod256 \\ . \\ . \\ . \\ E_{r(n-1)} = [B_{RM(n-1)} + Br_{(n-1)} + E_{r(n-2)} + IN_{k(n-1)}]mod256 \\ E_{rn} = [B_{RM(n)} + Br_n + E_{r(n-1)} + IN_{kn}]mod256, \end{cases} \tag{6}$$

where E_{rn} represents an encrypted pixel of the red channel block, IN_{kn} represents the integer value of a particular hexadecimal key entity, and mod is the modular function, which returns a pixel value between 0 and 255.

Step 6: For the green channel, repeat Step 5 over all blocks to encrypt the green blocks, as described below:

$$Green = \begin{cases} E_{g1} = [B_{RM1} + Bg_1 + IN_{k1}]mod256 \\ E_{g2} = [B_{RM2} + Bg_2 + E_{g1} + IN_{k2}]mod256 \\ E_{g3} = [B_{RM3} + Bg_3 + E_{g2} + IN_{k3}]mod256 \\ . \\ . \\ E_{g(n-1)} = [RM_{(n-1)} + Br_{(n-1)} + E_{g(n-2)} + IN_{k(n-1)}]mod256 \\ E_{gn} = [B_{RM(n)} + Br_n + E_{g(n-1)} + IN_{kn}]mod256. \end{cases} \tag{7}$$

Perform the modular addition on the blue channel to encrypt its pixels, as described in Step 5. In Equations (6) and (7) in each step, sum the original block, random block, key, and previous encrypted block resemble the cipher block chaining mode [40].

Step 7: Join all the encrypted sub-blocks to obtain an image matrix and divide each channel into binary bit-planes, as given below:

$$Image_{RGB} = \begin{cases} Red_{i,j} = Bin_{r1}, Bin_{r2}, Bin_{r3}, \ldots, Bin_{r8} \\ Green_{i,j} = Bin_{g1}, Bin_{g2}, Bin_{g3}, \ldots, Bin_{g8} \\ Blue_{i,j} = Bin_{b1}, Bin_{b2}, Bin_{b3}, \ldots, Bin_{b8}, \end{cases} \tag{8}$$

where Bin_{ij} represents the *i*th bit window of particular channel.

Step 8: Select the key bit-planes (i.e., *KL*, *QL*, *KP*, and so on) for each of the RGB channels and perform bit-plane scrambling, as described in Section 2.1.

Step 9: Join all the bit-planes to the respective sequences to obtain the cipher image:

$$Cipher_{(M,N,3)} = C^{red}_{(i,j)}; C^{green}_{(i,j)}; C^{blue}_{(i,j)}, \tag{9}$$

where *M* and *N* represent the height and width of the cipher image.

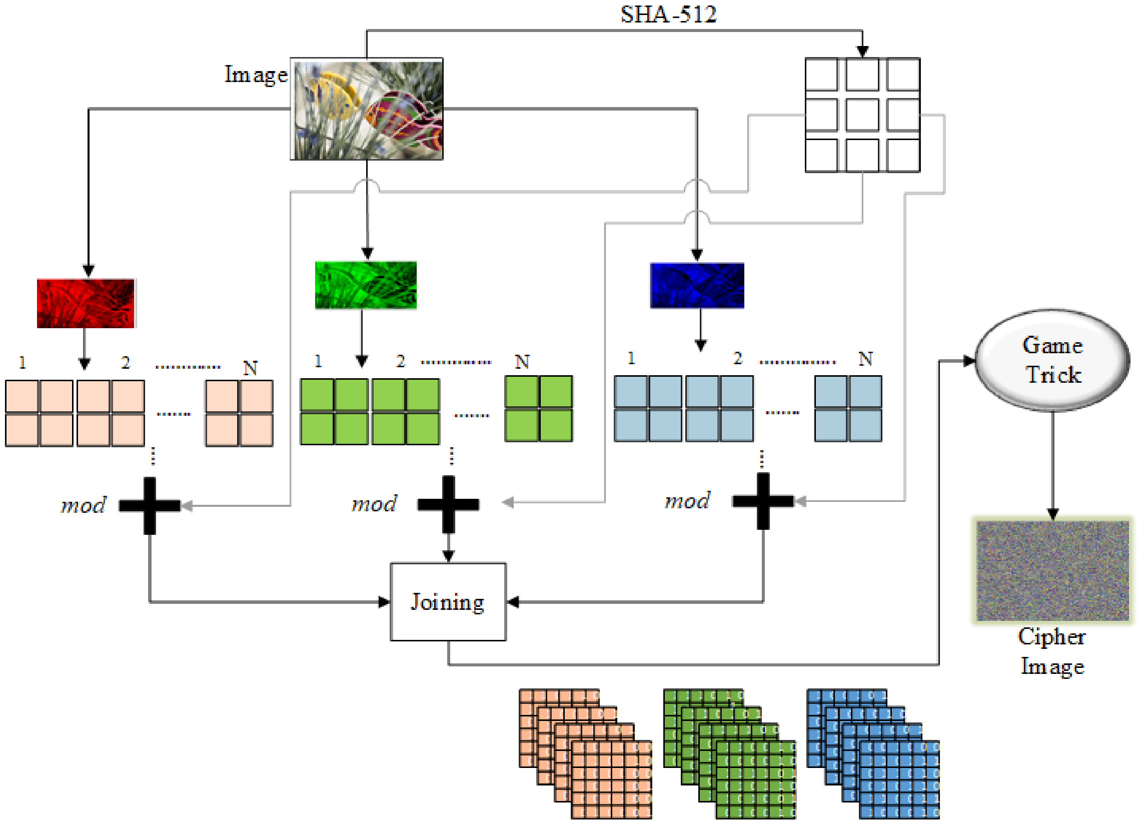


Figure 1. General block diagram.

4. Simulation Results and Discussion

Simulation was carried out using the JetBrains PyCharm Edu 2019.1.1 × 64 software installed on a PC with 4 GB memory, an Intel Core I5 Processor, and the Windows 10 Enterprise operating system. For the histograms, Matlab R2017a was used.

The main test image (fruits), along with its cipher and decrypted image, are displayed in Figure 2, while all the test images that we used are shown in Figure 3. The key and other experimental parameters are given in Table 1.

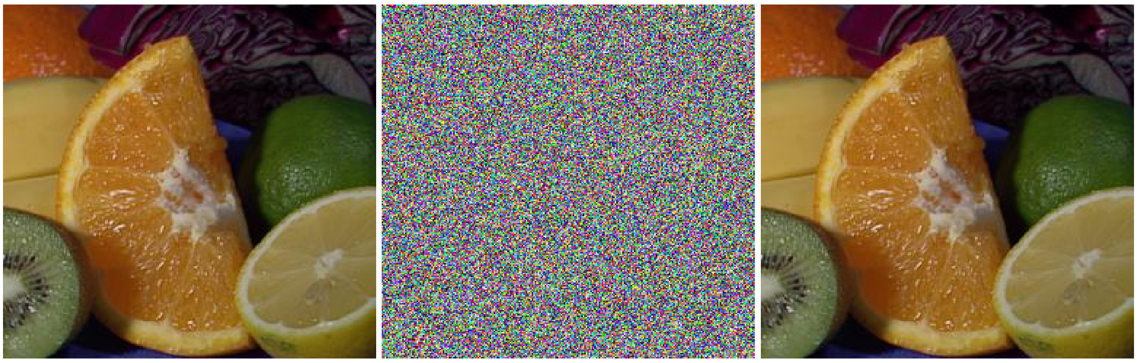
Before performing decryption, the secret key needs to be transferred to the receiver through a secure channel, and the three key bit-planes for the red, green, and blue channels. In decryption, all steps are performed in the reverse direction, while the modular addition phase equation in the decryption process is as follows:

$$Channel_{R,G,B} = \begin{cases} PI_{c(1)} = [E_{c(1)} - B_{RM1} - IN_{k1}]mod256 \\ PI_{c(2)} = [E_{c(2)} - B_{RM2} - PI_{c(1)} - IN_{k2}]mod256 \\ PI_{c(3)} = [E_{c(3)} - B_{RM3} - PI_{c(2)} - IN_{k3}]mod256 \\ . \\ . \\ . \\ PI_{c(n-1)} = [E_{c(n-1)} - B_{RM(n-1)} - PI_{c(n-2)} - IN_{k(n-1)}]mod256 \\ PI_{c(n)} = [E_{c(n)} - B_{RM(n)} - PI_{c(n-1)} - IN_{kn}]mod256, \end{cases} \tag{10}$$

where $PI_{c(i)}$ represents the plain image blocks of the channel $c \in R, G, B$ and $E_{c(i)}$ represents an encrypted block of the particular channel.

Table 1. Key and experimental parameters.

Terms	Key Parameters & Values
512-bit Hexadecimal key	E9BAD688008A995B22723D5565411F91769E9E72F6407B148D84B2DEDF1DF3996BFA27 5EBE13B9B9CA27D82C6440764A817BDA1A1A54DE542FD427AB39ADC292
Seed Value	$int(K, 16)$ = 1224142479849059493616438967241173867216550529300133027919602126895 5249538595902639015802274554590625310127458155867404943493171251136512933387 374143849106
Integer values	int(K[0 : 1], 16) = 14, int(K[1 : 2], 16) = 9,. . . , int(K[(n-1) : n], 16) = 2
Key bit-planes	Red$_{QD}$; Green$_{KH}$; Blue$_{QL}$



(**a**) Plain Image (**b**) Ciphered Image (**c**) Decrypted image

Figure 2. Encryption and decryption results. (**a**) Plain image of fruits; (**b**) cipher image of fruits; (**c**) decrypted image of fruits.

(**a**) Lena (**b**) Baby (**c**) Apricot (**d**) Fruits

(**e**) Tiffany (**f**) Nike (**g**) Panda (**h**) Vegetables

(**i**) Pepper (**j**) House

Figure 3. All test images.

5. Security Analysis and Tests

5.1. Key Space and Key Sensitivity

First, the key space of an encryption algorithm should be appropriate. In the proposed algorithm, we use SHA-512, which gives an output of 512 bits in hexadecimal form. In SPD, three key bit-planes and their resultant possible combinations, and the long seed value of the hexadecimal key for the random matrix, can be included in the key space. Together, these form the key space of the proposed algorithm; 2^{512} means 1.340×10^{154}. Thus, it has apt size and great capacity for protecting from brute-force attacks [41]. Secondly, the key must be tremendously sensitive to bit changes. If the key is not subtle enough, then a similar key may also be able to decrypt the cipher image [42,43].

