Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 3675459, 12 pages
https://doi.org/10.1155/2017/3675459

Research Article

Batch Image Encryption Using Generated Deep Features Based on Stacked Autoencoder Network

Fei Hu,[1,2] Jingyuan Wang,[1] Xiaofei Xu,[1] Changjiu Pu,[2] and Tao Peng[2]

[1]*School of Computer and Information Science, Southwest University, Chongqing, China*
[2]*Network Centre, Chongqing University of Education, Chongqing, China*

Correspondence should be addressed to Fei Hu; etz1@163.com

Received 8 November 2016; Accepted 30 January 2017; Published 28 February 2017

Academic Editor: Maria L. Gandarias

Chaos-based algorithms have been widely adopted to encrypt images. But previous chaos-based encryption schemes are not secure enough for batch image encryption, for images are usually encrypted using a single sequence. Once an encrypted image is cracked, all the others will be vulnerable. In this paper, we proposed a batch image encryption scheme into which a stacked autoencoder (SAE) network was introduced to generate two chaotic matrices; then one set is used to produce a total shuffling matrix to shuffle the pixel positions on each plain image, and another produces a series of independent sequences of which each is used to confuse the relationship between the permutated image and the encrypted image. The scheme is efficient because of the advantages of parallel computing of SAE, which leads to a significant reduction in the run-time complexity; in addition, the hybrid application of shuffling and confusing enhances the encryption effect. To evaluate the efficiency of our scheme, we compared it with the prevalent "logistic map," and outperformance was achieved in running time estimation. The experimental results and analysis show that our scheme has good encryption effect and is able to resist brute-force attack, statistical attack, and differential attack.

1. Introduction

Nowadays, though the communication system has been greatly developed, insecure communication channels such as the Internet are still prevalent and a growing number of digital images are transmitted through them. Nevertheless, as long as the images are transmitted and stored over public networks, they are easy to be intercepted and tampered by unauthorized IPs. If the images have confidential information, specially, they need to be encrypted before exchange across Internet. However, traditional encryption algorithms such as 3DES, AES, and IDEA are typically designed for text encryption and are not suitable for image encryption on account of the intrinsic features of images as high correlation between pixels and redundancy [1]. On the other hand, a lot of image encryption schemes have been suggested, for example, DNA cryptography, mathematical concept, compression methodology, and transform domain, but most of them have security vulnerabilities [2]. Over the past two decades, chaos-based cryptography has been studied by more and more researchers for the fundamental characteristics of chaos as ergodicity, pseudostochasticity, mixing, and high sensitivity to initial conditions/parameters, and so forth. [3–6]. In 1998, Fridrich firstly proposed permutation-diffusion method which encrypted images using chaos [7]. Based on his work, there have been many improvements for a wide variety of image encryption tasks, such as bit permutation [8–12], image pixel confusion [13, 14], extensive diffusion operations [15, 16], high-quality key-stream generation process [17, 18], and applications of plain-image features [19–21].

Chaotic cryptography with artificial neural networks (ANNs) has been extensively developed due to the following characteristics: nonlinear computation, associative memory, large-scale parallel processing, and highly fault tolerance. Those characteristics contribute much to enhancing chaos security. For example, in [22], a discrete Hopfield neural network was utilized to make a nonlinear sequential cipher generator, which could generate a pseudostochastic sequence to confuse the plain image given a small amount of stochastic parameters as the cipher codes. And, in [23–25], ANN models were also used for image encryption. Among kinds of ANN models, the SAE model provides a baseline method for

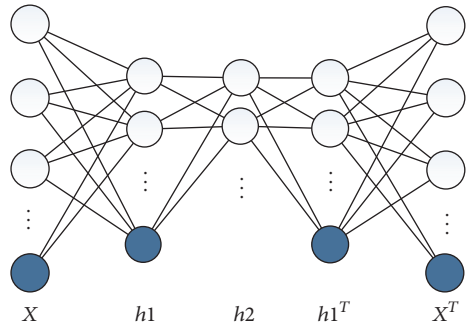


Figure 1: Stacked autoencoder (SAE) network.

unsupervised feature learning and additionally extracts a great number of feature parameters simultaneously, which are necessarily helpful for batch image encryption.

With previous chaos-based encryption schemes, a batch of images are usually encrypted using a single chaotic sequence. In this paper, we proposed a batch image encryption scheme that encrypts each image with an independent sequence. Firstly, a SAE based five-layer deep neural network was constructed to produce two chaotic matrices; and then the two matrices were used together for mixing encryption processing, that is, one for a total shuffling matrix generation and another for multiple chaotic sequence generation; finally the shuffling matrix was used to shuffle the pixel positions on each plain image, and each chaotic sequence was used to confuse the relationship between the corresponding shuffled image and the encrypted image. Several security evaluation results proved that the proposed scheme completely met the requirements of image encryption. Compared with traditional image schemes such as the logistic map, the proposed scheme is more powerful. In other words, the scheme is able to generate a large number of chaotic sequences in parallel and then to encrypt a lot of images simultaneously. Our scheme has a better performance in run-time complexity and a significant effect on batch image encryption.

2. Stacked Autoencoder

Autoencoder (AE) is a single-hidden-layer and unsupervised learning neural network. It is actually generated by two identical Restricted Boltzmann Machine models (RBMs) [26]. Several AEs compose the SAE network which is like a multilayer AE. For example, in the SAE network in Figure 1, the hidden layer of the previous AE is the input layer of the next AE; these joined layers are combined to make the encoding section of SAE (see layers [X, $h1$, $h2$] in Figure 1); then a reversed network of the encoding section makes the decoding section (see layers [$h2$, $h1^T$, X^T] in Figure 1); in addition, X and X' are input and output layers, respectively; the hollow cycles are neurons and the blue solid cycles are offsets. X^T is like a mirror of X, they have the same number of neurons, and so do $h1^T$ and $h1$ (For the sake of convenience, we will denote the input/output value of a certain layer by $X/h1/h2/h1^T/X^T$ in the following statement.). The encoding and decoding sections are combined to make the SAE network that has the functions of encoding and decoding. Greedy training methods such as back propagation (BP) can be directly used to train a monolayer AE to learn weight parameters; however, it is hard to train the SAE network because the multilayer network would consume much more memory and computation time. In order to alleviate the problem, a two-step training method is widely adopted for the SAE network training, that is, pretraining and fine-training.

