Expert Systems With Applications 257 (2024) 124891

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

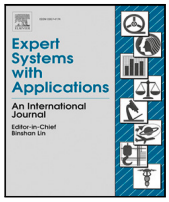


Image privacy protection scheme based on high-quality reconstruction DCT compression and nonlinear dynamics

Yiting Lin [a,b], Zhiyu Xie [a,b], Tingting Chen [a], Xiyuan Cheng [c], Heping Wen [a,b,*]

[a] University of Electronic Science and Technology of China, Zhongshan Institute, Zhongshan 528402, China
[b] School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[c] School of Automation, Guangdong University of Technology, Guangzhou 510006, China

ARTICLE INFO

Keywords:
Image compression
Image encryption
Multimedia security
Image processing
Information security

ABSTRACT

The protection of digital image privacy in Big Data environment is a hot issue of increasing concern. To address the contradiction between efficiency and security for privacy protection, this paper proposes a high-quality restored image privacy protection scheme based on Discrete Cosine Transform (DCT) frequency domain compression and nonlinear dynamics. First, the RGB space of the color image is converted to YCbCr domain with 4:2:0 sampling. Then, the image is divided into sub-blocks in the spatial domain, followed by block-wise DCT and quantization into frequency domain information. Next, both Alternating Current (AC) and Direct Current (DC) coefficients of all sub-blocks are individually extracted and subjected to compression encoding. Finally, a Rubik's Cube permutation and filtering diffusion encryption algorithm based on a nonlinear dynamical chaotic system is designed. Moreover, a mechanism for generating dynamic chaotic sequences associated with plaintext is introduced to effectively counter cryptographic attacks. The combined approach of compression and encryption significantly reduces computational complexity while improving encryption efficiency. Through experimental simulation results, the compression encryption scheme demonstrates high compression ratios as well as excellent image recovery quality while providing enhanced security against common cryptographic attacks. The work in this paper provides a preferred joint compression and encryption technical solution for digital image privacy protection in Big Data network.

0. Introduction

The advent of the era of big data has led to a significant increase in the complexity of information transmission, with the amount of data and diversity reaching unprecedented levels (Feng et al., 2024; Li et al., 2023; Qian et al., 2023). The security of digital images has gained significant attention due to their pivotal role as a medium for transmitting information in the current era of information explosion (Erkan et al., 2023; Toktas et al., 2023; Wang & Lo, 2024). Consequently, the reliable encrypted transmission of digital images becomes a matter of great importance. However, digital image information possesses distinctive characteristics that set it apart from traditional text information. These characteristics encompass high levels of information redundancy, as evidenced by Ref. Toktas, Erkan, Gao et al. (2024), strong correlation between pixels, as evidenced by Ref. Akbacak et al. (2023), and a discrete distribution of key information, as evidenced by Ref. Gao, Iu et al. (2024). As a result, the encryption methods used for traditional text information cannot be directly applied to digital images (Gao, Wu et al., 2023; Wen, Lin et al., 2024; Zhang, Ji et al., 2024). In addition, traditional global full encryption methods have become somewhat outdated in the current big data era (Li, Shen et al., 2024; Li et al., 2017; Zhang, Zhu et al., 2024). Therefore, the in-depth study of efficient digital image encryption algorithms is imminent (Gong et al., 2024; Ye, Du et al., 2023; Ye, Liu et al., 2023). In this context, the research of digital image encryption is not only to meet the current needs (Wen, Huang et al., 2024; Wen, Yuan et al., 2024), but also to better adapt to the evolving data environment in the future.

Considering the current status of international research, in an effort to address the efficiency and security challenges of digital image transmission in the era of big data (li Chai et al., 2023; Lu et al., 2023; Zhang et al., 2023), several scholars have made significant contributions to the fields of image encryption and compression (Chai et al., 2024; Chen et al., 2024; Mou et al., 2024). In recent years, numerous researchers have dedicated themselves to this area of study, leading to notable achievements (Gao et al., 2023; Kocak et al., 2023; Wen, Jiang et al.,

* Corresponding author at: University of Electronic Science and Technology of China, Zhongshan Institute, Zhongshan 528402, China.
E-mail addresses: YitingLin@zsc.edu.cn (Y. Lin), Zhiyuxie@stu.zsc.edu.cn (Z. Xie), 2006022024@stu.zsc.edu.cn (T. Chen), 2112304365@mail2.gdut.edu.cn (X. Cheng), wenheping@uestc.edu.com (H. Wen).

https://doi.org/10.1016/j.eswa.2024.124891
Received 11 May 2024; Received in revised form 29 June 2024; Accepted 24 July 2024
Available online 27 July 2024

2024). In 2021, Ref. Wang et al. (2021) introduced a novel triple image encryption and hiding algorithm based on 2D chaotic systems, compressed sensing, and 3D-DCT. Simulation experiments and comparative analysis illustrated that the proposed visual encryption scheme exhibits excellent visual security, robustness, decryption quality, and operational efficiency. Similarly, in 2022, Ref. Yamni et al. (2022) proposed a robust, secure, and blind audio watermarking scheme based on a novel hybrid transform, which combines the dual tree complex wavelet transform and the fractional Charlier moment transform. A comparative analysis confirmed the high robustness, security, and efficiency of the proposed watermarking method. Furthermore, in 2023, Ref. Wen, Huang et al. (2023) put forward a high-quality color image compression encryption scheme grounded in chaos and block arrangement. The results showcased the advantages of the joint compression encryption scheme, highlighting a high compression ratio, superior image restoration quality, and a very high security level, capable of resisting common password attacks. Subsequently, in 2024, Ref. Cao et al. (2023) proposed a universal screen-shotting robust image watermarking scheme designed to embed extractable information into on-screen images for copyright protection or additional information acquisition. Experimental results displayed that the proposed scheme outperformed some state-of-the-art schemes in terms of watermark embedding invisibility and watermark extraction accuracy. The research in digital image encryption has yielded fruitful outcomes (Jamal et al., 2023; Toktas, Erkan, Yetgin et al., 2024; Wen, Fan et al., 2023; Wen, Huang et al., 2023), resulting in the introduction of numerous encryption algorithms (Abdelfatah, 2024; Abdo et al., 2024; Devi et al., 2023; ur Rehman, 2024). It is evident from these findings that the majority of encryption algorithms have demonstrated satisfactory outcomes (Li, Cheng et al., 2024; Saberikamarposhti et al., 2024; Singh et al., 2024a; Tanaka, 2023), significantly advancing information security technology in certain aspects (Khalili et al., 2024; Sardar et al., 2024; Singh et al., 2024b; Singh & Singh, 2024). Regrettably, many research findings in the era of big data are subject to temporal limitations: (1) Although comprehensive encryption, iterative encryption, and multi-round encryption methods effectively enhance encryption quality, they present limitations in the era of big data due to issues such as low efficiency in encryption and high redundancy of information. (2) Evaluating encryption performance relies on image restoration quality, a crucial measure, but most studies solely compare plaintext and ciphertext without conducting systematic analyses on restored images. (3) Global or full-scale encryption, particularly for encrypting large-scale data sets, may significantly impact processing speed and response time.

With the aforementioned issues in mind, this paper proposes a high-quality image security communication scheme based on magic cube transform and image filtering diffusion. The specific innovative aspects are as follows:

(1) The proposed joint compression encryption scheme effectively reduces computational complexity by employing DCT frequency domain compression and compression coding techniques to minimize redundancy, shrink image size, and enhance overall encryption efficiency.

(2) The encryption method outlined in this article introduces a plaintext correlation mechanism to generate dynamic chaotic sequences, effectively resisting various cryptographic attacks. Targeted security enhancements will be implemented to counter cryptographic attacks, building on our existing cryptanalysis research base (Wen, Chen et al., 2023; Wen & Lin, 2023, 2024; Wen, Lin et al., 2024, 2024).

(3) The encryption scheme boasts a combination of high compression ratio, superior image recovery quality, and compact image compression size. Particularly, it exhibits exceptional recovery characteristics, especially after high compression ratio lossy compression encryption, as demonstrated by experimental results.

The subsequent sections of this paper are organized as follows: Section 1 briefly outlines the relevant theory behind the proposed scheme. Section 2 explains the precise details of our scheme. Section 3 presents experimental results and analysis discussion. The final section concludes the study.

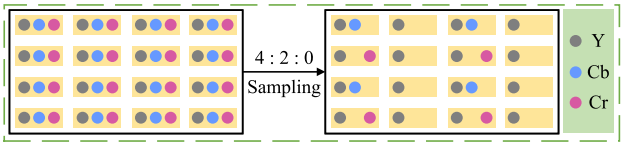


Fig. 1. The schematic diagram of chroma subsampling.

1. Related theories

1.1. Chaotic system

The security and efficiency of chaotic image encryption depend largely on the chaotic sequence. The proposed method chooses a 2D Logistic-Sine-coupling map (2D-LSCM) (Hua et al., 2018) due to its simplicity, easy implementation, and good chaotic performance. Its definition is as follows:

$$\begin{cases} x_i = \sin(\pi(4rx_{i-1}(1-x_{i-1}) + (1-r)\sin(\pi y_{i-1}))) \\ y_i = \sin(\pi(4ry_{i-1}(1-y_{i-1}) + (1-r)\sin(\pi x_i))) \end{cases} \tag{1}$$

where (x_i, y_i) are the state values produced by the ith iteration of 2D-LSCM, (x_{i-1}, y_{i-1}) are the inputs for the ith iteration, (x_0, y_0) represent the initial state values, and r is the system control parameter. The ranges of these values are all [0, 1].

1.2. Chroma subsampling

In the traditional RGB color space, an image is represented by three independent channels: red (R), green (G), and blue (B). However, to optimize compression effectiveness, chroma subsampling methods, specifically YCbCr encoding, are employed. The YCbCr color space divides image information into luminance (Y) and chrominance (Cb and Cr) components. The luminance channel (Y) captures the majority of the image's detail, while the chrominance channels (Cb and Cr) contain the color information. This separation facilitates more efficient processing of detail and color information. Consequently, the YCbCr color space is extensively utilized in image compression due to its capacity for enhancing data compression efficiency. The formula for converting RGB to YCbCr is:

$$\begin{cases} Y = 0.299R + 0.587G + 0.114B \\ Cb = 128 + (-0.168736R - 0.331264G + 0.5B) \\ Cr = 128 + (0.5R - 0.418688G - 0.081312B) \end{cases} \tag{2}$$

The formula for converting YCbCr to RGB is:

$$\begin{cases} R = Y + 1.402(Cr - 128) \\ G = Y - 0.344136(Cb - 128) - 0.714136(Cr - 128) \\ B = Y + 1.772(Cb - 128) \end{cases} \tag{3}$$

The human eye is more sensitive to variations in luminance than in chrominance. By separating the color information and compressing it to different extents, higher compression ratios can be achieved without significantly compromising image quality. In 4:2:0 chroma subsampling, the Y component maintains full resolution, while the Cb and Cr components are sampled at lower rates. Specifically, in each 2×2 pixel block, the 4:2:0 ratio ensures that the Y component is fully retained, whereas only one pixel from the Cb and Cr components is preserved, with the remaining pixels discarded. The specific operation is shown in Fig. 1.

1.3. 2D-discrete cosine transform (2D-DCT)

For an image with dimensions of $H \times W$, the 2D-DCT operation is equivalent to the 2D-DCT operation on a matrix of the same size. Therefore, an image can be considered as a real-valued matrix. The DCT coefficients of a matrix with dimensions of $M \times N$ can be obtained using the following equation:

$$C(u,v) = \frac{2}{\sqrt{MN}}\alpha(u)\alpha(v) \times \left[\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} I\left(x,y\right) \times cos\frac{(2x+1)u\pi}{2M} cos\frac{(2y+1)v\pi}{2N}\right] \tag{4}$$

where $0 \le u \le M-1$, $0 \le v \le N-1$, $x(q) = \begin{cases} 1/\sqrt{2}, q = 0 \\ 1, otherwise \end{cases}$, $I(x,y)$ represents the pixel value of the original image, and $C(u,v)$ represents the DCT coefficient.

The 2D-inverse discrete cosine transform (2D-IDCT) is the inverse process of the 2D-DCT. It can restore the coefficients to a matrix in the spatial domain. Its formula is defined by the following equation:

$$I\left(x,y\right) = \frac{2}{\sqrt{MN}}\left[\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\alpha(u)\alpha(v) \times C\left(u,v\right) \times cos\frac{(2x+1)u\pi}{2M} cos\frac{(2y+1)v\pi}{2N}\right] \tag{5}$$

The pixel values of an image are transformed from the spatial domain to the frequency domain using the DCT. In this scheme, the image is divided into blocks, each of which is processed individually with the DCT. To achieve optimal processing results and reduce computational complexity, this paper uses an 8×8 block size. Following the DCT, the low-frequency components of the image are concentrated in the first coefficient, located in the top-left corner, commonly referred to as the DC coefficient. The remaining coefficients, known as AC coefficients, represent the high-frequency components of the image. Thus, the primary objective of DCT processing is to separate low-frequency and high-frequency signals.

1.4. Image filtering diffusion

Image filtering is an image processing technique (Hua et al., 2019; Hua & Zhou, 2017) that involves convolution processing of image blocks using filters. Assuming that the size of the filter K is $(2m+1) \times (2n+1)$, the image filtering computation for each pixel of X is given by:

$$\mathbf{X}'(x,y) = \sum_{i=1}^{2m+1}\sum_{j=1}^{2n+1}\mathbf{K}(i,j)\mathbf{X}(x+i-m-1, y+j-n-1). \tag{6}$$

When using filtering techniques for frequency domain encryption, the filtering technique must be reversible because image encryption requires the restoration of the original image. This problem can be solved by setting the center element of the filter to 1 and the other elements to integers, i.e., $K(m+1, n+1) = 1$ and the other elements in K are integers. When performing image filtering on the image pixels $P(x,y)$, a matrix T of size $(2m+1) \times (2n+1)$ is obtained as follows.

$$\mathbf{T} = \begin{pmatrix} \mathbf{P}'(x-m, y-n) & \cdots & \mathbf{P}'(x-m, y) & \cdots & \mathbf{P}'(x-m, y+n) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{P}'(x, y-n) & \cdots & \mathbf{P}(x,y) & \cdots & \mathbf{P}(x, y+n) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{P}(x+m, y-n) & \cdots & \mathbf{P}(x+m, y) & \cdots & \mathbf{P}(x+m, y+n) \end{pmatrix} \tag{7}$$

Then we can apply the equation to filter the image pixels $P(x,y)$, as shown by the equation:

$$\mathbf{P}'(x,y) = \left(\sum_{i=1}^{2m+1}\sum_{j=1}^{2n+1}\mathbf{K}(i,j)\mathbf{T}(i,j)\right) \bmod K = \left(\mathbf{P}(x,y) + \sum_{(i,j)\in[1,2m+1]\times[1,2n+1]\cap(i,j)\neq(m+1,n+1)}\mathbf{K}(i,j)\mathbf{T}(i,j)\right) \bmod K \tag{8}$$

where K represents the gray level of P, from the matrix T, it can be seen that the upper-left neighboring pixels have been processed, and the lower-right neighboring pixels have been processed. The pixels are the original, unprocessed pixels. When performing the inverse operation on the current pixel, the elements of the matrix T are in the same state as the forward process. Therefore, this filtering diffusion is reversible, and its reverse operation can be obtained using the following equation.

$$\mathbf{P}(x,y) = \left(\mathbf{P}'(x,y) - \sum_{(i,j)\in[1,2m+1]\times[1,2n+1]\cap(i,j)\neq(m+1,n+1)}\mathbf{K}(i,j)\mathbf{T}(i,j)\right) \bmod K \tag{9}$$

2. Proposed image compression encryption scheme

This section presents the specific details of the encryption method. Fig. 2 shows the flow chart of the encryption method.

