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

Jui-Chuan Liu , Hengxiao Chi , Ching-Chun Chang , Chin-Chen Chang *

Posted Date: 30 December 2023

doi: 10.20944/preprints202312.2341.v1

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

Article

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

Jui-Chuan Liu [1], Hengxiao 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; hx9704@gmail.com;
[2] Information and Communication Security Research CenterFeng Chia University, Taichung 40724, Taiwan; ccc@fcu.edu.tw
* Correspondence: alan3c@gmail.com

Abstract: Information is uploaded and downloaded through the Internet day in and day out ever since we immersed ourselves in the Internet. Data security becomes an area demanding high attention and one of the efficient techniques to protect data is data hiding. Among various data hiding schemes, more applications prefer the schemes with reversibility. In recent reversible data hiding studies, 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 gets larger. 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 codeword index reordering technique to hide secret data in encrypted images. By observing our experimental results, the embedding capacity is significantly larger. Secret data can be extracted when a receiver owns a data hiding key and image can be recovered when a receiver owns an encryption key.

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

1. Introduction

In modern days, iCloud data transmission replaces the physical mails to speed up the time spent on exchanging information. Needless to mention, the Internet is a key element in the transmission process. Digital activities involve both virtual world and physical world can be foreseen in the near future. If a hospital technician uploads an x-ray image, a CT-scan photo, or its patient's information to a data center for a list of specific doctors to download them to discuss various treatments over an online meeting. To protect the privacy of the patient, no one should see the information other than related doctors. Encrypting and decrypting of these digital information are getting more and more important to increase the 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].

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 it may not be desirable for the data hider to see the image content, the owner encrypts the the image before sending the image to hide the message. A receiver holding the encryption key can recover the image and a receiver having the data hiding key can extract the secret information.

There are various data hiding schemes designed to protect data effectively in different application circumstances and demonstrated exceptional results. Data hiding is branched out to reversible data hiding (RDH) [2] and non-reversible data hiding depending on if the image content can be recovered or not. A wider range of applications adopted RDH schemes because of their reversibility and thus RDH studies became more popular in the recent decades. According to the hiding carriers, the hiding schemes can be catalogued 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 by wavelet transform methods before embedding data. Images are first compressed by compression technique prior to embedding information in the compression domain. In the encryption domain, images are encrypted using

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

encryption keys before sending to data hider to hide secret in the encrypted image [3–26]. Reversible data hiding in encrypted images (RDH-EI) combines cryptography and RDH technology to achieve higher levels of protection. In order to better secure information privacy, our scheme focuses on using encrypted images to hide data. Our research goal is to achieve larger embedding capacity without degrading visual quality of recovered images. The core contributions of the scheme are described as below:

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

The rest of the paper is structured as the follows: Section 2 discusses the related works which including the LSB encryption, the sorting applied 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. Related Work

Our novel scheme employs the principle idea of a vector quantization (VQ) [12] codeword index reordering scheme [13] 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.

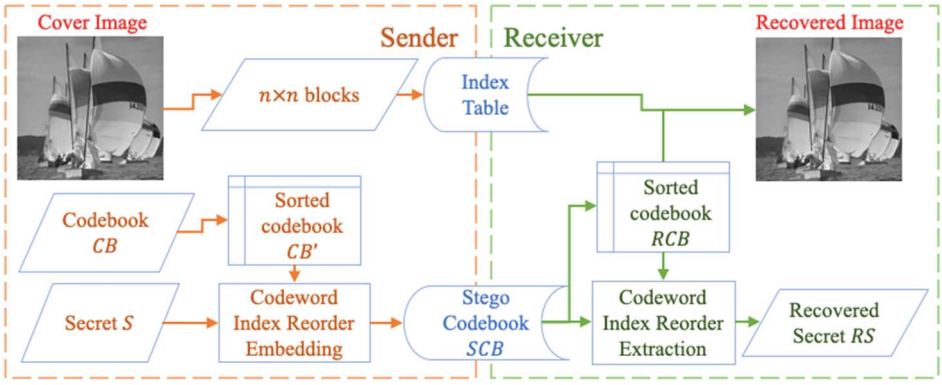


Figure 1. Flow of codeword index reordering scheme.

2.1. Sort Codewords by Projected Values

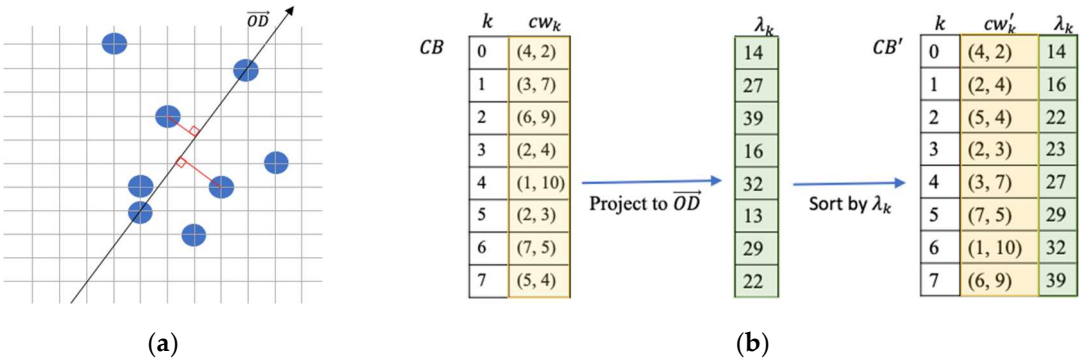
A sorted codebook is an essential component for a codeword index reordering scheme [13]. There is a VQ codebook $CB = \{cw_0, cw_1, \cdots, cw_{m-1}\}$, where m is the number of the codewords in the codebook. A data hider will find an n-dimensional point $D = \{r_1, r_2, \cdots, r_n\}$, $r_1, r_2, \cdots, r_n$ 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 get their projected values. These projected values are denoted as λ_1, λ_2, ..., λ_n using Eq. (1) and to be used to sort the codewords in the codebook.

