future internet

MDPI

Article

An Innovative Information Hiding Scheme Based on Block-Wise Pixel Reordering

Jui-Chuan Liu [1], Heng-Xiao Chi [1], Ching-Chun Chang [2] and Chin-Chen Chang [1,*]

[1] Department of Information Engineering and Computer Science, Feng Chia University, Taichung 40724, Taiwan; p1200318@o365.fcu.edu.tw (J.-C.L.); hx9704@gmail.com (H.-X.C.)
[2] Information and Communication Security Research Center, Feng Chia University, Taichung 40724, Taiwan; ccc@fcu.edu.tw
* Correspondence: alan3c@gmail.com

Abstract: Information has been uploaded and downloaded through the Internet, day in and day out, ever since we immersed ourselves in the Internet. Data security has become an area demanding high attention, and one of the most efficient techniques for protecting data is data hiding. In recent studies, it has been shown that the indices of a codebook can be reordered to hide secret bits. The hiding capacity of the codeword index reordering scheme increases when the size of the codebook increases. Since the codewords in the codebook are not modified, the visual performance of compressed images is retained. We propose a novel scheme making use of the fundamental principle of the codeword index reordering technique to hide secret data in encrypted images. By observing our experimental results, we can see that the obtained embedding capacity of 197,888 is larger than other state-of-the-art schemes. Secret data can be extracted when a receiver owns a data hiding key, and the image can be recovered when a receiver owns an encryption key.

Keywords: data hiding; vector quantization; codebook; codeword index reordering

Citation: Liu, J.-C.; Chi, H.-X.; Chang, C.-C.; Chang, C.-C. An Innovative Information Hiding Scheme Based on Block-Wise Pixel Reordering. *Future Internet* **2024**, *16*, 34. https://doi.org/10.3390/fi16010034

Academic Editor: Carlo Blundo

Received: 26 December 2023
Revised: 17 January 2024
Accepted: 19 January 2024
Published: 22 January 2024

1. Introduction

In the modern day, iCloud data transmission replaces the physical mail to speed up the time spent on exchanging information. Needless to mention, the Internet is a key element in the transmission process. Digital activities involving both the virtual world and the physical world can be foreseen in the near future, such as if a hospital technician uploads an X-ray image, a CT scan photo, or a patient's information to a data center for a list of specific doctors to download them in order to discuss various treatments over an online meeting. To protect the privacy of the patient, no one should be able to see the information other than the related doctors. Encrypting and decrypting digital information are becoming more and more important in order to increase security. At the same time, there can be diagnostics information to be passed to one or some particular doctors. Data hiding then plays another crucial role in these data transitions [1,2].

When a content owner wants to send an image to a data receiver, the owner would like to add a secret message in the image before transmitting it to the receiver. Adding the secret message normally needs to go through a data hider, and as it may not be desirable for the data hider to see the image content, the owner encrypts the image before sending it in order to hide the message. A receiver holding the encryption key can recover the image, and a receiver with the data hiding key can extract the secret information.

There are various data hiding schemes designed to protect data effectively in different application circumstances that have demonstrated exceptional results. Data hiding has branched out into reversible data hiding (RDH) [3–7] and non-reversible data hiding depending on whether the image content can be recovered or not. According to the hiding carriers, the hiding schemes can be cataloged into four different domains: spatial, frequency, compression, and encryption. When in the spatial domain, digital cover images

are modified directly to embed data. As for the frequency domain, images are transformed using wavelet transform methods before embedding data. Images are first compressed with compression techniques prior to embedding information in the compression domain. In the encryption domain, images are encrypted using encryption keys before being sent to the data hider to hide secrets in the encrypted image. Data hiding in encrypted images (DH-EI) combines cryptography and DH technology to achieve higher levels of protection. DH-EI has branched out into reserving room before encryption (RRBE) and vacating room after encryption (VRAE) depending on the timing of room vacating.

Ma et al. [8] proposed a reversible method for data hiding in encrypted images by preserving room before encryption in 2013. Yi and Zhou proposed a parameter-controlled method to embed data to encrypted images in 2018 [9,10]. Around the same time, Pauline and William [11] proposed a method using MSB prediction and gained high embedding capacity by hiding data in encrypted images. Chen et al. [12], in 2020, further improved the method by proposing a multi-MSB compression method. Puteaux and Puech [13] proposed a fully reversible method by including the MSB error prediction and a reversible adaption in 2020, and Wu et al. [14] proposed an improved version shortly after. Even though RRBE has a higher embedding capacity amount, owners may not have enough knowledge of how to go through the reserving process. Therefore, VRAE schemes still gain popularity among these types of applications to better secure information privacy.

In the VRAE scheme, the content owner performs only the operation of encrypting the image and then transmits the encrypted image to the data hider. After obtaining the encrypted image, the data hider vacates the room and embeds the data. Hong et al. [15] proposed a DH-EI method that combined side match and block-wise LSB flipping. Zhang [16] proposed a scheme to preserve space by utilizing compressed LSB planes, but the embedding capacity of this method was relatively low. Qian and Zhang [17] proposed a method to reserve embedded rooms using distributed source coding. In 2014, joint and separable DH-EI schemes using bit flipping and prediction error techniques were proposed by Wu and Sun [18]. Hung et al. [19], in 2016, proposed a DH-EI framework for encrypting images using block-wise stream cipher and shuffling. Since block-wise stream cipher and shuffling are used, the correlation between encrypted blocks of pixels is preserved so that DH techniques can be used directly in the encryption domain. In 2018, Ge et al. [20] combined the block-wise stream cipher and selected peaks for histogram shifting. In 2020, Bhardwaj and Aggarwal [21] used an improved block-based joint DHEI algorithm to obtain higher embedding rates. In 2022, Wang et al. [22] proposed a method to embed secret data using rotated pixel-blocks. Yu et al. [23] used MSB replacements to embed data and then used the complexity among neighboring pixels to restore the image.

Our novel scheme focuses on using VRAE to hide data and incorporates vector quantization codeword index reordering [4], Stream Cipher, and LSB replacement [5] techniques to implement the scheme. Our research goal is to achieve a larger embedding capacity than other methods without degrading the visual quality of recovered images. The core contributions of the scheme are described below:

- The scheme offers large embedding capacity.
- It sustains the visual quality of encrypted images.
- Extracting secret messages and recovering images can be independent.

The rest of the paper is structured as follows: Section 2 discusses the Background of the Works, including the LSB encryption, the applied sorting, and the key index reordering method. Section 3 describes the details of the novel scheme, and Section 4 shows the experiments conducted and their analyses. Finally, Section 5 is the conclusion.

2. Background of the Work

Our novel scheme employs the principal idea of a vector quantization (VQ) codeword index reordering scheme [4] as the basic technique to embed and extract data. This section describes the data embedding and data extraction using the codeword index reordering.

When there is a codebook, it is sorted first before embedding secret data. The data is embedded through the new indices in a stego codebook. Figure 1 indicates the flow of the codeword index reordering scheme including data embedding, data extraction, and image recovery. The stego codebook with the new ordered indices is then sent to receivers to extract the data.

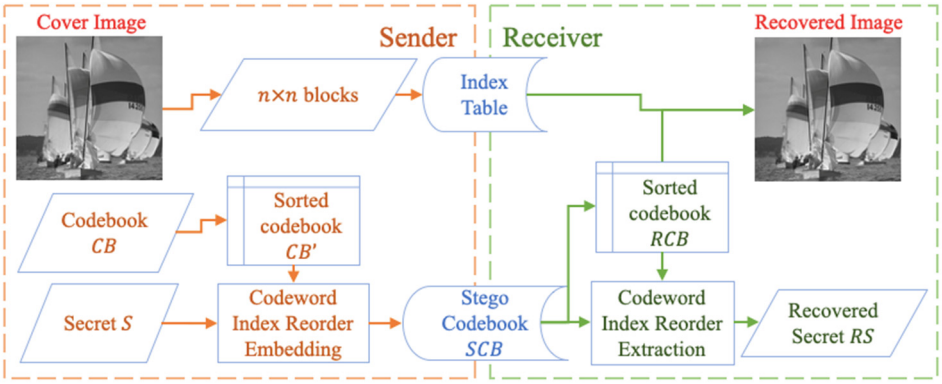


Figure 1. Flow of codeword index reordering scheme.