In the pretraining period, each layer is trained in turn with a lay-wise approach: the SAE network encodes the input data X and produces $h1$ and immediately decodes $h1$ and produces X'; then we get the error $e = X' - X$ which is utilized to tune the weight parameters between the layers X and $h1$; the network continues to encode $h1$ and produce $h2$ and immediately to decode $h2$ and produce $h1'$; the weights between the layers $h1$ and $h2$ are tuned by $e = h1' - h1$; if there is any more layer, the encoding and decoding operations will be repeated. Now we get the optimized weight set (W, b) which will save training time in the following training. In the fine-training period, the network is trained using the BP algorithm and is gradually close to the optimal solution. See detailed fine-tuning procedures as follows [27]:

(1) The activation of each hidden layer is computed successively in the feed-forward processing.

(2) Firstly, in the output layer, the grade error is calculated by the following function:

$$G^{(n)} = -\left(y - \delta\left(z^{(n)}\right)\right) \cdot \delta'\left(z^{(n)}\right); \tag{1}$$

(3) The grade error of each hidden layer is as follows:

$$G^{(l)} = \left(\left(W^{(l)}\right)^T G^{(l+1)}\right) \cdot \delta'\left(z^{(l)}\right); \tag{2}$$

(4) The partial derivatives are

$$\begin{aligned} \nabla_{W^{(l)}} J(W, b; x, y) &= G^{(l+1)}\left(\delta\left(z^{(l)}\right)\right)^T, \\ \nabla_{b^{(l)}} J(W, b; x, y) &= G^{(l+1)}, \end{aligned} \tag{3}$$

where $z = WT * x + b$, W is the set of weights, x is the input, b is the offset value, y is the expected output, $\delta(\cdot)$ is the

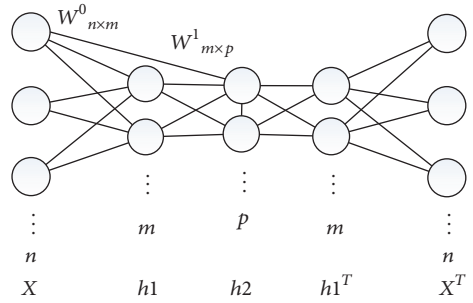


Figure 2: The chaotic matrix generation network.

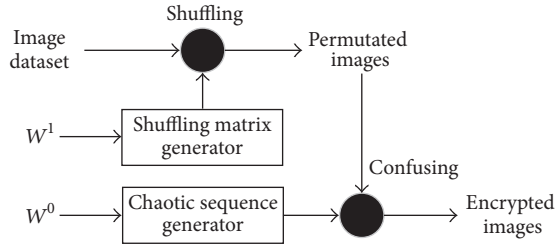


Figure 3: The process of batch image encryption.

sigmoid function which will be described in the following, $\delta'(\cdot)$ is the derivative of $\delta(\cdot)$, and the superscript is the number of a certain layer.

$$\delta(z) = \frac{1}{1 - e^{-z}}. \tag{4}$$

Then ∇_W and ∇_b are used to tune W and b. And steps (1) to (4) are repeated until the network is close to the optimal solution.

The SAE network can learn levels of features. And given the same hyperparameters, the extracted feature information can be reproduced. Sequences composed of those features satisfy the basic requirements of chaos and can be utilized to encrypt images. In this paper, we extracted two sets of features to encrypt images in batch using a five-layer neural network based on SAE, and experimental results show that our scheme is effective and efficient.

3. The Batch Image Encryption Scheme

3.1. The Chaotic Matrix Generation Network. In order to obtain practicable chaotic matrices, we modified the SAE model in Figure 1 by removing the offset neurons and constructed a chaotic matrix generation network (Figure 2). As mentioned in Section 2, $X/h1/h2/h1^T/X^T$ is the layer name which also denotes the input/output value of the layer. X is the input layer with n neurons, $h1$ is the first hidden layer with m neurons, $h2$ is the second hidden layer with p neurons, $h1^T$ and X^T are mirror layers of $h1$ and X, respectively, W^0 is the weight matrix between X and $h1$, W^1 is the weight matrix between $h1$ and $h2$, and the weight matrices between $h2$, $h1^T$, and X^T are not marked in the figure; they are transposed matrices of W^0 and W^1, respectively. The procedures of the chaotic matrix generation are described as follows.

(1) Parameter Initialization. The pixel components of a given image compose a vector and the values in the vector are normalized as float point numbers in the fixed range of 0 to 1; W^0 and W^1 are initialized using a cipher code C which is a decimal between 0 and 1; n is the number of the image pixel components; m is the number of images to be encrypted; p is a preappointed number that is not equal to m, such as $p = m - 1$; and the iteration times is predefined as it_num; accordingly, these parameters are prepared for the following chaotic matrix generation, where C, p, and it_num are cipher codes.

(2) Chaotic Matrix Generation. Given the set of cipher codes, the network is trained it_num times, and two chaotic matrices are obtained which are W^1 and W^0; W^1 is used to produce the set of chaotic sequences and W^0 is used to produce the total shuffling matrix, which will be used in the following subsection.

3.2. Batch Image Encryption. As we can see in Figure 3, W^1 and W^0 feed into two generators, respectively, and then a total shuffling matrix and a set of chaotic sequences are obtained; in addition, there are two operations where one with the total shuffling matrix shuffles each image of the image dataset and another with each chaotic sequence confuses the corresponding permutated image; finally, every image is encrypted. The detailed encryption procedures are as follows.

(1) Image Dataset Preparation. We assume that all plain images have the same number of pixels. The assumption

conforms to the rules of most image processing tasks such as real-time telemedicine. And each image has n pixels.

(2) Total Shuffling Matrix Generation. W^1 is an m by p matrix; that is, it has $m * p$ digital numbers, which is converted to a sequence as $\{x_1, x_2 \cdots x_{m*p}\}$. Then let

$$\mathrm{sl} = \mathrm{mod}\left(\mathrm{abs}\left(x_i\right) \times 10^4, n\right), \quad i = 1, 2 \cdots m * p. \tag{5}$$

Obviously, sl is an integer in the range of 0 to $n - 1$, to do iteration of (5) until we get n different integers. If W^1 is exhausted before we get n different integers, the updated W^1 at previous iteration could be used and it_num could be adjusted accordingly. Finally, the n different integers make a matrix that has the same dimension of the plain image.