$$\lambda_k = cw_k \cdot \overrightarrow{OD},\ 0 \le k \le m-1. \quad (1)$$

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 consists of eight 2-dimensional codewords is demonstrated in Figure 2 for better understanding of projected values and the sorting result according to their projected values.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1



(**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 $S = \{s_0, s_1, \cdots, s_k\}$, $s_k = \{0, 1\}$, embedding procedure is stepped as the following:

Step 1: Initialize a stego codebook SCB.

Step 2: Calculate the number of bits to embed

$$b = \lfloor log_2(length(SCB)) \rfloor. \quad (2)$$

Step 3: Convert secret bits to a decimal index

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

Step 4: Move the codeword scw_{indx} out from CB' and add to SCB.

Step 5: Repeat Step 2 thru Step 4 until there is no codeword in CB'.

Step 6: Send SCB to receivers.

Using the example in Figure 1, the embedding results is illustrated in Figure 3.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

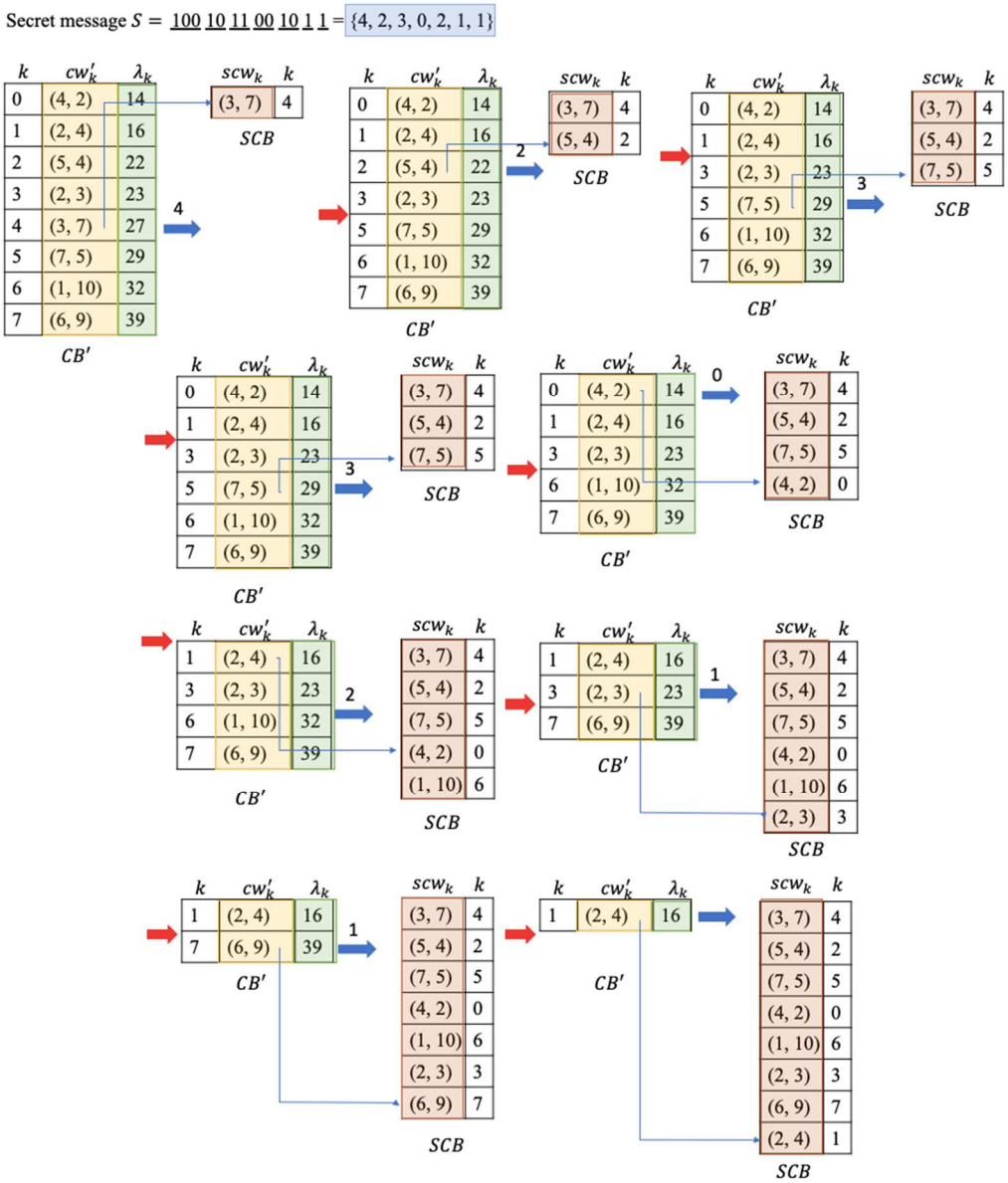


Figure 3. Data embedding of a codeword index reordering scheme.