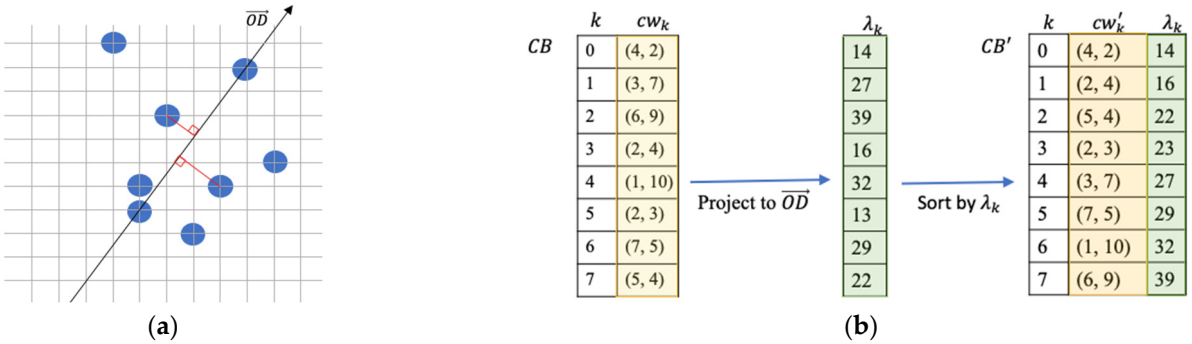
2.1. Sort Codewords by Projected Values

A sorted codebook is an essential component for a codeword index reordering scheme [4]. There is a VQ codebook $CB = \{cw_0, cw_1, \cdots, cw_{m-1}\}$, where m is the number of the codewords in the codebook. A data hider will find an n-dimensional point $D = \{r_1, r_2, \cdots, r_n\}$, where $r_1, r_2, \cdots, r_n$ are randomly generated using a random seed. A line $\overrightarrow{OD}$ connecting D and the origin $O = \{0, 0, \cdots, 0\}$ is a line that codewords in a codebook can project and obtain their projected values. These projected values are denoted as $\lambda_1, \lambda_2, \ldots, \lambda_n$ using Equation (1) and are to be used to sort the codewords in the codebook.

$$\lambda_k = cw_k \cdot \overrightarrow{OD}, \quad 0 \leq k \leq m-1. \tag{1}$$

The codebook CB is sorted and resulted to a sorted codebook $CB' = \{cw'_0, cw'_1, \cdots, cw'_{m-1}\}$.

An example of a codebook consisting of eight two-dimensional codewords is demonstrated in Figure 2 to provide a better understanding of the projected values and the sorting result according to their projected values.



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

Figure 2. (**a**) Project to line $\overrightarrow{OD}$; (**b**) sort codewords using line-projected values.

2.2. Data Embedding of the Codeword Index Reordering

When there is a secret data $S = \{s_0, s_1, \cdots, s_k\}$, $s_k = \{0, 1\}$, the embedding procedure is organized as follows:

Step 1: Initialize a stego codebook SCB.

Step 2: Calculate the number of bits to embed

$$b = \lfloor log_2(length(SCB)) \rfloor. \tag{2}$$

Step 3: Convert secret bits to a decimal index

$$(indx)_{10} = (s_0 \parallel s_1 \parallel \cdots \parallel s_{b-1})_2. \tag{3}$$

Step 4: Move the codeword scw_{indx} out from $CB\prime$ and add to SCB.
Step 5: Repeat Step 2 through Step 4 until there are no codewords in $CB\prime$.
Step 6: Send SCB to receivers.
Using the example in Figure 1, the embedding results are illustrated in Figure 3.

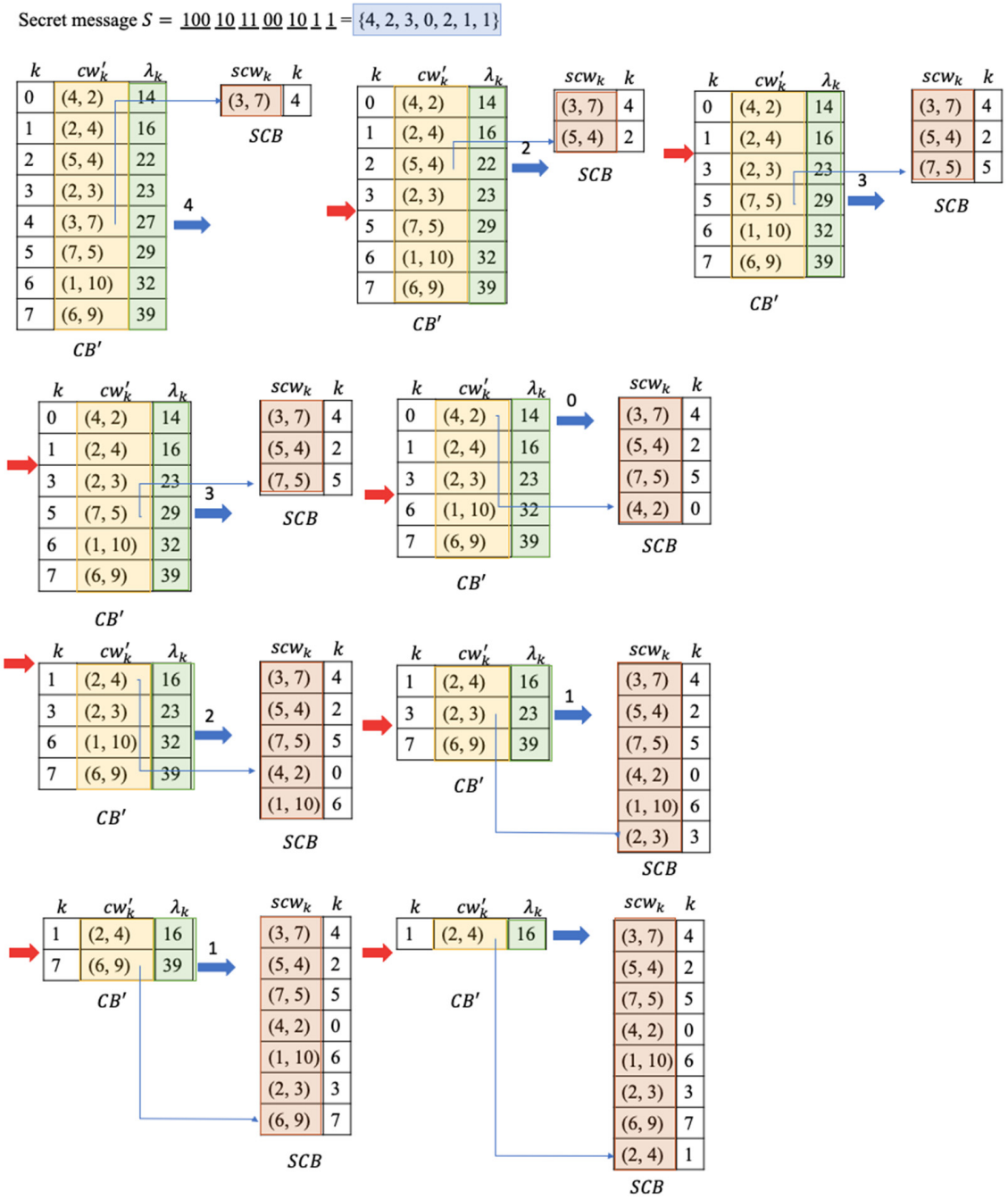


Figure 3. Data embedding of a codeword index reordering scheme.

2.3. Data Extraction of the Codeword Index Reordering

When a receiver receives a stego codebook $SCB = \{scw_0, scw_1, \cdots, scw_{m-1}\}$, the stego codebook needs to be sorted by the projected values of codewords using the received projecting line $\overrightarrow{OD}$ before extracting secret data. The following steps showing how the secret data are extracted:

Step 1: Project the codewords in SCB to the line $\overrightarrow{OD}$ and sort the codebook by using the projected values of the codewords to obtain the recovered codebook RCB.

Step 2: Initialize recovered secret RS.

Step 3: Initialize the current codeword index ci to 0.

Step 4: Match the current codeword scw_i with the codewords in the recovered codebook RCB and obtain the index si in the RCB.

Step 5: Convert $(si)_{10}$ to a binary bit stream $RS' = (rs_0 \parallel rs_1 \parallel \cdots \parallel rs_{k\prime})_2$.

Step 6: Append RS' to RS.

Step 7: Increase the current codeword index ci by 1.

Step 8: Repeat Step 4 through Step 7 until all codewords in SCB are exhausted.

Continuing to use the example above, Figure 4 demonstrates how the data are extracted from the stego codebook SCB.

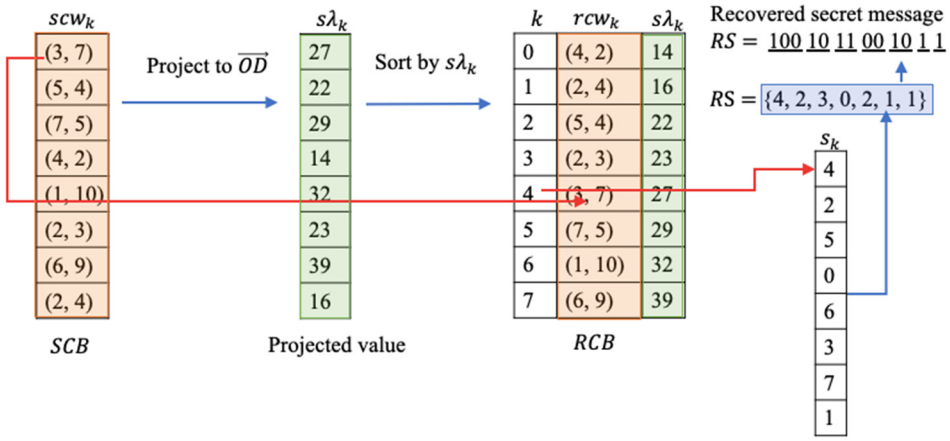


Figure 4. Data extraction of a codeword index reordering scheme.