(3) Chaotic Sequence Generation. W^0 is an n by m matrix; that is, it produces m n-length sequences which can be used to encrypt m plain images. To confuse each image with an independent chaotic sequence, we set m as the number of images in the image dataset, and each column of W^0 makes an n-length chaotic sequence; then we get the set of independent chaotic sequences.

(4) Shuffling the Plain Image. With the total shuffling matrix, each plain image is shuffled. Pixel values of the plain image are changed according to the number of the corresponding position in the total shuffling matrix. For example, with respect to a 512 by 512 image, if it is 522 in the position [1, 1] of the total shuffling matrix, the pixel value in the position [1, 1] of the plain image will be replaced by the pixel value in the position [2, 10] of the plain image.

(5) Confusing the Relationship between the Permutated Image and Encrypted Image. With each of the chaotic sequences, the corresponding permutated image is confused. The confusion process is mathematically abstracted as follows, where C is the confused one, V is the permuted one, w is a chaotic sequence, bitxor(·) is an element-wise XOR function:

$$C = \mathrm{bitxor}\left(V, w\right). \tag{6}$$

Now, we get the encrypted images.

The following pseudocode is presented to depict the process of batch image encryption.

```
# Shuffling the plain image
shuffled_images = [ ]
for image in plain_images:
    shuffled_images.append(shuffling(image,
    sl_matrix))
# Confusing the permutated image
encrypted_images = [ ]
for pimage, index in enumerate(shuffled_images):
    encrypted_images.append(confusing(pimage,
    chaotic_sequences[index]))
return encrypted_images
```

Here, sl_matrix is the total shuffling matrix, chaotic_sequences is the set of chaotic sequences, the function shuffling(·) is described as step (4), and the function confusing(·) is as (6).

4. Experiments and Results

4.1. Experimental Settings. This encryption scheme was evaluated over a group of gray-level images. Those images have the same dimension of 512 by 512 and are all 256 gray-level. Pixel values were normalized, that is, each image was represented by a $512 * 512$ matrix and each element of the matrix had a decimal value in the range of 0 to 1. The iteration number it_num is 10 and one of the images was selected as the cipher code. In order to facilitate the presentation, we only discussed the encryption effect of four randomly selected images. Figure 4 lists the plain images and the corresponding encrypted images and decrypted images, and the plain and the decrypted images are almost identical. At the next subsection, the encryption effect was verified through several evaluation methods, and comparative study was performed with other image encryption schemes; besides, we applied the proposed scheme to encrypt four color images, and the plain, encrypted, and decrypted images are listed in the figure. In Section 4.3, a comparative study on running time was performed with the prevalent image encryption scheme—"logistic map."

4.2. Performance and Security Analysis

(1) Key Space Analysis. Any encryption scheme should have a large-enough key space to resist brute-force attack. The cipher codes of the proposed scheme are the following: key = $\{C, p, \mathrm{it\_num}\}$, where C—the calculation precision—is 10^{16} by default, p is the size of third layer in Figure 2 and it approximates 10^5 in this experimental task, and it_num is the iteration times which is 10. Hence, the key space is 10^{21}. With so huge a key space, it is extremely hard to crack through a brute-force effort.

(2) Histogram Analysis. The plain image has rich-semantic information and it can be read because of the frequency of each gray-level. Hence, a strong encryption scheme should reduce such statistical information as much as possible. Histograms of the plain and the encrypted images are listed in Figure 5. The histogram of each encrypted image is pretty uniform and does not reveal any statistical information.

(3) Correlation Analysis. Similar to the histogram, the correlation reveals statistical information of the image; besides, it reflects the relationship between adjacent pixels. An encryption scheme should decrease the correlations of the image. The correlation coefficient is a quantitative indicator reflecting the correlation of a pair of adjacent pixels and is mathematically abstracted as follows [28]:

$$\mathrm{cor}_{xy} = \frac{\mathrm{cov}\left(x, y\right)}{\sqrt{V\left(X\right)}\sqrt{V\left(y\right)}},$$

$$M\left(x\right) = \frac{1}{N}\sum_{i=1}^{N} x_i,$$

Figure 4: Image encryption and decryption.