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 get 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 get the index si in the RCB.

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

Step 6: Append RS' to RS.

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

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

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

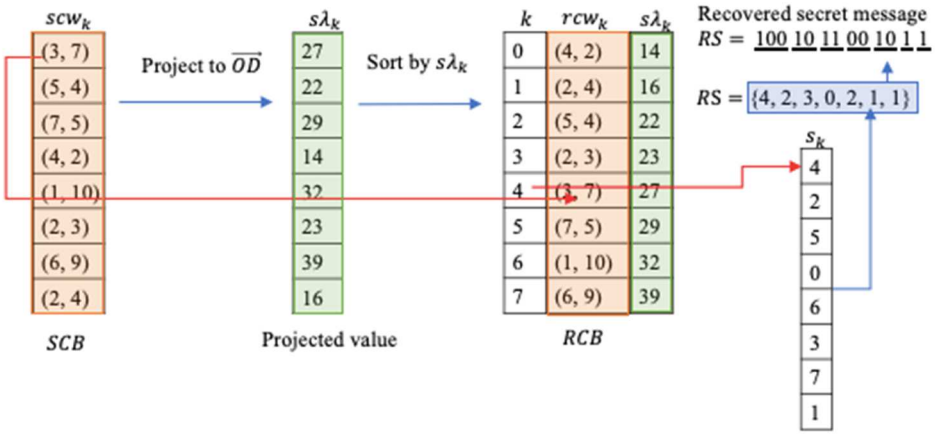


Figure 4. Data extraction of a codeword index reordering scheme.

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.

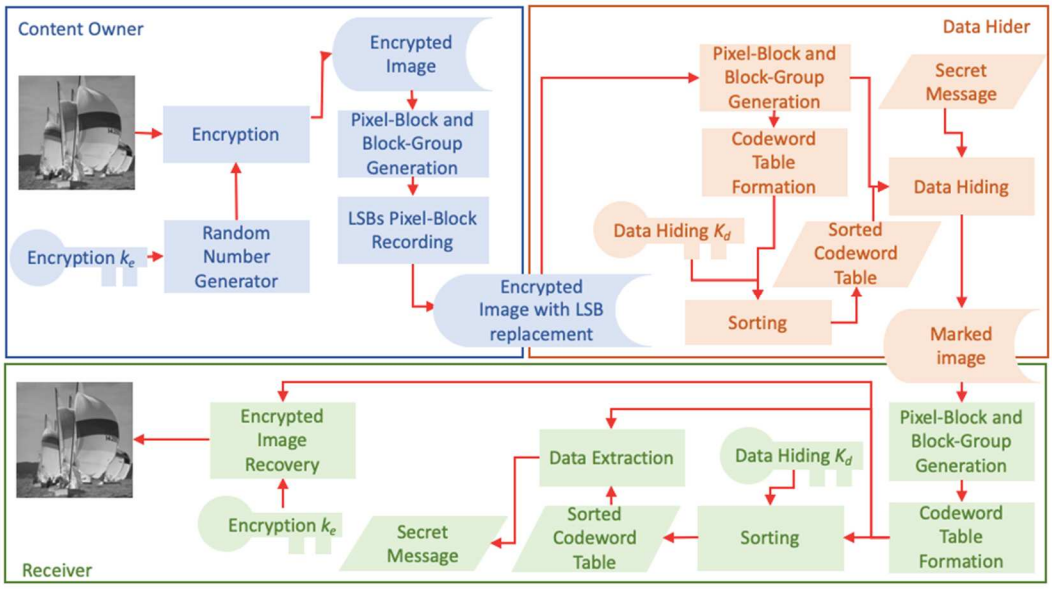


Figure 5. Framework of block group index reordering scheme.

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 [14], 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 are ranged between 0 and 255 is generated by using a random seed value. The stream cipher is a symmetrical 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 Eqs. (4) and (5),

$$I_{ij}^{x} = \left\lfloor \frac{I_{ij}}{2^{x-1}} \right\rfloor mod\ 2, \tag{4}$$

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

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

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 can be represented as x=1,2,⋯,8. After binary conversion, a bit-level XOR as Eq. (6) is applied,

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

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

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

3.2. Codeword Table Generation

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.

After obtaining the encrypted image I' which consists of $M \times N$ pixels, it is denoted as $I' = \{p_0, p_1, \cdots, p_{(M \times N)-1}\}$. The encrypted image is cut into pixel-blocks and each pixel-block contains n contiguous pixels. A current pixel-block can be denoted as $b_c = \{p_{n(c-1)}, p_{n(c-1)-1}, \cdots, p_{n(c-1)-(n-1)}\}$, where c is the number of the current pixel-block. After pixel-blocks are formed, m pixel-blocks are grouped together to be a block-group $g_l = \{b_{m(l-1)}, b_{m(l-1)+1}, \cdots, b_{m(l-1)+(m-1)}\}$, 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 on how pixel-blocks and block-groups are created for a 512 × 512 image. When a pixel-block consisting of 8 pixels and a block-group combining 255 blocks, there are 32768 pixel-blocks and 128 block-groups are formed in total for the encrypted image.