Thus, to test the key sensitivity, we checked the number of bit change rate (NBCR) [44]. The NBCRs of the two images B_1 and B_2 are defined as

$$NBCR(B_1, B_2) = \frac{ham(B_1, B_2)}{T_b}, \tag{11}$$

where ham (B_1, B_2) indicates the Hamming distance between B_1 and B_2, whereas T_b is the overall number of bits from B_1 or B_2. If the acquired NBCR is near to 50%, then B_1 and B_2 are entirely dissimilar images, with no relationship.

$$K_1 = E9BAD688008A995B2.......B39ADC292.$$

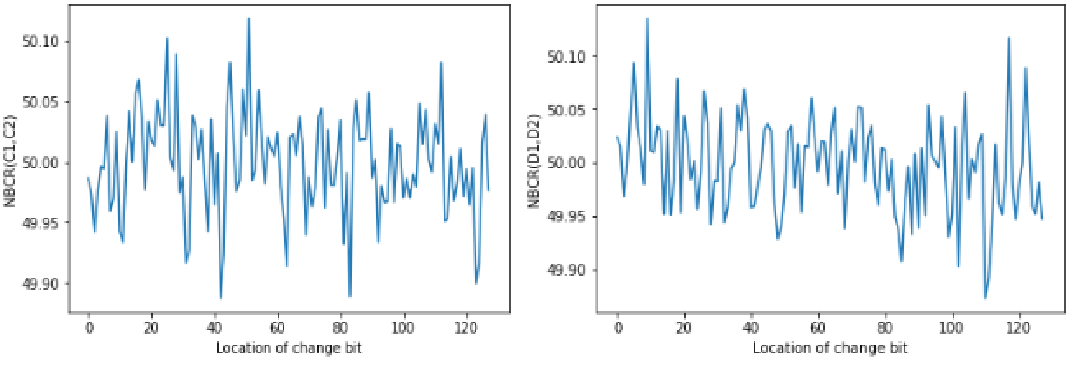
For the key sensitivity of the proposed algorithm, our experiments were designed as follows:

1st: Alter one bit of K_1 to obtain K_2.

2nd: Using K_1 and K_2, encrypt the plain-image P and generate two cipher-images C_1 and C_2. Then, compute their NBCR.

3rd: Decrypt the particular cipher-image C_1 by using K_1 and K_2 to get two decrypted images D_1 and D_2. Then, compute the NBCR.

Figure 4 demonstrates the key sensitivity exploration result of the secret key for the fruit test image. We can observe from the graph that any change in the secret key will result in obtaining two cipher-images in the encryption phase and the two decrypted results in the decryption phase in entirely different forms. This shows that our proposed algorithm is extremely sensitive to deviation of the cipher key.



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

Figure 4. Key sensitivity analysis. (**a**) Number of bit change rate (NBCR) between C1 and C2; (**b**) NBCR between D1 and D2.

Furthermore, if the encryption algorithm has a less sensitive key, then any insignificant change in the real secret key will also lead to proper decryption of the actual image. Thus, a secret key of algorithm may seem corrupted, and as a result, the real key space will be smaller than the theoretical one [45,46] .

Therefore, in order to further analyze the key sensitivity of our algorithm, we performed another experiment over the fruit image, in the encryption and decryption steps, by altering only one bit in the actual key, as shown below:

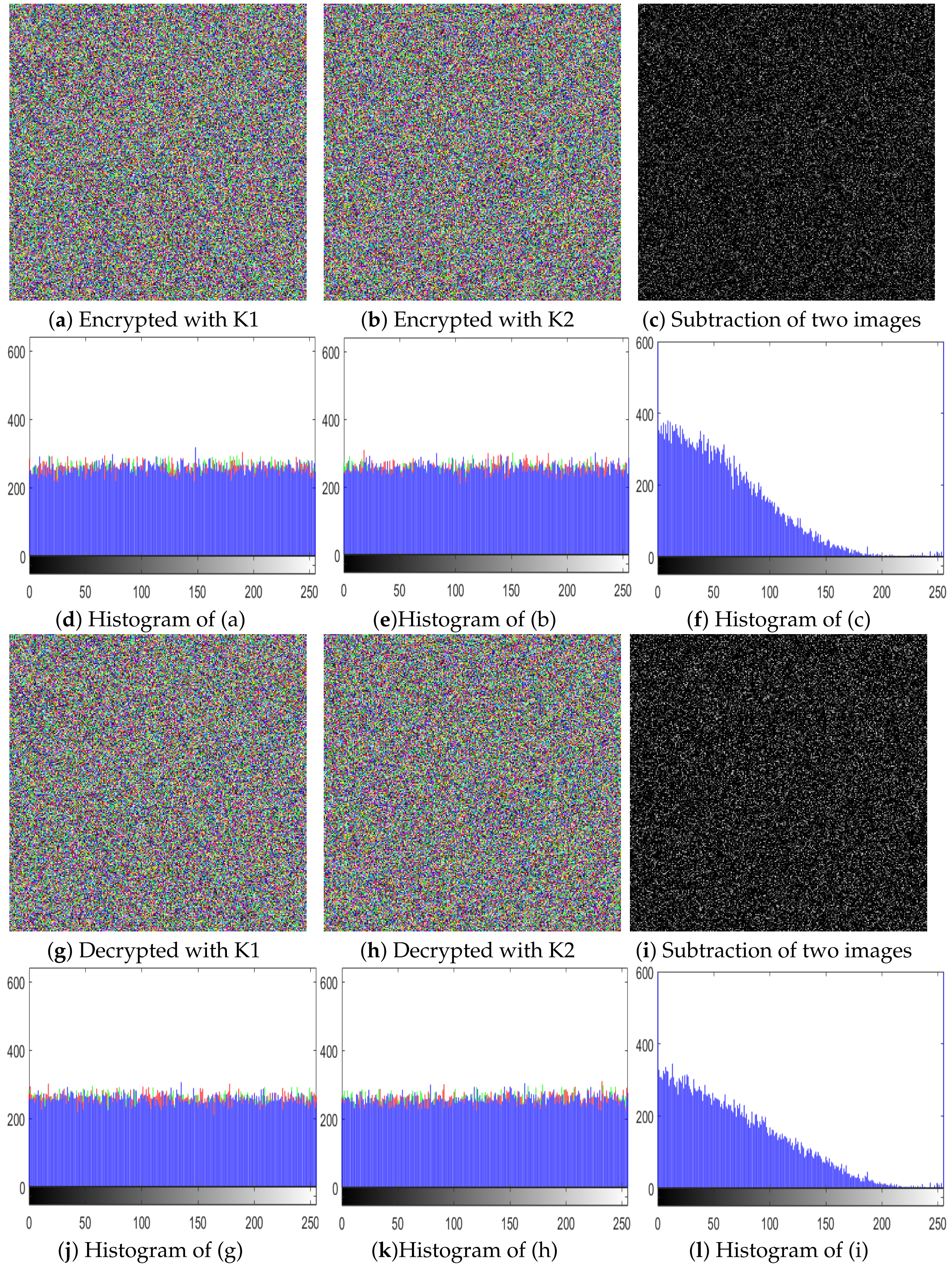
$\mathbf{Key_o}$ = "E 9B AD 68 80 08 A9 95 B2 27 23 D5 56 54 11 F9 17 69 E9 E7 2F 64 07 B1 48 D8 4B 2D ED F1 DF 39 96 BF A2 75 EB E1 3B 9B 9C A2 7D 82 C6 44 07 64 A8 17 BD A1 A1 A5 4D E5 42 FD 42 7A B3 9A DC 29 2."

$\mathbf{Key_1}$ = "F 9B AD 68 80 08 A9 95 B2 27 23 D5 56 54 11 F9 17 69 E9 E7 2F 64 07 B1 48 D8 4B 2D ED F1 DF 39 96 BF A2 75 EB E1 3B 9B 9C A2 7D 82 C6 44 07 64 A8 17 BD A1 A1 A5 4D E5 42 FD 42 7A B3 9A DC 29 2."

$\mathbf{Key_2}$ = "E 9B AD 68 80 08 A9 95 B2 27 23 D5 56 54 11 F9 17 69 E9 E7 2F 64 07 B1 48 D8 4B 2D ED F1 DF 39 96 BF A2 75 EB E1 3B 9B 9C A2 7D 82 C6 44 07 64 A8 17 BD A1 A1 A5 4D E5 42 FD 42 7A B3 9A DC 29 3."

$\mathbf{Key_o}$ is the actual key, while $\mathbf{Key_1}$ and $\mathbf{Key_2}$ have only a one-bit key difference from the left-most side and the right-most side of the bit, respectively.

We performed experiments for both the encryption and decryption steps. In both cases, we selected keys with only a one-bit difference from the real key. K_1 is the key having a one-bit difference from the left side, while K_2 has a one-bit difference from the right side. Figure 5a,b shows the encrypted images with K_1 and K_2, respectively. Figure 5c is the subtraction result of (a) and (b). Figure 5d–f are the histograms of Figure 5a–c, respectively. Similarly, on the decryption side, we attempted to decrypt the actual cipher image with a key having one bit difference from the original key; the results are listed in Figure 5g,h. Figure 5i is the subtracted image of (g) and (h), and the respective histograms are shown in in Figure 5j–l. The results show that our proposed algorithm is highly sensitive to the key, and even a bit difference in the key will result in a totally noise-like image.



(**a**) Encrypted with K1 (**b**) Encrypted with K2 (**c**) Subtraction of two images

(**d**) Histogram of (a) (**e**)Histogram of (b) (**f**) Histogram of (c)

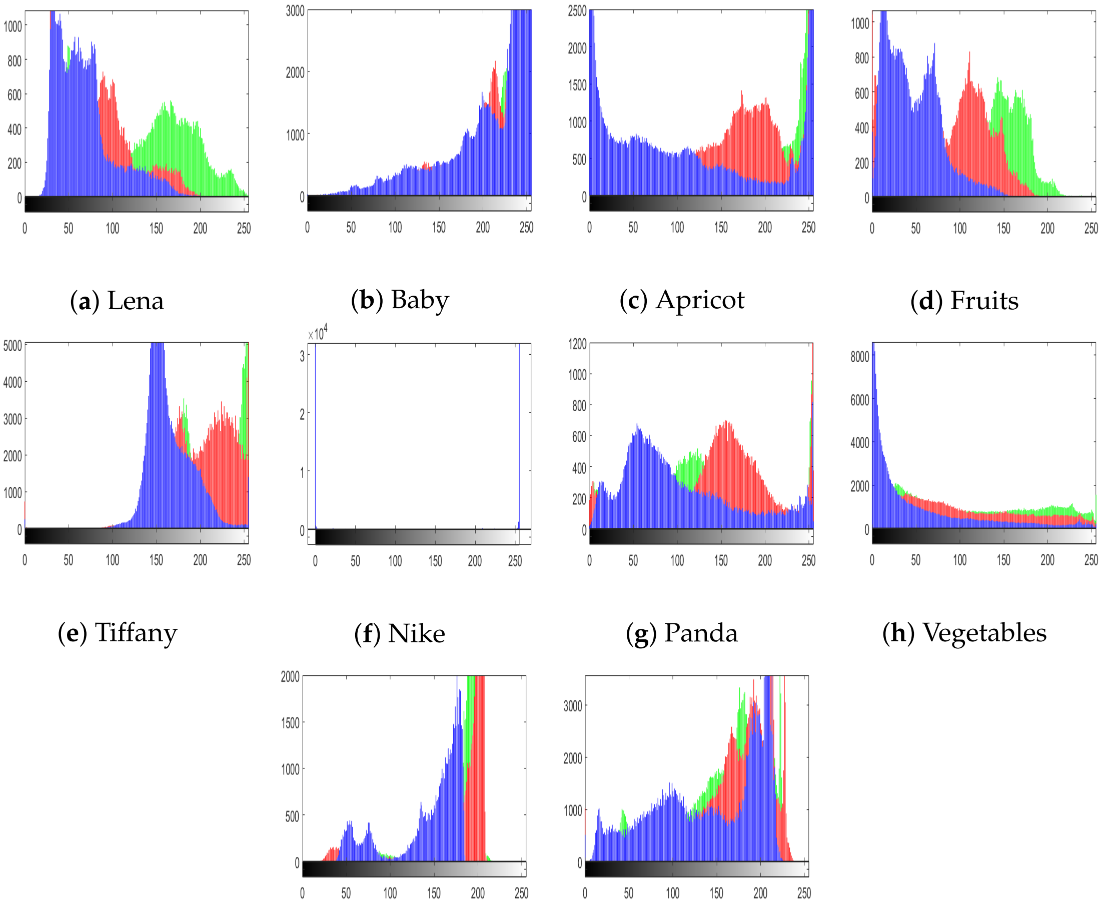
(**g**) Decrypted with K1 (**h**) Decrypted with K2 (**i**) Subtraction of two images

(**j**) Histogram of (g) (**k**)Histogram of (h) (**l**) Histogram of (i)

Figure 5. Key sensitivity test: (**a**) encrypted with K1; (**b**) encrypted with K2; (**c**) subtraction of (**a**,**b**); (**d**) histogram of (**a**); (**e**) histogram of (**b**); (**f**) histogram of (**c**); (**g**) decrypted with K1; (**h**) decrypted with K2; (**i**) subtraction of (**g**,**h**); (**j**) histogram of (**g**); (**k**) histogram of (**h**); (**l**) histogram of (**i**).