3. Proposed Scheme

To protect the privacy of original cover images, our novel scheme encrypts the cover image before data embedding using an encryption key. A content owner sends the encrypted image to a data hider to hide secret messages. The data hider sends the marked images, the encryption key and the data hiding key to designated receivers after hiding data. In order to be able to simulate the codeword table to manipulate index order as the codeword index reordering scheme in Section 2 Background of the Work, Section 3.2 Codeword Table Formation describes how codeword tables are generated.

Figure 5 shows the framework of the proposed scheme. The content owner uses an encryption key k_e to activate a random number generator and uses the stream Cipher to encrypt an image and generate an encrypted image. The encrypted image is then divided into pixel-blocks and block-groups. In order to be reversible, the LSBs are used to record the pixel-block index in its block-group. The LSB-replaced encrypted image is then sent to the data hider. The data hider used the same method to divide pixel-blocks and block-groups to create codeword tables. The codeword table is sorted by using projected values to a line generated based on a data hiding key k_d. After sorting, the table indices are used to hide secret messages. A marked image is generated and sent to receivers after secret messages are embedded. A marked image is divided into pixel-blocks and block-groups to form the codeword tables using the same method as the data hider after receiving. When a receiver owns a data hiding key k_d, the codeword table is sorted using the key and the

secret message can be extracted by finding indices of matched codewords. If a receiver has an encryption key k_e, the encrypted image can be recovered partially. If a receiver obtains both keys, both the secret message and the encrypted image can be recovered with minor distortions.

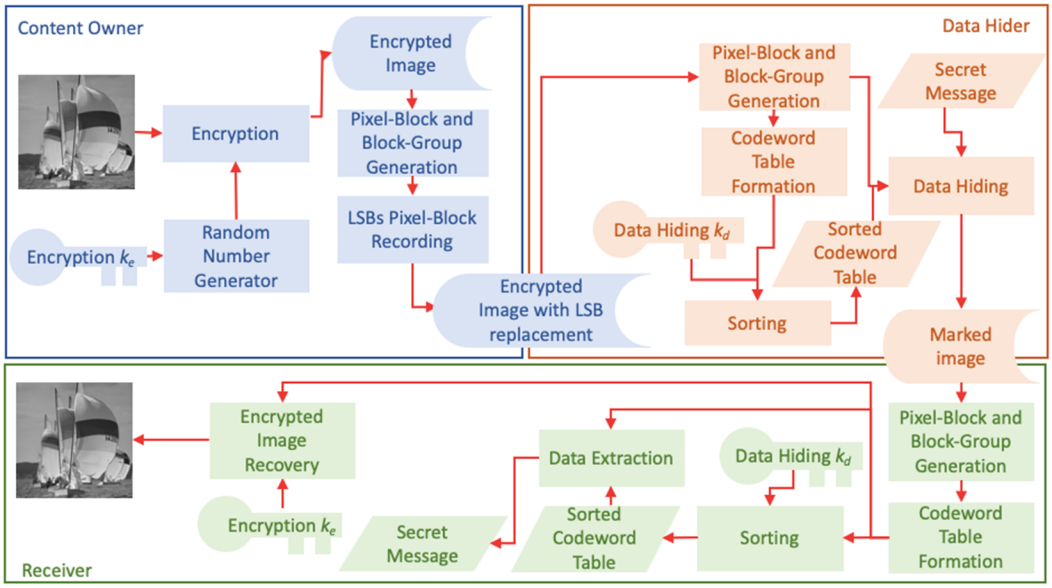


Figure 5. Framework of block-group index reordering scheme.

3.1. Image Encryption

Content owners can use any of the existing encryption methods to protect the privacy of the original images and send the encrypted images to third-party data hiders. The Stream Cipher technique, an exclusive OR operation to encrypt a cipher stream generated by using an encryption key k_e, is a well-known, simple, and efficient method to encrypt images. When there is an original image I of size $M \times N$, a pseudorandom matrix used as an encryption key k_e with the matching size of the image I and whose values range between 0 and 255 is generated by using a random seed value. The stream cipher is an encryption algorithm which uses the encryption key k_e in both encryption and decryption. A pixel value and its corresponded encryption key value are converted to 8-bit binary system using Equations (4) and (5):

$$I_{ij}^{x} = \left\lceil \frac{I_{ij}}{2^{x-1}} \right\rceil \; mod \; 2, \tag{4}$$

where ij represents the (i, j) coordinates, $1 \leq i \leq M$ and $1 \leq j \leq N$.

$$k_{e_{ij}}^{x} = \left\lceil \frac{k_{eij}}{2^{x-1}} \right\rceil \; mod \; 2, \tag{5}$$

where ij represents the (i, j) coordinates, $1 \leq i \leq M$, $1 \leq j \leq N$, and x is the bit order from right to left and can be represented as $x = 1, 2, \cdots, 8$. After binary conversion, a bit-level XOR as Equation (6) is applied:

$$I_{e_{ij}}^{x} = I_{ij}^{x} \oplus k_{e_{ij}}^{x}. \tag{6}$$

The encrypted image $I_{e_{ij}}^{x}$ is converted back to a decimal encrypted image

$$I'_{ij} = \sum_{x=1}^{8} I_{e_{ij}}^{x} \times 2^{x-1}. \tag{7}$$

3.2. Codeword Table Formation

After a data hider obtains the encrypted image I', it is converted into a pixel stream. Depending on the block size desired, it is broken into pixel-blocks. An owner can set the number of blocks in a group to perform the index reordering technique which is derived from the codeword index reordering detailed in Background of the Work. We are using the same technique as the codeword index reordering to embed data into the block-groups generated from the encrypted image.

After obtaining the encrypted image I' which consists of $M \times N$ pixels, it is denoted as $I' = \left\{p_0,\ p_1,\ \cdots, p_{(M\times N)-1}\right\}$. The encrypted image is cut into pixel-blocks, and each pixel-block contains n contiguous pixels. A current pixel-block can be denoted as $b_c = \left\{p_{n(c-1)},\ p_{n(c-1)-1},\ \cdots, p_{n(c-1)-(n-1)}\right\}$, where c is the number of the current pixel-block. After the pixel-blocks are formed, m pixel-blocks are grouped toobtainher to be a block-group $g_l = \left\{b_{m(l-1)},\ b_{m(l-1)+1},\ \cdots, b_{m(l-1)+(m-1)}\right\}$, where l is the number of the current block-group and m is the length of a desired codeword table for the codeword index reordering detailed in Section 2. The total number of block-groups is $K = \frac{(M\times N)}{(m\times n)}$ and the block-groups in the encrypted image I' can be represented as $G = \{g_0,\ g_1,\ \cdots,\ g_{K-1}\}$. A codeword table containing m n-dimensional codewords is generated for each corresponding block-group g_l.

Figure 6 is an example of how pixel-blocks and block-groups are created for a 512×512 image. When a pixel-block consists of 8 pixels and a block-group combines 255 blocks, there are 32,768 pixel-blocks and 128 block-groups are formed in total for the encrypted image.

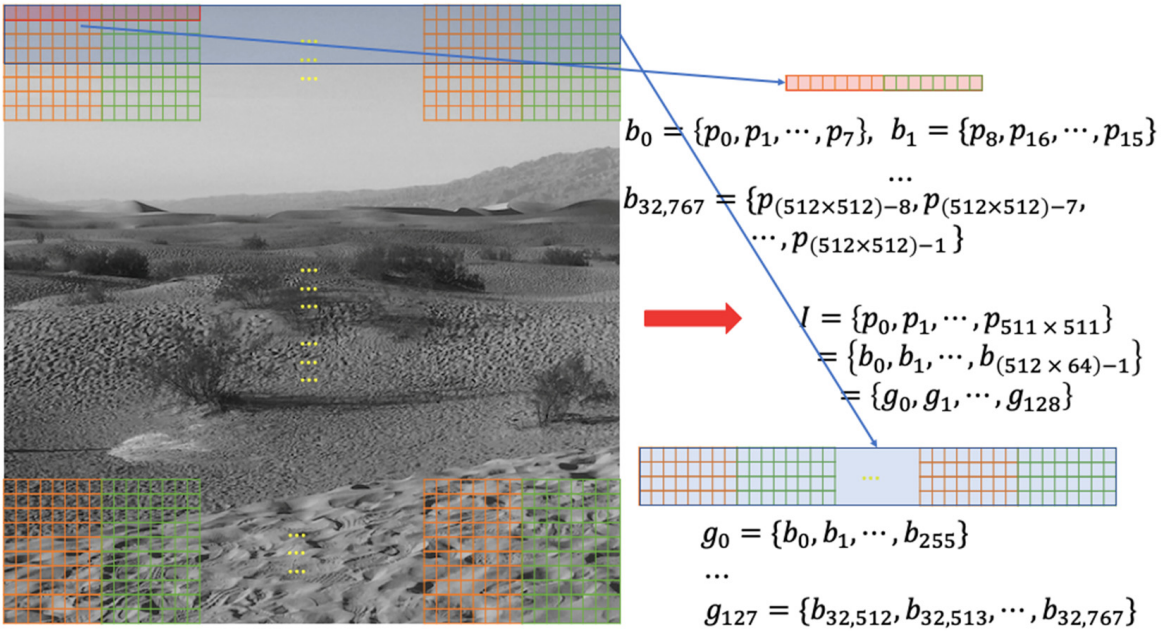
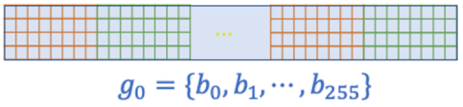


Figure 6. Cutting the pixel stream into pixel-blocks and forming block-groups.