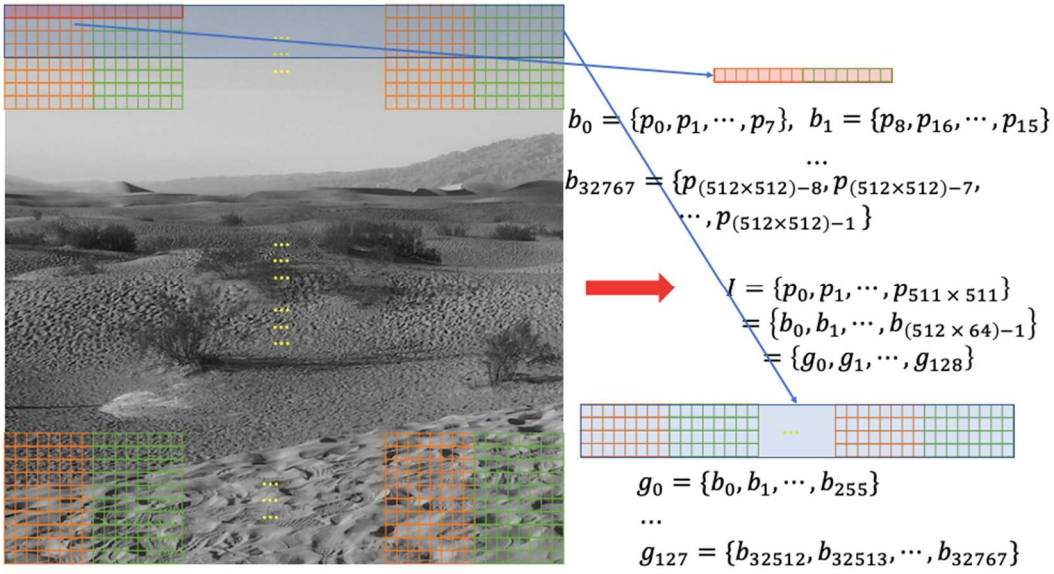


Figure 6. Cutting pixel stream into pixel-blocks and forming block-groups.

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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

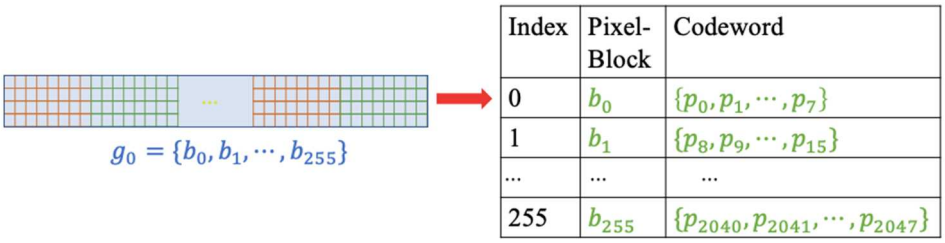


Figure 7. Constructing a codeword table for a block-group.

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

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

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.

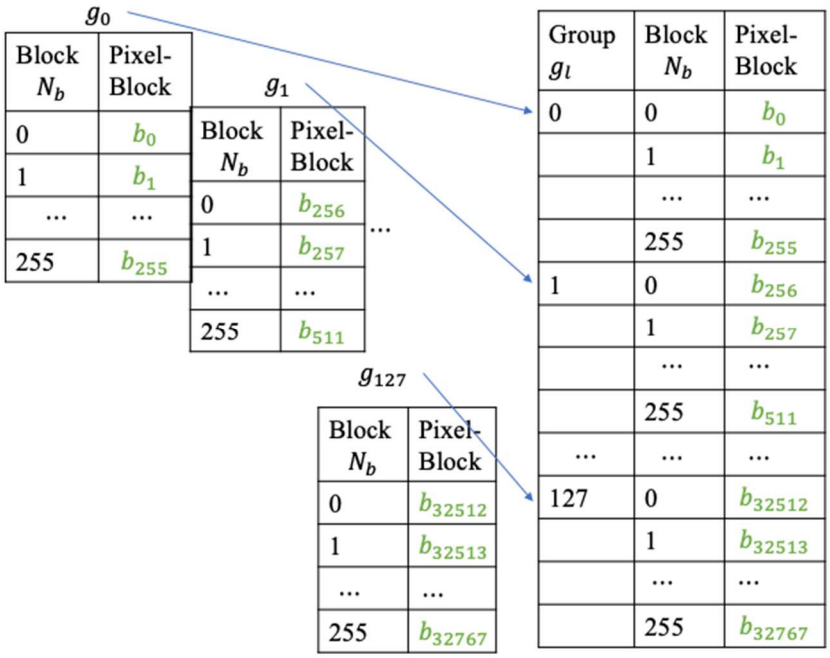


Figure 8. Block numbers in block groups.

Since there are n pixels in a pixel-block, the least significant bit (LSB) [15] 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

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_s 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 extraction key k_s. The pixels in each pixel-block form a codeword and get 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 sorted by their projected values. The sorted codeword table T' is then used to embed the secret bits by applying the codeword index reordering technique described in Subsection 2.2.