5.2. Histogram Analysis

The histograms of the images reveal the dissemination of the pixel values. Generally, an interloper can recuperate important information about the cipher image from its (inconsistent) histogram. Therefore, to thwart an impostor from recuperating useful information, it is essential that the histograms of images generated by the cipher show no numerical resemblance with the plain image; in addition, it must have an unvarying distribution of pixels. Figure 6a–j shows the histograms of all the test images that we used, and Figure 7 shows the histograms of corresponding cipher-generated images in the sequence of all test images. The histogram of main test image (fruit) is given in Figure 8. Figure 8a–c shows the histograms of the plain channels of the fruit image, and Figure 8d–f shows the histograms of the blue, green, and red channels, respectively, of the image encrypted by the proposed algorithm. It can be observed that the histograms of the cipher-generated images are nearly flat and uniform.

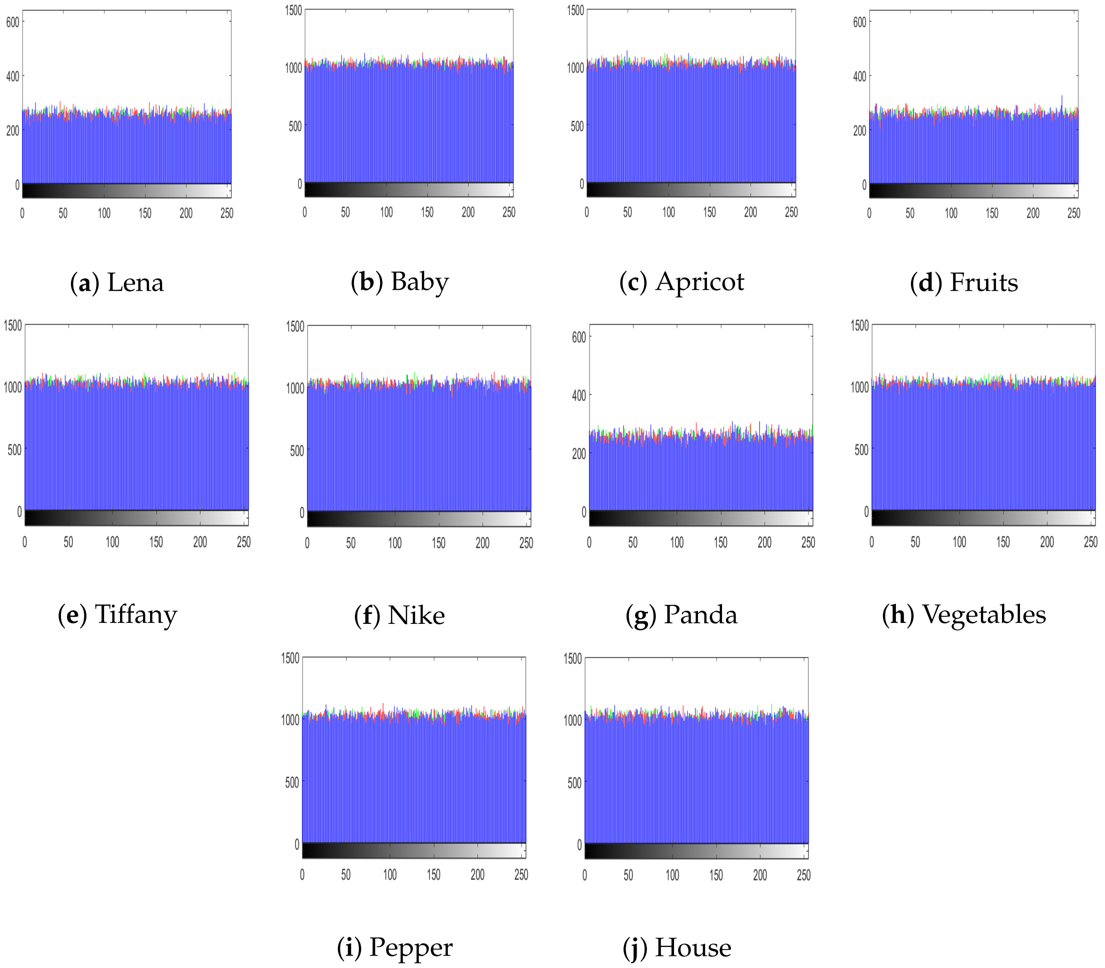


(**a**) Lena (**b**) Baby (**c**) Apricot (**d**) Fruits

(**e**) Tiffany (**f**) Nike (**g**) Panda (**h**) Vegetables

(**i**) Pepper (**j**) House

Figure 6. Histograms of plain test images.

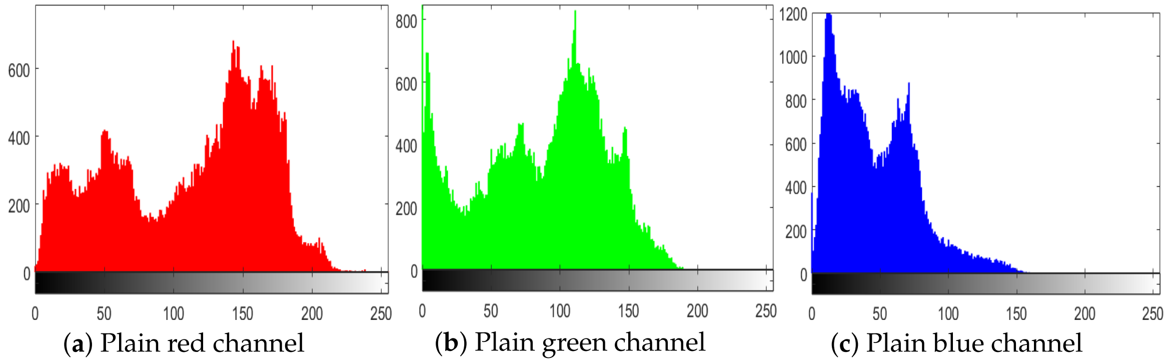


(**a**) Lena (**b**) Baby (**c**) Apricot (**d**) Fruits

(**e**) Tiffany (**f**) Nike (**g**) Panda (**h**) Vegetables

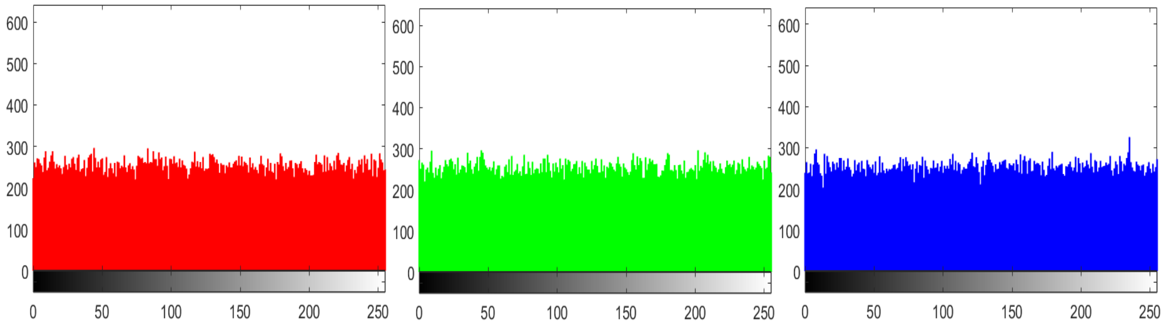
(**i**) Pepper (**j**) House

Figure 7. Histograms of cipher images.



(**a**) Plain red channel (**b**) Plain green channel (**c**) Plain blue channel

Figure 8. *Cont*.



(**d**) Cipher red channel (**e**) Cipher green channel (**f**) Cipher Blue Channel

Figure 8. Histograms of RGB channels of fruits. (**a**) Plain red channel; (**b**) plain green channel; (**c**) plain blue channel; (**d**) ciphered red channel; (**e**) ciphered green channel; (**f**) ciphered blue channel.

Likewise, by calculating the variances of the histograms, we assessed the consistency of the encrypted images. If the variance of an encrypted images is smaller, then the uniformity (and the security level) of the particular image encryption algorithm is greater [45,46].

$$H_{variance}(W) = \frac{1}{n^2} \sum_{r=1}^{n} \sum_{c=1}^{n} \frac{1}{2}(w_r - w_c)^2, \tag{12}$$

where $W = w_1, w_2, \ldots, w_{256}$ represent the vectors of the histogram values, and w_r and w_c are pixels with gray values equal to r and c, correspondingly.

Table 2 lists the variances of some test images. From the table values, it can be concluded that the variances of the plain images were too high, but for the ciphered images, they were very low. For example, the average variance of the cipher image of Lena (256 × 256) was 229, but it was 81,231 for the plain Lena image. The given comparison also shows that for the majority of test images, the histogram variances of images encrypted by the proposed algorithm were lower than those in [41,43]. On the basis of this comparison, we can say that our proposed algorithm has the ability to make the encryption algorithm more secure.

Table 2. Histogram variance comparison.

Images	**Plain**			**Proposed**			**[42] Cipher**			**[44] Cipher**		
	Red	**Green**	**Blue**	**Red**	**Green**	**Blue**	**Red**	**Green**	**Blue**	**Red**	**Green**	**Blue**
Lena	123,072.500	87,100.835	33,522.734	219.513	231.052	221.119	247.78	279.62	265.71	527.32	504.75	501.68
Female	620,306.750	860,899.312	790,776.56	262.265	233.211	255.813	280.64	280.46	230.42	-	-	-
Couple	289,630.656	337,863.062	210,359.812	275.095	243.858	238.196	284.35	247.37	260.76	-	-	-
House	99,2034.125	1,330,180.125	768,126.75	1136.569	1119.742	1060.540	1070.2	1231.2	941.65	-	-	-
Tree	129,825.531	57,011.605	81,373.710	250.877	215.093	258.754	282.81	254.87	225.79	-	-	-
Bean	129,765.984	349,251.718	537,500.062	214.782	259.715	238.384	232.98	279.61	245.61	-	-	-

5.3. Pixel Correlation Analysis

To check the relationships amongst neighboring pixels in the encrypted and plain images, a pixel correlation analysis test was performed. For an encryption algorithm, it is essential to diminish the correlation amongst neighboring pixels, in order to prevent the leakage of authentic information.