A codeword table for the block-group g_0 of the example in Figure 6 is illustrated in Figure 7. The codeword table is treated as a codebook used in the codeword index reordering to hide data at a later stage.



Index	Pixel-Block	Codeword
0	b_0	$\{p_0, p_1, \cdots, p_7\}$
1	b_1	$\{p_8, p_9, \cdots, p_{15}\}$
...	...	...
255	b_{255}	$\{p_{2,040}, p_{2,041}, \cdots, p_{2,047}\}$

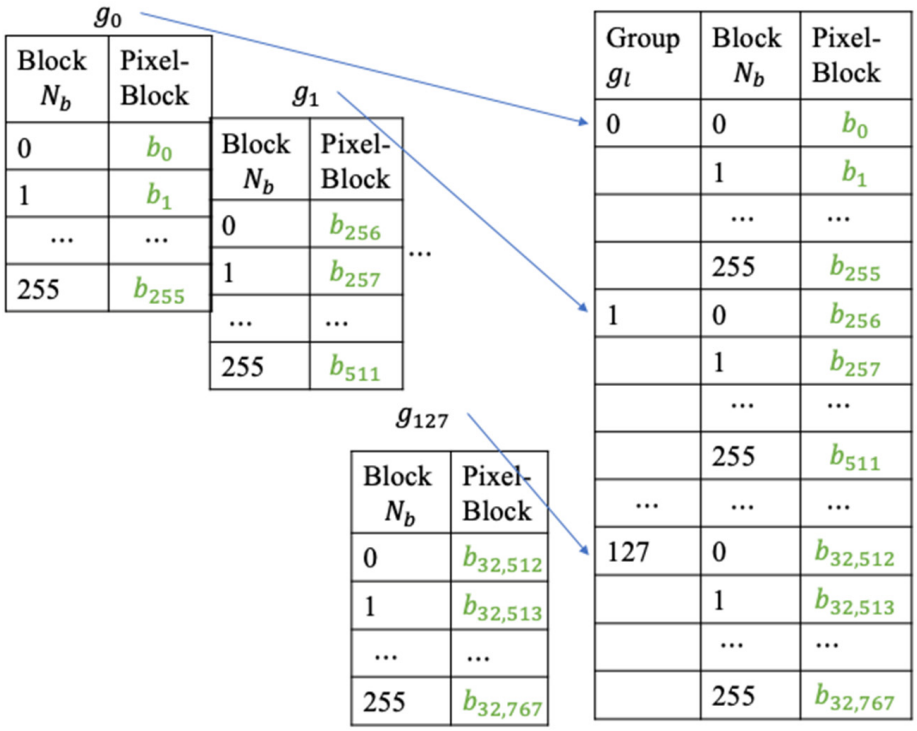
Figure 7. Constructing a codeword table for a block-group.

3.3. Pixel-Block Number Embedding

Before hiding data, recording the indices of pixel-blocks in a block-group is needed for the image recovery. The original index of a pixel-block N_b in a block-group is calculated based on the following:

$$N_b = b_i \ mod \ m, \tag{8}$$

where m is the number of pixel-blocks in a block-group and b_i is the current pixel-block. For example, the number of pixel-block b_2 in block-group g_0 is 2 and the number of pixel-block b_{257} in block-group g_1 is 1. Figure 8 shows a diagram on how pixel-blocks are numbered in each block-group.



Block N_b	Pixel-Block
0	b_0
1	b_1
...	...
255	b_{255}

Block N_b	Pixel-Block
0	b_{256}
1	b_{257}
...	...
255	b_{511}

Block N_b	Pixel-Block
0	$b_{32,512}$
1	$b_{32,513}$
...	...
255	$b_{32,767}$

Group g_l	Block N_b	Pixel-Block
0	0	b_0
	1	b_1
	...	...
	255	b_{255}
1	0	b_{256}
	1	b_{257}
	...	...
	255	b_{511}
...	...	...
127	0	$b_{32,512}$
	1	$b_{32,513}$
	...	...
	255	$b_{32,767}$

Figure 8. Block numbers in block-groups.

Since there are n pixels in a pixel-block, the least significant bit (LSB) [5] of each pixel in the block is used to indicate its pixel-block number in a block-group. Figure 9 demonstrates how to embed a calculated block number N_b to LSBs. The LSB bit in each pixel inside of a pixel-block is replaced with the binary bit value of its calculated block number in its group. The example shows the decimal value of 1 which is the calculated block number for b_1 in g_0.

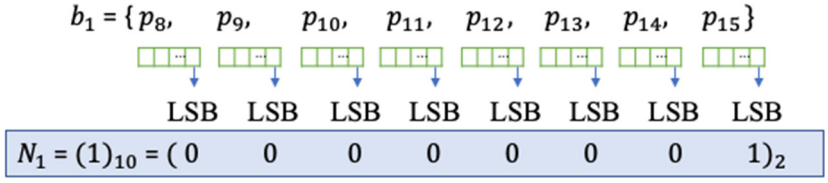


Figure 9. LSB replacements in pixels according to the calculated block number in a block- group.

After embedding the calculated block numbers as metadata into the LSBs, the encrypted image I_e is now an encrypted image embedded with image recovery information. It is then sent to the data hider to hide secret data.

3.4. Data Hiding

When the data hider receives the encrypted image I_e with embedded block numbers, a data hiding key k_d, and a secret message $SM = \{sm_0, sm_1, \cdots, sm_k\}$, $sm_k = \{0, 1\}$, an n-dimensional line L is generated according to the data hiding key k_d. The pixels in each pixel-block form a codeword and obtain a projected value by projecting the codeword to the line L. A codeword table T is created for each block-group by using these codewords and is sorted by their projected values. The sorted codeword table $T\prime$ is then used to embed the secret bits by applying the codeword index reordering technique described in Section 2.2.

Algorithm 1 details the steps of how to embed the secret message SM:

Algorithm 1: Data Hiding.

Input	The encrypted image I_e, the pixel-block size n, the block-group size m, the data hiding key k_d, and the secret message SM.
Output	A marked image I_m.
1:	Obtain an n-dimensional line L for pixel-blocks in a block-group to project to by using the data hiding key k_d.
2:	Initialize a marked image $I_m = \{\}$.
3:	Form pixel-blocks and block-groups based on sizes n and m.
4:	FOR each block-group g_c in the encrypted image I_e
5:	Generate a codeword table $T = \{cw_0, cw_1, \cdots, cw_{m-1}\}$ using g_c
6:	Initialize projected values PV = {}
	FOR each pixel-block b_c in g_c
7:	Obtain projected value $\lambda_c = b_c \cdot L$
	Append λ_c to PV.
	END
8:	Sort codeword table T according to PV to obtain the sorted codeword table $T\prime = \{cw'_0, cw'_1, \cdots, cw'_{m-1}\}$.
9:	Initialize a stego codeword table ST = {}.
	WHILE SM is not empty and $T\prime$ is not empty
	Calculate the number of secret bits that can be embedded by using $nsm = \lfloor \log_2(length(T\prime)) \rfloor$,
10:	Convert secret bits to a decimal index $(sm_0 \parallel sm_1 \parallel \cdots \parallel sm_{nsm-1})_2 = (indx)_{10}$.
	Move codeword cw'_{indx} out from $T\prime$ and add to ST.
	END
	FOR each codeword $stcw_i$ in ST
11:	Append $stcw_i$ to the marked image I_m.
	END
	END
12:	Export I_m.

3.5. Data Extraction and Image Recovery

After receiving the marked image I_m, the secret message can be extracted when a receiver has the data hiding key k_d. When a receiver has the encryption key k_e, a recovered

image with high visual quality can be obtained. If a receiver has both keys, the secret message can be extracted and the recovered image can be obtained.

3.5.1. Data Extraction

If a receiver has a data hiding key k_d, the secret data can be extracted. The marked image I_m is converted into a pixel stream first. Depending on the size of a pixel-block n and the number of blocks m in a block-group received, it is divided into pixel-blocks and block-groups first. A n-dimensional line L is generated by using the data hiding key k_d. The pixels in each pixel-block form a codeword and obtain a projected value by projecting the codeword to the line L. A codeword table T is created for each block-group of the marked image by using these codewords and is sorted by their projected values. Both the sorted codeword table $T\prime$ and the codeword table T created from the marked image are used to extract the secret data. The detailed steps are provided in Algorithm 2.

Algorithm 2: Data Extraction.

Input The marked image I_m, the size of pixel-block n, the size of block-group m, and the data hiding key k_d.

Output A recovered secret message RSM.

1: Obtain the n-dimensional line L for pixel-blocks in a block-group to project to by using data hiding key k_d.