2.1. Chaotic initial value disturbance and sequences preprocessing

Step 1: For the plaintext image P, calculate its hash value $Hash$ using SHA256. Then, divide $Hash$ equally into 32 components, namely $Value_1$, $Value_2$, $Value_3$, …, $Value_{32}$.

Step 2: Obtain interference parameters based on the components of the hash value as follows:

$$\begin{cases} G_1^{(Value)} = (Value_1 \oplus Value_2 \oplus \cdots \oplus Value_{12}) \sum_{i=1}^{32} Value_i \\ G_2^{(Value)} = (Value_5 \oplus Value_6 \oplus \cdots \oplus Value_{16}) \sum_{i=1}^{32} Value_i \\ G_3^{(Value)} = (Value_9 \oplus Value_{10} \oplus \cdots \oplus Value_{20}) \sum_{i=1}^{32} Value_i \\ G_4^{(Value)} = (Value_{13} \oplus Value_{14} \oplus \cdots \oplus Value_{24}) \sum_{i=1}^{32} Value_i \\ G_5^{(Value)} = (Value_{17} \oplus Value_{18} \oplus \cdots \oplus Value_{28}) \sum_{i=1}^{32} Value_i \\ G_6^{(Value)} = (Value_{21} \oplus Value_{22} \oplus \cdots \oplus Value_{32}) \sum_{i=1}^{32} Value_i \end{cases} \tag{10}$$

Step 3: Add the six components $G_1^{(Value)}$, $G_2^{(Value)}$, …, $G_3^{(Value)}$ of the key $G^{(Key)}$ to get.

$$G^{(Key)} = \sum_{i=1}^{6} G_i^{(Value)} \tag{11}$$

Step 4: Generate two sets of control parameters and initial state values for iterating the 2D-LSCM.

$$\begin{cases} r^{(1)} = (Key_1 + G_1^{(Value)} G^{(Key)}) \bmod 1 \\ x_0^{(1)} = (Key_2 + G_2^{(Value)} G^{(Key)}) \bmod 0.5 \\ y_0^{(1)} = (Key_3 + G_3^{(Value)} G^{(Key)}) \bmod 0.5 \\ r^{(2)} = (Key_4 + G_4^{(Value)} G^{(Key)}) \bmod 1 \\ x_0^{(2)} = (Key_5 + G_5^{(Value)} G^{(Key)}) \bmod 0.5 \\ y_0^{(2)} = (Key_6 + G_6^{(Value)} G^{(Key)}) \bmod 0.5 \end{cases} \tag{12}$$

Step 5: In the first iteration, use $(r^{(1)}, x_0^{(1)}, y_0^{(1)})$, and in the second iteration, use $(r^{(2)}, x_0^{(2)}, y_0^{(2)})$ to generate chaotic pseudorandom sequences $S^{(1)}$, $S^{(2)}$, $S^{(3)}$.

$$\begin{cases} S^{(1)} = 2D - LSCM(r^{(1)}, x_0^{(1)}, y_0^{(1)}, H+16) \\ S^{(2)} = 2D - LSCM(r^{(2)}, x_0^{(2)}, y_0^{(2)}, W+16) \\ S^{(3)} = [S^{(1)}(H+1:H+16), S^{(2)}(W+1:W+16)] \end{cases} \tag{13}$$

In this case, H represents the height of the image, W represents the width of the image. The function $2D - LSCM$ $(r^{(0)}, x_0^{(0)}, y_0^{(0)}, L)$ is used to generate chaotic pseudorandom sequences, where $r^{(0)}$, $x_0^{(0)}$ and $y_0^{(0)}$ are the initial parameters of the chaotic system, and L is the length of the generated chaotic sequence.

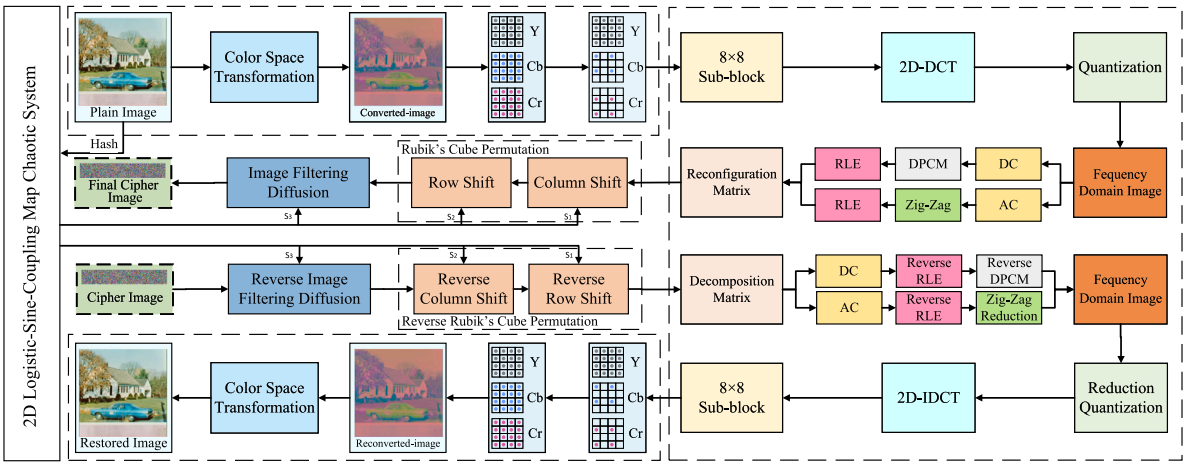


Fig. 2. The flowchart of the image compression encryption scheme.

2.2. Color gamut space transformation and sampling

The input color plaintext image is initially converted from the RGB color space to the YCbCr color space. Using 4:2:0 chroma subsampling, the luminance component (Y) is maintained at full resolution, with each pixel having a corresponding Y value. However, the chrominance components (Cb and Cr) are subsampled, meaning that within each 2×2 pixel block, there is only one Cb value and one Cr value, effectively reducing the resolution by half in both the horizontal and vertical directions. Consequently, each 2×2 block contains 4 Y values and only 1 value each for Cb and Cr. Below is an example using a 4×2 matrix to illustrate 4:2:0 chroma subsampling at this step. Suppose we have a 4×2 pixel block where each pixel has a luminance (Y) value and two chrominance (Cb and Cr) values, and the raw data appears as follows:

$$Y = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \\ Y_{31} & Y_{32} \\ Y_{41} & Y_{42} \end{bmatrix} \tag{14}$$

$$Cb = \begin{bmatrix} Cb_{11} & Cb_{12} \\ Cb_{21} & Cb_{22} \\ Cb_{31} & Cb_{32} \\ Cb_{41} & Cb_{42} \end{bmatrix} \tag{15}$$

$$Cr = \begin{bmatrix} Cr_{11} & Cr_{12} \\ Cr_{21} & Cr_{22} \\ Cr_{31} & Cr_{32} \\ Cr_{41} & Cr_{42} \end{bmatrix} \tag{16}$$

In 4:2:0 sampling, the luminance component (Y) retains full resolution, while the chrominance components (Cb and Cr) are downsampled. The downsampled chroma matrices are:

$$Cb_{4:2:0} = \begin{bmatrix} Cb_{11} \\ Cb_{31} \end{bmatrix} \tag{17}$$

$$Cr_{4:2:0} = \begin{bmatrix} Cr_{21} \\ Cr_{41} \end{bmatrix} \tag{18}$$

Specifically, for each 2×2 pixel block, we only retain one Cb and one Cr value. For example:

- For the $(1,1)$, $(1,2)$, $(2,1)$, $(2,2)$ 2×2 block, we retain Cb_{11} and Cr_{21}.
- For the $(3,1)$, $(3,2)$, $(4,1)$, $(4,2)$ 2×2 block, we retain Cb_{31} and Cr_{41}.

Finally, we can express 4:2:0 sampling with the following matrix formulas:

$$Y_{4:2:0} = Y \tag{19}$$

$$Cb_{4:2:0} = \begin{bmatrix} Cb_{11} \\ Cb_{31} \end{bmatrix} \tag{20}$$

$$Cr_{4:2:0} = \begin{bmatrix} Cr_{21} \\ Cr_{41} \end{bmatrix} \tag{21}$$

Therefore, in each 2×2 pixel block, the luminance component retains full resolution, while the chrominance components are downsampled by one both horizontally and vertically.

2.3. Temporal domain block shifting and DCT operation

To comply with the symmetric interval of DCT, the pixel values of the three sampled layers are shifted by subtracting 128 from each pixel value, transforming the value range of the shifted pixels to $[-128, 127]$. Subsequently, the image is divided into multiple 8×8 pixel blocks, arranged from left to right and top to bottom, within each of the three layers. These blocks do not overlap. If the number of rows or columns in the image is not a multiple of 8, zero-padding is required to fill the gaps. Then, each sub-block undergoes DCT transformation for frequency domain processing of the image.

2.4. Quantitative processing

Each sub-block of P' after DCT processing is quantized separately. The equation is as follows:

$$\begin{cases} I'_Y = round(\frac{I_Y}{Q_Y}) \\ I'_C = round(\frac{I_C}{Q_C}) \end{cases} \tag{22}$$

where $round(\cdot)$ represents the rounding function, I_Y and I_C represent the luminance and chrominance matrices in the DCT coefficient matrix, respectively. Q_Y refers to the quantization matrix of luminance, and Q_C corresponds to the quantization matrix of chrominance. I'_Y and I'_C represent the quantization matrices of luminance and chrominance after processing, respectively. The definitions of Q_Y and Q_C can be

found in formulas (23) and (24).

$$Q_Y = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{23}$$

$$Q_C = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix} \tag{24}$$

Suppose an 8×8 DCT coefficient matrix F is as follows:

$$F = \begin{bmatrix} 154 & 16 & 10 & 12 & 14 & 15 & 7 & 9 \\ 16 & 18 & 21 & 23 & 19 & 20 & 18 & 20 \\ 10 & 11 & 17 & 22 & 24 & 19 & 16 & 15 \\ 12 & 15 & 19 & 23 & 25 & 22 & 19 & 17 \\ 14 & 17 & 20 & 25 & 28 & 24 & 22 & 18 \\ 15 & 18 & 21 & 23 & 26 & 25 & 22 & 19 \\ 7 & 10 & 15 & 17 & 18 & 18 & 15 & 12 \\ 9 & 12 & 16 & 19 & 20 & 19 & 17 & 15 \end{bmatrix} \tag{25}$$

Here, the luminance quantization matrix Q_Y is used to quantize the DCT coefficient matrix F to obtain the quantized DCT coefficient matrix F_q. Let us take the first column of the quantized DCT coefficient matrix F as an example:

$$\begin{cases} F_q(0,0) = \text{round}\left(\frac{154}{16}\right) = 10 \\ F_q(0,1) = \text{round}\left(\frac{16}{11}\right) = 1 \\ F_q(0,2) = \text{round}\left(\frac{10}{10}\right) = 1 \\ F_q(0,3) = \text{round}\left(\frac{12}{16}\right) = 1 \\ F_q(0,4) = \text{round}\left(\frac{14}{24}\right) = 1 \\ F_q(0,5) = \text{round}\left(\frac{15}{40}\right) = 0 \\ F_q(0,6) = \text{round}\left(\frac{7}{51}\right) = 0 \\ F_q(0,7) = \text{round}\left(\frac{9}{61}\right) = 0 \end{cases} \tag{26}$$

Continue to perform the same calculation on all elements to obtain the quantized matrix F_q:

$$F_q = \begin{bmatrix} 10 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 2 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{27}$$

In order to maintain generality, the formula for the dequantization process is provided here:

$$\begin{cases} I_Y \approx I'_Y Q_Y \\ I_C \approx I'_C Q_C \end{cases} \tag{28}$$

Multiply the quantized DCT coefficient matrix F_q by the element of the quantization matrix at the corresponding position to restore the approximate DCT coefficient matrix F_c. Similarly, the first column of the DCT coefficient matrix F_q is taken as an example for the dequantization processing:

$$\begin{cases} F_c(0,0) \approx 10 \times 16 = 160 \\ F_c(0,1) \approx 1 \times 11 = 11 \\ F_c(0,2) \approx 1 \times 10 = 10 \\ F_c(0,3) \approx 1 \times 16 = 16 \\ F_c(0,4) \approx 1 \times 24 = 24 \\ F_c(0,5) \approx 0 \times 40 = 0 \\ F_c(0,6) \approx 0 \times 51 = 0 \\ F_c(0,7) \approx 0 \times 61 = 0 \end{cases} \tag{29}$$

Continue to perform the same calculation on all elements to obtain the dequantized matrix F_c:

$$F_c \approx \begin{bmatrix} 160 & 11 & 10 & 16 & 24 & 0 & 0 & 0 \\ 12 & 24 & 28 & 19 & 26 & 0 & 0 & 0 \\ 14 & 13 & 16 & 24 & 40 & 0 & 0 & 0 \\ 14 & 17 & 22 & 29 & 51 & 0 & 0 & 0 \\ 18 & 22 & 37 & 56 & 0 & 0 & 0 & 0 \\ 24 & 35 & 55 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{30}$$

2.5. Extract coefficients and compress into binary sequence

The DC coefficients of the sub-blocks of each layer are extracted in order to create a first-order row matrix. Subsequently, operations of Differential Pulse Code Modulation (DPCM) and Run-Length Encoding (RLE) are applied to this matrix to generate a binary bitstream based on the DC coefficients. Similarly, the AC coefficients of the sub-blocks of each layer are extracted and formed into a first-order row matrix. This matrix is then subjected to zigzag scanning and RLE processing to produce a binary bit stream based on the AC coefficients. The following example illustrates this process:

Given an image block I (e.g., an 8×8 pixel block), first perform Discrete Cosine Transform to obtain the DCT coefficient matrix C. For each 8×8 block, the DC coefficient is the top-left element of the DCT coefficient matrix C:

$$DC = \mathbf{C}_{0,0} \tag{31}$$

Then use DPCM to encode the DC coefficient. Assume we have multiple adjacent 8×8 blocks, with DC coefficients $DC_1, DC_2, \ldots, DC_n$. For the ith block, the difference ΔDC_i is defined as:

$$\Delta DC_i = \begin{cases} DC_i & \text{if } i = 1 \\ DC_i - DC_{i-1} & \text{if } i > 1 \end{cases} \tag{32}$$

Suppose we have 5 adjacent 8×8 blocks with DC coefficients:

$$DC = [1020, 980, 1000, 1010, 1030] \tag{33}$$

Calculate the DC coefficient difference using DPCM:

$$\begin{cases} \Delta DC_1 = 1020 \\ \Delta DC_2 = 980 - 1020 = -40 \\ \Delta DC_3 = 1000 - 980 = 20 \\ \Delta DC_4 = 1010 - 1000 = 10 \\ \Delta DC_5 = 1030 - 1010 = 20 \end{cases} \tag{34}$$

Therefore, the difference sequence is:

$$\Delta DC = [1020, -40, 20, 10, 20] \tag{35}$$

These differential values are further RLE encoded for the purpose of data compression. The basic idea of RLE encoding is to represent consecutive occurrences of the same character (or data) as a combination of a count value and that character. For example, for a sequence of

data containing consecutive characters, RLE encoding compresses these consecutive characters into the form of a symbol and the number of times it is repeated. The example here will not be repeated to illustrate the results because the amount of data is too small.