The correlation coefficient $C_{orr.}(i,j)$ between the adjacent pixels of an image can be computed by the following equations:

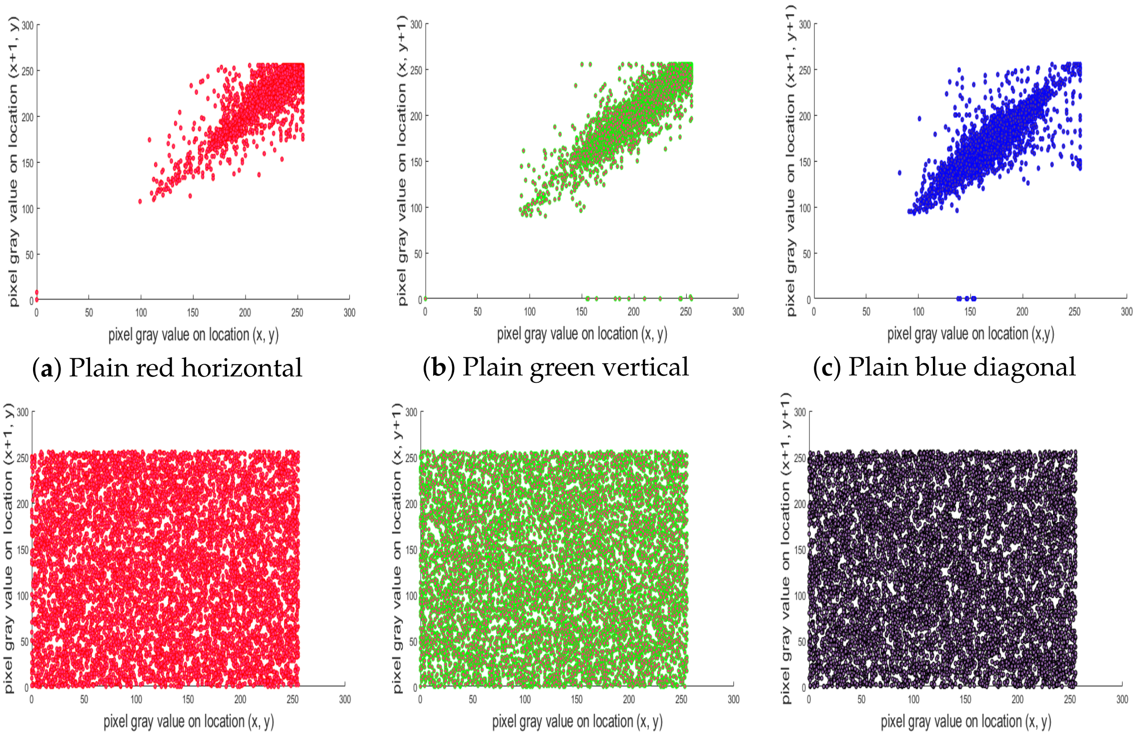
$$C(\hbar) = \frac{1}{L} \sum_{i=1}^{L} \hbar_i \tag{13}$$

$$D(\hbar) = \frac{1}{L} \sum_{i=1}^{L} (\hbar_i - E(\hbar))^2 \tag{14}$$

$$C_{var.}(\hbar, w) = \frac{1}{L}\sum_{i=1}^{L}(\hbar_i - E(\hbar))(w_i - E(w)) \tag{15}$$

$$C_{r(\hbar,w)} = \frac{C_{var.}(\hbar, w)}{\sqrt{D(\hbar) \times D(w)}}, \tag{16}$$

where ($\hbar$, w) is the gray value of neighboring pixels, L is the entire number of pixels that have been selected from an image, $C_{var.}(\hbar, w)$ is the covariance, $D(w)$ is the variance, and $E(\hbar)$ and $E(w)$ denote the expectation of the variables $\hbar$ and w, respectively. Table 3 lists the correlations of 1000 pixels in the Lena image. The average value of the pixel correlation in the horizontal, vertical, and diagonal directions is −0.0092375. Comparison with earlier algorithms indicates that the correlation between neighboring pixels in our ciphered image is less than those in [47–49], while it was comparable to those of [50,51]. In Table 4, the pixel correlation values of the images of USC-SIPI database, as well as other test images, are given. From the table values, we can see that the correlation in the plain image is too high (almost equal to "1" for each channel), but in the images encrypted by our proposed algorithm, it is very low. This confirms the satisfactory security level of our algorithm. Figure 9a–f shows plots of the correlation of two neighboring pixels in the plain image "tree" and its cipher-generated image. The plot shows the red channel horizontally, the green channel vertically, and the blue channel diagonally distributed. These results also show that the correlation is highly reduced in the cipher image, and thus, we can say that attackers cannot obtain any information from the cipher image in this way.



(**a**) Plain red horizontal (**b**) Plain green vertical (**c**) Plain blue diagonal

(**d**) Cipher red horizontal (**e**) Cipher green vertical (**f**) Cipher blue diagonal

Figure 9. Pixel correlation of test image tree. (**a**) Plain red channel pixels, horizontal; (**b**) plain green channel pixels, vertical; (**c**) plain blue channel, diagonal; (**d**) ciphered red channel pixels, horizontal; (**e**) ciphered green channel pixels, vertical; (**f**) ciphered blue channel pixels, diagonal.

Table 3. Pixel correlation between 1000 random pixels of Lena.

Algorithm	Horizontal	Vertical	Diagonal
[47]	−0.0098	−0.0050	−0.0013
[50]	−0.0023	0.0019	−0.0034
[51]	0.0020	−0.0007	−0.0014
[48]	−0.0237	−0.0178	−0.0284
[49]	0.0080	0.0098	−0.0058
Proposed	−0.0042707	−0.0032498	−0.020192

Table 4. Pixel correlation of USC−SIPI.

Images	Channel	Plain			Ciphered		
		Horizontal	Vertical	Diagonal	Horizontal	Vertical	Diagonal
4.1.01	Red	0.97318	0.96395	0.94653	−0.0053517	0.010961	−0.0049967
	Green	0.9692	0.96409	0.94567	−0.0047375	−0.0012887	0.0059078
	Blue	0.95666	0.94848	0.93846	0.0021301	0.0023748	−0.00036386
4.1.02	Red	0.94829	0.95259	0.91853	−0.00331	−0.0041589	0.016235
	Green	0.9237	0.95515	0.89936	−0.012505	−0.00032095	0.0032421
	Blue	0.91566	0.94324	0.88315	0.018044	−0.00053454	0.019847
4.1.03	Red	0.98061	0.92609	0.90982	−0.01801	−0.012732	−0.01593
	Green	0.97722	0.91153	0.88157	0.0009692	0.0024241	0.0076471
	Blue	0.97383	0.90929	0.89348	−0.0072574	0.0084441	0.0036835
4.1.04	Red	0.97653	0.98664	0.97151	−0.0063601	0.021812	−0.020985
	Green	0.96491	0.98351	0.95113	−0.0085227	−0.01142	−0.0089817
	Blue	0.95215	0.97005	0.93335	0.0018656	0.0041814	−0.011218
4.1.05	Red	0.96786	0.93611	0.91023	0.0025253	−0.0070104	−0.001377
	Green	0.98024	0.95152	0.92792	0.0015667	−0.013295	0.0082844
	Blue	0.98215	0.97459	0.96133	−0.0077072	−0.0074648	−0.0048926
4.1.06	Red	0.95813	0.93145	0.91129	−0.0092083	−0.01433	0.0011584
	Green	0.96761	0.94832	0.92968	−0.0020563	0.0052499	0.0073594
	Blue	0.96175	0.94106	0.92656	0.00096206	−0.0032515	0.013931
4.1.07	Red	0.97582	0.97547	0.94929	0.0032059	−0.0079411	−0.0082824
	Green	0.9777	0.98171	0.95789	−0.0037171	−0.0054693	0.0013666
	Blue	0.98938	0.98803	0.97825	−0.010115	0.00021278	−0.00074795
4.1.08	Red	0.97278	0.97356	0.95062	−0.0035631	−0.0040268	0.004974
	Green	0.97332	0.97477	0.94604	−0.002521	−0.0092897	−0.0079288
	Blue	0.97943	0.98097	0.95977	0.0090987	−0.0020318	−0.0036913
Apricot	Red	0.98386	0.9863	0.96945	−0.013663	−0.0018757	−0.0054083
	Green	0.97884	0.98512	0.96538	−0.010663	0.0015055	0.0023429
	Blue	0.98725	0.99154	0.98349	0.0048932	−0.0057104	−0.011769
Vegetables	Red	0.97887	0.98013	0.96109	0.001243	−0.0032611	0.00075363
	Green	0.9775	0.97953	0.95948	−0.00075568	−0.0024519	0.0044758
	Blue	0.97225	0.97092	0.9473	−0.0010255	−0.0049976	0.0015237
Fruits	Red	0.9868	0.98516	0.97407	−0.0044958	−0.0014318	0.0047764
	Green	0.98084	0.98078	0.96405	−0.0057875	−0.00091906	−0.0066637
	Blue	0.95394	0.9494	0.90847	−0.0040615	0.0012303	−0.005508
baby	Red	0.97834	0.97816	0.96278	−0.006159	0.0021686	0.0025641
	Green	0.99107	0.99428	0.97854	−0.0021032	0.0011388	0.0099095
	Blue	0.99371	0.99472	0.98814	−0.0036711	−0.012494	0.0046702
Panda	Red	0.95176	0.96553	0.93162	0.005131	−0.00076785	−0.0049536
	Green	0.95216	0.96437	0.93067	0.0078787	−0.00079504	0.00027378
	Blue	0.95543	0.97087	0.94266	7.0511×10^{-5}	−0.010969	0.01059
Nike	Red	0.98682	0.9922	0.97288	−0.0043816	−0.016514	0.0066848
	Green	0.98852	0.99104	0.97353	0.0023924	0.0017956	−0.0034512
	Blue	0.98676	0.9905	0.9719	0.0087522	0.0057947	−0.0030273
Tiffany	Red	0.9584	0.95431	0.91675	−0.0014434	−0.0019299	0.0075856
	Green	0.89597	0.95016	0.87161	−0.012698	0.0075576	−0.001533
	Blue	0.91305	0.93013	0.86384	−0.0014111	0.0010208	−0.0038235

5.4. Differential Attack

Usually, a cracker performs a little amendment to the pixels of the plain image and then uses the same encryption approach to encrypt similar images. With this approach, they try to find the association between a plain image and an encrypted image. To check the robustness of our proposed method against differential attacks, we employed the number of pixels change rate (NPCR) and unified average changing intensity (UACI). The NPCR and UACI can be computed by the following equations:

$$NPCR_{C_1,C_2} = [\frac{\sum_{i,j} \tilde{I}(i,j)}{H \times W}] \times 100 \tag{17}$$

$$UACI_{C_1,C_2} = \frac{1}{H \times W}[\sum_{i,j} \frac{|C_1(i,j) - C_2(i,j)|}{2^8 - 1}] \times 100, \tag{18}$$

where C_1 and C_2 represent two dissimilar ciphered images, before and after one pixel in the plain image is altered, respectively; H and W are the height and width of the image; and $\tilde{I}(i,j)$ can be defined as follows:

$$\tilde{I}(i,j) = \begin{cases} 0 & C_1(i,j) \neq C_2(i,j) \\ 1 & otherwise \end{cases} \tag{19}$$

The NPCR and UACI values for two images (Lena, pepper) are given in Table 5, from which we can see that $NPCR \geq 99.6$ and $UACI \geq 33.4$. From these values, it can be seen that the NPCR values of the proposed encryption algorithm were better than those of [52–55], while the UACI value was comparable to the others.