The Algorithm 1 detailed 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_s, and the secret message SM.
Output	A marked image I_m.
1:	Get an n-dimensional line L for pixel-blocks in a block-group to project to by using the data hiding key k_s.
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 = {}
7:	FOR each pixel-block b_c in g_c
	Get projected value $\lambda_c = b_c \cdot L$
	Append λ_c to PV
	END
8:	Sort codeword table T according to PV to get the sorted codeword table $T' = \{cw'_0, cw'_1, \cdots, cw'_{m-1}\}$
9:	Initialize a stego codeword table ST = {}
10:	WHILE SM is not empty and T' is not empty
	Calculate the number of secret bits can be embedded by using $nsm = \lfloor \log_2(length(T')) \rfloor$,
	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' and add to ST.
	END
11:	FOR each codeword $stcw_i$ in ST
	Append $stcw_i$ to the marked image I_m
	END
	END

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

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_s. 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 extraction key k_s, 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 extraction key k_s. The pixels in each pixel-block form a codeword and get 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' and the codeword table T created from the marked image are used to extract the secret data. The detailed steps are 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 projected line L.
Output	A recovered secret message RSM.
1:	Get the n-dimensional line L for pixel-blocks in a block-group to project to by using extraction key k_s.
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
	Get projected value $\lambda_c = b_c \cdot L$
	Append λ_c to PV
	END
8:	Sort codeword table T according to PV to get the sorted codeword table $T' = \{cw'_0, cw'_1, \cdots, cw'_{m-1}\}$
9:	FOR each codeword cw_c in the codeword table T
	Find the index $indx$ of cw_c in T' and remove cw_c from T'.
	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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

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 details.

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 = {}.
7:	FOR each pixel-block b_c in g_c
	Get block value bv_c from the LSB of each pixel in b_c.
	Convert bv_c to decimal v.
	Append v to BV.
	END
8:	Sort codeword table T according to PV to get the sorted codeword table $T' = \{cw'_0, cw'_1, \cdots, cw'_{m-1}\}$
9:	FOR each codeword cw'_c in the codeword table T'
	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

n this section, we conducted experiments utilizing the MATLAB environment, version 2017a, within the Windows PC operating system, 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:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

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

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

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

where W and H represent the width and height of the images, $O_{i,j}$ 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; and 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.; and c_1, and c_2 are two constants close to zero.

A higher PSNR value indicates less distortion caused by the hidden data. Typically, with PSNR values exceeding 30 dB suggests imperceptible image distortion 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 Table 1 and Table 2 is what we call the codebook table size.

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.

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	197888	51.1497	0.9956
	2 × 4	256	197888	51.1327	0.9956

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

	4×2	256	197888	51.1491	0.9956
	2×8	256	98944	54.1349	0.9978
	3×8	256	64932	55.9329	0.9985
Baboon	1×8	256	197888	51.1499	0.9987
	2×4	256	197888	51.1469	0.9987
	4×2	256	197888	51.1329	0.9987
	2×8	256	98944	54.1787	0.9994
	3×8	256	64932	55.9347	0.9996
Barbara	1×8	256	197888	51.1360	0.9972
	2×4	256	197888	51.1542	0.9972
	4×2	256	197888	51.1450	0.9972
	2×8	256	98944	54.1392	0.9986
	3×8	256	64932	55.9208	0.9991
Boat	1×8	256	197888	51.1419	0.9972
	2×4	256	197888	51.1469	0.9972
	4×2	256	197888	51.1410	0.9972
	2×8	256	98944	54.1624	0.9986
	3×8	256	64932	55.9081	0.9991
Couple	1×8	256	197888	51.1399	0.9975
	2×4	256	197888	51.1316	0.9975
	4×2	256	197888	51.1489	0.9975
	2×8	256	98944	54.1454	0.9987
	3×8	256	64932	55.9446	0.9992
Lena	1×8	256	197888	51.1344	0.9960
	2×4	256	197888	51.1447	0.9960
	4×2	256	197888	51.1509	0.9960
	2×8	256	98944	54.1487	0.9980
	3×8	256	64932	55.9430	0.9987
Peppers	1×8	256	197888	51.1362	0.9963
	2×4	256	197888	51.1452	0.9963
	4×2	256	197888	51.1435	0.9963
	2×8	256	98944	54.1419	0.9981
	3×8	256	64932	55.8990	0.9988

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:

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

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

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

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

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

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	197888	51.15	49.90	0.1957	0.4990
	1×6	64	180048	51.15	49.89	0.1957	0.4989
	1×4	8	155648	51.14	49.98	0.1960	0.4998
	1×2	4	131072	51.14	50.06	0.1963	0.5006
Baboon	1×8	256	197888	51.15	49.90	0.1957	0.4990
	1×6	64	180048	51.12	50.22	0.1969	0.5022
	1×4	8	155648	51.14	50.05	0.1963	0.5005
	1×2	4	131072	51.14	50.01	0.1961	0.5001
Barbara	1×8	256	197888	51.14	50.06	0.1963	0.5006
	1×6	64	180048	51.15	49.95	0.1959	0.4995
	1×4	8	155648	51.14	49.98	0.1960	0.4998
	1×2	4	131072	51.14	50.02	0.1961	0.5002
Boat	1×8	256	197888	51.14	49.99	0.1960	0.4999
	1×6	64	180048	51.15	49.88	0.1956	0.4988
	1×4	8	155648	51.14	50.00	0.1961	0.5000
	1×2	4	131072	51.13	50.07	0.1964	0.5007
Couple	1×8	256	197888	51.14	50.01	0.1961	0.5001
	1×6	64	180048	51.14	49.97	0.1960	0.4997
	1×4	8	155648	51.14	49.99	0.1960	0.4999
	1×2	4	131072	51.13	50.08	0.1964	0.5008
Lena	1×8	256	197888	51.13	50.08	0.1964	0.5008
	1×6	64	180048	51.14	49.98	0.1960	0.4998
	1×4	8	155648	51.14	49.97	0.1960	0.4997
	1×2	4	131072	51.14	49.97	0.1960	0.4997
Peppers	1×8	256	197888	51.14	50.06	0.1963	0.5006
	1×6	64	180048	51.14	50.04	0.1962	0.5004
	1×4	8	155648	51.15	49.93	0.1958	0.4993
	1×2	4	131072	51.14	50.05	0.1963	0.5005