The purpose of using zigzag scanning for AC coefficients is to convert the data in the two-dimensional matrix to a one-dimensional array for more efficient entropy coding. Zigzag scanning traverses the matrix elements in diagonal order, starting from the top left corner and moving to the bottom right corner, first taking a diagonal down path, then a diagonal up path, and repeating this process until the entire matrix has been traversed. Assume there is a random 8 × 8 AC coefficient matrix as follows:

$$AC = \begin{bmatrix} 52 & 55 & 61 & 66 & 70 & 61 & 64 & 73 \\ 63 & 59 & 55 & 90 & 109 & 85 & 69 & 72 \\ 62 & 59 & 68 & 113 & 144 & 104 & 66 & 73 \\ 63 & 58 & 71 & 122 & 154 & 106 & 70 & 69 \\ 67 & 61 & 68 & 104 & 126 & 88 & 68 & 70 \\ 79 & 65 & 60 & 70 & 77 & 68 & 58 & 75 \\ 85 & 71 & 64 & 59 & 55 & 61 & 65 & 83 \\ 87 & 79 & 69 & 68 & 65 & 76 & 78 & 94 \end{bmatrix} \tag{36}$$

According to the order of zigzag scanning, we extract the elements from the matrix and convert them into a one-dimensional array. The index matrix I is used for scanning, and the scanning order is as follows:

$$I = \begin{bmatrix} 0 & 1 & 5 & 6 & 14 & 15 & 27 & 28 \\ 2 & 4 & 7 & 13 & 16 & 26 & 29 & 42 \\ 3 & 8 & 12 & 17 & 25 & 30 & 41 & 43 \\ 9 & 11 & 18 & 24 & 31 & 40 & 44 & 53 \\ 10 & 19 & 23 & 32 & 39 & 45 & 52 & 54 \\ 20 & 22 & 33 & 38 & 46 & 51 & 55 & 60 \\ 21 & 34 & 37 & 47 & 50 & 56 & 59 & 61 \\ 35 & 36 & 48 & 49 & 57 & 58 & 62 & 63 \end{bmatrix} \tag{37}$$

Use the index matrix I to scan and obtain the array:

$$\Delta AC = \begin{bmatrix} 52 & 55 & 63 & 62 & 59 & 61 & 66 & 55 \\ 59 & 63 & 67 & 58 & 68 & 90 & 70 & 61 \\ 109 & 113 & 71 & 61 & 79 & 85 & 65 & 68 \\ 122 & 144 & 85 & 64 & 73 & 69 & 104 & 154 \\ 104 & 60 & 71 & 87 & 79 & 64 & 70 & 126 \\ 106 & 66 & 72 & 73 & 70 & 88 & 77 & 59 \\ 69 & 68 & 55 & 68 & 68 & 69 & 70 & 58 \\ 61 & 65 & 76 & 65 & 75 & 83 & 78 & 94 \end{bmatrix} \tag{38}$$

Finally, the array ΔAC is converted to a one-dimensional array as the result.

2.6. Reconstruction of a binary sequence into a compressed matrix

The three color channels undergo compression and encoding, which produce six binary bitstreams. These bitstreams are concatenated to reconstruct a binary matrix with a column length eight times that of the plaintext. Subsequently, every eight columns of binary numbers are converted into a single column of decimal numbers, resulting in a matrix of the same length as the plaintext but with a different width. Random sequences consisting of the numbers 0 and 1 are added as necessary and recorded. This matrix is then evenly divided into three parts by rows, which are allocated to the R, G, and B channels, forming the compressed ciphertext C. The specific operations are shown in Algorithm 1.

Algorithm 1 Reconstruction of a binary sequence into a compressed matrix

Input: DCT coefficient matrix: DCY, $DCCb$, $DCCr$, ACY, $ACCb$, $ACCr$.

Output: Reconstruct the ciphertext matrixt: C

$P = [DCY, DCCb, DCCr, ACY, ACCb, ACCr]$;
$L = length(P)$;
$DP = ceil(L/row/8) * row * 8 - L$;
$R = rand(1, DP)$;
for $i \leftarrow 1$ to DP **do**
  if $R(1,i) <= 0.5$ **then**
    $R(1,i) = 0$;
  else
    $R(1,i) = 1$;
  end if
end for
$C1 = reshape([P, R], row * 8, [])$;
$C2 = blkproc(C1', [1, 8], ' two2ten')$;
$[h, w] = size(C3_a ll)$;
if $if ceil(h/3) * 3 - h \ = 0$ **then**
  $buh = ceil(rand(ceil(h/3) * 3 - h, row) * 256)$;
  for $i \leftarrow 1$ to h **do**
    $C3(i, :) = C2(i, :)$;
  end for
  for $j \leftarrow 1$ to $ceil(h/3) * 3 - h$ **do**
    $C3(h + j, :) = buh(j, :)$;
  end for
end if
$CR(1 : ceil(h/3), :) = C3(1 : ceil(h/3), :)$;
$CG(1 : ceil(h/3), :) = C3(ceil(h/3) + 1 : 2 * ceil(h/3), :)$;
$CB(1 : ceil(h/3), :) = C3(2 * ceil(h/3) + 1 : 3 * ceil(h/3), :)$;
$C(:, :, 1) = CR$;
$C(:, :, 2) = CG$;
$C(:, :, 3) = CB$;

2.7. Rubik's Cube permutation

First, the ciphertext image C obtained in Section 2.6 is subjected to RGB layering. Taking the encryption of one layer as an example, let R_o represent an α-bit grayscale image of size $H \times W$. The specific encryption algorithm is as follows.

Step 1. Take the H bits from the chaotic pseudo-random sequence $S^{(1)}$ generated in Section 2.1, and the W bits from the chaotic pseudo-random sequence $S^{(2)}$, and ensure that their values range from 0 to $2^{\alpha} - 1$;

Step 2. Determine the maximum number of iterations $Iteration_{max}$ and initialize the counter $Iteration$ to 0;

Step 3. Increment the counter by one;

$$Iteration = Iteration + 1 \tag{39}$$

Step 4. For each row m of the image R_o, perform the following operations:

(1) Compute the sum of all elements in the mth row and denote it as $\gamma(m)$

$$\gamma(m) = \sum_{j=1}^{W} R_o(m, n) \tag{40}$$

where $m = 1, 2, \ldots, H$.

(2) Compute the modulo of $\gamma(m)$ with 2 and denote it as $M\mu(m)$

$$M\mu(m) = \gamma(m) \mod 2 \tag{41}$$

(3) Shift the mth row cyclically to the left or right by $S^{(1)}(m)$ positions, i.e., the image pixels are shifted to the left or right by $S^{(1)}(m)$ positions, with the first pixel moving to the last pixel.

Step 5: For each column n of the image R_o, there are the following operations:

(1) Calculate the sum of all elements in the nth column and denote it as $\delta(n)$

$$\delta(n) = \sum_{m=1}^{H} R_o(m,n) \tag{42}$$

where $m = 1, 2, \ldots, W$.

(2) Calculate the modulus of $\delta(n)$ with 2 and denote it as $M\lambda(n)$

$$M\lambda(n) = \delta(n) \mod 2 \tag{43}$$

(3) Shift the nth column downwards or upwards by $S^{(2)}(m)$ positions, following the same rules as mentioned above. At this point, we will obtain the scrambled image I_{Per}.

Step 6: Use the chaotic pseudo-random sequence $S^{(2)}$, apply the bitwise $\oplus$ operator to each row of the scrambled image I_{Per} using the following expression:

$$\mathrm{I}_L(2m-1,n) = I_{Per}(2m-1,n) \oplus S^{(2)}(n) \tag{44}$$

$$\mathrm{I}_L(2\ m,n) = I_{Per}(2\ m,n) \oplus \mathrm{rot}180(S^{(2)}(n)) \tag{45}$$

where $\oplus$ and $rot180(K_H)$ represent the bitwise XOR operator and the left-to-right flip using the chaotic pseudo-random sequence $S^{(2)}$, respectively.

Step 7: Use the chaotic pseudo-random sequence $S^{(1)}$, apply the bitwise $\oplus$ operator to each column of the scrambled image I_{Per} using the following expression:

$$I_{End}(m,2n-1) = \mathrm{I}_L(m,2n-1) \oplus S^{(1)}(n) \tag{46}$$

$$I_{End}(m,2n) = \mathrm{I}_L(m,2n) \oplus \mathrm{rot}180(S^{(1)}(n)) \tag{47}$$

where $rot180(S^{(1)})$ represents the left-to-right flip using the chaotic pseudo-random sequence $S^{(1)}$.

Step 8: If $Iteration = Iteration_{max}$, the encryption of the image R_0 is complete, and the permuted cipher image I_{End} is generated, marking the end of the encryption process. Otherwise, repeat the operations from Step 3.

Step 9: Encrypt the remaining two layers of images. Once each layer of the image has been encrypted, synthesize the permuted cipher image P_{Per}.

2.8. Image filtering diffusion

In order to achieve better diffusion effects, we set the filter size to 2×2 for our operation, with the bottom-right weight corresponding to the current pixel, i.e., $K(2,2) = 1$. The other three weights are generated as subkeys using the following chaotic sequence $S^{(3)}$.

(1) Take the 96-bit $S^{(1)}$ and divide it into three 32-bit sequences, then convert them into integers $e = (e_1, e_2, e_3)$;

(2) Sort the elements of e and obtain the index vector $v = (v_1, v_2, v_3)$;

(3) Perturb the three integers with the vector v to obtain $e' = (e_1 + v_1, e_2 + v_2, e_3 + v_3)$. The resulting filter can be written as:

$$\mathbf{K} = \begin{pmatrix} e_1 + v_1 & e_2 + v_2 \\ e_3 + v_3 & 1 \end{pmatrix} \tag{48}$$

Next, the diffusion process is performed using the filter shown above. First, initialize the filtering diffusion result C using the scrambled result P_{Per}. Then, update the value of each pixel in C using the filtering operation, as follows:

$$\mathbf{C}'(x,y) = (y + \sum_{i,j\in\{1,2\}} \mathbf{K}(i,j)\mathbf{C}(x+i-2, y+j-2)) \bmod K \tag{49}$$

Since the weight $K(2,2) = 1$, the above equation can be rewritten as follows:

$$\begin{aligned}\mathbf{C}'(x,y) &= \left(\mathbf{y} + \mathbf{C}(x,y)\mathbf{K}(2,2) + \sum_{i,j\in\{1,2\}\cap(i,j)\neq(2,2)} \mathbf{K}(i,j)\mathbf{C}'(x+i-2, y+j-2)\right) \\ &\mod K \\ &= \left(y + \mathbf{C}(x,y) + \sum_{i,j\in\{1,2\}\cap(i,j)\neq(2,2)} \mathbf{K}(i,j)\mathbf{C}'(x+i-2, y+j-2)\right) \bmod K\end{aligned} \tag{50}$$

where $\mathbf{C}'(x,y)$ indicates the processed cipher image and K represents the grayscale levels of the cipher image. For the pixels on the top row and left column, their neighboring pixels are insufficient. To address this issue, we extend the image by using the boundary pixels in the opposite direction when processing these pixels. As the extended pixels do not need to be stored, the size of the encrypted image remains unchanged. The specific process of image filtering diffusion is described next with an example:

First construct an 8×8 image matrix P, and use the three chaotic initial parameters $r^{(0)} = 0.52$, $x_0^{(0)} = 1.52$, $y_0^{(0)} = 0.5111$ to generate a pseudo-random sequence for image filtering encryption. The following is the image matrix P:

$$P = \begin{bmatrix} 22 & 43 & 219 & 203 & 103 & 52 & 219 & 185 \\ 249 & 91 & 159 & 6 & 92 & 16 & 212 & 254 \\ 67 & 138 & 251 & 188 & 195 & 134 & 69 & 34 \\ 153 & 54 & 13 & 16 & 49 & 177 & 248 & 51 \\ 30 & 57 & 97 & 222 & 162 & 150 & 50 & 112 \\ 206 & 197 & 50 & 84 & 191 & 148 & 61 & 172 \\ 220 & 229 & 155 & 72 & 252 & 18 & 82 & 66 \\ 21 & 4 & 28 & 11 & 180 & 242 & 25 & 42 \end{bmatrix} \tag{51}$$

Then use image filtering diffusion to encrypt the image matrix P and obtain the encrypted image matrix C':

$$C' = \begin{bmatrix} 120 & 5 & 197 & 163 & 64 & 124 & 9 & 133 \\ 83 & 121 & 145 & 212 & 205 & 150 & 242 & 32 \\ 236 & 65 & 13 & 162 & 244 & 185 & 160 & 72 \\ 102 & 152 & 136 & 86 & 17 & 59 & 149 & 136 \\ 99 & 104 & 214 & 160 & 43 & 253 & 34 & 191 \\ 100 & 114 & 38 & 106 & 228 & 36 & 27 & 196 \\ 226 & 52 & 76 & 224 & 78 & 118 & 193 & 87 \\ 238 & 177 & 61 & 211 & 2 & 178 & 241 & 4 \end{bmatrix} \tag{52}$$

The same filter K can be used in the decryption process to recover the ciphertext image P. The inverse formula is as follows:

$$\mathbf{C}(x,y) = \left(\mathbf{C}'(x,y) - y - \sum_{i,j\in\{1,2\}\cap(i,j)\neq(2,2)} \mathbf{K}(i,j)\mathbf{C}'(x+i-2, y+j-2)\right) \bmod F \tag{53}$$