Table 5. Number of pixels change rate (NPCR) and unified average changing intensity (UACI) comparison.

Algorithm	Lena		Pepper	
	NPCR$_{R,G,B}$	**UACI$_{R,G,B}$**	**NPCR$_{R,G,B}$**	**UACI$_{R,G,B}$**
Proposed	0.9968	0.3346	0.9967	0.3348
[52]	0.9966	0.3344	0.9963	0.3347
[53]	0.9959	0.3346	-	-
[56]	0.9962	0.3365	-	-
[57]	0.9960	0.3348	-	-
[54]	0.9960	0.3344	-	-
[55]	0.9961	0.3346	0.9960	0.3349

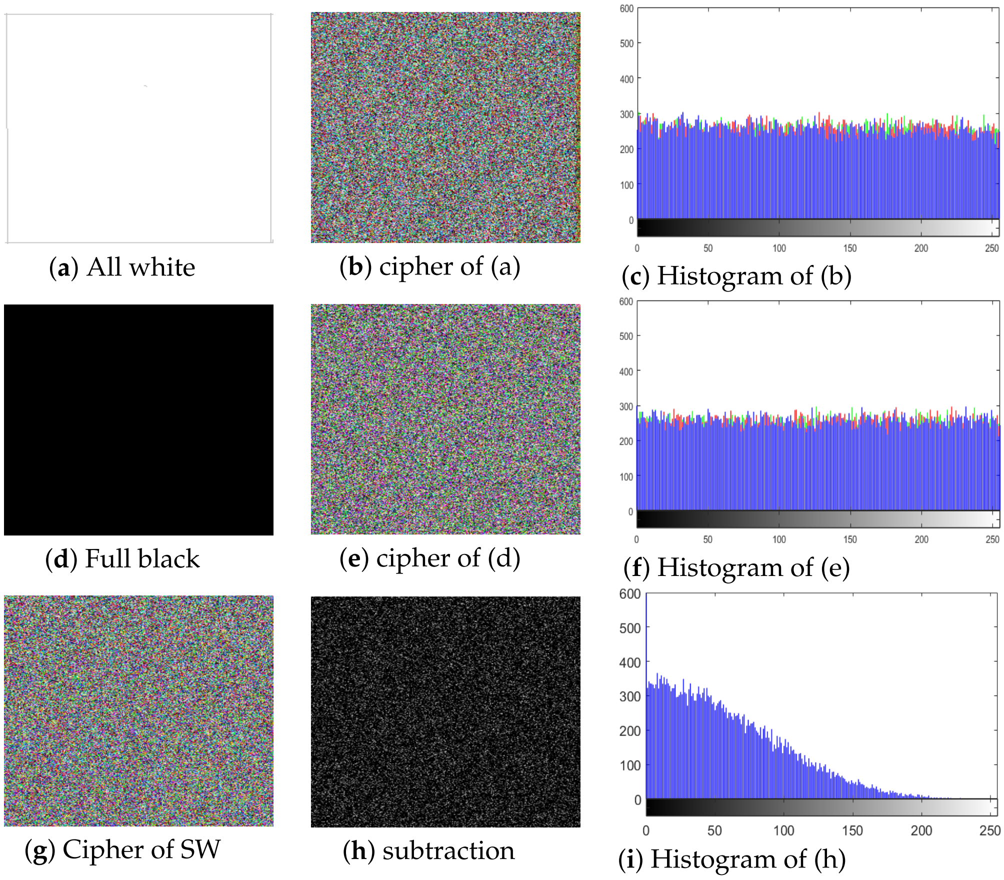
5.5. Known and Chosen Plain Text Analysis

Generally, there are four types of cryptanalysis attacks which can be performed to break an image encryption algorithm: chosen-plain text attack, chosen-cipher text attack, known-plain text attack, and cipher text-only attack [58–60]. The cryptanalysis method where the attacker picks certain plain text to obtain the related cipher text is called chosen-plain text attack. By probing the plain text and the related cipher text, they can attempt to deduce some valuable encrypted information. Finally, by means of that information, they can try to convalesce the actual images [61,62].

Of the above-mentioned attacks, the chosen-plain text attack is the most influential, so if an encryption algorithm can successfully withstand this attack, it is believed to have a satisfactory security level against the other three attacks as well. We performed this attack by using all-white and full-black images. We also created special white (SW) and special black (SB) images, having all pixels 255 in the white image and 0 in the black image, except for one pixel at the location 200 × 200, which was different. We created the cipher images and the results are given in Figure 10. In Figure 10a,d, the full white and full black images are shown; Figure 10b,e are the corresponding cipher images, respectively. Additionally, Figure 10c,f shows the histograms of Figure 10b,e; Figure 10g is the cipher image of the special image; and Figure 10h is the subtraction result of the cipher of the all-white image and the cipher of the SW image. Figure 10i shows the histogram of the subtracted image. We can see that the histograms of the cipher-generated images of both the all-white and full-black images are almost uniform and flat, and that the subtracted image is also a noise-like image showing no useful information. In Table 6, the entropy, NPCR, UACI, and correlation values of the all-white, SW, and full-black images are also given. From the results, it is clear that the ciphered image of our proposed algorithm had higher entropy values, better correlation results (in comparison to [63]), and ideal NPCR and UACI values. On the basis of these results, we can conclude that our proposed algorithm has high security against known/chosen plain text attacks, and hence, it can keep images more secure.

Table 6. Known/chosen plain-text attack result.

Algorithms	**Images**	**Entropy**	**Correlation**			**NPCR**	**UACI**
			Horizontal	**Vertical**	**Diagonal**	**SW,SB**	**SW,SB**
Ours	All white	0	-	-	-	-	-
	Cipher white	7.9988	−0.01140	−0.00706	0.00631	0.9962	0.3349
	Full black	0	-	-	-	-	-
	Cipher black	7.9989	0.00965	−0.00307	−0.00172	0.9959	0.3343
[63]	All white	0	-	-	-	-	-
	Cipher white	7.9971	0.0071	0.0144	−0.0068	-	-
	Full black	0	-	-	-	-	-
	Cipher black	7.9970	−0.0101	0.0135	−0.0010	-	-



(**a**) All white (**b**) cipher of (a) (**c**) Histogram of (b)

(**d**) Full black (**e**) cipher of (d) (**f**) Histogram of (e)

(**g**) Cipher of SW (**h**) subtraction (**i**) Histogram of (h)

Figure 10. Encryption result of all white and full black images. (**a**) All white image; (**b**) full black image; (**c**) cipher image of (**a**); (**d**) cipher image of (**b**); (**e**) histogram of (**c**); (**f**) histogram of (**d**); (**g**) cipher of one pixel change of (**a**); (**h**) subtraction of (**b**,**g**); (**i**) histogram of (**g**).

5.6. Encryption Quality

The image encryption quality can be determined by the below equation, where $E_{(r,c)}$ and $I_{(r,c)}$ can be taken as the gray values of the image pixels in a grid $G_{(r,c)}$ in the cipher and the plain images, respectively, each having $H \times W$ pixels with n gray levels; $I_{(r,c)}$ and $E_{(r,c)} \in (0, 1, \ldots, n-1)$; $H_n(E)$ and $H_n(I)$ are the occurrences of the gray level n in the cipher and plain images, respectively; and E_Q depicts the average change rate for each gray level n. The higher the $\mathrm{E_Q}$ value, the more secure the image is. E_Q can be computed by the following formula:

$$E_Q = \sum_{n=0}^{255} \frac{(H_n(E) - H_n(I))^2}{256}. \tag{20}$$

Table 7 lists the E_Q values of USC-SIPI database images, along with their comparison to [64]. The table values show that the E_Q values of the proposed algorithm are higher that those of [64] in the majority of images. Thus, we can say that the proposed algorithm provides a more secure encryption algorithm.

Table 7. E_Q comparison of USC-SIPI database.

USC-SIPI	**Proposed**			**[64]**		
	Red	**Green**	**Blue**	**Red**	**Green**	**Blue**
4.1.01	358	385	292	312	317	326
4.1.02	229	289	272	336	347	346
4.1.03	393	370	403	308	346	339
4.1.04	272	210	284	203	199	282
4.1.05	219	278	231	290	273	312
4.1.06	210	252	260	189	181	246
4.1.07	285	289	244	372	358	291
4.1.08	242	264	261	331	309	259
House	284	317	276	183	219	155

5.7. Robustness Against Occlusion Attack

In image processing, the extensively used parameters to check the encryption quality are peak signal to noise ratio (PSNR) and mean square error (MSE). The PSNR and MSE values between the plain (P), decrypted (D), and ciphered images can be computed as follows:

$$PSNR = 10 \times log_{10}(\frac{(2^L - 1)^2}{MSE}(dB), \tag{21}$$

where L is the bit-depth of the particular image. The MSE can be defined as

$$MSE = \frac{1}{HW} \sum_{r=1}^{H} \sum_{c=1}^{W} [P_{(r,c)} - D_{(r,c)}]^2. \tag{22}$$