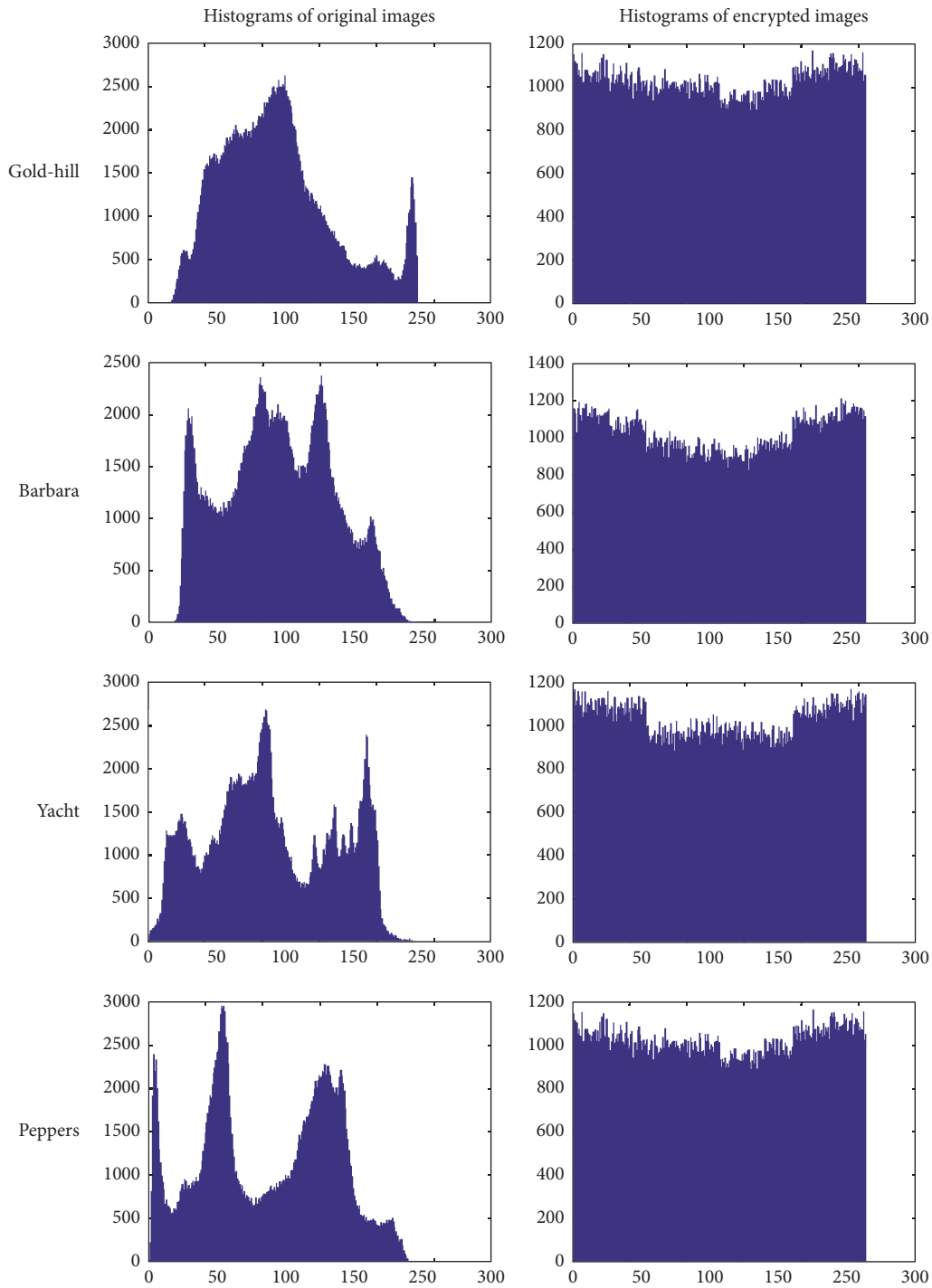


Figure 5: Histograms of original and encrypted images.

$$V(x) = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - M(x)\right)^2,$$

$$\operatorname{cov}(x, y) = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - M(x)\right)\left(y_i - M(y)\right), \tag{7}$$

where N is the number of selected pixels in the image, x and y are values of two adjacent pixels, cor_{xy} is the correlation coefficient of x and y, $M(\cdot)$ is the mean function, $V(\cdot)$ is the variance function, and $\mathrm{cov}(\cdot)$ is the covariance function.

We selected ten pairs of adjacent pixels in each plain and encrypted image, and we listed the correlation coefficients of these pairs in Table 1. There is rich-semantic information

Table 1: Correlation analysis.

Images	Correlation coefficients				
Plain Gold-hill	0.9871	0.9833	0.9847	0.9865	0.9872
	0.9841	0.9852	0.9835	0.9877	0.9928
Plain Barbara	0.9859	0.9828	0.9852	0.9874	0.9865
	0.9888	0.9849	0.9843	0.9857	0.9873
Plain Yacht	0.9856	0.9847	0.9865	0.9838	0.9869
	0.9868	0.9838	0.9856	0.9877	0.9838
Plain Peppers	0.9856	0.9871	0.9835	0.9875	0.9873
	0.9849	0.9862	0.9864	0.9849	0.9882
Encrypted Gold-hill	0.0038	−0.0044	0.0182	−0.0054	−0.0231
	0.0085	0.0057	−0.0248	0.0058	0.0105
Encrypted Barbara	0.0047	0.0157	0.0287	0.0108	0.0091
	−0.0092	0.0038	−0.0217	−0.0098	0.0124
Encrypted Yacht	−0.0157	−0.0134	0.0158	0.0064	−0.0158
	−0.0085	0.0068	−0.0263	0.0097	−0.0215
Encrypted Peppers	−0.0074	0.0238	0.0398	0.0087	0.0378
	−0.0159	−0.0105	0.0485	−0.0182	−0.0082

in the plain image, so adjacent pixels of it are strongly correlated and the coefficient value is high; on the contrary, the correlation between adjacent pixels of the encrypted image is deeply decreased and the coefficient value is very small.

The correlation distribution of an image is commonly depicted using a scattered diagram. We randomly selected 1000 pairs of adjacent pixels from each image in vertical, horizontal, and diagonal directions, and the correlation is illustrated in Figure 6. As we can see, the correlation is strong for the plain image in any direction, and the encryption scheme conceals the significant characteristics of the plain image in three directions.

(4) Sensitivity Analysis. The encrypted image would not be decrypted with a tiny change in the cipher code; that is, the encryption scheme is robust. In the experiment, we changed the cipher code C by increasing a very small number, and the changed cipher code $C' = C + 10^{-15}$. The plain image was encrypted using C and then decrypted using C'. The encrypted images are listed in Figure 4 and the decrypted images are shown in Figure 7. The encrypted ones completely cannot be identified. That is to say, our scheme is very sensitive.

(5) Differential Attack. Differential attack would fail if the encrypted image is significantly different with a tiny change in the plain image. The following measures are usually used to measure this capability: number of pixels change rate (NPCR) and unified average changing intensity (UACI) [29, 30].

$$\mathrm{NPCR} = \frac{\sum_{i,j} \mathrm{Dist}(i,j)}{\mathrm{Wid} \times \mathrm{Hei}} \times 100\%,$$

$$\mathrm{UACI} = \frac{1}{\mathrm{Wid} \times \mathrm{Hei}} \left[\sum_{i,j} \frac{|M_1(i,j) - M_2(i,j)|}{255} \right] \times 100\%,$$

$$\mathrm{Dist}(i,j) = \begin{cases} 1, & \text{if } M_1(i,j) \neq M_2(i,j), \\ 0, & \text{if } M_1(i,j) = M_2(i,j), \end{cases} \tag{8}$$

where M_1 and M_2 are two encrypted images whose original plain images have only one different pixel, $M_1(i, j)$ and $M_2(i, j)$ are the pixel values at position (i, j) of the encrypted images, respectively, and Hei and Wid are the size of the height and the width of the image, respectively.

According to the two measures, our scheme is compared with several other encryption schemes and results are listed in Table 2. Of our scheme, the NPCR is more than 99% and the UACI is more than 33%, which proves that the proposed scheme is sensitive.

In addition to the gray-level images, our scheme is also used to encrypt color images. The color image can be considered to be three channels overlapping together which are Red, Green, and Blue channels. Each channel is like a gray-level image. The three channels are encrypted, respectively, and then merged together to become an encrypted color image. Figure 8 lists four color images and their encrypted and decrypted images.