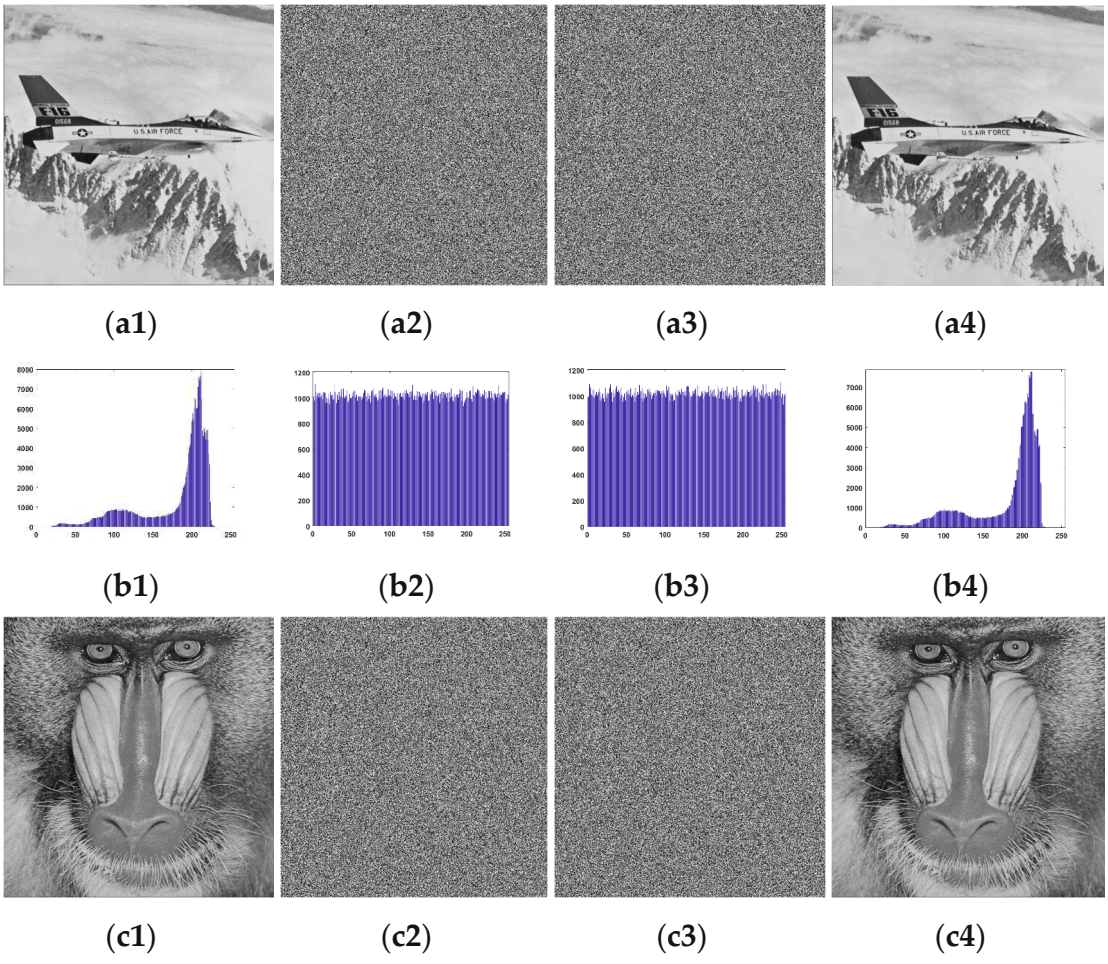
NPCR is employed to compute the number of differing pixels between two images, while UACI is utilized to calculate the average change in pixels between two images. Unlike Table 1, Table 2 is a test of the proposed scheme with various codebook table sizes. From Table 2, it can be clearly found that, as 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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

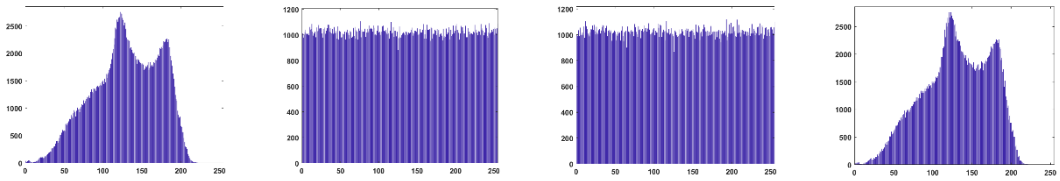
each pixel in a pixel block was used to record the original pixel block index in a group and the PSNR values of all tested scenarios were around 51dB. The PSNR values indicated that our recovered image had a good visual quality. As shown in Table 2, the proposed scheme involves replacing the LSB of pixels 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 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 find 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 uniform consequently. 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.



Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1



(d1) (d2) (d3) (d4)

Figure 10. Execution results of our proposed scheme: (**a1**) is original "Airplane" image with 512×512 pixels. (**a2**) and (**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 (**c**) and (**d**), 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

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

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, N denotes the number of pixel pairs, and then calculate the correlation coefficient between them. Tables 3, 4 and 5 were tested with the pixel block size of 8 and the codebook table size of 256. The information entropy and the pixel correlation results for horizontal and vertical pairs are shown in Table 3 and Table 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 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 scramble 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.

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

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

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.

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

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 subsection 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.

4.3. Comparison with state-of-the-art schemes

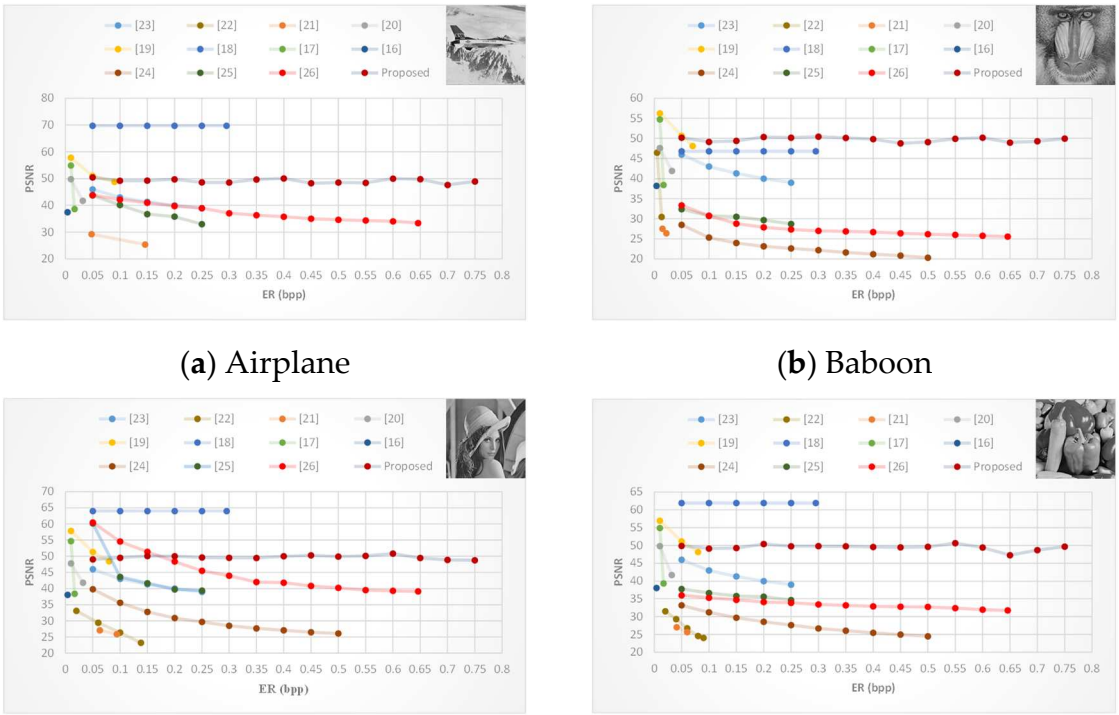
In this section, we primarily compare our proposed scheme with some state-of-the-art schemes for data hiding in encrypted images in terms of PSNR, SSIM, and embedding capacity. In Table 6, the embedding capabilities of our proposed schemes all outperform the other schemes. Moreover, our proposed scheme 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 6, it is evident that our proposed scheme also outperforms most of SOTA methods based on the PSNR values and SSIM values. Even though the PSNR is slightly lower than [18], our proposed scheme achieves a PSNR of approximately 51 dB, which is adequate for visual quality of a recovered image.

Table 6. EC, PSNR and SSIM comparisons.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 December 2023 doi:10.20944/preprints202312.2341.v1

Image		[18]	[23]	[24]	Group size=16	Group size=64	Group size=256
Lena	EC	77385	65536	131072	155648	180048	197888
	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	77385	65536	131072	155648	180048	197888
	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	77385	65536	131072	155648	180048	197888
	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	77385	65536	130915	155648	180048	197888
	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 18 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.

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 secret message and create a marked image by 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 our experiment results, we assured that the proposed scheme could achieve our goal of increasing embedding capacity by significant amount comparing to the SOTA schemes. Our experiments showed that the consistency of embedding capability with any cover image as long as the sizes of pixel-blocks and block groups were the same. As for PSNRs, the results showed the proposed scheme could outperform the SOTA schemes as well.

Author Contributions: Conceptualization and methodology, Jui-Chuan Liu, Hengxiao Chi, Ching-Chun Chang and Chin-Chen Chang; software, Hengxiao Chi; validation, Jui-Chuan Liu, Hengxiao Chi, Ching-Chun Chang and Chin-Chen Chang; data curation, Hengxiao Chi; writing—original draft preparation, Jui-Chuan Liu, and Hengxiao Chi; writing—review and editing, Jui-Chuan Liu, and Hengxiao Chi; supervision, 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: Not applicable.