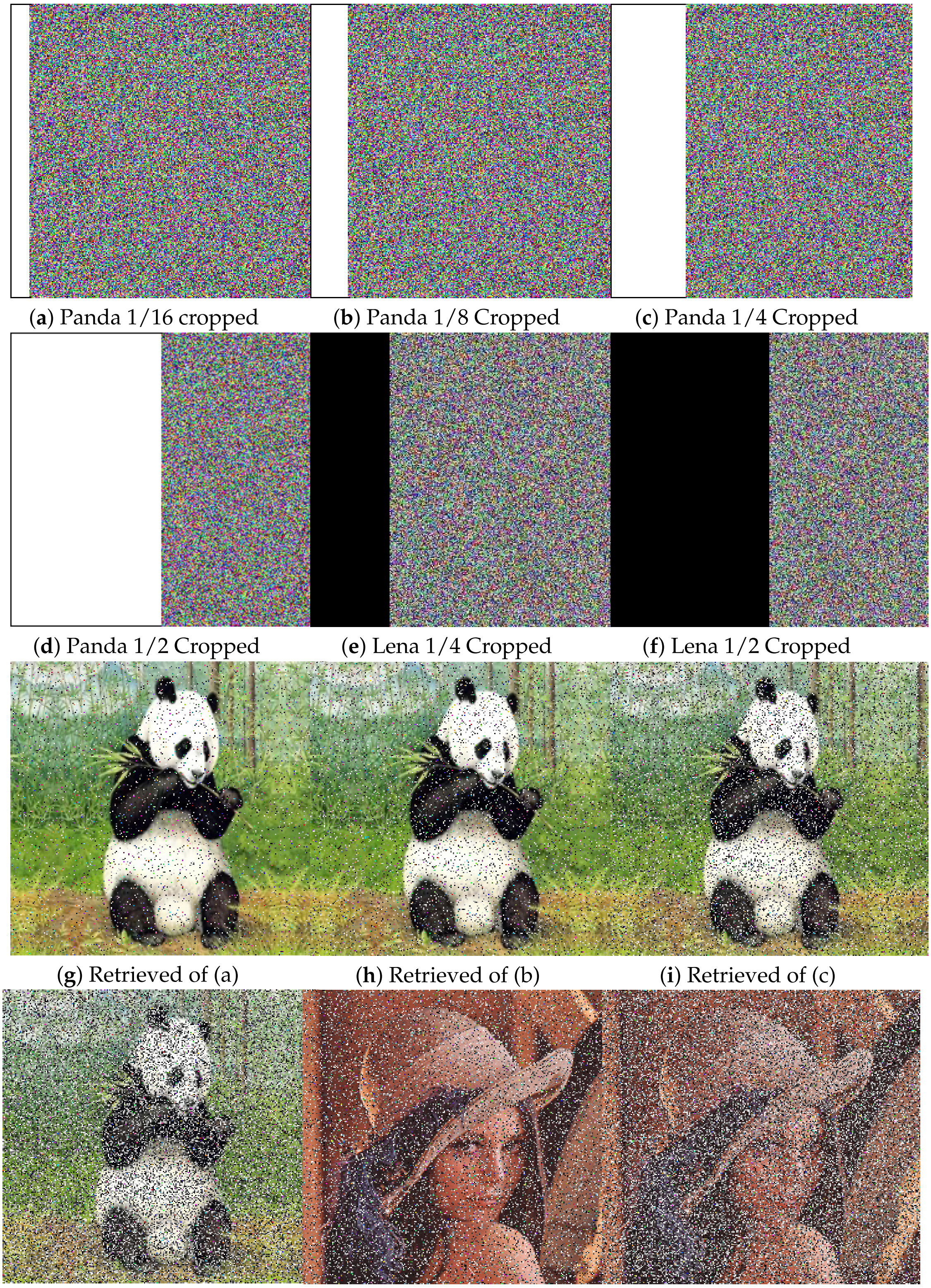
A higher PSNR value indicates a smaller difference between the plain and decrypted images. If P and D are the same, their PSNR will be infinity. Table 8 lists the PSNR values between the plain and ciphered images and plain and decrypted images for the different test images, along with comparisons to other methods. The infinity value of PSNR for plain to decrypted shows that the proposed algorithm can efficiently decrypt the actual image, while the method of [65] resulted in loss of some information. Similarly, lower values of PSNR between the plain and cipher images indicate more secure image encryption, as lower values indicate greater differences. The lower PSNR values of plain to ciphered images in Table 8 indicate that the proposed algorithm encrypts better than those of [65–67]. An alternative means of quality checking is at the receiver side, where a ciphered image may lose some data or get blurred. Thus, a reliable encryption algorithm or method should be able to get back the actual image without dropping too much substantial information. Figure 11a–d shows the test image Panda with different cropped portions (i.e., 1/16, 1/8, 1/4, and 1/2, respectively), while Figure 11g–j shows the images retrieved from the respective cropped image. Similarly, Figure 11e,f shows the Lena image cropped by 1/4 and 1/2, respectively, with the corresponding retrieved images in Figure 11k,l, respectively. We can see that retrieved image still has visual information, even after half of the ciphered image is destroyed. Table 9 lists a comparison of the PSNR and MSE of the Lena image with different cropped portions. The higher PSNR values and lower MSE values, as compared to [64,68], prove that proposed algorithm can retrieve images without losing substantial information.

Table 8. Peak signal to noise ratio (PSNR) (dB) between plain (P) ↦ cipher (C) image and plain ↦ decrypted (D) image.

Algorithm	PSNR type	Lena	Baboon	Panda	Vegetables
Proposed	P to D	∞	∞	∞	∞
	P to C	8.1102	8.7776	8.1648	6.8760
[66]	P to C	8.1300	7.8569	7.7410	7.4395
[65]	P to D	96.295	-	-	-
	P to C	9.0348	-	-	-
[67]	P to C	8.3655	8.8532	-	-
[69]	P to C	8.2522	8.8223	-	-

Table 9. PSNR (dB) and mean square error (MSE) comparison with different cropped size.

Cropped Size	Image	Proposed		[68]		[64]	
		PSNR	MSE	PSNR	MSE	PSNR	MSE
1/2	Lena	17.4286	1.1755×10^3	12.88	3121.1	11.58	4578.34
	Panda	13.9065	2.6450×10^3	-	-	-	-
1/4	Lena	20.2816	609.4230	14.722	2192.2	14.59	2289.9
	Panda	16.7175	1.3846×10^3	-	-	-	-
1/8	Lena	23.0749	320.3273	16.75	1375.9	17.57	1155.3
	Panda	19.4912	731.0670	-	-	-	-
1/16	Lena	25.6379	177.5376	19.25	772.65	20.57	579.9
	Panda	21.9180	418.0993	-	-	-	-



(**a**) Panda 1/16 cropped (**b**) Panda 1/8 Cropped (**c**) Panda 1/4 Cropped

(**d**) Panda 1/2 Cropped (**e**) Lena 1/4 Cropped (**f**) Lena 1/2 Cropped

(**g**) Retrieved of (a) (**h**) Retrieved of (b) (**i**) Retrieved of (c)

(**j**) Retrieved of (d) (**k**) Retrieved of (e) (**l**) Retrieved of (f)

Figure 11. Occlusion attack test. (**a**) Cropped image of Lena; (**b**) cropped image of a panda; (**c**) cropped image of a house; (**d**) cropped image of baboon; (**e**) cropped image of a panda; (**f**) cropped image of Lena; (**g**) retrieved image of (**a**); (**h**) retrieved image of (**b**); (**i**) retrieved image of (**c**); (**j**) retrieved image of (**d**); (**k**) retrieved image of (**e**); (**l**) retrieved image of (**f**).

5.8. Local and Shannon Information Entropy

Entropy (E) values indicate the degree of chaos in an encryption system by calculating its gray value probability. Information entropy (IE) or entropy can be defined in the following way:

Suppose an information source is $\hbar$. Then, E can be calculated with the following formula:

$$E(\hbar) = \sum_{i=0}^{2^l-1} \rho(\hbar_i) log_2 \frac{1}{\rho(\hbar_i)}. \tag{23}$$

In the above equation, l depicts the gray intensity level and $\rho(\hbar_i)$ represents the probability of the symbol $\hbar_i$. For an image with an intensity level of 2^8, the ideal value of E is 8 [61]. Therefore, the closer the entropy is to this value, the better the randomness of the image pixels. A higher value ensures that less information will be discovered from the particular encrypted image. Table 10 shows the entropy value of our test images, along with a comparison to some existing image encryption algorithms. The values in the table show that $E \geq 7.996$, which is higher than those of [10,58,62,70,71].

In [72], a new uncertainty test was proposed for native image blocks by means of the Shannon entropy. The Shannon entropy for native image pixel blocks $(\tau, \vec{a}_k)$ can be computed in following way:

(1): Randomly choose (non-overlapping) blocks in an image (i.e., $N_1, N_2, N_3, \ldots, N_k$) with $\vec{a}_k$ pixels within the image I having intensity L^n. (2): Calculate the Shannon entropy for all $i \in ((k-1), (k-2), \ldots, 3, 2, 1)$ using Equation (23). (3): Compute the Shannon entropy by sample mean over all k image pixel blocks $N_1, N_2, N_3, \ldots, N_k$ with the following formula:

$$\hbar_{(\tau,\vec{a}_k)}(N) = \sum_{i=1}^{k} \frac{H(N_i)}{k}. \tag{24}$$

We calculated the Shannon entropy values for the ciphered images with $k = 32$ and $\vec{a}_k$ = 1940 pixels. Table 11 lists the global and Shannon entropy values for the USC-SIPI image database, and some well-known test images. The values were $E \geq 7.998$, which is equivalent to the ideal value; furthermore, the Shannon entropy values were between 7.9018 and 7.9034. These results are comparable with those of [73].

Table 10. Information entropy comparison.

Algorithm	Test	Ciphered		
	Image	**R**	**G**	**B**
Our	Lena	7.9974	7.9974	7.9971
	Pepper	7.9993	7.9994	7.9992
	Baboon	7.9993	7.9992	7.9993
	Panda	7.9972	7.9971	7.9966
	Vegetable	7.9992	7.9994	7.9994
[58]	Lena	7.989	7.989	7.989
	Pepper	7.989	7.988	7.989
	Baboon	7.989	7.989	7.988
	Panda	7.988	7.989	7.989
	Vegetable	7.989	7.989	7.989
[62]	Lena	7.987	7.987	7.986
[70]	Lena	7.927	7.974	7.970
[71]	Lena	7.973	7.975	7.971
[10]	Lena	7.987	7.988	7.987

Table 11. Global entropy and Shannon entropy.

Images	**Our**	**Global$_{R,G,B}$**		[73]
	Shannon$_C$	**Plain**	**Cipher**	**Shannon$_C$**
Vegetables	7.9032	7.5060	7.9997	-
Baby	7.9029	6.3988	7.9997	-
Fruits	7.9023	7.5322	7.9991	-
Panda	7.9026	7.8679	7.9990	-
Pepper	7.9030	7.6698	7.9998	7.9024
Apricot	7.9029	5.7492	7.9997	-
Nike	7.9022	1.1969	7.9997	
Tiffany	7.9023	6.4288	7.9998	-
House	7.9032	7.4858	7.9997	7.9021
Baboon	7.9027	7.2428	7.9998	7.9023
Lena	7.9032	7.4517	7.9991	7.9024
4.1.01	7.9023	6.8981	7.9991	-
4.1.02	7.9032	6.2945	7.9990	-
4.1.03	7.9019	5.9709	7.9991	
4.1.04	7.9021	7.4270	7.9992	-
4.1.05	7.9033	7.0686	7.9990	-
4.1.06	7.9031	7.5371	7.9991	-
4.1.07	7.9020	6.5835	7.9989	-
4.1.08	7.9027	6.8527	7.9991	-

5.9. Gray Value Degree (GVD) Analysis

GVD analysis is a well-known statistical test to check the randomness of an image. It can be calculated by comparing the cipher-generated image of a particular algorithm with the corresponding plain image. The ideal value is **1**, and the higher the value, the better the haphazardness and security. The GVD can be calculated using the following Equation:

$$GVD(i,j) = \sum[\Gamma(i,i) - \Gamma(\acute{i},\acute{j})], \tag{25}$$

where $\Gamma(i,j)$ symbolizes the gray score at position (i,j), and $(\acute{i},\acute{j})$ is as given below:

$$(\acute{i},\acute{j}) = \begin{cases} (i, j+1) \\ (i, j-1) \\ (i+1, j) \\ (i-1, j). \end{cases} \tag{26}$$

The average neighborhood gray difference of an image can be calculated as follows:

$$[GVD(i,j)]_{avg.} = \frac{\sum_{i=2}^{M-1} \sum_{j=2}^{N-1} GVD(i,j)}{(M-2)(N-2)} \tag{27}$$

$$GVD_{avg.} = \frac{\acute{\chi}[G_D(r,c)] - \chi[G_D(r,c)]}{\acute{\chi}[G_D(r,c)] + \chi[G_D(r,c)]}. \tag{28}$$

In the above equations, M and N represent the rows and columns of the image, respectively, and $\acute{\chi}$ and χ symbolize the average score of the neighborhood gray value, where $\acute{\chi}$ denotes after encrypting the image and χ denotes before encrypting the image. The final value is termed the GVD, which is **1** for completely dissimilar images but **0** for identical images. The GVD scores for the plain images of the **USC-SIPI** database and their cipher-generated images by the proposed algorithm are given in Table 12, along with comparisons to existing methods. In our case, the GVD scores indirectly varied with the key bit-planes (i.e., the GVD scores were higher when we selected the least significant bit plane as a key, and lower when we selected the most significant bit-planes as a secret key). The listed comparison shows that the GVD scores of the proposed algorithm were comparable those of [8,64].