Therefore, image filtering decryption can be used to decrypt the ciphertext image C' and restore it to the original image C:

$$C = \begin{bmatrix} 22 & 43 & 219 & 203 & 103 & 52 & 219 & 185 \\ 249 & 91 & 159 & 6 & 92 & 16 & 212 & 254 \\ 67 & 138 & 251 & 188 & 195 & 134 & 69 & 34 \\ 153 & 54 & 13 & 16 & 49 & 177 & 248 & 51 \\ 30 & 57 & 97 & 222 & 162 & 150 & 50 & 112 \\ 206 & 197 & 50 & 84 & 191 & 148 & 61 & 172 \\ 220 & 229 & 155 & 72 & 252 & 18 & 82 & 66 \\ 21 & 4 & 28 & 11 & 180 & 242 & 25 & 42 \end{bmatrix} \tag{54}$$

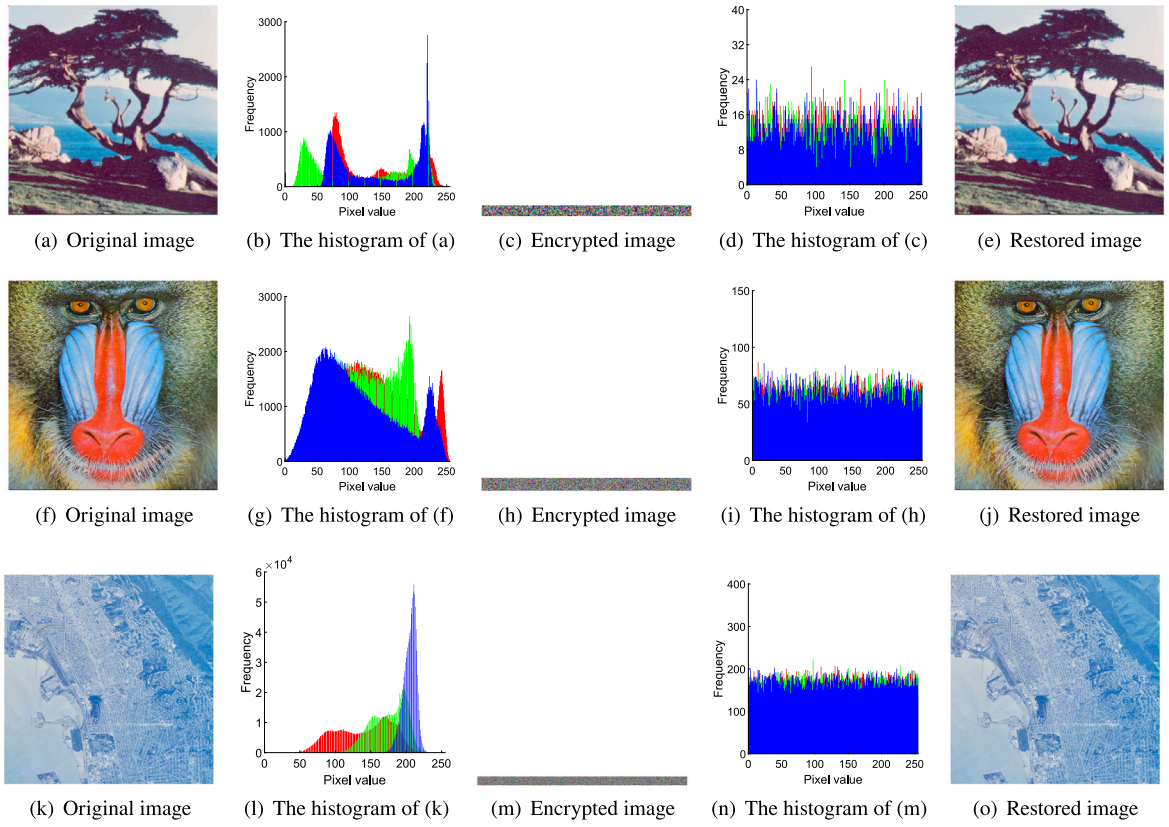


Fig. 3. Compressed and encrypted plaintext and ciphertext images and corresponding histograms.

3. Analysis of results

3.1. Experimental environment and platform

In this experiment, we used a computer running MATLAB R2023b experimental software. The host is equipped with an 12th Gen Intel Core i5-12490F @ 3.00 GHz 6-core processor and 64 GB of memory. All images used in this article are from The USC-SIPI Image Database (1977).

3.2. Statistical analysis

3.2.1. Histogram analysis

The image encryption algorithms can cleverly hide the main information by simulating the noise through processing the image distribution. The histogram of the processed image is similar to the noise distribution after encryption, which effectively prevents attackers from obtaining valid information. Fig. 3 displays the compressed and encrypted image along with its corresponding histogram. Fig. 4 displays the directly encrypted image and its corresponding histogram. Finally, Fig. 5 shows the 3D histogram of the compressed and encrypted image.

3.2.2. Information entropy analysis

Information entropy serves as a crucial indicator that reflects the level of uncertainty in image information. As the uncertainty in an image increases, the information entropy also increases, thereby enhancing confidentiality. Conversely, a decrease in information entropy indicates a reduction in the uncertainty of the image, resulting in reduced confidentiality. Eq. (55) demonstrates the specific calculation equation used for information entropy.

$$H(x) = -\sum_{i=1}^{L} P(x_i) log_2 P(x_i) \tag{55}$$

Table 1
Information entropy of different images.

Filename	Description	Size	HI	HC
2.1.01	San Diego (Miramar NAS)	512 × 512	7.9604	7.9954
2.1.02	San Diego	512 × 512	7.9677	7.9961
2.1.04	Oakland	512 × 512	7.9418	7.9953
2.1.06	Woodland Hills, Ca.	512 × 512	7.9521	7.9959
2.1.07	Foster City, Ca.	512 × 512	7.8587	7.9926
2.1.10	San Diego (Shelter Island)	512 × 512	7.9309	7.9947
2.1.12	San Diego (Downtown)	512 × 512	7.9233	7.9939
2.2.01	San Diego	1024 × 1024	7.9383	7.9991
2.2.04	Richmond, Ca.	1024 × 1024	7.9290	7.9988
2.2.05	San Diego (Miramar NAS)	1024 × 1024	7.9544	7.9987
2.2.08	San Diego	1024 × 1024	7.9433	7.9988
4.1.06	Tree	256 × 256	7.9420	7.9809
4.2.03	Mandrill (a.k.a. Baboon)	512 × 512	7.9669	7.9961
4.2.06	Sailboat on lake	512 × 512	7.9467	7.9951
4.2.07	Peppers	512 × 512	7.9286	7.9934

where L is the total number of symbols $x_i \in x$ and $P(x_i)$ denotes the probability of the symbols.

The information entropy results obtained in this paper are presented in Table 1. Table 2 shows the comparison of encrypting only the original image using the algorithm proposed in this paper with the data from other algorithms. Following encryption, the information entropy of the ciphertext approaches the ideal value of 8, signifying the strong performance of our scheme in terms of confidentiality.

3.2.3. Differential statistical analysis

The evaluation of permutation and diffusion properties of image encryption algorithms commonly employs the Number of Pixel Changes Rate (NPCR) and the Unified Average Changing Intensity (UACI) values. These values are widely used to assess the effectiveness of image encryption algorithms against differential attacks. Mathematically,

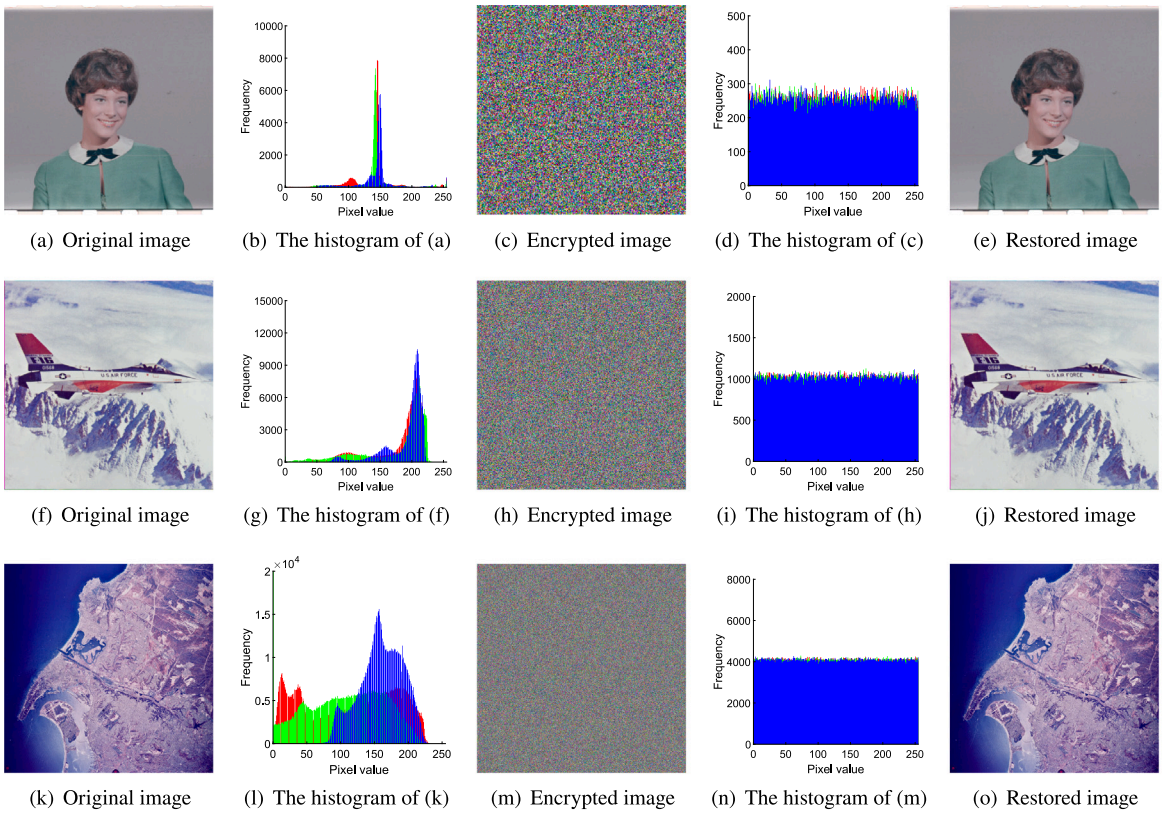


Fig. 4. Directly encrypted plaintext and ciphertext images and corresponding histograms.

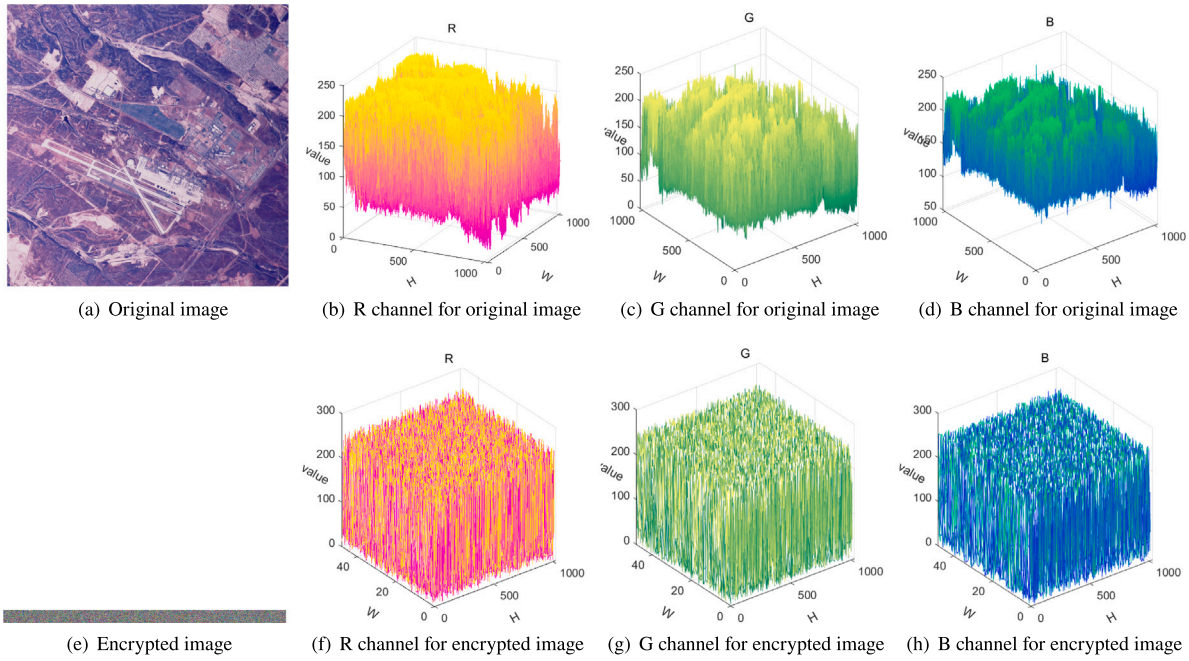


Fig. 5. Plaintext and ciphertext images and corresponding 3D histograms.

NPCR and UACI between two images can be defined as:

$$\begin{cases} NPCR = \frac{1}{MN}\sum_{i=1}^{M}\sum_{i=1}^{N} D(i,j) \times 100\% \\ UACI = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} \frac{|v_1(i,j)-v_2(i,j)|}{255} \times 100\% \end{cases} \tag{56}$$

where $M \times N$ is the size of the image, v_1 and v_2 are the ciphertext image before and after the plaintext image is changed by one pixel respectively. D can be defined by the following equation:

$$D(i,j) = \begin{cases} 0, v_1(i,j) = v_2(i,j) \\ 1, v_0(i,j) \neq v_2(i,j) \end{cases} \tag{57}$$

This study selected some open source images for testing and collected the data of RGB channels, as shown in Table 3. Table 4 shows

Table 2
Comparison of information entropy between different algorithms.

Filename	Proposed	Zhang, Hu et al. (2024)	Peng et al. (2023)	Zhang and Hu (2023)	Song et al. (2024)
Airplane	7.9998	7.9983	7.9994	7.9992	/
Couple	7.9990	7.9987	/	/	7.9973
House	7.9992	7.9988	7.9978	7.9994	7.9968
Mandrill	7.9997	7.9986	/	7.9992	7.9992
Peppers	7.9998	7.9992	7.9994	7.9989	7.9971
San Diego	7.9998	7.9995	7.9998	/	/
Tree	7.9991	7.9994	/	/	/
Female	7.9990	/	7.9974	/	7.9971
Oakland	7.9999	/	7.9998	/	/
Stockton	7.9999	/	7.9998	/	/

Table 3
NPCR and UACI values of different images.