4.3. Comparative Study on Running Time. The comparative study on running time is performed with "logistic map" [31], which has been widely used to encrypt images. To simplify the comparative process, we just take into account the calculation process of generating the total shuffling matrix and generating the chaotic sequences; and the consumption time of the encryption process is stable, so it is excluded. Table 3 lists the comparative results. As we can see, the running time of the logistic map increases with the increment of the generated sequences; on the contrary, the running time of our scheme decreases slightly when more than 50 sequences are generated. We think it is the following reasons that cause the result: the logistic map generates the shuffling

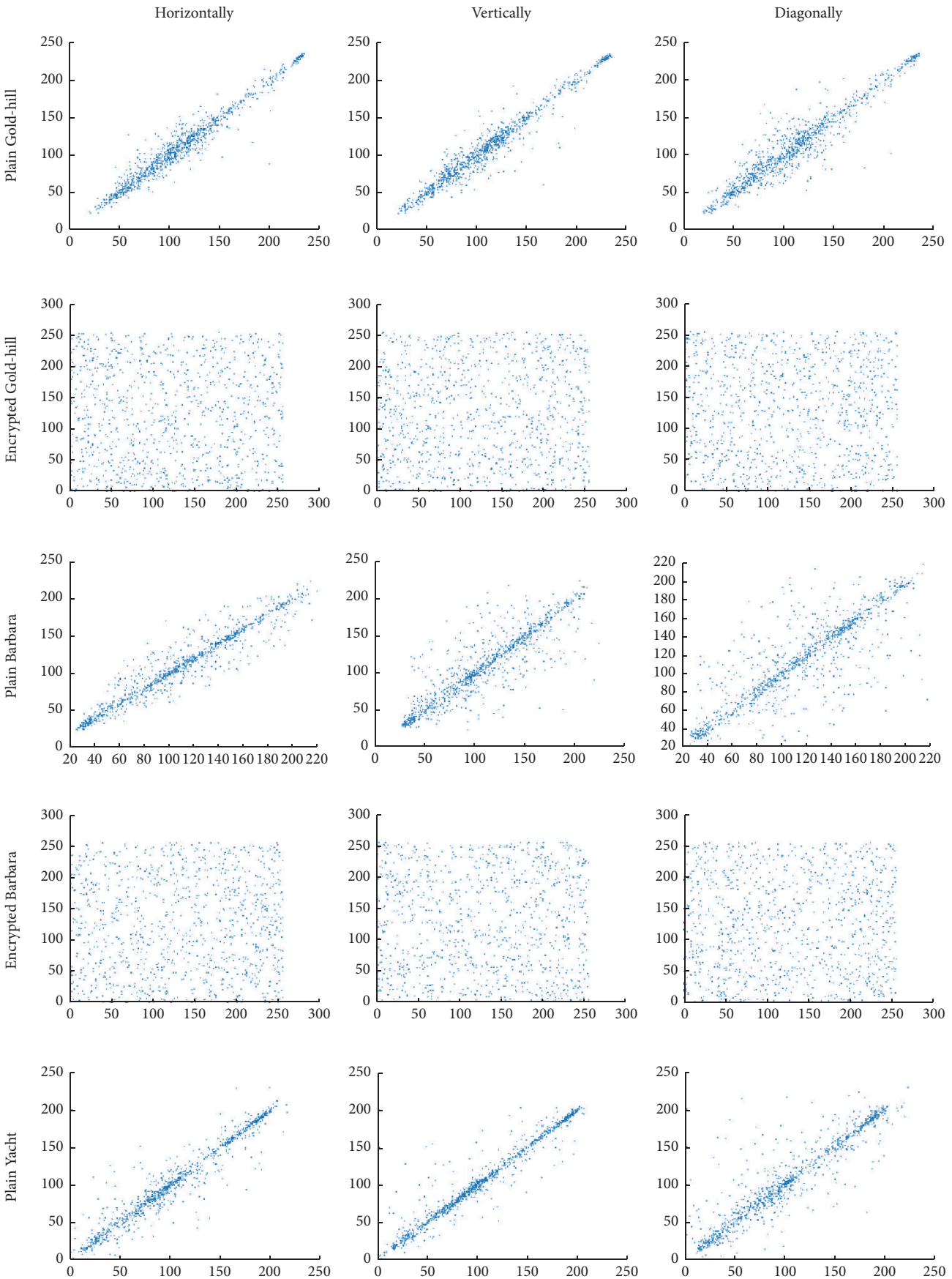


Figure 6: Continued.