Table 12. Gray value difference comparison.

USC-SIPI	Proposed			[64]			[74]			[8]		
	GVD			GVD			GVD			GVD		
	Red	Green	Blue	Red	Green	Blue	Red	Green	Blue	Red	Green	Blue
4.1.01	0.995	0.943	0.996	0.977	0.979	0.975	-	-	-	-	-	-
4.1.02	0.970	0.999	0.999	0.978	0.979	0.979	-	-	-	-	-	
4.1.03	0.672	0.407	0.408	0.978	0.976	0.977	-	-	-	-	-	-
4.1.04	0.383	0.348	0.532	0.979	0.975	0.980	-	-	-	-	-	-
4.1.05	0.673	0.918	0.719	0.982	0.966	0.969	-	-	-	-	-	-
4.1.06	0.988	0.995	0.989	0.943	0.912	0.934	-	-	-	-	-	-
4.1.07	0.245	0.382	0.794	-	-	-	-	-	-	-	-	-
4.1.08	0.564	0.355	0.291	0.985	0.973	0.983	-	-	-	-	-	-
4.2.01	0.782	0.804	0.905	-	-	-	-	-	-	-	-	-
4.2.03	0.998	0.985	0.986	-	-	-	-	-	-	0.9801	0.989	0.9865
4.2.05	0.934	0.460	0.565	-	-	-	-	-	-	-	-	-
4.2.06	0.453	0.261	0.879	-	-	-	-	-	-	-	-	-
4.2.07	0.511	0.329	0.571	-	-	-	-	-	-	-	-	-
Lena	0.979	0.997	0.995	-	-	-	0.980	0.981	0.987	0.970	0.970	0.969

5.10. Performance Comparison

Efficiency and time are key characteristics of an encryption algorithm. In chaotic, map-based image encryption schemes, the number of rounds for diffusion and permutation impact the encryption time. More rounds means a longer time taken by the algorithm to encrypt an image. As our proposed algorithm was based on modular addition, it took less time compared to many of the other algorithms listed in Table 13. Additionally, the encryption and decryption times (in seconds) for different-sized test images are shown in Figure 12.

Table 13. Encryption time (in seconds) comparison.

Image Size	Proposed	[75]	[76]	[77]
256 × 256	**2.09**	4.7795	3.617	4.221
512 × 512	**3.77**	8.670	14.811	21.178

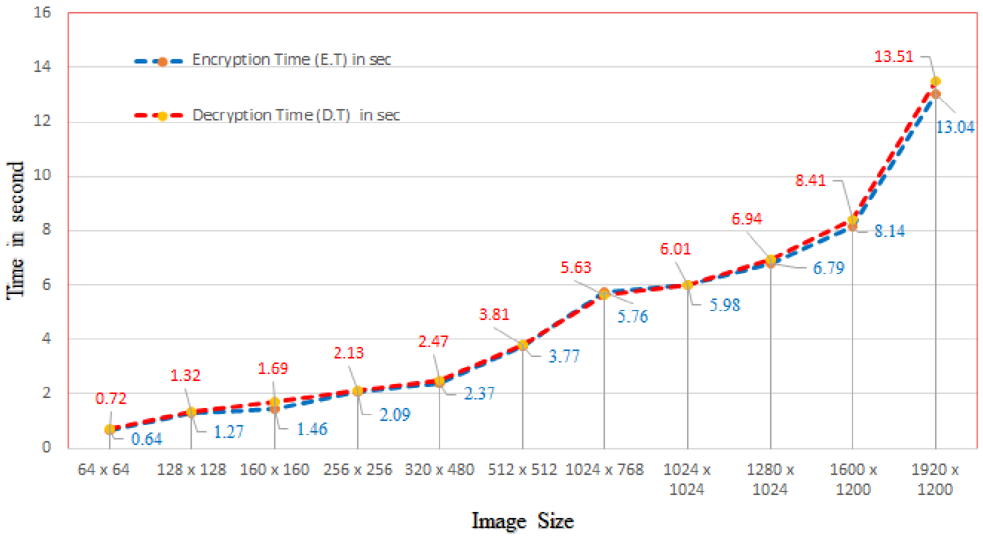


Figure 12. Encryption time in seconds.

6. Conclusions

To tackle the problems of complexity and encryption time, we have proposed a simple, fast, and secure image encryption algorithm which can ensure better security in less time, in comparison to some older algorithms. Our novel scrambling plus diffusion (SPD) technique is effective in two ways: it diffuses the pixels and scrambles the bit-planes in an efficient manner. The proposed algorithm can withstand many well-known attacks to ensure the security of the ciphered images. The use of modular addition between the pixels of image blocks and bit-plane scrambling provides an advantage in the sense of fast processing compared to binary addition between the pixels and pixel-based scrambling. Experimental results verified that our proposed algorithm can withstand some well-known attacks, such as key sensitivity tests, occlusion attacks, and known/chosen plain text attacks, and obtain satisfactory results from the GVD test. Therefore, we can conclude that the proposed encryption algorithm has demonstrated satisfactory security and efficiency. In further research, we will investigate its prospective applications in image communication.

Supplementary Materials: The following are available online at http://www.mdpi.com/1099-4300/22/1/112/s1.

Author Contributions: K.K.B. wrote the manuscript, and conceived and performed the simulation; G.L. supervised the research and delivered funding; S.K. prepared original draft, designed the methodology, and performed software and simulation work; S.M., editing, visualization, and formal analysis. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 61572215.

Conflicts of Interest: The authors declare no conflict of interest in regard to the manuscript, and in the decision to publish the results.

References

1. Wong, K.; Kwok, B.; Law, W. A fast image encryption scheme based on chaotic standard map. *Phys. Lett. A* **2008**, *372*, 2645–2652. [CrossRef]
2. Fridrich, J. Symmetric ciphers based on two-dimensional chaotic maps. *Int. J. Bifurcation Chaos* **1998**, *8*, 1259–1284. [CrossRef]
3. Zhu, H.; Zhao, C.; Zhang, X.; Yang, L. An image encryption scheme using generalized Arnold map and affine cipher. *Optik* **2014**, *125*, 6672–6677. [CrossRef]
4. Hua, Z.; Zhou, Y. Image encryption using 2D Logistic-adjusted-Sine map. *Inf. Sci.* **2016**, *339*, 237–253. [CrossRef]
5. Li, Y.; Wang, C.; Chen, H. A hyper-chaos-based image encryption algorithm using pixel-level permutation and bit-level permutation. *Opt. Lasers Eng.* **2017**, *90*, 238–246. [CrossRef]
6. Belazi, A.; El-Latif, A.A.A.; Belghith, S. A novel image encryption scheme based on substitution-permutation network and chaos. *Signal Process.* **2016**, *128*, 155–170. [CrossRef]
7. 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]
8. Kalpana, J.; Murali, P. An improved color image encryption based on multiple DNA sequence operations with DNA synthetic image and chaos. *Optik* **2015**, *126*, 5703–5709. [CrossRef]
9. Huang, X.; Ye, G. An efficient self-adaptive model for chaotic image encryption algorithm. *Commun. Nonlinear Sci. Numer. Simul.* **2014**, *19*, 4094–4104. [CrossRef]
10. Liu, H.; Wang, X. Color image encryption using spatial bit-level permutation and high-dimension chaotic system. *Opt. Commun.* **2011**, *284*, 3895–3903. [CrossRef]
11. Mao, Y.; Chen, G.; Lian, S. A novel fast image encryption scheme based on 3D chaotic baker maps. *Int. J. Bifurcation Chaos* **2004**, *14*, 3613–3624. [CrossRef]
12. Wu, Y. Image encryption using the two-dimensional logistic chaotic map. *J. Electron. Imag.* **2012**, *21*, 013014. [CrossRef]
13. Chen, G.; Mao, Y.; Chui, C.K. A symmetric image encryption scheme based on 3D chaotic cat maps. *Chaos Solitons Fractals* **2004**, *21*, 749–761. [CrossRef]