Filename	Description	Size	Channel	NPCR	UACI
2.1.01	San Diego (Miramar NAS)	512 × 512	Red	99.5824	33.4702
2.1.01	San Diego (Miramar NAS)	512 × 512	Green	99.6835	33.8241
2.1.01	San Diego (Miramar NAS)	512 × 512	Blue	99.6835	33.5667
2.1.02	San Diego	512 × 512	Red	99.5634	33.0952
2.1.02	San Diego	512 × 512	Green	99.5519	33.5654
2.1.02	San Diego	512 × 512	Blue	99.6553	33.1521
2.1.04	Oakland	512 × 512	Red	99.6931	33.3724
2.1.04	Oakland	512 × 512	Green	99.4489	33.6115
2.1.04	Oakland	512 × 512	Blue	99.6164	33.5182
2.1.06	Woodland Hills, Ca.	512 × 512	Red	99.6094	32.9102
2.1.06	Woodland Hills, Ca.	512 × 512	Green	99.5326	33.1490
2.1.06	Woodland Hills, Ca.	512 × 512	Blue	99.6582	33.2885
2.1.07	Foster City, Ca.	512 × 512	Red	99.6826	33.1635
2.1.07	Foster City, Ca.	512 × 512	Green	99.6704	33.4156
2.1.07	Foster City, Ca.	512 × 512	Blue	99.4995	33.5196
2.1.10	San Diego (Shelter Island)	512 × 512	Red	99.6183	33.2134
2.1.10	San Diego (Shelter Island)	512 × 512	Green	99.6094	33.2628
2.1.10	San Diego (Shelter Island)	512 × 512	Blue	99.5117	32.8078
2.1.12	San Diego (Downtown)	512 × 512	Red	99.7248	33.0737
2.1.12	San Diego (Downtown)	512 × 512	Green	99.5916	33.2319
2.1.12	San Diego (Downtown)	512 × 512	Blue	99.6183	33.2165
2.2.01	San Diego	1024 × 1024	Red	99.6419	33.2667
2.2.01	San Diego	1024 × 1024	Green	99.6247	33.6924
2.2.01	San Diego	1024 × 1024	Blue	99.6304	33.6272
2.2.04	Richmond, Ca.	1024 × 1024	Red	99.6457	33.1074
2.2.04	Richmond, Ca.	1024 × 1024	Green	99.6412	33.3919
2.2.04	Richmond, Ca.	1024 × 1024	Blue	99.5753	32.8114
2.2.05	San Diego (Miramar NAS)	1024 × 1024	Red	99.5605	33.1410
2.2.05	San Diego (Miramar NAS)	1024 × 1024	Green	99.6175	33.2751
2.2.05	San Diego (Miramar NAS)	1024 × 1024	Blue	99.6155	33.1647
2.2.08	San Diego	1024 × 1024	Red	99.6496	33.5996
2.2.08	San Diego	1024 × 1024	Green	99.5653	33.5839
2.2.08	San Diego	1024 × 1024	Blue	99.5960	33.1005
4.1.06	Tree	256 × 256	Red	99.6394	33.4950
4.1.06	Tree	256 × 256	Green	99.5793	33.4422
4.1.06	Tree	256 × 256	Blue	99.5192	33.8414
4.2.03	Mandrill (a.k.a. Baboon)	512 × 512	Red	99.6157	33.2724
4.2.03	Mandrill (a.k.a. Baboon)	512 × 512	Green	99.5842	33.4596
4.2.03	Mandrill (a.k.a. Baboon)	512 × 512	Blue	99.6157	33.4684
4.2.06	Sailboat on lake	512 × 512	Red	99.6373	32.7649
4.2.06	Sailboat on lake	512 × 512	Green	99.7117	33.5429
4.2.06	Sailboat on lake	512 × 512	Blue	99.5908	32.9162
4.2.07	Peppers	512 × 512	Red	99.6582	32.6309
4.2.07	Peppers	512 × 512	Green	99.6338	32.8462
4.2.07	Peppers	512 × 512	Blue	99.5972	33.2896

the comparison of encrypting only the original image using the algorithm proposed in this paper with the data from other algorithms. The experimental data indicate that the encryption results of the algorithm are very close to the theoretical values.

3.2.4. Correlation analysis

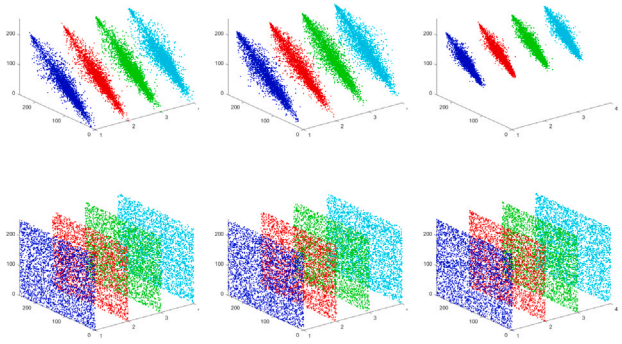
One of the key tasks of image encryption algorithms is to disturb the correlation between pixels to effectively resist attacks based on pixel correlation. Taking the "Pepper" image as an example, this paper analyzes the images before and after encryption by randomly selecting 3000 pairs of adjacent pixels in horizontal, vertical, diagonal and anti-diagonal directions, and compares their pixel correlation. The results of the correlation analysis are shown in Fig. 6 and the experimental data are shown in Table 5. The experiment shows that our encryption algorithm successfully weakens the correlation between pixels, and there is almost no recognizable correlation in the ciphertext. This strongly indicates that the encryption algorithm we propose has excellent security.

Table 4
Comparison of NPCR values between different algorithms.

Filename	Proposed	Zhang, Hu et al. (2024)	Peng et al. (2023)	Zhang and Hu (2023)	Song et al. (2024)
Airplane	99.6105	99.6283	99.6330	99.6092	/
Couple	99.6094	99.5845	/	/	99.6130
House	99.6078	99.6296	99.6399	99.6128	99.6110
Mandrill	99.6113	99.6296	/	99.6131	99.6110
Peppers	99.6075	99.6236	99.6174	99.6071	/
San Diego	99.6052	99.6291	99.6172	/	/
Tree	99.6033	99.6074	/	/	/
Female	99.6002	/	99.5880	/	/
Oakland	99.6106	/	99.6147	/	/
Stockton	99.6042	/	99.6066	/	/

Table 5
Neighboring pixel correlation values for plaintext and ciphertext image in different directions.

Component	Direction	Original image	The proposed
R channel	Horizontal	0.8626	0.0284
	Vertical	0.8770	−0.0066
	Diagonal	0.8478	0.0049
	Anti-diagonal	0.8041	−0.0031
G channel	Horizontal	0.8659	0.0183
	Vertical	0.8911	−0.0787
	Diagonal	0.8586	−0.0184
	Anti-diagonal	0.8209	−0.0359
B channel	Horizontal	0.8717	−0.0114
	Vertical	0.8793	0.0363
	Diagonal	0.8528	−0.0168
	Anti-diagonal	0.8117	−0.0161



(a) R channel (b) G channel (c) B channel

Fig. 6. Correlation of neighboring pixels in different directions between plaintext and ciphertext image.

3.2.5. Analysis of image quality restoration

Peak Signal to Noise Ratio (PSNR) and Structural Similarity (SSIM) are commonly used as a tool to weigh the quality of encryption in the image processing field. Mean Square Error (MSE) is a part of PSNR which is defined as:

$$\begin{cases} \mathrm{MSE} = \frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}(X(i,j)-Y(i,j))^2 \\ \mathrm{PSNR} = 10\log_{10}\left(\frac{Q^2}{MSE}\right) \end{cases} \tag{58}$$

where MSE represents the mean square error between the plaintext image X and the ciphertext image Y. The height and width of the image are denoted by H and W respectively. Q signifies the pixel level of the image. SSIM is a metric that quantifies the similarity between two images, defined as:

$$\mathrm{SSIM}(X,Y) = \frac{(2\mu_X\mu_Y + (0.01L)^2)(2\mu_{XY} + (0.03L)^2)}{(\mu_X^2 + \mu_Y^2 + (0.01L)^2)(\mu_X^2 + \mu_Y^2 + (0.03L)^2)} \tag{59}$$

where the mean values of the images X and Y are denoted by μ_X and μ_Y, respectively, while their respective standard deviations are represented as well. The dynamic range of pixel values is denoted as L.

In the case of digital images, PSNR value higher than 40 dB generally indicates excellent image restoration quality, while a range between 30 and 40 dB typically signifies good image restoration quality. Experimental results show that our image reconstruction quality performs excellently in terms of PSNR and SSIM, indicating that our scheme successfully reduces information loss and distortion while compressing and encrypting images. This provides strong support for the practical application of image restoration, especially in cases where high-quality restoration is required.

3.2.6. The influence of quantization matrix on compression ratio and recovery quality

In proposed encryption algorithm, the compression rate can be customized by adjusting the quantization matrix. The specific relationship between the customized quantization matrix and the standard quantization matrix is defined by the following equation:

$$\begin{cases} Q_1 = \alpha Q_Y \\ Q_2 = \alpha Q_C \end{cases} \tag{60}$$

where Q_1 represents the customized luminance quantization matrix, Q_2 represents the customized chrominance quantization matrix, Q_Y and Q_C refer to the standard quantization matrices, respectively. α denotes the quantization scaling factor, with α taking values from the set $\alpha = 1, 2^{-1}, 2^{-2}, 2^{-3}$ and 2^{-4}.

Generally, as the value of parameter α increases, the compression rate of the image also increases. However, this increase comes at the expense of sacrificing the quality of image recovery. By adjusting the value of parameter α, it can be observed that as α increases, the image compression rate decreases. Figs. 7–9 represent the compression effect of different sized images at different α values, respectively. This trend indicates the significant influence of parameter α on the image compression rate, while also demonstrating the favorable recovery performance of the proposed encryption algorithm under high compression ratios.

In order to better support the scheme proposed in this paper, we have taken the same image compressed using different quantization matrices in open source libraries. Figs. 10–14 show the quality of image compression under different quantization matrices respectively. Tables 6–10 represent the MSE, PSNR and SSIM values of image recovery quality under different quantization matrices; Tables 11–15 represent the size of the image after compression and the compression ratio under different quantization matrices.

3.3. Analysis of sensitivity to the key

Key sensitivity refers to the significant difference between the ciphertext obtained when the same image is encrypted with slightly different keys, and is an important aspect of encryption technology. In this section, we use the original keys ($r^{(1)}$, $x_0^{(1)}$, $y_0^{(1)}$, $r^{(2)}$, $x_0^{(2)}$, $y_0^{(2)}$) and

(a) $2^{-4}Q$ (b) $2^{-3}Q$ (c) $2^{-2}Q$ (d) $2^{-1}Q$ (e) $2^{0}Q$

Fig. 7. Compressed and recovered images of 256 × 256 size image under different quantization matrix.

(a) $2^{-4}Q$ (b) $2^{-3}Q$ (c) $2^{-2}Q$ (d) $2^{-1}Q$ (e) $2^{0}Q$

Fig. 8. Compressed and recovered images of 512 × 512 size image under different quantization matrix.

(a) $2^{-4}Q$ (b) $2^{-3}Q$ (c) $2^{-2}Q$ (d) $2^{-1}Q$ (e) $2^{0}Q$

Fig. 9. Compressed and recovered images of 1024 × 1024 size image under different quantization matrix.

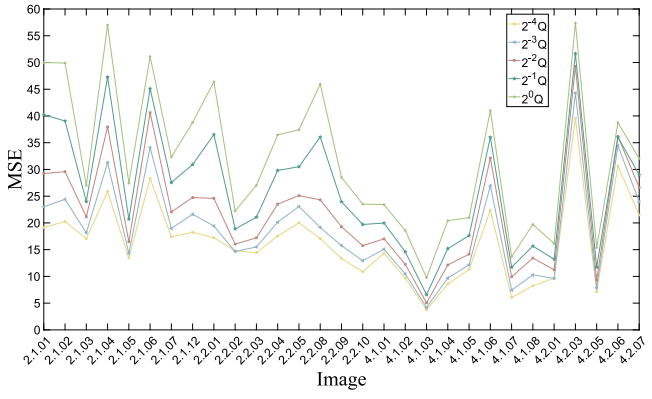


Fig. 10. Comparative analysis of MSE values after image restoration using different quantization matrix compression.

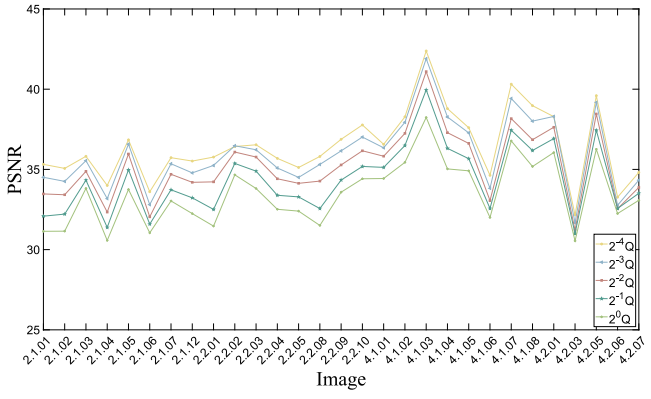


Fig. 11. Comparative analysis of PSNR values after image restoration using different quantization matrix compression.

the key with small perturbation ($r^{(1)} + 10^{-14}$, $x_0^{(1)} + 10^{-14}$, $y_0^{(1)} + 10^{-14}$, $r^{(2)} + 10^{-14}$, $x_0^{(2)} + 10^{-14}$, $y_0^{(2)} + 10^{-14}$) encrypts the same plaintext. The experimental results are shown in Table 16. The results demonstrate a notable disparity between the two ciphertext images obtained following key perturbation, as evidenced by the NPCR and UACI values approaching the ideal values of 99.6094 and 33.4635, respectively. This indicates that the encryption scheme exhibits a high degree of sensitivity to the key.

3.4. Key space

The key space represents the range of encryption keys. Having a large enough key space can effectively resist exhaustive attacks. The scheme selects six key parameters with an accuracy up to 10^{14} and perturbs the key by introducing hash values of plaintext and ciphertext correlation. Therefore, the key space size of this scheme can be

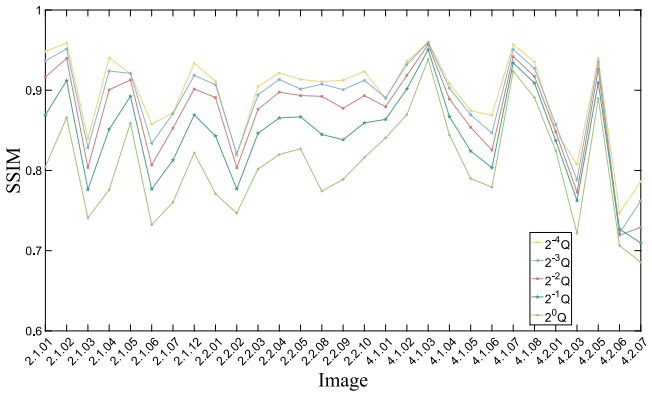


Fig. 12. Comparative analysis of SSIM values after image restoration using different quantization matrix compression.