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

References

1. U. Fiore, "Selective Redundancy Removal: A Framework for Data Hiding", *Future Internet*, 2:30-40, 2010, doi:10.3390/fi2010030.
2. Y. Q. Shi, X. Li, X. Zhang, H. T. Wu, and B. Ma, "Reversible data hiding: Advances in the past two decades," *IEEE Access*, 4:3210–3237, 2016.
3. W. Zhang, H. Wang, D. Hou, and N. Yu, "Reversible data hiding in encrypted images by reversible image transformation," *IEEE Trans. Multimedia*, 18(8):1469–1479, 2016.
4. K. Ma, W. Zhang, X. Zhao, N. Yu, and F. Li, "Reversible data hiding in encrypted images by reserving room before encryption," *IEEE Trans. Inf. Forensics Secur.*, 8(3):553–562, 2013.
5. P. Puteaux and W. Puech, "A recursive reversible data hiding in encrypted images method with a very high payload," *IEEE Trans. Multimedia*, 23:636–650, 2020, doi: 10.1109/TMM.2020.2985537.
6. Y. Wu, Y. Xiang, Y. Guo, J. Tang, and Z. Yin, "An improved reversible data hiding in encrypted images using parametric binary tree labeling," *IEEE Trans. Multimedia*, 22(8):1929–1938, 2020.
7. P. Pauline and P. William, "An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images," *IEEE Trans. Inf. Forensics Secur.*, 13(7):1670–1681, 2018.
8. F. Chen, Y. Yuan, H. He, M. Tian, and H. M. Tai, "Multi-MSB compression based reversible data hiding scheme in encrypted images," *IEEE Trans. Circuits Syst. Video Technol.*, 31(3):905–916, 2021, doi:10.1109/TCSVT.2020.2992817.
9. S. Yi and Y. Zhou, "Separable and reversible data hiding in encrypted images using parametric binary tree labeling," *IEEE Trans. Multimedia*, 21(1):51–64, 2019.
10. X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Trans. Inf. Forensics Secur.*, 7(2):826–832, 2012.
11. S. Yi and Y. Zhou, "Parametric reversible data hiding in encrypted images using adaptive bit-level data embedding and checkerboard-based prediction," *Signal Process.*, 150:171–182, 2018.
12. C. C. Chang, J. F. Chang, W. J. Kao, and J. H. Horng, 2021, "Two-Layer Reversible Data Hiding for VQ-Compressed Images Based on De-Clustering and Indicator-Free Search-Order Coding," *Future Internet*, 13(8):215, doi:10.3390/fi13080215.
13. J. C. Liu, C. C. Chang, C. C. Lin, and C. C. Chang. Hiding Information in a Well-Trained Vector Quantization Codebook. *ACM International Conference on Signal Processing and Machine Learning (SPML)*, July 2023, doi:10.1145/3614008.3614052.
14. S. D. Mohd Satar, M. Hussin, Z. M. Hanapi, and M. A. Mohamed, "Towards Virtuous Cloud Data Storage Using Access Policy Hiding in Ciphertext Policy Attribute-based Encryption", *Future Internet*, *13*(11):279, 2021, doi:10.3390/fi13110279.
15. Celik MU, Sharma G, Tekalp AM, and Sable E, "Lossless generalized-LSB data embedding," *IEEE Trans Image Process*, 14(2):253–266, 2005.

16. W. Hong, T. S. Chen, and H. Y. Wu, "An improved reversible data hiding in encrypted images using side match," *IEEE Signal Processing Letters*, 19(4):199-202, 2012.
17. X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Transactions on Information Forensics and Security*, 7(2):826–832, 2011.
18. Z. Qian and X. Zhang, "Reversible data hiding in encrypted images with distributed source encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, 26(4):636–646, 2015.
19. X. Wu and W. Sun, "High-capacity reversible data hiding in encrypted images by prediction error, " *Signal Processing*, vol. 104, pp. 387–400, 2014.
20. C. Qin, W. Zhang, F. Cao, X. Zhang, and C.-C. Chang, "Separable reversible data hiding in encrypted images via adaptive embedding strategy with block selection," *Signal Processing*, vol. 153, pp. 109–122, 2018.
21. F. Huang, J. Huang, and Y.-Q. Shi, "New framework for reversible data hiding in encrypted domain," *IEEE Transactions on Information Forensics and Security*, 11(12):2777–2789, 2016.
22. H. Ge, Y. Chen, Z. Qian, and J. Wang, "A high capacity multi-level approach for reversible data hiding in encrypted images," *IEEE Transactions on Circuits and Systems for Video Technology*, 29(8):2285–2295, 2018.
23. R. Bhardwaj and A. Aggarwal, "An improved block based joint reversible data hiding in encrypted images by symmetric cryptosystem," *Pattern Recognition Letters*, 139:60–68, 2020.
24. X. Wang, C. C. Chang, C. C. Lin, and C. C. Chang, "Reversal of pixel rotation: A reversible data hiding system towards cybersecurity in encrypted images," *Journal of Visual Communication and Image Representation*, 82:103421, 2022.
25. M. Yu, H. Yao, and C. Qin, "Reversible data hiding in encrypted images without additional information transmission," *Signal Processing: Image Communication*, 105:116696, 2022.
26. K. Gao, J. H. Horng, and C. C. Chang, "Dual Mode Data Hiding in Fully Encrypted Images with Pixel-Shuffling for Cloud Applications," *Displays*, 102609. 2023.