14. Lian, S.; Sun, J.; Wang, Z. A block cipher based on a suitable use of the chaotic standard map. *Chaos Solitons Fractals* **2005**, *26*, 117–129. [CrossRef]
15. Norouzi, B.; Seyedzadeh, S.M.; Mirzakuchaki, S.; Mosavi, M.R. A novel image encryption based on row-column, masking and main diffusion processes with hyper chaos. *Multimed. Tools Appl.* **2015**, *74*, 781–811. [CrossRef]
16. Ping, P.; Xu, F.; Mao, Y.; Wang, Z. Designing permutation substitution image encryption networks with Henon map. *Neurocomputing* **2018**, *283*, 53–63. [CrossRef]
17. Liu, H.; Wang, X.; Kadir, A. Image encryption using DNA complementary rule and chaotic maps. *Appl. Soft Comput.* **2012**, *12*, 1457–1466. [CrossRef]
18. Huang, X.; Ye, G. An image encryption algorithm based on hyperchaos and DNA sequence. *Multimed. Tools Appl.* **2014**, *72*, 57–70. [CrossRef]
19. Liu, H.; Wang, X. Color image encryption based on one-time keys and robust chaotic maps. *Comput. Math. Appl.* **2010**, *59*, 3320–3327. [CrossRef]
20. Chai, X.; Zheng, X.; Gan, Z.; Han, D.; Chen, Y. An image encryption algorithm based on chaotic system and compressive sensing. *Signal Process.* **2018**, *148*, 124–144. [CrossRef]
21. Wang, X.-Y.; Yang, L.; Liu, R.; Kadir, A. A chaotic image encryption algorithm based on perceptron model. *Nonlinear Dyn.* **2010**, *62*, 615–621. [CrossRef]
22. Wang, X.; Liu, L.; Zhang, Y. A novel chaotic block image encryption algorithm based on dynamic random growth technique. *Opt. Lasers Eng.* **2015**, *66*, 10–18. [CrossRef]
23. Hua, Z.; Yi, S.; Zhou, Y. Medical image encryption using highspeed scrambling and pixel adaptive diffusion. *Signal Process.* **2018**, *144*, 134–144. [CrossRef]
24. Parvin, Z.; Seyedarabi, H.; Shamsi, M. A new secure and sensitive image encryption scheme based on new substitution with chaotic function. *Multimed. Tools Appl.* **2016**, *75*, 10631–10648. [CrossRef]
25. Li, C.; Lin, D.; Lü, J. Cryptanalyzing an image-scrambling encryption algorithm of pixel bits. *IEEE Multimed.* **2017**, *24*, 64–71. [CrossRef]
26. Li, C.; Lo, K.-T. Optimal quantitative cryptanalysis of permutation only multimedia ciphers against plaintext attacks. *Signal Process.* **2011**, *91*, 949–954. [CrossRef]
27. Tao, X.; Wong, K.; Liao, X. Selective image encryption using a spatiotemporal chaotic system. *Chaos* **2007**, *17*, 023115.
28. Teng, L.; Wang, X. A bit-level image encryption algorithm based on spatiotemporal chaotic system and self-adaptive. *Opt. Commun.* **2012**, *285*, 4048–4054. [CrossRef]
29. Fu, C.; Lin, B.-B.; Miao, Y.-S.; Liu, X.; Chen, J.-J. A novel chaos based bit-level permutation scheme for digital image encryption. *Opt. Commun.* **2011**, *284*, 5415–5423. [CrossRef]
30. Zhu, Z.-L.; Zhang, W.; Wong, K.-W.; Yu, H. A chaos-based symmetric image encryption scheme using a bit-level permutation. *Inf. Sci.* **2011**, *181*, 1171–1186. [CrossRef]
31. Zhang, W.; Wong, K.-W.; Yu, H.; Zhu, Z.-L. A symmetric color image encryption algorithm using the intrinsic features of bit distributions. *Commun. Nonlinear Sci. Numer. Simul.* **2013**, *18*, 584–600. [CrossRef]
32. Chen, J.-X.; Zhu, Z.-L.; Fu, C.; Zhang, L.-B.; Zhang, Y. An image encryption scheme using nonlinear inter-pixel computing and swapping based permutation approach. *Commun. Nonlinear Sci. Numer. Simul.* **2015**, *23*, 294–310. [CrossRef]
33. Bao, L.; Zhou, Y. Image encryption: Generating visually meaningful encrypted images. *Inf. Sci.* **2015**, *324*, 197–207. [CrossRef]
34. Chen, J.-X.; Zhu, Z.-L.; Fu, C.; Yu, H.; Zhang, Y. Reusing the permutation matrix dynamically for efficient image cryptographic algorithm. *Signal Process.* **2015**, *111*, 294–307. [CrossRef]
35. Zhang, Y.; Xiao, D.; Wen, W.; Wong, K.-W. On the security of symmetric ciphers based on DNA coding. *Inf. Sci.* **2014**, *289*, 254–261. [CrossRef]
36. Wu, Y.; Zhou, Y.; Noonan, J.P.; Agaian, S. Design of image cipher using latin squares. *Inf. Sci.* **2014**, *264*, 317–339. [CrossRef]
37. Chen, J.-X.; Zhu, Z.-L.; Fu, C.; Yu, H.; Zhang, L.-B. An efficient image encryption scheme using gray code based permutation approach. *Opt. Lasers Eng.* **2015**, *67*, 191–204. [CrossRef]
38. Khan, S.; Han, L.; Lu, H.; Butt, K.K.; Bachira, G.; Khan, N. A New Hybrid Image Encryption Algorithm Based on 2D-CA, FSM-DNA Rule Generator, and FSBI. *IEEE Access* **2019**, *7*, 81333–81350. [CrossRef]

39. Gonzalez, R.C.; Woods, R.E. *Digital Image Processing*, 3rd ed.; Pearson Prentice Hall: Upper Saddle River, NJ, USA, 2008.
40. Tanenbaum, A.S.; Wetherall, D.J. 8.2.3 Cipher Modes. In *Computer Networks*, 5th ed.; Pearson: Upper Saddle River, NJ, USA, 2011.
41. Alvarez, G.; Li, S. Some basic cryptographic requirements for chaos-based cryptosystems. *Int. J. Bifur. Chaos* **2016**, *16*, 2129–2151. [CrossRef]
42. Li, C. Cracking a hierarchical chaotic image encryption algorithm based on permutation. *Signal Process.* **2016**, *118*, 203–210. [CrossRef]
43. Arroyo, D.; Hernandez, F.; Orue, A.B. Cryptanalysis of a classical chaos-based cryptosystem with some quantum cryptography features. *Int. J. Bifur. Chaos* **2017**, *27*, 1750004. [CrossRef]
44. Castro, J.C.H.; Sierra, J.M.; Seznec, A.; Izquierdo, A.; Ribagorda, A. The strict avalanche criterion randomness test. *Math. Comput. Simul.* **2005**, *68*, 1–7. [CrossRef]
45. Ramasamy, P.; Ranganathan, V.; Kadry, S.; Damasevicius, R.; Blazauskas, T. An Image Encryption Scheme Based on Block Scrambling, Modified Zigzag Transformation and Key Generation Using Enhanced Logistic-Tent Map. *Entropy* **2019**, *21*, 656. [CrossRef]
46. Wu, X.; Kurths, J.; Kan, H. A robust and lossless DNA encryption scheme for color images. *Multimed. Tools Appl.* **2017**, *77*, 12349–12376. [CrossRef]
47. 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*, 1–19. [CrossRef]
48. 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]
49. Wang, X.Y.; Zhang, Y.Q.; Bao, X.M. A novel chaotic image encryption scheme using DNA sequence operations. *Opt. Lasers Eng.* **2015**, *73*, 53–61. [CrossRef]
50. CLi, Q.; Li, S.J.; Chen, G.R.; Halang, W.A. Cryptanalysis of an image encryption scheme based on a compound chaotic sequence. *Image Vis. Comput.* **2009**, *27*, 1035–1039.
51. Wang, X.; Zhang, H.L. A color image encryption with heterogeneous bit-permutation and correlated chaos. *Opt. Commun.* **2015**, *342*, 51–60. [CrossRef]
52. Niyat, A.Y.; Moattar, M.H.; Torshiz, M.N. Color image encryption based on hybrid hyper chaotic system and cellular automata. *Opt. Lasers Eng.* **2017**, *90*, 225–237. [CrossRef]
53. Hu, T.; Liu, Y.; Gong, L.; Guo, S.; Yuan, H. Chaotic image crypto-system using DNA deletion and DNA insertion. *Signal Process.* **2017**, *134*, 234–243. [CrossRef]
54. Chen, J.; Zhu, Z.; Zhang, L.; Zhang, Y.; Yang, B. Exploiting self-adaptive permutation-diffusion and DNA random encoding for secure and efficient image encryption. *Signal Process.* **2018**, *142*, 340–353. [CrossRef]
55. Wu, X.; Kan, H.; Kurths, J. A new color image encryption scheme based on DNA sequences and multiple improved 1D chaotic maps. *Appl. Soft Comput. J.* **2015**, *37*, 24–39. [CrossRef]
56. Ye, G.; Huang, X. An efficient symmetric image encryption algorithm based on an intertwining logistic map. *Neuro-Computing* **2017**, *251*, 45–53. [CrossRef]
57. Toughi, S.; Fathi, M.H.; Sekhavat, Y.A. An image encryption scheme based on elliptic curve pseudo random and advanced encryption system. *Signal Process.* **2017**, *141*, 217–227. [CrossRef]
58. Van den Assem, R.; van Elk, W. A chosen-plaintext attack on the microsoft basic protection. *Comput. Secur.* **1986**, *5*, 36–45. [CrossRef]
59. Fu, C.; Chen, J.; Zou, H.; Meng, W.; Zhan, Y.; Yu, Y. A chaos-based digital image encryption scheme with an improved diffusion strategy. *Opt. Express* **2012**, *20*, 2363–2378. [CrossRef]
60. Boopathy, D.; Sundaresan, M. A novel multi-dimensional encryption technique to secure the grayscale images and color images in public cloud storage. *Innov. Syst. Softw. Eng.* **2019**, *15*, 43–64. [CrossRef]
61. Wang, X.; Teng, L.; Qin, X. A novel colour image encryption algorithm based on chaos. *Signal Process.* **2012**, *92*, 1101–1108. [CrossRef]
62. Liu, H.; Wang, X.; Kadir, A. Color image encryption using Choquet fuzzy integral and hyper chaotic system. *Opt. Int. J. Light Electron. Opt.* **2013**, *124*, 3527–3533. [CrossRef]
63. Chai, X.; Gan, Z.; Yang, K.; Chen, Y.; Liu, X. An image encryption algorithm based on the memristive hyperchaotic system, cellular automata and DNA sequence operations. *Signal Process. Image Commun.* **2017**, *52*, 6–19. [CrossRef]

64. Rehman, A.U.; Liao, X.; Ashraf, R.; Ullah, S.; Wang, H. A color image encryption technique using exclusive-OR with DNA complementary rules based on chaos theory and SHA-2. *Optik* **2018**, *159*, 348–367. [CrossRef]
65. Zhu, C.X. A novel image encryption scheme based on improved hyper chaotic sequences. *Opt. Commun.* **2012**, *285*, 29–37. [CrossRef]
66. Wu, X.;Wang, K.;Wang, X.; Kan, H.; Kurths, J. Color image DNA encryption using NCA map based CML and one time keys. *Signal Process.* **2018**, *148*, 272–287. [CrossRef]
67. Ahmad, J.; Hwang, S.O. A secure image encryption scheme based on chaotic maps and affine transformation. *Multimed. Tools Appl.* **2016**, *75*, 13951–13976. [CrossRef]
68. Khan, S.; Han, L.; Mudassir, G.; Guehguih, B.; Ullah, H. 3C3R, an Image Encryption Algorithm Based on BBI, 2D-CA, and SM-DNA. *Entropy* **2019**, *21*, 1075. [CrossRef]
69. Liu, X.; Xiao, D.; Xiang, Y. Quantum Image Encryption Using Intra and Inter Bit Permutation Based on Logistic Map. *IEEE Access* **2019**, *7*, 6937–6946. [CrossRef]
70. Kadir, A.; Hamdulla, A.; Guo, W. Color image encryption using skew tent map and hyper chaotic system of 6th-order CNN. *Optik* **2014**, *125*, 1671–1675. [CrossRef]
71. Rhouma, R.; Meherzi, S.; Belghith, S. OCML-based colour image encryption. *Chaos Solitons Fract.* **2009**, *40*, 309–318. [CrossRef]
72. Wu, Y.; Zhou, Y.; Saveriades, G.; Agaian, S.; Noonan, J.P.; Natarajan, P. Local Shannon entropy measure with statistical tests for image randomness. *Inf. Sci.* **2013**, *222*, 323–342. [CrossRef]
73. Wang, X.Y.; Zhang, Y.Q.; Bao, X.M. A Colour Image Encryption Scheme Using Permutation-Substitution Based on Chaos. *Entropy* **2015**, *17*, 3877–3897. [CrossRef]
74. Wei, X.; Guo, L.; Zhang, Q.; Zhang, J.; Lian, S. A novel color image encryption algorithm based on DNA sequence operation and hyper-chaotic system. *J. Syst. Softw.* **2012**, *85*, 290–299. [CrossRef]
75. Pareek, N.K.; Patidar, V.; Sud, K.K. Image encryption using chaotic logistic map. *Image Vis. Comput.* **2006**, *24*, 926–934. [CrossRef]
76. 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.
77. Luo, Y.; Zhou, R.; Liu, J.; Qiu, S.; Cao, Y. An efficient and self-adapting colour-image encryption algorithm based on chaos and interactions among multiple layers. *Multimed. Tools Appl.* **2018**, *77*, 26191–26217. [CrossRef]