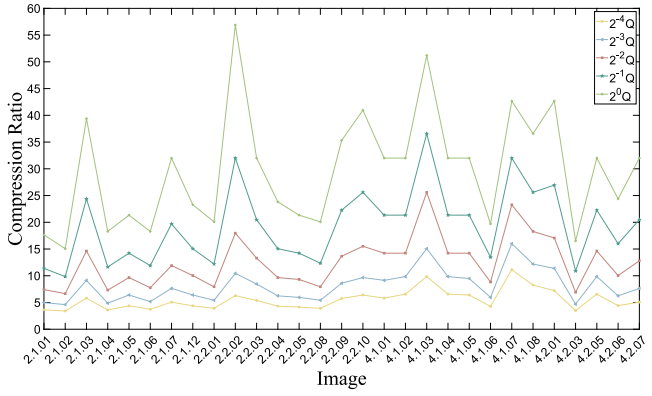


Fig. 13. Comparative analysis of image compression ratios using different quantization matrices.

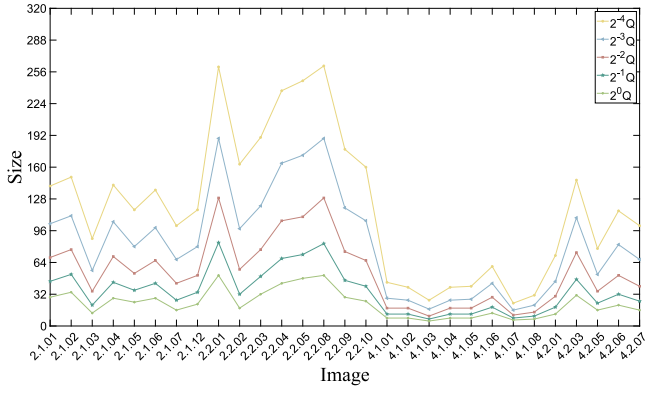


Fig. 14. Comparative analysis of image compression size using different quantization matrices.

estimated to be approximately $10^{14\times6} \approx 2^{278}$, which far exceeds the theoretical requirement of 2^{100}.

3.5. Running efficiency

Runtime is an important indicator of whether a compression encryption algorithm is feasible and practical. Obviously, an algorithm with a short running time is more acceptable. In the experimental environment shown in Section 3.1, the "Lena" images with sizes of 512 × 512 and 256 × 256 are processed respectively, and the time consumption of the four main steps of hyperchaotic system generation, frequency domain compression, block permutation and row-column diffusion is tested. The results are shown in Table 17.

In order to facilitate comparison, we refer to the running efficiency in the literature, as shown in Table 18. In comparison with other algorithms, we can observe that our permutation and diffusion speed

Table 6
Recovery quality analysis for image compression using the $2^{-4}Q$ quantization matrix.

Filename	Size	MSE	PSNR	SSIM
2.1.01	512 × 512	19.1281	35.3141	0.9483
2.1.02	512 × 512	20.2676	35.0628	0.9590
2.1.03	512 × 512	17.0352	35.8173	0.8394
2.1.04	512 × 512	25.8913	33.9993	0.9403
2.1.05	512 × 512	13.4551	36.8419	0.9206
2.1.06	512 × 512	28.3481	33.6056	0.8577
2.1.07	512 × 512	17.3805	35.7302	0.8719
2.1.10	512 × 512	20.1772	35.0822	0.9008
2.1.12	512 × 512	18.2421	35.5200	0.9337
2.2.01	1024 × 1024	17.2279	35.7685	0.9110
2.2.02	1024 × 1024	14.8091	36.4255	0.8187
2.2.03	1024 × 1024	14.4414	36.5347	0.9047
2.2.04	1024 × 1024	17.5483	35.6885	0.9215
2.2.05	1024 × 1024	20.0032	35.1198	0.9136
2.2.08	1024 × 1024	17.0727	35.8078	0.9106
2.2.09	1024 × 1024	13.3447	36.8777	0.9125
2.2.10	1024 × 1024	10.8650	37.7705	0.9236
4.1.01	256 × 256	14.3419	36.5647	0.8893
4.1.02	256 × 256	9.6680	38.2774	0.9358
4.1.03	256 × 256	3.7576	42.3817	0.9602
4.1.04	256 × 256	8.5932	38.7892	0.9087
4.1.05	256 × 256	11.3058	37.5978	0.8745
4.1.06	256 × 256	22.3813	34.6319	0.8687
4.1.07	256 × 256	6.0509	40.3126	0.9569
4.1.08	256 × 256	8.2388	38.9722	0.9353
4.2.01	512 × 512	9.6437	38.2884	0.8489
4.2.03	512 × 512	39.5645	32.1577	0.8076
4.2.05	512 × 512	7.1382	39.5949	0.9397
4.2.06	512 × 512	30.6385	33.2681	0.7470
4.2.07	512 × 512	21.4709	34.8123	0.7863

Table 7
Recovery quality analysis for image compression using the $2^{-3}Q$ quantization matrix.

Filename	Size	MSE	PSNR	SSIM
2.1.01	512 × 512	23.0236	34.5091	0.9367
2.1.02	512 × 512	24.4144	34.2543	0.9515
2.1.03	512 × 512	18.1434	35.5436	0.8284
2.1.04	512 × 512	31.2757	33.1787	0.9240
2.1.05	512 × 512	14.3085	36.5749	0.9211
2.1.06	512 × 512	34.0942	32.8040	0.8338
2.1.07	512 × 512	18.9629	35.3517	0.8708
2.1.10	512 × 512	23.3175	34.4540	0.8891
2.1.12	512 × 512	21.6179	34.7827	0.9187
2.2.01	1024 × 1024	19.4235	35.2475	0.9068
2.2.02	1024 × 1024	14.6439	36.4742	0.8207
2.2.03	1024 × 1024	15.5105	36.2245	0.8942
2.2.04	1024 × 1024	20.1439	35.0894	0.9136
2.2.05	1024 × 1024	23.0625	34.5017	0.9014
2.2.08	1024 × 1024	19.1463	35.3099	0.9077
2.2.09	1024 × 1024	15.7603	36.1552	0.9007
2.2.10	1024 × 1024	12.9256	37.0163	0.9123
4.1.01	256 × 256	15.0821	36.3462	0.8906
4.1.02	256 × 256	10.4539	37.9380	0.9322
4.1.03	256 × 256	4.1989	41.8995	0.9594
4.1.04	256 × 256	9.6902	38.2675	0.9027
4.1.05	256 × 256	12.1701	37.2779	0.8693
4.1.06	256 × 256	26.9583	33.8239	0.8467
4.1.07	256 × 256	7.4315	39.4200	0.9509
4.1.08	256 × 256	10.2820	38.0100	0.9272
4.2.01	512 × 512	9.6157	38.3010	0.8571
4.2.03	512 × 512	44.2905	31.6677	0.7881
4.2.05	512 × 512	7.8561	39.1787	0.9356
4.2.06	512 × 512	34.4108	32.7639	0.7218
4.2.07	512 × 512	23.9849	34.3314	0.7623

Table 8
Recovery quality analysis for image compression using the $2^{-2}Q$ quantization matrix.

Filename	Size	MSE	PSNR	SSIM
2.1.01	512 × 512	29.2359	33.4716	0.9166
2.1.02	512 × 512	29.5865	33.4199	0.9397
2.1.03	512 × 512	21.1372	34.8803	0.8037
2.1.04	512 × 512	37.9453	32.3392	0.9005
2.1.05	512 × 512	16.4769	35.9621	0.9129
2.1.06	512 × 512	40.6145	32.0440	0.8068
2.1.07	512 × 512	22.0530	34.6961	0.8527
2.1.09	512 × 512	16.7517	35.8902	0.8058
2.1.12	512 × 512	24.7531	34.1945	0.9015
2.2.01	1024 × 1024	24.5837	34.2243	0.8909
2.2.02	1024 × 1024	16.0290	36.0817	0.8033
2.2.03	1024 × 1024	17.2154	35.7716	0.8762
2.2.04	1024 × 1024	23.5208	34.4163	0.8976
2.2.05	1024 × 1024	25.1089	34.1325	0.8934
2.2.08	1024 × 1024	24.3194	34.2713	0.8924
2.2.09	1024 × 1024	19.2891	35.2777	0.8774
2.2.10	1024 × 1024	15.7418	36.1603	0.8933
4.1.01	256 × 256	17.0146	35.8226	0.8795
4.1.02	256 × 256	12.2579	37.2466	0.9187
4.1.03	256 × 256	5.0532	41.0951	0.9568
4.1.04	256 × 256	12.1363	37.2899	0.8891
4.1.05	256 × 256	14.1599	36.6202	0.8538
4.1.06	256 × 256	32.1375	33.0607	0.8254
4.1.07	256 × 256	9.9221	38.1648	0.9419
4.1.08	256 × 256	13.4298	36.8501	0.9172
4.2.01	512 × 512	11.2281	37.6277	0.8481
4.2.03	512 × 512	49.2820	31.2039	0.7729
4.2.05	512 × 512	9.2769	38.4568	0.9260
4.2.06	512 × 512	36.1700	32.5473	0.7194
4.2.07	512 × 512	26.6273	33.8775	0.7291

Table 9
Recovery quality analysis for image compression using the $2^{-1}Q$ quantization matrix.

Filename	Size	MSE	PSNR	SSIM
2.1.01	512 × 512	40.2260	32.0857	0.8691
2.1.02	512 × 512	39.0622	32.2132	0.9120
2.1.03	512 × 512	24.0082	34.3272	0.7763
2.1.04	512 × 512	47.2766	31.3843	0.8513
2.1.05	512 × 512	20.7015	34.9708	0.8924
2.1.06	512 × 512	45.1061	31.5884	0.7768
2.1.07	512 × 512	27.5560	33.7286	0.8128
2.1.10	512 × 512	32.1108	33.0643	0.8447
2.1.12	512 × 512	30.9206	33.2283	0.8692
2.2.01	1024 × 1024	36.5108	32.5066	0.8430
2.2.02	1024 × 1024	18.8746	35.3720	0.7769
2.2.03	1024 × 1024	21.0792	34.8923	0.8463
2.2.04	1024 × 1024	29.8111	33.3870	0.8654
2.2.05	1024 × 1024	30.5100	33.2864	0.8669
2.2.08	1024 × 1024	36.0669	32.5597	0.8449
2.2.09	1024 × 1024	23.9640	34.3352	0.8383
2.2.10	1024 × 1024	19.7055	35.1849	0.8593
4.1.01	256 × 256	19.9742	35.1261	0.8637
4.1.02	256 × 256	14.5935	36.4892	0.9017
4.1.03	256 × 256	6.5684	39.9562	0.9506
4.1.04	256 × 256	15.1978	36.3130	0.8672
4.1.05	256 × 256	17.6314	35.6679	0.8242
4.1.06	256 × 256	36.0205	32.5653	0.8034
4.1.07	256 × 256	11.7173	37.4425	0.9341
4.1.08	256 × 256	15.6671	36.1809	0.9090
4.2.01	512 × 512	13.2128	36.9208	0.8372
4.2.03	512 × 512	51.6547	30.9997	0.7625
4.2.05	512 × 512	11.7253	37.4395	0.9094
4.2.06	512 × 512	36.0336	32.5637	0.7268
4.2.07	512 × 512	28.9707	33.5112	0.7097

are at the leading, which indicates that the algorithm is satisfactory and effective, and has higher advantages in some existing algorithms.

Table 10
Recovery quality analysis for image compression using the $2^{0}Q$ quantization matrix.

Filename	Size	MSE	PSNR	SSIM
2.1.01	512 × 512	49.9985	31.1412	0.8046
2.1.02	512 × 512	49.8984	31.1499	0.8659
2.1.03	512 × 512	27.0022	33.8168	0.7409
2.1.04	512 × 512	57.0055	30.5716	0.7760
2.1.05	512 × 512	27.4270	33.7490	0.8590
2.1.06	512 × 512	51.1006	31.0465	0.7323
2.1.07	512 × 512	32.3383	33.0336	0.7604
2.1.10	512 × 512	39.0214	32.2178	0.8011
2.1.12	512 × 512	38.7743	32.2454	0.8220
2.2.01	1024 × 1024	46.3956	31.4660	0.7710
2.2.02	1024 × 1024	22.2308	34.6613	0.7468
2.2.03	1024 × 1024	27.0455	33.8099	0.8017
2.2.04	1024 × 1024	36.4495	32.5139	0.8199
2.2.05	1024 × 1024	37.4166	32.4002	0.8270
2.2.08	1024 × 1024	45.9612	31.5069	0.7742
2.2.09	1024 × 1024	28.5121	33.5805	0.7888
2.2.10	1024 × 1024	23.5235	34.4158	0.8162
4.1.01	256 × 256	23.4316	34.4328	0.8406
4.1.02	256 × 256	18.5804	35.4402	0.8696
4.1.03	256 × 256	9.7532	38.2393	0.9388
4.1.04	256 × 256	20.4176	35.0308	0.8442
4.1.05	256 × 256	20.9854	34.9116	0.7900
4.1.06	256 × 256	41.0101	32.0019	0.7791
4.1.07	256 × 256	13.6605	36.7761	0.9238
4.1.08	256 × 256	19.7240	35.1809	0.8907
4.2.01	512 × 512	16.1419	36.0513	0.8248
4.2.03	512 × 512	57.3895	30.5425	0.7216
4.2.05	512 × 512	15.3885	36.2588	0.8899
4.2.06	512 × 512	38.7486	32.2482	0.7064
4.2.07	512 × 512	32.0341	33.0747	0.6855

Table 11
Comparison of image compression rate and compressed size using $2^{-4}Q$ quantization matrix for compression.