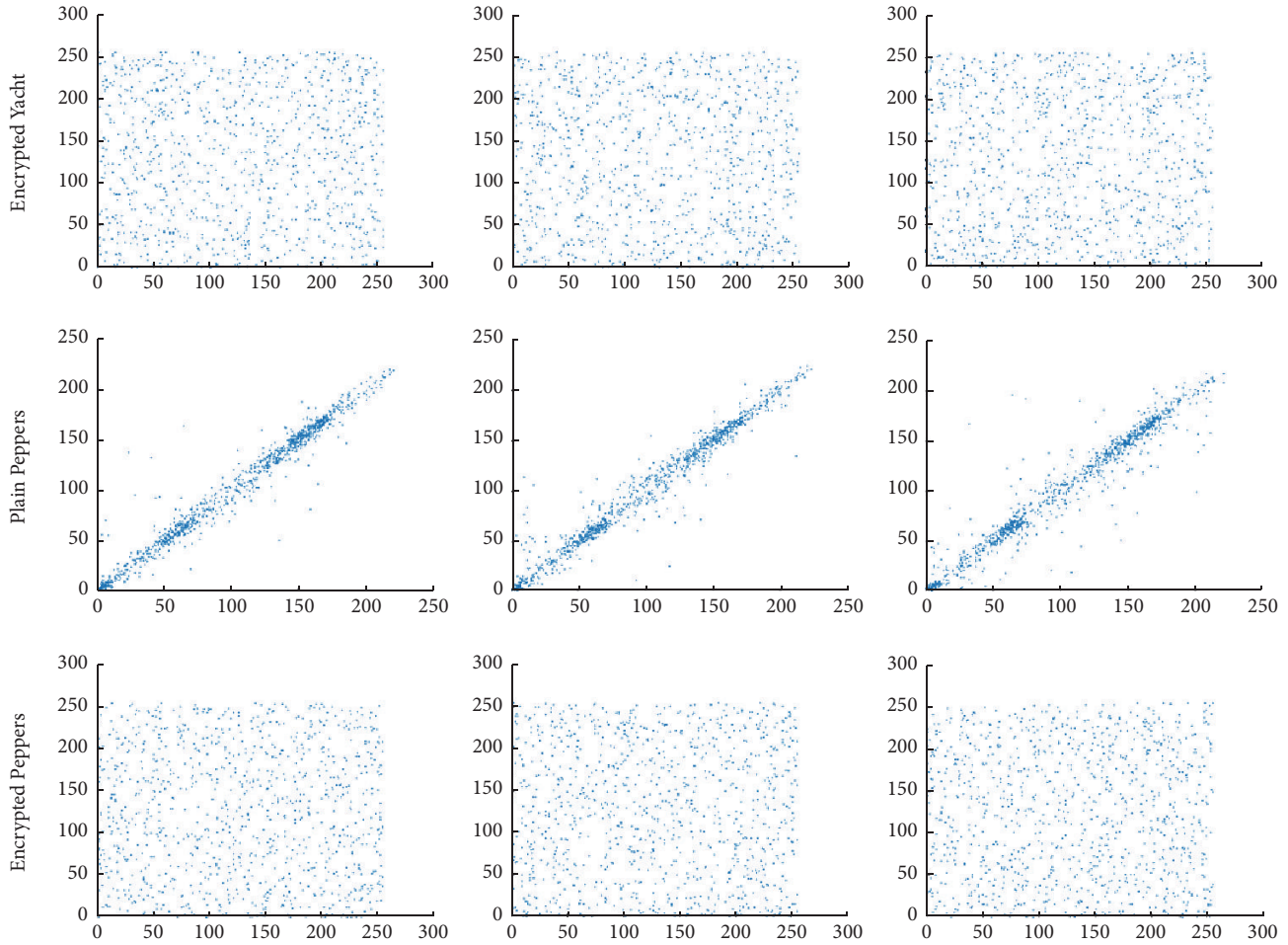


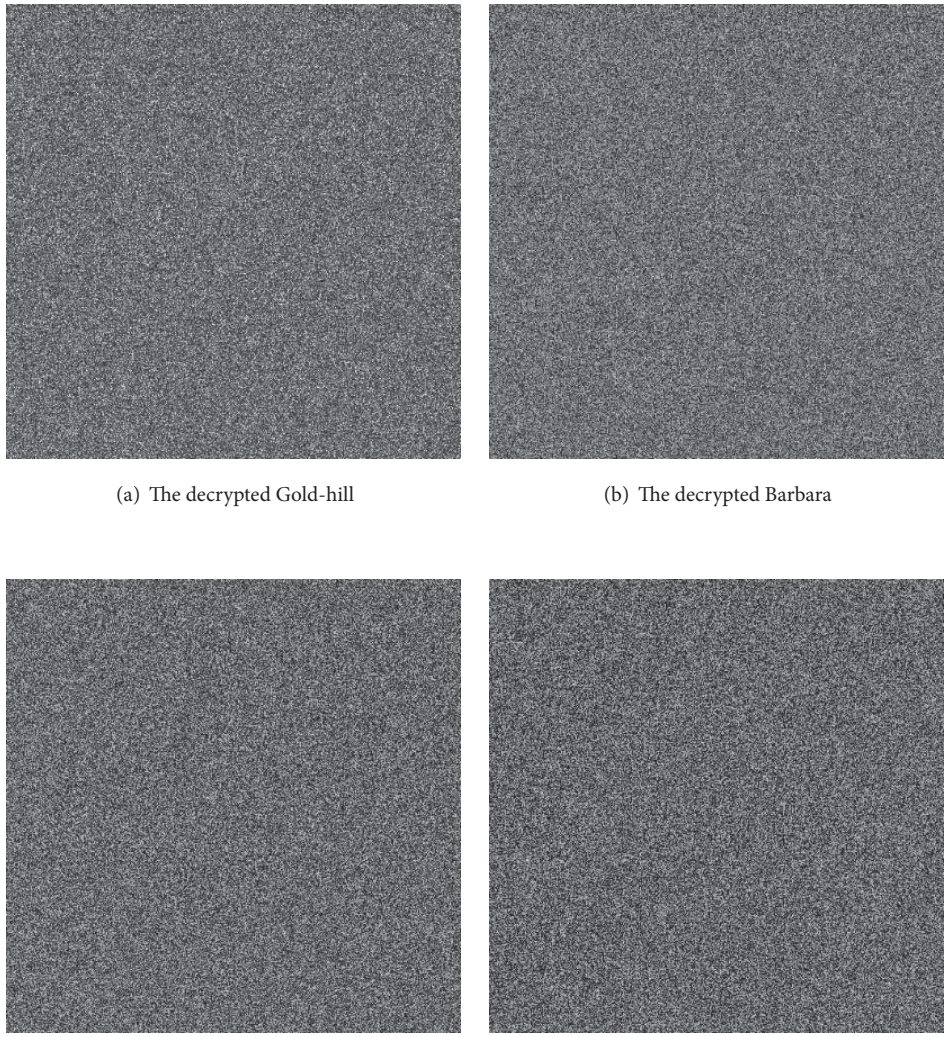
Figure 6: Correlation distribution of two adjacent pixels.

Table 2: Comparative study of NPCR and UACI of the proposed algorithm to some existing algorithms.

Images	NPCR (%)	UACI (%)	Images	NPCR (%)	UACI (%)
Ref. [18] (Lena)	99.89	33.68	Ref. [19] (Lena)	99.61	33.48
Ref. [20] (Lena)	99.6972	33.5086	Ref. [21] (Lena)	99.6086	33.4273
Ref. [22] (Lena)	99.6000	33.5000	Ref. [23] (Lena)	99.6100	33.4400
Ref. [24] (Lena)	99.5983	33.6883	Ref. [25] (Lena)	99.5865	33.4834
Ref. [26] (Lena)	99.6399	33.5916	Ref. [27] (Lena)	99.5926	33.3386
Ours (Gold-hill)	99.6108	33.5852	Ours (Barbara)	99.6432	33.5454
Ours (Yacht)	99.4968	33.6011	Ours (Peppers)	99.4205	33.3396

Table 3: Comparative study on running time of the proposed scheme and the logistic map.