2: Initialize the recovered secret message $RSM = \{\}$.

3: Form pixel-blocks and block-groups based on sizes n and m.

4: FOR each block-group g_c in the marked image I_m

5: Generate a codeword table $T = \{cw_0, cw_1, \cdots, cw_{m-1}\}$ using g_c.

6: Initialize projected values PV = {}.

7: FOR each pixel-block b_c in g_c
Obtain projected value $\lambda_c = b_c \cdot L$
Append λ_c to PV.
END

8: Sort codeword table T according to PV to obtain the sorted codeword table $T\prime = \{cw'_0, cw'_1, \cdots, cw'_{m-1}\}$.

9: FOR each codeword cw_c in the codeword table T
Find the index $indx$ of cw_c in $T\prime$ and remove cw_c from $T\prime$.
Convert decimal index $indx$ to binary secret bits
$(indx)_{10} = (sm_0 \parallel sm_1 \parallel \cdots \parallel sm_{nsm-1})_2$.
Append secret bits to RSM.
END
END

10: Export RSM.

3.5.2. Image Recovery

If a receiver has an encryption key k_e, a recovered image with high visual quality can be obtained. The marked image I_m is converted into a pixel stream first. Depending on the size of a pixel-block n and the number of blocks m in a block-group received, the marked pixel stream is broken into pixel-blocks and block-groups. The pixels in each pixel-block form a codeword, and a codeword table T is created for each block-group of the marked image by using these codewords. We can extract the original pixel-blocks' indices by extracting the values from the LSBs of pixels in pixel-blocks and recover the pixel values by using the encryption key k_e. Algorithm 3 goes through the recovery steps in more detail.

Algorithm 3: Image Recovery.

Input	The marked image I_m, the size of pixel-block n, the size of block-group m, and the encryption key k_e.
Output	A recovered image RI.
1:	Initialize the recovered image $I_r = \{\}$.
2:	Form pixel-blocks and block-groups based on sizes n and m.
3:	FOR each block-group g_c in the marked image I_m
5:	Generate a codeword table $T = \{cw_0, cw_1, \cdots, cw_{m-1}\}$ using g_c.
6:	Initialize block values BV = {}.
	FOR each pixel-block b_c in g_c
	Obtain block value bv_c from the LSB of each pixel in b_c.
7:	Convert bv_c to decimal v.
	Append v to BV.
	END
8:	Sort codeword table T according to PV to obtain the sorted codeword table $T\prime = \{cw'_0, cw'_1, \cdots, cw'_{m-1}\}$.
	FOR each codeword cw'_c in the codeword table $T\prime$
9:	Append cw'_c to I_r.
	END
	END
	Recover image RI by using encryption key k_e to decrypt image I_r with bit XOR operations.
10:	Export RI.

4. Experimental Results

In this section, we conducted experiments on an Intel(R) Core(TM) i5-9500 CPU utilizing the MATLAB environment, version 2017a, within the Windows PC operating system, in order to assess the performance of the proposed solution and compare it with some state-of-the-art (SOTA) schemes.

We evaluated the performance of the proposed data hiding scheme through experimental analysis. A binary data stream S, generated by a random number generator, was employed as the secret information. The two key factors that we pay attention during data hiding are good visual quality and high embedding quantity. To keep the visual quality means that the recovered images look as similar to the original cover images as possible. To efficiently measure the embedding quantity, embedding capacity (EC) is calculated, representing the total number of bits that the proposed scheme could embed in the image. With the evolution of data hiding techniques, numerous metrics have been utilized to assess the visual quality of the restored images. The commonly employed metrics include Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). The calculation formulas are provided below:

$$\mathrm{MSE} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} \left(O_{i,j} - R_{i,j} \right)^2, \tag{9}$$

$$\mathrm{PSNR} = 10 log_{10} \frac{(255)^2}{\mathrm{MSE}} (\mathrm{dB}), \tag{10}$$

$$\mathrm{SSIM} = \frac{(2\mu_O\mu_D + c_1)(2\sigma_{OD} + c_2)}{\left[(\mu_O)^2 + (\mu_D)^2 + c_1 \right] \left[(\sigma_O)^2 + (\sigma_D)^2 + c_2 \right]}, \tag{11}$$

where W and H represent the width and height of the images, and $O_{i,j}$ and $R_{i,j}$ denote the pixel values at position (i, j) for the cover image and the restored image. μ represents the mean value used as an estimate for luminance; σ is the standard deviation used as an estimate of contrast; σ_{OD} denotes the covariance between the original image O and

the restored image R and serves as a metric for structural similarity; c_1, and c_2 are two constants close to zero.

A higher PSNR value indicates less distortion caused by the hidden data. Typically, PSNR values exceeding 30 dB suggest image distortion imperceptible to the human eye. SSIM combines three factors—luminance, contrast, and structure—to assess the similarity between two images. The SSIM range is from -1 to 1. As the value of SSIM approaches 1, it demonstrates a higher degree of similarity between the two images.

At the same time, information entropy is selected to test the security of the encrypted image, that is, the randomness of the image histogram is calculated by the following method:

$$\text{entropy} = \sum_{pi=0}^{255} P(pi)\log\frac{1}{P(pi)}, \tag{12}$$

where pi represents the image pixel value between 0 and 255, and $P(pi)$ is the probability of the image pixel value pi occurring. An information entropy value closer to 8 means that the encrypted image has higher randomness.

In Section 4.1, we evaluate the performance of our proposed scheme under different test images. We give some execution results and security analysis in Section 4.2. Section 4.3 gives comparisons with other SOTA schemes.

4.1. Performances of Our Proposed Scheme

The embedding capacity of the proposed scheme is contingent upon the pixel-block size and the codeword table size. Therefore, we initially conducted performance tests on different test images by varying the pixel-block sizes in a codeword table. The group size in Tables 1 and 2 is what we call the codebook table size.

Table 1. Performance of different block sizes with the same group size of 256.

Image Name	Block Size	Group Size	EC	PSNR	SSIM
Airplane	1×8	256	197,888	51.1497	0.9956
	2×4	256	197,888	51.1327	0.9956
	4×2	256	197,888	51.1491	0.9956
	2×8	256	98,944	54.1349	0.9978
	3×8	256	64,932	55.9329	0.9985
Baboon	1×8	256	197,888	51.1499	0.9987
	2×4	256	197,888	51.1469	0.9987
	4×2	256	197,888	51.1329	0.9987
	2×8	256	98,944	54.1787	0.9994
	3×8	256	64,932	55.9347	0.9996
Barbara	1×8	256	197,888	51.1360	0.9972
	2×4	256	197,888	51.1542	0.9972
	4×2	256	197,888	51.1450	0.9972
	2×8	256	98,944	54.1392	0.9986
	3×8	256	64,932	55.9208	0.9991
Boat	1×8	256	197,888	51.1419	0.9972
	2×4	256	197,888	51.1469	0.9972
	4×2	256	197,888	51.1410	0.9972
	2×8	256	98,944	54.1624	0.9986
	3×8	256	64,932	55.9081	0.9991
Couple	1×8	256	197,888	51.1399	0.9975
	2×4	256	197,888	51.1316	0.9975
	4×2	256	197,888	51.1489	0.9975
	2×8	256	98,944	54.1454	0.9987
	3×8	256	64,932	55.9446	0.9992

Table 1. *Cont.*

Image Name	Block Size	Group Size	EC	PSNR	SSIM
Lena	1×8	256	197,888	51.1344	0.9960
	2×4	256	197,888	51.1447	0.9960
	4×2	256	197,888	51.1509	0.9960
	2×8	256	98,944	54.1487	0.9980
	3×8	256	64,932	55.9430	0.9987
Peppers	1×8	256	197,888	51.1362	0.9963
	2×4	256	197,888	51.1452	0.9963
	4×2	256	197,888	51.1435	0.9963
	2×8	256	98,944	54.1419	0.9981
	3×8	256	64,932	55.8990	0.9988

Table 2. Performance with different block-group sizes and different pixel-block.