Filename	Original image size	Compressed image size	Compression ratio
2.1.01	512 × 512	141 × 512	3.6312
2.1.02	512 × 512	150 × 512	3.4133
2.1.03	512 × 512	88 × 512	5.8182
2.1.04	512 × 512	142 × 512	3.6056
2.1.05	512 × 512	117 × 512	4.3761
2.1.06	512 × 512	137 × 512	3.7372
2.1.07	512 × 512	101 × 512	5.0693
2.1.10	512 × 512	121 × 512	4.2314
2.1.12	512 × 512	117 × 512	4.3761
2.2.01	1024 × 1024	261 × 1024	3.9234
2.2.02	1024 × 1024	163 × 1024	6.2822
2.2.03	1024 × 1024	190 × 1024	5.3895
2.2.04	1024 × 1024	237 × 1024	4.3207
2.2.05	1024 × 1024	247 × 1024	4.1457
2.2.08	1024 × 1024	262 × 1024	3.9084
2.2.09	1024 × 1024	178 × 1024	5.7528
2.2.10	1024 × 1024	160 × 1024	6.4000
4.1.01	256 × 256	44 × 256	5.8182
4.1.02	256 × 256	39 × 256	6.5641
4.1.03	256 × 256	26 × 256	9.8462
4.1.04	256 × 256	39 × 256	6.5641
4.1.05	256 × 256	40 × 256	6.4000
4.1.06	256 × 256	60 × 256	4.2667
4.1.07	256 × 256	23 × 256	11.1304
4.1.08	256 × 256	31 × 256	8.2581
4.2.01	512 × 512	71 × 512	7.2113
4.2.03	512 × 512	147 × 512	3.4830
4.2.05	512 × 512	78 × 512	6.5641
4.2.06	512 × 512	116 × 512	4.4138
4.2.07	512 × 512	101 × 512	5.0693

3.6. Robustness analysis

Robustness is an important index in the evaluation of image encryption algorithm. It measures whether the encryption algorithm can effectively protect the content of the image from damage or leakage in

Table 12
Comparison of image compression rate and compressed size using $2^{-3}Q$ quantization matrix for compression.

Filename	Original image size	Compressed image size	Compression ratio
2.1.01	512 × 512	103 × 512	4.9709
2.1.02	512 × 512	111 × 512	4.6126
2.1.03	512 × 512	56 × 512	9.1429
2.1.04	512 × 512	105 × 512	4.8762
2.1.05	512 × 512	80 × 512	6.4000
2.1.06	512 × 512	99 × 512	5.1717
2.1.07	512 × 512	67 × 512	7.6418
2.1.10	512 × 512	83 × 512	6.1687
2.1.12	512 × 512	80 × 512	6.4000
2.2.01	1024 × 1024	189 × 1024	5.4180
2.2.02	1024 × 1024	98 × 1024	10.4490
2.2.03	1024 × 1024	121 × 1024	8.4628
2.2.04	1024 × 1024	164 × 1024	6.2439
2.2.05	1024 × 1024	172 × 1024	5.9535
2.2.08	1024 × 1024	189 × 1024	5.4180
2.2.09	1024 × 1024	119 × 1024	8.6050
2.2.10	1024 × 1024	106 × 1024	9.6604
4.1.01	256 × 256	28 × 256	9.1429
4.1.02	256 × 256	26 × 256	9.8462
4.1.03	256 × 256	17 × 256	15.0588
4.1.04	256 × 256	26 × 256	9.8462
4.1.05	256 × 256	27 × 256	9.4815
4.1.06	256 × 256	43 × 256	5.9535
4.1.07	256 × 256	16 × 256	16.0000
4.1.08	256 × 256	21 × 256	12.1905
4.2.01	512 × 512	45 × 512	11.3778
4.2.03	512 × 512	109 × 512	4.6972
4.2.05	512 × 512	52 × 512	9.8462
4.2.06	512 × 512	82 × 512	6.2439
4.2.07	512 × 512	67 × 512	7.6418

Table 13
Comparison of image compression rate and compressed size using $2^{-2}Q$ quantization matrix for compression.

Filename	Original image size	Compressed image size	Compression ratio
2.1.01	512 × 512	69 × 512	7.4203
2.1.02	512 × 512	77 × 512	6.6494
2.1.03	512 × 512	35 × 512	14.6286
2.1.04	512 × 512	70 × 512	7.3143
2.1.05	512 × 512	53 × 512	9.6604
2.1.06	512 × 512	66 × 512	7.7576
2.1.07	512 × 512	43 × 512	11.9070
2.1.09	512 × 512	33 × 512	15.5152
2.1.12	512 × 512	51 × 512	10.0392
2.2.01	1024 × 1024	129 × 1024	7.9380
2.2.02	1024 × 1024	57 × 1024	17.9649
2.2.03	1024 × 1024	77 × 1024	13.2987
2.2.04	1024 × 1024	106 × 1024	9.6604
2.2.05	1024 × 1024	110 × 1024	9.3091
2.2.08	1024 × 1024	129 × 1024	7.9380
2.2.09	1024 × 1024	75 × 1024	13.6533
2.2.10	1024 × 1024	66 × 1024	15.5152
4.1.01	256 × 256	18 × 256	14.2222
4.1.02	256 × 256	18 × 256	14.2222
4.1.03	256 × 256	10 × 256	25.6000
4.1.04	256 × 256	18 × 256	14.2222
4.1.05	256 × 256	18 × 256	14.2222
4.1.06	256 × 256	29 × 256	8.8276
4.1.07	256 × 256	11 × 256	23.2727
4.1.08	256 × 256	14 × 256	18.2857
4.2.01	512 × 512	30 × 512	17.0667
4.2.03	512 × 512	74 × 512	6.9189
4.2.05	512 × 512	35 × 512	14.6286
4.2.06	512 × 512	51 × 512	10.0392
4.2.07	512 × 512	40 × 512	12.8000

Table 14
Comparison of image compression rate and compressed size using $2^{-1}Q$ quantization matrix for compression.

Filename	Original image size	Compressed image size	Compression ratio
2.1.01	512 × 512	45 × 512	11.3778
2.1.02	512 × 512	52 × 512	9.8462
2.1.03	512 × 512	21 × 512	24.3810
2.1.04	512 × 512	44 × 512	11.6364
2.1.05	512 × 512	36 × 512	14.2222
2.1.06	512 × 512	43 × 512	11.9070
2.1.07	512 × 512	26 × 512	19.6923
2.1.10	512 × 512	34 × 512	15.0588
2.1.12	512 × 512	34 × 512	15.0588
2.2.01	1024 × 1024	84 × 1024	12.1905
2.2.02	1024 × 1024	32 × 1024	32.0000
2.2.03	1024 × 1024	50 × 1024	20.4800
2.2.04	1024 × 1024	68 × 1024	15.0588
2.2.05	1024 × 1024	72 × 1024	14.2222
2.2.08	1024 × 1024	83 × 1024	12.3373
2.2.09	1024 × 1024	46 × 1024	22.2609
2.2.10	1024 × 1024	40 × 1024	25.6000
4.1.01	256 × 256	12 × 256	21.3333
4.1.02	256 × 256	12 × 256	21.3333
4.1.03	256 × 256	7 × 256	36.5714
4.1.04	256 × 256	12 × 256	21.3333
4.1.05	256 × 256	12 × 256	21.3333
4.1.06	256 × 256	19 × 256	13.4737
4.1.07	256 × 256	8 × 256	32.0000
4.1.08	256 × 256	10 × 256	25.6000
4.2.01	512 × 512	19 × 512	26.9474
4.2.03	512 × 512	47 × 512	10.8936
4.2.05	512 × 512	23 × 512	22.2609
4.2.06	512 × 512	32 × 512	16.0000
4.2.07	512 × 512	25 × 512	20.4800

Table 15
Comparison of image compression rate and compressed size using $2^{0}Q$ quantization matrix for compression.

Filename	Original image size	Compressed image size	Compression ratio
2.1.01	512 × 512	29 × 512	17.6552
2.1.02	512 × 512	34 × 512	15.0588
2.1.03	512 × 512	13 × 512	39.3846
2.1.04	512 × 512	28 × 512	18.2857
2.1.05	512 × 512	24 × 512	21.3333
2.1.06	512 × 512	28 × 512	18.2857
2.1.07	512 × 512	16 × 512	32.0000
2.1.10	512 × 512	22 × 512	23.2727
2.1.12	512 × 512	22 × 512	23.2727
2.2.01	1024 × 1024	51 × 1024	20.0784
2.2.02	1024 × 1024	18 × 1024	56.8889
2.2.03	1024 × 1024	32 × 1024	32.0000
2.2.04	1024 × 1024	43 × 1024	23.8140
2.2.05	1024 × 1024	48 × 1024	21.3333
2.2.08	1024 × 1024	51 × 1024	20.0784
2.2.09	1024 × 1024	29 × 1024	35.3103
2.2.10	1024 × 1024	25 × 1024	40.9600
4.1.01	256 × 256	8 × 256	32.0000
4.1.02	256 × 256	8 × 256	32.0000
4.1.03	256 × 256	5 × 256	51.2000
4.1.04	256 × 256	8 × 256	32.0000
4.1.05	256 × 256	8 × 256	32.0000
4.1.06	256 × 256	13 × 256	19.6923
4.1.07	256 × 256	6 × 256	42.6667
4.1.08	256 × 256	7 × 256	36.5714
4.2.01	512 × 512	12 × 512	42.6667
4.2.03	512 × 512	31 × 512	16.5161
4.2.05	512 × 512	16 × 512	32.0000
4.2.06	512 × 512	21 × 512	24.3810
4.2.07	512 × 512	16 × 512	32.0000

the face of various interference noises. In the real world, images may be affected by a variety of disturbances. Therefore, it is very important to analyze and evaluate the anti-interference ability of image encryption algorithm. This section selects salt and pepper noise and occlusion attacks to analyze the only-encryption algorithm.

3.6.1. Salt and pepper noise analysis

Separately add 1%, 3% and 5% salt and pepper noise into the ciphertext image. We can see from Fig. 15 that the ciphertext image adding noise can still have effective recognizable image information after decryption.

Table 16
Sensitivity analysis of different keys.

	R		G		B	
	NPCR	UACI	NPCR	UACI	NPCR	UACI
$r^{(1)} + 10^{-14}$	99.5884	33.4087	99.5710	33.6021	99.6198	33.4655
$x_0^{(1)} + 10^{-14}$	99.6286	33.4639	99.6408	33.4932	99.6076	33.3180
$y_0^{(1)} + 10^{-14}$	99.6076	33.3326	99.5972	33.6335	99.5780	33.5065
$r^{(2)} + 10^{-14}$	99.6111	33.3651	99.5762	33.3844	99.6059	33.4201
$x_0^{(2)} + 10^{-14}$	99.6111	33.5104	99.6443	33.4358	99.5989	33.3865
$y_0^{(2)} + 10^{-14}$	99.5745	33.5302	99.5675	33.6344	99.6129	33.5442

Table 17
Encryption time statistics of each module (Unit: second).

Module	512 × 512 × 3	256 × 256 × 3
Chaos	0.000064	0.000016
Compression	0.305211	0.098118
Permutation	0.005351	0.003886
Diffusion	0.075429	0.023302

Table 18
Comparison results with some state-of-the-art algorithms (Unit: second).

Algorithm	Time
Proposed	0.125322
Raghuvanshi et al. (2024)	3.069000
Li (2024)	0.243100
Su et al. (2024)	0.700100
Tang et al. (2024)	0.858302
Lai and Liu (2023)	0.151330

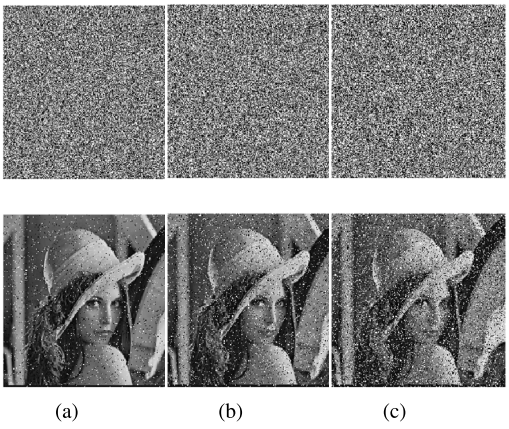


Fig. 15. The ciphertext and decryption image after adding salt and pepper noise: (a) Scrambling 1% salt and pepper noise ciphertext image and decrypted image; (b) Scrambling 3% salt and pepper noise ciphertext image and decrypted image; (c) Scrambling 5% salt and pepper noise ciphertext image and decrypted image.

3.6.2. Occlusion attack analysis

Respectively add occlusion noise, whose sizes are 32 × 32,48 × 48, and 64 × 64, into ciphertext image and we can see from Fig. 16 that the ciphertext image adding noise can still have effective recognizable image information after decryption.

4. Conclusion

This paper proposes a security-enhanced image privacy protection scheme that combines high compression ratio and high-quality decryption restoration, leveraging Rubik's Cube transformation and image filtering diffusion. This approach is a well-coordinated solution to the tension between security and efficiency. The scheme integrates color space conversion, DCT frequency domain transformation, compression coding, Rubik's Cube permutation, and image filtering diffusion for image encryption, ensuring robust image security and privacy. By incorporating mechanisms related to the plaintext, the scheme generates dynamic chaotic sequences, enhancing the encryption's resistance. Additionally, combining compression and encryption methods reduces computational complexity, boosting encryption efficiency. Simulations demonstrate the algorithm's superior performance across various key aspects, including high compression ratio, excellent image restoration quality, and strong resistance against common cryptographic attacks. The work in this paper provides a competitive and excellent technical idea for the increasingly serious digital image privacy problem in the new generation network communication environment.

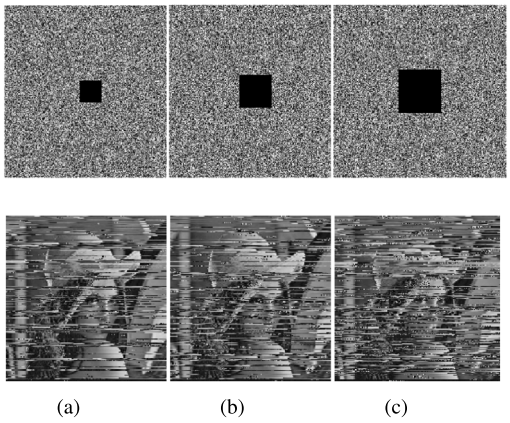


Fig. 16. The ciphertext and decryption image after adding occlusion noise: (a) Scrambling 32 × 32 occlusion noise ciphertext image and decrypted image; (b) Scrambling 48 × 48 occlusion noise ciphertext image and decrypted image; (c) Scrambling 64 × 64 occlusion noise ciphertext image and decrypted image.

CRediT authorship contribution statement

Yiting Lin: Conceptualization, Methodology, Project administration, Supervision, Resources, Software, Writing – original draft, Writing – review & editing. **Zhiyu Xie:** Data curation, Visualization, Investigation, Writing – review & editing. **Tingting Chen:** Visualization, Formal Analysis, Writing – review & editing. **Xiyuan Cheng:** Investigation, Formal Analysis, Writing – review & editing. **Heping Wen:** Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grant 62271130, the Science and Technology Foundation of Guangdong Province under Grant 2023A1515010066, Guangdong Basic and Applied Basic Research Foundation under Grant 2022A1515010317 and 2023A1515011717, Special Projects for Key Fields of the Education Department of Guangdong Province under Grant 2023ZDZX1041, the Key Laboratory of Guangdong Higher Education Institutes under Grant 2023KSYS011, and Special Funds for the Cultivation of Guangdong College Students' Scientific and Technological Innovation ("Climbing Program" Special Funds) under Grant pdjh2024a428 and pdjh2023b0600.