Schemes		Number of generated sequences			
		10	50	100	200
Ours (seconds)	Total time	483.2845	782.8963	546.7157	523.7215
	Average time	48.3285	15.6579	5.4672	2.6186
Logistic map (seconds)	Total time	6314.5490	6373.6024	6449.8472	6601.2951
	Average time	631.4549	127.4720	64.4985	33.0065



(a) The decrypted Gold-hill

(b) The decrypted Barbara

(c) The decrypted Yacht

(d) The decrypted Peppers

Figure 7: The decrypted images with a changed cipher code.

matrix and the chaotic sequences by linear calculation, and most time (almost 6300 seconds) is consumed on the generation of the shuffling matrix; our scheme does the work in parallel: the running time of the generation of the shuffling matrix decreases with the increment of the generated sequences.

5. Conclusions and Future Work

Our scheme has the following advantages: fast batch image encryption and independent chaotic sequence for each image, which ensures the efficiency and security of our scheme. The experimental results prove that our scheme can be used for batch image encryption in variant applications as real-time remote image transmission and image protection.

There is a drawback to our scheme that all plain images must have the same size ($512 * 512$ pixels in the experiments). In the future, we would seek to generate multilength shuffling matrices and chaotic sequences for different sizes of images. We would like, specially, to generate a large-enough shuffling matrix, from which the minor size of matrix would be extracted to shuffle the corresponding size of images.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by Scientific and Technological Research Program of Chongqing Municipal Education Commission (no. KJ1501405, no. KJ1501409); Scientific and Technological Research Program of Chongqing University

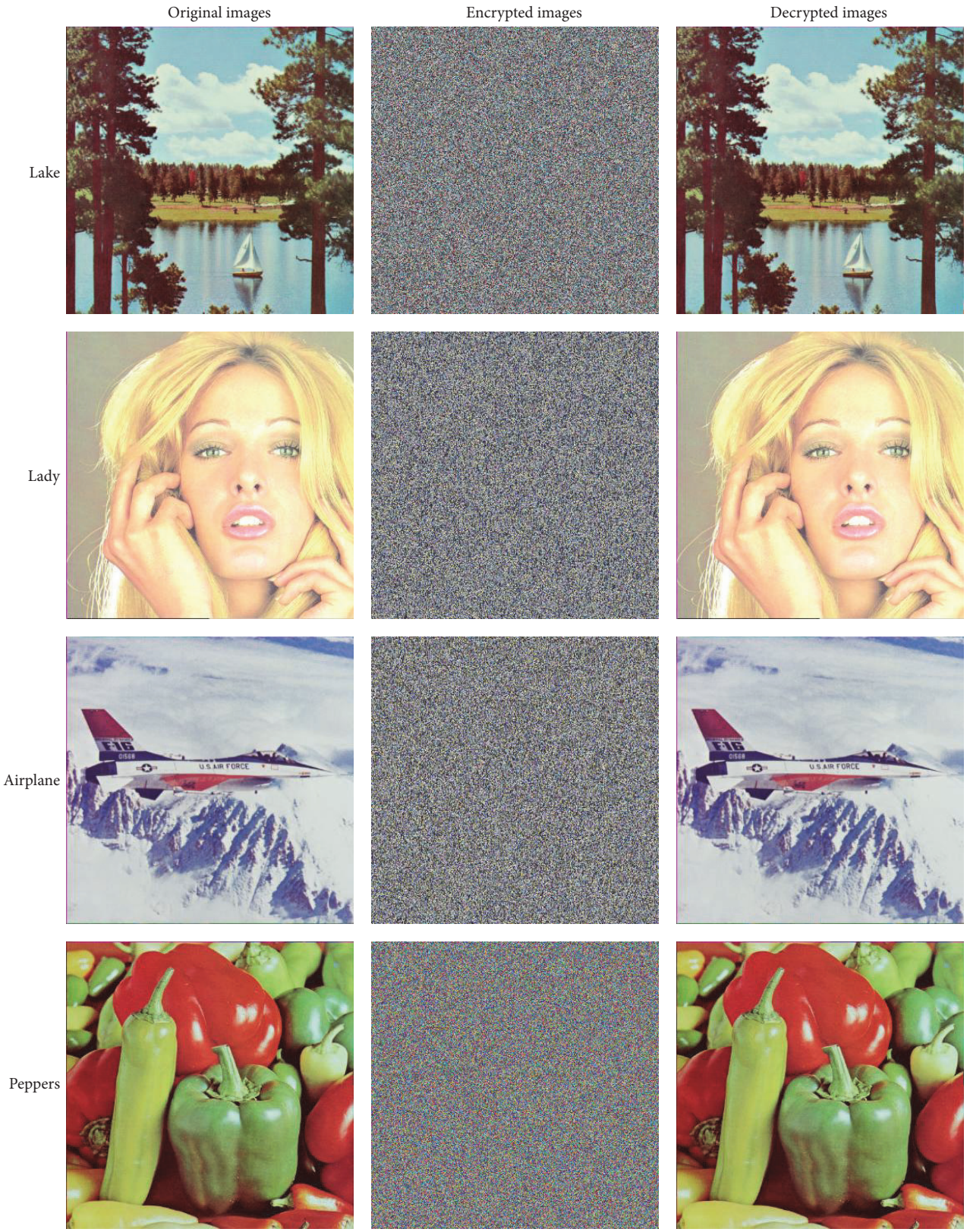


Figure 8: Color image encryption and decryption.

of Education (no. KY201522B, no. KY201520B); and Fundamental Research Funds for the Central Universities (no. XDJK2016E068).

References

[1] S. Li, G. Chen, and X. Zheng, "Chaos-based encryption for digital images and videos," in *Multimedia Security Handbook*, Kirovski, Ed., vol. 4, CRC Press, 2004.

[2] S. Zhou, Q. Zhang, X. Wei, and C. Zhou, "A summarization on image encryption," *IETE Technical Review*, vol. 27, no. 6, pp. 503–510, 2010.

[3] E. Inzunza-González and C. Cruz-Hernández, "Double hyperchaotic encryption for security in biometrie systems," *Nonlinear Dynamics and Systems Theory*, vol. 13, no. 1, pp. 55–68, 2013.

[4] H. Liu and X. Wang, "Color image encryption based on one-time keys and robust chaotic maps," *Computers and Mathematics with Applications*, vol. 59, no. 10, pp. 3320–3327, 2010.