Image Name	Block Size	Group Size	EC	PSNR	NPCR	UACI	MAE
Airplane	1×8	256	197,888	51.15	49.90	0.1957	0.4990
	1×6	64	180,048	51.15	49.89	0.1957	0.4989
	1×4	8	155,648	51.14	49.98	0.1960	0.4998
	1×2	4	131,072	51.14	50.06	0.1963	0.5006
Baboon	1×8	256	197,888	51.15	49.90	0.1957	0.4990
	1×6	64	180,048	51.12	50.22	0.1969	0.5022
	1×4	8	155,648	51.14	50.05	0.1963	0.5005
	1×2	4	131,072	51.14	50.01	0.1961	0.5001
Barbara	1×8	256	197,888	51.14	50.06	0.1963	0.5006
	1×6	64	180,048	51.15	49.95	0.1959	0.4995
	1×4	8	155,648	51.14	49.98	0.1960	0.4998
	1×2	4	131,072	51.14	50.02	0.1961	0.5002
Boat	1×8	256	197,888	51.14	49.99	0.1960	0.4999
	1×6	64	180,048	51.15	49.88	0.1956	0.4988
	1×4	8	155,648	51.14	50.00	0.1961	0.5000
	1×2	4	131,072	51.13	50.07	0.1964	0.5007
Couple	1×8	256	197,888	51.14	50.01	0.1961	0.5001
	1×6	64	180,048	51.14	49.97	0.1960	0.4997
	1×4	8	155,648	51.14	49.99	0.1960	0.4999
	1×2	4	131,072	51.13	50.08	0.1964	0.5008
Lena	1×8	256	197,888	51.13	50.08	0.1964	0.5008
	1×6	64	180,048	51.14	49.98	0.1960	0.4998
	1×4	8	155,648	51.14	49.97	0.1960	0.4997
	1×2	4	131,072	51.14	49.97	0.1960	0.4997
Peppers	1×8	256	197,888	51.14	50.06	0.1963	0.5006
	1×6	64	180,048	51.14	50.04	0.1962	0.5004
	1×4	8	155,648	51.15	49.93	0.1958	0.4993
	1×2	4	131,072	51.14	50.05	0.1963	0.5005

Table 1 lists the results of applying the proposed scheme to the test images for different pixel-block size configurations with the same codebook table size. We observed that when pixel-blocks have an identical number of pixels, the shapes of the pixel-blocks had no effect at all on EC values depicted in Table 1; the shapes also had no significant impact on PSNR and SSIM. Therefore, we could simplify our scheme by converting the encrypted images to a pixel stream first and specifying the size of a pixel-block without the necessity of specifying the width and the height of a pixel-block. However, the number of pixels within a pixel-block had a significant effect on EC, and the larger the size of the pixel-block, the less EC the proposed scheme could provide. Moreover, the larger the pixel-block, the higher the PSNR and SSIM values of the recovered image when the size of the codebook

table was the same. Simultaneously, we observed that the most efficient partitioning is to have 2^n pixel-blocks within a block-group when the size of a pixel-block is n.

In addition to these metrics, we also employed the number of pixels changing rate (NPCR), unified average changed intensity (UACI), and mean absolute error (MAE) to evaluate image distortion between the original and the restored images. The calculation formulas for these metrics are as follows:

$$\text{NPCR} = \frac{\sum_{i,j} D(i,j)}{W \times H} \times 100\%, \tag{13}$$

$$D(i,j) = \begin{cases} 1, & O(i,j) \neq R(i,j) \\ 0, & otherwise \end{cases}, \tag{14}$$

$$\text{UACI} = \frac{\sum |O(i,j) - R(i,j)|}{W \times H \times 255} \times 100\%, \tag{15}$$

$$\text{MAE} = \frac{1}{W \times H} \sum_{i=1}^{W} \sum_{j=1}^{H} |O(i,j) - R(i,j)|, \tag{16}$$

where W and H represent the width and height of the images, and $O(i,j)$ and $R(i,j)$ denote the pixel values at position (i,j) for the cover image and the restored image.

NPCR is employed to compute the number of differing pixels between two images, while UACI is utilized to calculate the average change in pixels between two images. Unlike Table 1, Table 2 is a test of the proposed scheme with various codebook table sizes. From Table 2, it can be clearly seen that as the size of the codebook table becomes larger, the EC of the proposed scheme also becomes larger. Since the efficient pairing of codebook table size and pixel-block size was utilized, the LSB of each pixel in a pixel-block was used to record the original pixel-block index in a group, and the PSNR values of all tested scenarios were around 51 dB. The PSNR values indicated that our recovered image had a good visual quality. As shown in Table 2, the proposed scheme involves replacing the LSB of pixels with the initial block index in a group after encrypting the image. Consequently, after extracting information and decrypting the image, the LSB of each pixel cannot be fully restored, with a 50% probability of being flipped. As a result, NPCR values are around 50% across various test images. UACI values are notably low, with the highest UACI not exceeding 0.2%, indicating that the average pixel changes induced by the proposed scheme are small, even if the images cannot be fully recovered. Mean Absolute Error (MAE) is also an indicator for assessing image quality, where a smaller MAE value corresponds to better image quality. From Table 2, we can see that the MAE values are small for all images. In summary, the proposed scheme causes very little damage to the cover images when embedding the secret data and the recovered images are very similar to the cover images.

4.2. Execution Results and Security Analysis

Figure 10 presents the performance results of various test images at different stages. After image encryption, pixel values become disordered, rendering meaningful information indiscernible, and the pixel distribution in the histogram of the encrypted image is consequently uniform. Since our proposed data embedding scheme involves merely scrambling the positions of encrypted pixels after recording the original block indices using LSB replacements, the distribution of pixel values in the encrypted image does not undergo substantial changes. The histogram distributions of the recovered images that we are able to obtain after extracting the information are also very similar to the original histogram distributions.

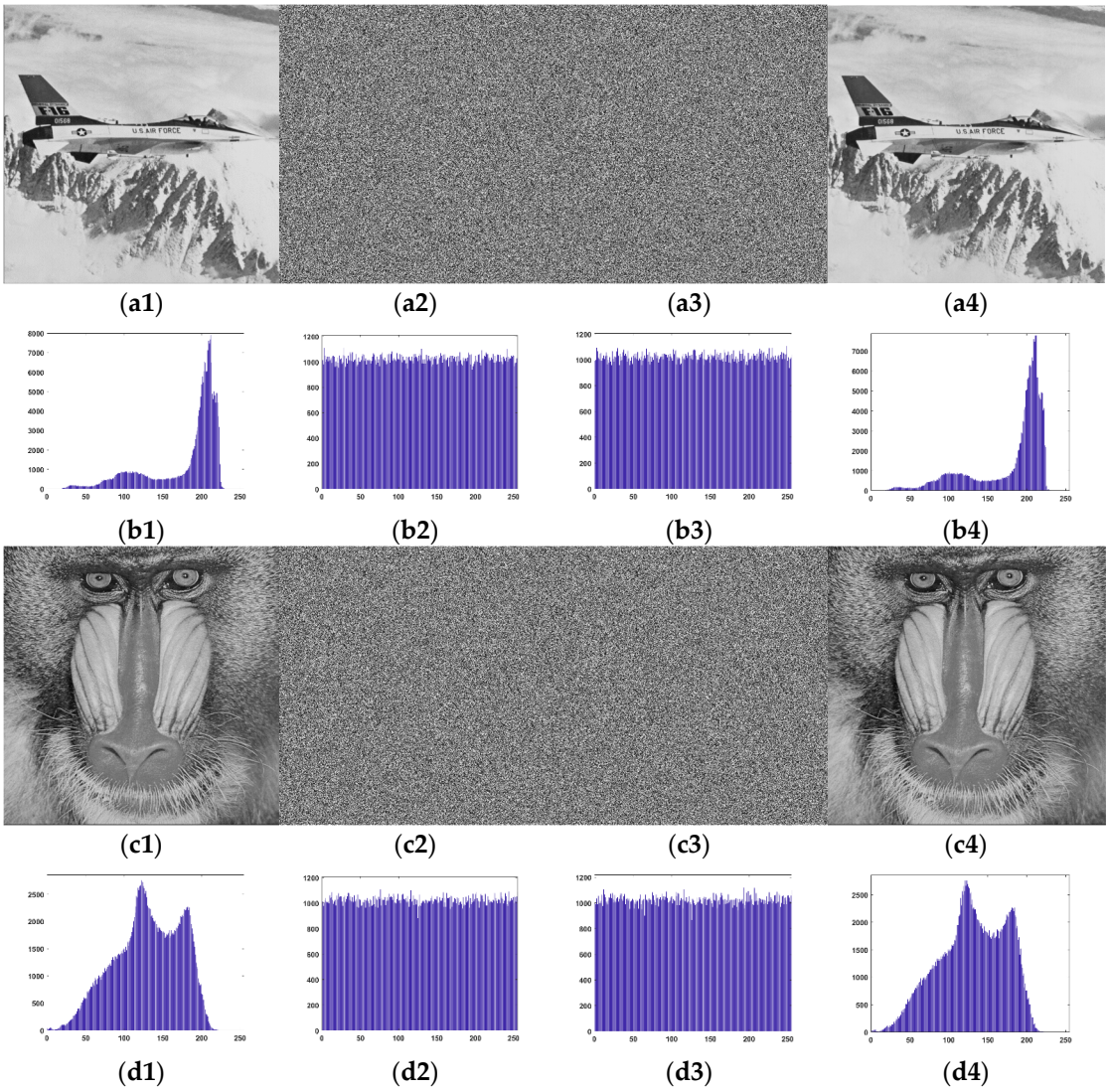


Figure 10. Execution results of our proposed scheme: (**a1**) is original "Airplane" image with 512 × 512 pixels. (**a2**,**a3**) are encrypted image and embedded encrypted image (block size = 1 × 8, codeword table size = 256). (**a4**) is the decrypted image. (**b1**–**b4**) are the corresponding histograms of (**a1**–**a4**). In (**c1**–**c4**,**d1**–**d4**), the results of "Baboon" (block size = 1 × 8, codeword table size = 256), are given.