References

Abdelfatah, R. I. (2024). Robust biometric identity authentication scheme using quantum voice encryption and quantum secure direct communications for cybersecurity. *Journal of King Saud University - Computer and Information Sciences*, *36*(5), Article 102062.

Abdo, A., Karamany, T. S., & Yakoub, A. (2024). A hybrid approach to secure and compress data streams in cloud computing environment. *Journal of King Saud University - Computer and Information Sciences*, *36*(3), Article 101999.

Akbacak, E., Toktas, A., Erkan, U., & Gao, S. (2023). MLMQ-IR: Multi-label multi-query image retrieval based on the variance of hamming distance. *Knowledge-Based Systems*, *283*, Article 111193.

Cao, F., Guo, D., Wang, T., Yao, H., Li, J., & Qin, C. (2023). Universal screen-shooting robust image watermarking with channel-attention in DCT domain. *Expert Systems with Applications*, *238*, Article 122062.

li Chai, X., Song, S., Gan, Z., Long, G., Tian, Y., & He, X. (2023). CSENMT: A deep image compressed sensing encryption network via multi-color space and texture feature. *Expert Systems with Applications*, *241*, Article 122562.

Chai, X., Tang, Z., Gan, Z., Lu, Y., Wang, B., & Zhang, Y. (2024). SE-NDEND: A novel symmetric watermarking framework with neural network-based chaotic encryption for internet of medical things. *Biomedical Signal Processing and Control*, *90*, Article 105877.

Chen, Z., li Chai, X., Gan, Z., Wang, B., & Zhang, Y. (2024). RAE-VWP: A reversible adversarial example-based privacy and copyright protection method of medical images for internet of medical things. *IEEE Internet of Things Journal*, 1.

Devi, K. J., Singh, P., Bilal, M., & Nayyar, A. (2023). Enabling secure image transmission in unmanned aerial vehicle using digital image watermarking with H-grey optimization. *Expert Systems with Applications*, *236*, Article 121190.

Erkan, U., Toktas, A., & Lai, Q. (2023). Design of two dimensional hyperchaotic system through optimization benchmark function. *Chaos, Solitons & Fractals*, *167*, Article 113032.

Feng, W., Zhang, J., Chen, Y., Qin, Z., Zhang, Y., Ahmad, M., & Woźniak, M. (2024). Exploiting robust quadratic polynomial hyperchaotic map and pixel fusion strategy for efficient image encryption. *Expert Systems with Applications*, *246*, Article 123190.

Gao, S., Iu, H. H.-C., Wang, M., Jiang, D., El-latif, A., Wu, R., & Tang, X. (2024). Design, hardware implementation, and application in video encryption of the 2D memristive cubic map. *IEEE Internet of Things Journal*, 1.

Gao, S., Liu, S., Wang, X., Wu, R., Wang, J., Li, Q., & Tang, X. (2023). New image encryption algorithm based on hyperchaotic 3D-IHAL and a hybrid cryptosystem. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *53*(22), 27826–27843.

Gao, S., Wu, R., Wang, X., Liu, J., Li, Q., Wang, C., & Tang, X. (2023). Asynchronous updating boolean network encryption algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*, *33*(8), 4388–4400.

Gong, M., li Chai, X., Lu, Y., & Zhang, Y. (2024). Exploiting four-dimensional chaotic systems with dissipation and optimized logical operations for secure image compression and encryption. *IEEE Transactions on Circuits and Systems for Video Technology*, 1.

Hua, Z., Jin, F., Xu, B., & Huang, H. (2018). 2D logistic-Sine-coupling map for image encryption. *Signal Processing*, *149*, 148–161.

Hua, Z., Xu, B., Jin, F., & Huang, H. (2019). Image encryption using josephus problem and filtering diffusion. *IEEE Access*, *7*, 8660–8674.

Hua, Z., & Zhou, Y. (2017). Design of image cipher using block-based scrambling and image filtering. *Information Sciences*, *396*, 97–113.

Jamal, S. S., Hazzazi, M. M., Khan, M. F., Bassfar, Z., Aljaedi, A., & Islam, Z. U. (2023). Region of interest-based medical image encryption technique based on chaotic S-boxes. *Expert Systems with Applications*, *238*, Article 122030.

Khalili, H., Chien, H.-J., Hass, A., & Sehatbakhsh, N. (2024). Context-aware hybrid encoding for privacy-preserving computation in IoT devices. *IEEE Internet of Things Journal*, *11*(1), 1054–1064.

Kocak, O., Erkan, U., Toktas, A., & Gao, S. (2023). PSO-based image encryption scheme using modular integrated logistic exponential map. *Expert Systems with Applications*, *237*, Article 121452.

Lai, Q., & Liu, Y. (2023). A cross-channel color image encryption algorithm using two-dimensional hyperchaotic map. *Expert Systems with Applications*, *223*, Article 119923.

Li, L. (2024). A novel chaotic map application in image encryption algorithm. *Expert Systems with Applications*, *252*, Article 124316.

Li, G.-D., Cheng, W.-C., Wang, Q., Cheng, L., Mao, Y., & Jia, H.-Y. (2024). Enhanced quantum secret sharing protocol for anonymous secure communication utilizing W states. *iScience*, *27*(6), Article 109836.

Li, C., Shen, X., & Liu, S. (2024). Cryptanalyzing an image encryption algorithm underpinned by 2D lag-complex logistic map. *IEEE MultiMedia*, 1–11.

Li, C., Tan, K.-Y., Feng, B., & Lü, J. (2017). The graph structure of the generalized discrete Arnold's cat map. *Institute of Electrical and Electronics Engineers. Transactions on Computers*, *71*(2), 364–377.

Li, H., Yu, S., Feng, W., Chen, Y., Zhang, J., Qin, Z., Zhu, Z., & Woźniak, M. (2023). Exploiting dynamic vector-level operations and a 2D-enhanced logistic modular map for efficient chaotic image encryption. *Entropy*, *25*(8), 1147.

Lu, Z., Feng, Q., Li, P., Lo, K.-T., & Huang, F. (2023). A privacy-preserving image retrieval scheme based on 16 × 16 DCT and deep learning. *IEEE Transactions on Cloud Computing*, *11*(3), 3314–3325.

Mou, J., Ma, T., Banerjee, S., & Zhang, Y. (2024). A novel memcapacitive-synapse neuron: Bionic modeling, complex dynamics analysis and circuit implementation. *IEEE Transactions on Circuits and Systems. I. Regular Papers*, *71*(4), 1771–1780.

Peng, Y., Lan, Z., Sun, K., & Xu, W. (2023). A simple color image encryption algorithm based on a discrete memristive hyperchaotic map and time-controllable operation. *Optics and Laser Technology*, *165*, Article 109543.

Qian, K., Xiao, Y., Wei, Y., Liu, D., Wang, Q., & Feng, W. (2023). A robust memristor-enhanced polynomial hyper-chaotic map and its multi-channel image encryption application. *Micromachines*, *14*(11), 2090.

Raghuvanshi, K., Kumar, S., Kumar, S., & Kumar, S. (2024). Image encryption algorithm based on DNA encoding and CNN. *Expert Systems with Applications*, *252*, Article 124287.

Saberikamarposhti, M., Ghorbani, A., & Yadollahi, M. (2024). A comprehensive survey on image encryption: Taxonomy, challenges, and future directions. *Chaos, Solitons & Fractals*, *178*, Article 114361.

Sardar, A., Umer, S., Rout, R. K., Sahoo, K. S., & Gandomi, A. H. (2024). Enhanced biometric template protection schemes for securing face recognition in IoT environment. *IEEE Internet of Things Journal*, 1.

Singh, M., Baranwal, N., Singh, K. N., & Singh, A. K. (2024a). Using GAN-based encryption to secure digital images with reconstruction through customized super resolution network. *IEEE Transactions on Consumer Electronics*, *70*(1), 3977–3984.

Singh, H. K., Baranwal, N., Singh, K. N., & Singh, A. K. (2024b). Using multimodal biometric fusion for watermarking of multiple images. *IEEE Transactions on Consumer Electronics*, *70*(1), 3487–3494.

Singh, H. K., & Singh, A. K. (2024). Using deep learning to embed dual marks with encryption through 3-D chaotic map. *IEEE Transactions on Consumer Electronics*, *70*(1), 3056–3063.

Song, W., Fu, C., Zheng, Y., Zhang, Y., Chen, J., & Wang, P. (2024). Batch image encryption using cross image permutation and diffusion. *Journal of Information Security and Applications*, *80*, Article 103686.

Su, Q., Chen, S., Wang, H., Cao, H., & Hu, F. (2024). An efficient watermarking scheme for dual color image with high security in 5G environment. *Expert Systems with Applications*, *249*, Article 123818.

Tanaka, H. K. M. (2023). Cosmic coding and transfer for ultra high security near-field communications. *iScience*, *26*(2), Article 105897.

Tang, S., Xu, X., Jiang, Z.-H., Meng, D., & Sun, K. (2024). An image encryption scheme without additional key transmission based on an N-dimensional closed-loop coupled triangular wave model. *Chaos, Solitons & Fractals*, *185*, Article 115039.

The USC-SIPI Image Database (1977). https://sipi.usc.edu/database/database.php.

Toktas, A., Erkan, U., Gao, S., & Pak, C. (2024). A robust bit-level image encryption based on Bessel map. *Applied Mathematics and Computation*, *462*, Article 128340.

Toktas, A., Erkan, U., Ustun, D., & Wang, X. (2023). Parameter optimization of chaotic system using Pareto-based triple objective artificial bee colony algorithm. *Neural Computing and Applications*, *35*(18), 13207–13223.

Toktas, F., Erkan, U., & Yetgin, Z. (2024). Cross-channel color image encryption through 2D hyperchaotic hybrid map of optimization test functions. *Expert Systems with Applications*, *249*, Article 123583.

ur Rehman, M. (2024). Quantum-enhanced chaotic image encryption: Strengthening digital data security with 1-D Sine-based chaotic maps and quantum coding. *Journal of King Saud University - Computer and Information Sciences*, *36*(3), Article 101980.

Wang, X., Liu, C., & Jiang, D. (2021). A novel triple-image encryption and hiding algorithm based on chaos, compressive sensing and 3D DCT. *Information Sciences*, *574*, 505–527.

Wang, B., & Lo, K.-T. (2024). Autoencoder-based joint image compression and encryption. *Journal of Information Security and Applications*, *80*, Article 103680.

Wen, H., Chen, R., Yang, J., Zheng, T., Wu, J., Lin, W., Jian, H., Lin, Y., Ma, L., Liu, Z., & Zhang, C. (2023). Security analysis of a color image encryption based on bit-level and chaotic map. *Multimedia Tools and Applications*, *83*(2), 4133–4149.

Wen, W., Fan, J., Zhang, Y., & Fang, Y. (2023). APCAS: Autonomous privacy control and authentication sharing in social networks. *IEEE Transactions on Computational Social Systems*, *10*(6), 3169–3180.

Wen, H., Huang, Y., & Lin, Y. (2023). High-quality color image compression-encryption using chaos and block permutation. *Journal of King Saud University - Computer and Information Sciences*, *35*(8), Article 101660.

Wen, W., Huang, H., Qi, S., Zhang, Y., & Fang, Y. (2024). Joint coverless steganography and image transformation for covert communication of secret messages. *IEEE Transactions on Network Science and Engineering*, *11*(3), 2951–2962.

Wen, W., Huang, M., Zhang, Y., Fang, Y., & Zuo, Y. (2023). Visual security index combining CNN and filter for perceptually encrypted light field images. *ACM Transactions on Multimedia Computing, Communications and Applications*, *20*(1).

Wen, W., Jiang, Q., Huang, H., Zhang, Y., & Fang, Y. (2024). TPE-DF: Thumbnail preserving encryption via dual-2DCS fusion. *IEEE Signal Processing Letters*, 1–5.

Wen, H., & Lin, Y. (2023). Cryptanalyzing an image cipher using multiple chaos and DNA operations. *Journal of King Saud University - Computer and Information Sciences*, *35*(7), Article 101612.

Wen, H., & Lin, Y. (2024). Cryptanalysis of an image encryption algorithm using quantum chaotic map and DNA coding. *Expert Systems with Applications*, *237*, Article 121514.

Wen, H., Lin, Y., & Feng, Z. (2024). Cryptanalyzing a bit-level image encryption algorithm based on chaotic maps. *Engineering Science and Technology, an International Journal*, *51*, Article 101634.

Wen, H., Lin, Y., Kang, S., Zhang, X., & Zou, K. (2024). Secure image encryption algorithm using chaos-based block permutation and weighted bit planes chain diffusion. *iScience*, *27*(1), Article 108610.

Wen, H., Lin, Y., Yang, L., & Chen, R. (2024). Cryptanalysis of an image encryption scheme using variant hill cipher and chaos. *Expert Systems with Applications*, Article 123748.

Wen, W., Yuan, Z., Qi, S., Zhang, Y., & Fang, Y. (2024). PPM-SEM: A privacy-preserving mechanism for sharing electronic patient records and medical images in telemedicine. *IEEE Transactions on Multimedia*, *26*, 5795–5806.

Yamni, M., Karmouni, H., Sayyouri, M., & Qjidaa, H. (2022). Robust audio watermarking scheme based on fractional Charlier moment transform and dual tree complex wavelet transform. *Expert Systems with Applications*, *203*, Article 117325.

Ye, G., Du, S., & ling Huang, X. (2023). Image compression-hiding algorithm based on compressive sensing and integer wavelet transformation. *Applied Mathematical Modelling*.

Ye, G., Liu, M., Yap, W.-S., & Goi, B.-M. (2023). Reversible image hiding algorithm based on compressive sensing and deep learning. *Nonlinear Dynamics*, *111*, 13535–13560.

Zhang, H., & Hu, H. (2023). An image encryption algorithm based on a compound-coupled chaotic system. *Digital Signal Processing*, *146*, Article 104367.

Zhang, H., Hu, H., & Ding, W. (2024). VSDHS-CIEA: Color image encryption algorithm based on novel variable-structure discrete hyperchaotic system and cross-plane confusion strategy. *Information Sciences*, *665*, Article 120332.

Zhang, Y., Ji, J., Wen, W., Zhu, Y., Xia, Z., & Weng, J. (2024). Understanding visual privacy protection: A generalized framework with an instance on facial privacy. *IEEE Transactions on Information Forensics and Security*, 1.

Zhang, Y., Ye, X., Xiao, X., Xiang, T., Li, H., & Cao, X. (2023). A reversible framework for efficient and secure visual privacy protection. *IEEE Transactions on Information Forensics and Security*, *18*, 3334–3349.

Zhang, Y., Zhu, J., Xue, M., Zhang, X., & Cao, X. (2024). Adaptive 3D mesh steganography based on feature-preserving distortion. *IEEE Transactions on Visualization and Computer Graphics*, 1–13.