[5] K.-W. Wong, B. S.-H. Kwok, and W.-S. Law, "A fast image encryption scheme based on chaotic standard map," *Physics Letters A*, vol. 372, no. 15, pp. 2645–2652, 2008.

[6] M. Usama, M. K. Khan, K. Alghathbar, and C. Lee, "Chaos-based secure satellite imagery cryptosystem," *Computers & Mathematics with Applications*, vol. 60, no. 2, pp. 326–337, 2010.

[7] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps," *International Journal of Bifurcation and Chaos in Applied Sciences and Engineering*, vol. 8, no. 6, pp. 1259–1284, 1998.

[8] Z.-L. Zhu, W. Zhang, K.-W. Wong, and H. Yu, "A chaos-based symmetric image encryption scheme using a bit-level permutation," *Information Sciences*, vol. 181, no. 6, pp. 1171–1186, 2011.

[9] C. Fu, B.-B. Lin, Y.-S. Miao, X. Liu, and J.-J. Chen, "A novel chaos-based bit-level permutation scheme for digital image encryption," *Optics Communications*, vol. 284, no. 23, pp. 5415–5423, 2011.

[10] W. Zhang, K.-W. Wong, H. Yu, and Z.-L. Zhu, "A symmetric color image encryption algorithm using the intrinsic features of bit distributions," *Communications in Nonlinear Science & Numerical Simulation*, vol. 18, no. 3, pp. 584–600, 2013.

[11] X. Wang and D. Luan, "A novel image encryption algorithm using chaos and reversible cellular automata," *Communications in Nonlinear Science & Numerical Simulation*, vol. 18, no. 11, pp. 3075–3085, 2013.

[12] Y. Zhang and D. Xiao, "An image encryption scheme based on rotation matrix bit-level permutation and block diffusion," *Communications in Nonlinear Science & Numerical Simulation*, vol. 19, no. 1, pp. 74–82, 2014.

[13] Y. Mao, G. Chen, and S. Lian, "A novel fast image encryption scheme based on 3D chaotic baker maps," *International Journal of Bifurcation & Chaos*, vol. 14, no. 10, pp. 3613–3624, 2004.

[14] O. Mirzaei, M. Yaghoobi, and H. Irani, "A new image encryption method: parallel sub-image encryption with hyper chaos," *Nonlinear Dynamics*, vol. 67, no. 1, pp. 557–566, 2012.

[15] W. Zhang, K.-W. Wong, H. Yu, and Z.-L. Zhu, "An image encryption scheme using reverse 2-dimensional chaotic map and dependent diffusion," *Communications in Nonlinear Science & Numerical Simulation*, vol. 18, no. 8, pp. 2066–2080, 2013.

[16] C. Fu, J.-J. Chen, H. Zou, W.-H. Meng, Y.-F. Zhan, and Y.-W. Yu, "A chaos-based digital image encryption scheme with an improved diffusion strategy," *Optics Express*, vol. 20, no. 3, pp. 2363–2378, 2012.

[17] C. Zhu, "A novel image encryption scheme based on improved hyperchaotic sequences," *Optics Communications*, vol. 285, no. 1, pp. 29–37, 2012.

[18] M. Saberikamarposhti, D. Mohammad, M. S. M. Rahim, and M. Yaghobi, "Using 3-cell chaotic map for image encryption based on biological operations," *Nonlinear Dynamics*, vol. 75, no. 3, pp. 407–416, 2014.

[19] Y. Wang, K.-W. Wong, X. Liao, T. Xiang, and G. Chen, "A chaos-based image encryption algorithm with variable control parameters," *Chaos, Solitons & Fractals*, vol. 41, no. 4, pp. 1773–1783, 2009.

[20] J.-X. Chen, Z.-L. Zhu, C. Fu, and H. Yu, "An improved permutation-diffusion type image cipher with a chaotic orbit perturbing mechanism," *Optics Express*, vol. 21, no. 23, pp. 27873–27890, 2013.

[21] Y. Zhang, D. Xiao, Y. Shu, and J. Li, "A novel image encryption scheme based on a linear hyperbolic chaotic system of partial differential equations," *Signal Processing: Image Communication*, vol. 28, no. 3, pp. 292–300, 2013.

[22] Q. Ding, Z.-M. Lu, and X.-J. Sun, "Image encryption based on neural network cipher," *Acta Electronica Sinica*, vol. 32, no. 4, pp. 677–679, 2004.

[23] S. Lian, "A block cipher based on chaotic neural networks," *Neurocomputing*, vol. 72, no. 4–6, pp. 1296–1301, 2009.

[24] N. Bigdeli, Y. Farid, and K. Afshar, "A robust hybrid method for image encryption based on Hopfield neural network," *Computers and Electrical Engineering*, vol. 38, no. 2, pp. 356–369, 2012.

[25] W. Jian, "The research of neural network mixed with image encryption based on chaotic encryption technology," *Advances in Information Sciences and Service Sciences*, vol. 4, no. 9, pp. 1–8, 2012.

[26] A. Fischer and C. Igel, "Training restricted Boltzmann machines: an introduction," *Pattern Recognition*, vol. 47, no. 1, pp. 25–39, 2014.

[27] P. P. Sarangi, A. Sahu, and M. Panda, "A hybrid differential evolution and back-propagation algorithm for feedforward neural network training," *International Journal of Computer Applications*, vol. 84, no. 14, pp. 1–9, 2013.

[28] G. Chen, Y. Mao, and C. K. Chui, "A symmetric image encryption scheme based on 3D chaotic cat maps," *Chaos, Solitons and Fractals*, vol. 21, no. 3, pp. 749–761, 2004.

[29] H. S. Kwok and W. K. S. Tang, "A fast image encryption system based on chaotic maps with finite precision representation," *Chaos, Solitons & Fractals*, vol. 32, no. 4, pp. 1518–1529, 2007.

[30] J. Peng, D. Zhang, and X. Liao, "A digital image encryption algorithm based on hyper-chaotic cellular neural network," *Fundamenta Informaticae*, vol. 90, no. 3, pp. 269–282, 2009.

[31] N. K. Pareek, V. Patidar, and K. K. Sud, "Image encryption using chaotic logistic map," *Image and Vision Computing*, vol. 24, no. 9, pp. 926–934, 2006.