In addition to information entropy, another way to analyze the differences between adjacent pixels is to use the correlation coefficient to assess the correlation between adjacent pixels. A natural image has a high correlation between neighboring pixels normally, but for a secure encrypted image, the lower the correlation between its neighboring pixels, the more secure it is. It is defined as follows:

$$\mathrm{Corr} = \frac{\sum_{i=1}^{N}\left(x_i - \frac{1}{N}\sum_{i=1}^{N} x_i\right)\left(y_i - \frac{1}{N}\sum_{i=1}^{N} y_i\right)}{\sqrt{\sum_{i=1}^{N}\left(x_i - \frac{1}{N}\sum_{i=1}^{N} x_i\right)^2 \times \sum_{i=1}^{N}\left(y_i - \frac{1}{N}\sum_{i=1}^{N} y_i\right)^2}}. \tag{17}$$

We divide the image into pixel pairs, take the former of all pixel pairs as x_i and the latter of all pixel pairs as y_i, with N denoting the number of pixel pairs, and then calculate the correlation coefficient between them. Tables 3–5 were tested with a pixel-block size of 8 and a codebook table size of 256. The information entropy and the pixel correlation results for horizontal and vertical pairs are shown in Tables 3 and 4, respectively.

Table 3. Entropy values of test images.

Image Name	Original Image Entropy	Encrypted Image Entropy	Marked Image Entropy
Airplane	6.705888	7.9993	7.9993
Baboon	7.357949	7.9993	7.9993
Barbara	7.632119	7.9993	7.9992
Boat	7.19137	7.9994	7.9993
Couple	7.058103	7.9992	7.9993
Lena	7.445507	7.9992	7.9992
Peppers	7.594429	7.9993	7.9993

Table 4. Horizontal and vertical correlation analysis.

Image Name	Original Image		Encrypted Image		Marked Image	
	Hor	Ver	Hor	Ver	Hor	Ver
Airplane	0.9606	0.9584	−0.0019	−0.0033	−0.0019	−0.0080
Baboon	0.8667	0.7498	0.0010	−0.0034	0.0010	−0.0131
Barbara	0.8956	0.9588	−0.0009	−0.0028	−0.0009	−0.0138
Boat	0.9383	0.9715	0.0000	−0.0052	0.0000	−0.0087
Couple	0.9433	0.9534	−0.0035	−0.0027	−0.0034	−0.0091
Lena	0.9719	0.9850	0.0002	−0.0012	0.0001	−0.0112
Peppers	0.9730	0.9762	−0.0012	−0.0028	−0.0012	−0.0103

Table 5. NPCR and UACI analysis of test images.

Image Name	Encrypted Image		Marked Image	
	NPCR	UACI	NPCR	UACI
Airplane	99.8096	32.3892	99.61624	32.43519
Baboon	99.8096	27.9032	99.60213	27.58626
Barbara	99.8096	29.8222	99.60213	29.86593
Boat	99.8096	28.5168	99.61014	28.54057
Couple	99.8096	28.2082	99.63608	27.8268
Lena	99.8096	28.6750	99.61281	28.81635
Peppers	99.8096	29.6210	99.60632	29.64995

Table 3 provides measurements for the original, encrypted, and embedded encrypted images of different test images. Clearly, the entropy values for the encrypted and embedded encrypted images of each image are quite similar, as our proposed scheme only employs one-bit LSB replacement on encrypted pixels and subsequently scrambles the pixel positions. On the other hand, the entropy values for different encrypted images are consistently close to 8, significantly higher than the entropy values of the original images. Consequently, the distribution of pixel values in encrypted images exhibits heightened randomness, thereby enhancing security.

As shown in Table 4, neighboring pixel values in the original image have strong positive correlation both horizontally and vertically, while the pixels of encrypted and marked images have only very low correlation both horizontally and vertically. Also, the correlation between the marked image and the encrypted image does not change much compared to the encrypted image both horizontally and vertically. This indicates that the Stream Cipher encryption completely encrypts the image, and the proposed scheme that utilizes the reordering of the codebook table to embed the secret data inherits this feature and does not make the correlation between the pixels higher due to the embedded data.

The security of the proposed scheme is further analyzed in Table 5 using two metrics, NPCR and UACI. The definition and computation of both NPCR and UACI have been mentioned in Section 4.1. In terms of security analysis, the higher the NPCR value, the more effective the encryption algorithm is, because the higher the number of different pixels,

the harder the encrypted image is to crack. For two uncorrelated images, the theoretical value of UACI is 33.33%. As shown in Table 5, for both encrypted and marked images, the NPCR value and UACI value of the proposed scheme are close to the theoretical optimal value, which indicates that our scheme can provide a high level of security against potential attacks.

Table 6 shows the time in seconds spent by different stages of the proposed scheme on different test images. The pixel-block size of the proposed scheme is 8, the codebook size is 256, and 197,888 bits of secret data are embedded. From the table, we can see that the proposed scheme spends 6.30 s in the embedding stage at most, and 2.85 s in the extraction and recovering stage at most. Overall, the cost of time spent is acceptable.

Table 6. Experimental runtime on different test images.

Image Name	Embedding	Recovered Image	Extraction	Extraction and Recovery
Airplane	5.98	1.95	0.83	2.80
Baboon	6.30	2.00	0.78	2.85
Barbara	6.19	1.98	0.80	2.78
Boat	6.13	1.95	0.82	2.83
Couple	6.17	1.97	0.79	2.73
Lena	6.29	1.96	0.81	2.77
Peppers	6.18	1.98	0.77	2.77

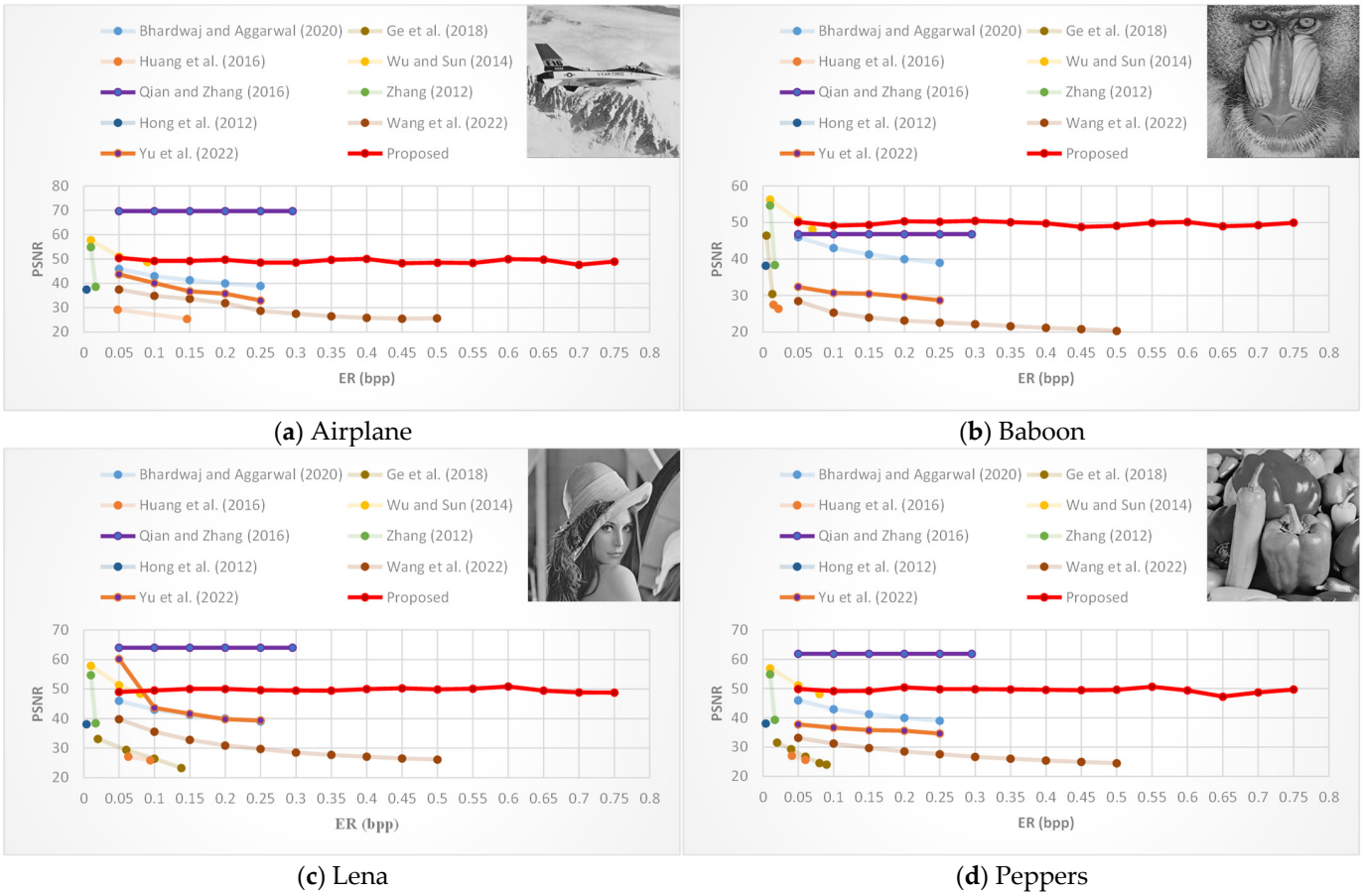
4.3. Comparison with State-of-the-Art Schemes

In this section, we primarily compare our proposed scheme with some state-of-the-art schemes for data hiding in encrypted images in terms of PSNR, SSIM, and embedding capacity. In Table 7, it can be seen that the embedding capabilities of our proposed schemes outperform all the other schemes. Moreover, our proposed scheme consistently provides stable embedding capacity for any image, as long as the pixel-block size and the number of blocks within a group remain unchanged. The independence of embedding capacity from image content is attributed to our utilization of indices within each group for data embedding. So, as long as the number of blocks in the group and the size of the pixel-blocks are consistent, images of the same size will have the same embedding capacity. From Table 7, it is evident that our proposed scheme also outperforms most SOTA methods based on the PSNR values and SSIM values. Even though the PSNR is slightly lower than [17], our proposed scheme achieves a PSNR of approximately 51 dB, which is adequate for visual quality of a recovered image.

Table 7. EC, PSNR, and SSIM comparisons.

Image		Qian and Zhang [17]	Bhardwaj and Aggarwal [21]	Wang et al. [22]	Group Size = 16	Group Size = 64	Group Size = 256
Lena	EC	77,385	65,536	131,072	155,648	180,048	197,888
	PSNR	64	39	26.1	51.14	51.14	51.13
	SSIM	0.9781	0.9417	0.8965	0.9960	0.9960	0.9960
Peppers	EC	77,385	65,536	131,072	155,648	180,048	197,888
	PSNR	61.9	39	24.5	51.15	51.14	51.14
	SSIM	0.9837	0.9446	24.5	0.9963	0.9963	0.9963
Baboon	EC	77,385	65,536	131,072	155,648	180,048	197,888
	PSNR	46.8	39	20.31	51.15	51.12	51.15
	SSIM	0.9918	0.9805	0.7739	0.9987	0.9987	0.9987
Airplane	EC	77,385	65,536	130,915	155,648	180,048	197,888
	PSNR	69.7	39	25.73	51.14	51.15	51.15
	SSIM	0.9824	0.9403	0.9073	0.9956	0.9956	0.9956

In Figure 11, we further compare the PSNR of our scheme with more SOTA schemes under different embedding capacities, with a pixel-block size of 8 and 256 blocks within a codeword table. In Figure 11, we use the embedding ratio (ER) to represent the embedding capacity (ER = EC/total number of pixels). As depicted in Figure 11, in most cases, the PSNR values of other schemes tends to decrease with increasing embedding capacity. In contrast, our proposed scheme consistently has a stable PSNR for the recovered image. Therefore, as the amount of embedded secret data increases, the advantages of our proposed scheme become more pronounced in terms of the PSNR and the SSIM of the recovered image. As long as the embedding rate is greater than 0.25 bpp, the proposed scheme has better visual quality and higher embedding capability than other schemes.



(**a**) Airplane (**b**) Baboon

(**c**) Lena (**d**) Peppers

Figure 11. PSNR comparisons with SOTA schemes [15–23].

5. Conclusions

In the proposed scheme, a content owner encrypts an image first using the Stream Cipher technique with an encryption key and embedding pixel-block numbers in each block-group by replacing LSBs of pixels in each pixel-block. The encrypted image with embedding pixel-block numbers is then sent to a data hider to embed a secret message and create a marked image using a data hiding key and the codeword table index reordering technique. The marked image is sent to the receivers. When a receiver has the data hiding key, the secret message can be extracted. When a receiver has the encrypted key, the image can be reconstructed with minor distortions because of LSB replacements.

With our experiment results, we are confident that the proposed scheme could achieve our goal of increasing embedding capacity by a significant amount compared to the SOTA schemes. Our experiments showed that the embedding capability with any cover image is consistent as long as the sizes of pixel-blocks and block-groups were the same. As for PSNRs, the results showed that the proposed scheme could outperform the SOTA schemes

as well. Because of LSB replacements, our scheme has a limitation in that the images are not fully recovered, yet the PSNR reaches 50 dB, which indicates that the images have good visual performance.

Author Contributions: Conceptualization and methodology, J.-C.L., H.-X.C., C.-C.C. (Ching-Chun Chang) and C.-C.C. (Chin-Chen Chang); software, H.-X.C.; validation, J.-C.L., H.-X.C., C.-C.C. (Ching-Chun Chang) and C.-C.C. (Chin-Chen Chang); data curation, H.-X.C.; writing—original draft preparation, J.-C.L. and H.-X.C.; writing—review and editing, J.-C.L. and H.-X.C.; supervision, C.-C.C. (Chin-Chen Chang). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

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

References

1. Fiore, U. Selective Redundancy Removal: A Framework for Data Hiding. *Future Internet* **2010**, *2*, 30–40. [CrossRef]
2. Fiore, U.; Rossi, F. Embedding an Identity-Based Short Signature as a Digital Watermark. *Future Internet* **2015**, *7*, 393–404. [CrossRef]
3. Pilania, U.; Tanwar, R.; Zamani, M.; Manaf, A.A. Framework for Video Steganography Using Integer Wavelet Transform and JPEG Compression. *Future Internet* **2022**, *14*, 254. [CrossRef]
4. Liu, J.C.; Chang, C.C.; Lin, C.C.; Chang, C.C. Hiding Information in a Well-Trained Vector Quantization Codebook. In Proceedings of the ACM International Conference on Signal Processing and Machine Learning (SPML), Tianjin, China, 14–16 July 2023. [CrossRef]
5. Celik, M.U.; Sharma, G.; Tekalp, A.M.; Sable, E. Lossless generalized-LSB data embedding. *IEEE Trans. Image Process* **2005**, *14*, 253–266. [CrossRef] [PubMed]
6. Shi, Y.Q.; Li, X.; Zhang, X.; Wu, H.T.; Ma, B. Reversible data hiding: Advances in the past two decades. *IEEE Access* **2016**, *4*, 3210–3237. [CrossRef]
7. Zhang, W.; Wang, H.; Hou, D.; Yu, N. Reversible data hiding in encrypted images by reversible image transformation. *IEEE Trans. Multimed.* **2016**, *18*, 1469–1479. [CrossRef]
8. Ma, K.; Zhang, W.; Zhao, X.; Yu, N.; Li, F. Reversible data hiding in encrypted images by reserving room before encryption. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 553–562. [CrossRef]
9. Yi, S.; Zhou, Y. Separable and reversible data hiding in encrypted images using parametric binary tree labeling. *IEEE Trans. Multimed.* **2019**, *21*, 51–64. [CrossRef]
10. Yi, S.; Zhou, Y. Parametric reversible data hiding in encrypted images using adaptive bit-level data embedding and checkerboard-based prediction. *Signal Process.* **2018**, *150*, 171–182. [CrossRef]
11. Pauline, P.; William, P. An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1670–1681.
12. Chen, F.; Yuan, Y.; He, H.; Tian, M.; Tai, H.M. Multi-MSB compression based reversible data hiding scheme in encrypted images. *IEEE Trans. Circuits Syst. Video Technol.* **2021**, *31*, 905–916. [CrossRef]
13. Puteaux, P.; Puech, W. A recursive reversible data hiding in encrypted images method with a very high payload. *IEEE Trans. Multimed.* **2020**, *23*, 636–650. [CrossRef]
14. Wu, Y.; Xiang, Y.; Guo, Y.; Tang, J.; Yin, Z. An improved reversible data hiding in encrypted images using parametric binary tree labeling. *IEEE Trans. Multimed.* **2020**, *22*, 1929–1938. [CrossRef]
15. Hong, W.; Chen, T.S.; Wu, H.Y. An improved reversible data hiding in encrypted images using side match. *IEEE Signal Process. Lett.* **2012**, *19*, 199–202. [CrossRef]
16. Zhang, X. Separable reversible data hiding in encrypted image. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 826–832. [CrossRef]
17. Qian, Z.; Zhang, X. Reversible data hiding in encrypted images with distributed source encoding. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 636–646. [CrossRef]
18. Wu, X.; Sun, W. High-capacity reversible data hiding in encrypted images by prediction error. *Signal Process.* **2014**, *104*, 387–400. [CrossRef]
19. Huang, F.; Huang, J.; Shi, Y.Q. New framework for reversible data hiding in encrypted domain. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 2777–2789. [CrossRef]
20. Ge, H.; Chen, Y.; Qian, Z.; Wang, J. A high capacity multi-level approach for reversible data hiding in encrypted images. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *29*, 2285–2295. [CrossRef]
21. Bhardwaj, R.; Aggarwal, A. An improved block based joint reversible data hiding in encrypted images by symmetric cryptosystem. *Pattern Recognit. Lett.* **2020**, *139*, 60–68. [CrossRef]

22. Wang, X.; Chang, C.C.; Lin, C.C.; Chang, C.C. Reversal of pixel rotation: A reversible data hiding system towards cybersecurity in encrypted images. *J. Vis. Commun. Image Represent.* **2022**, *82*, 103421. [CrossRef]
23. Yu, M.; Yao, H.; Qin, C. Reversible data hiding in encrypted images without additional information transmission. *Signal Process. Image Commun.* **2022**, *105*, 116696. [CrossRef]