Reversible Data Hiding in Encrypted Image with Secret Sharing

Reading and Implementation

電機碩一 r11921072 謝子涔 | 醫工碩一 r11528026 陳品如

Introduction

Nowadays, with the rapid development of cloud storage and cloud computing, the integrity of data transmission to the cloud server are threatened. After uploading the encrypted image to the cloud, the image might undergo tampering attacks from the third party. The original image is unable to be recovered on the receiver side.

To solve this problems, secret sharing strategy was introduced. We focus on the approach of Qin *et al* [1]. According to their schemes, when partial share images are damaged, original image can be recovered under certain condition. Since, the secret message is embedded in each share, even partial shares are damaged, additional data can still be extracted. Different from most of the methods, based on the secret sharing and data hiding over *GF* (2^8), the method proposed in this paper can process pixels with pixel value greater than 250. The higher embedding rate and perfect recovery performance can be exhibited.

In this research, we implement the whole algorithm to reproduce the concept. Additionally, we also refer to another paper [2] to compare different methods of image encryption toward this data embedding method and analysis the safety of image encryption.

Problem addressed by the paper

Generally, reversible data hiding in encrypted images (RDHEI) contains vacating room after encryption (VRAE) for data embedding and reserving room before encryption (RRBE) for data embedding. However, for the state-of-the-art method, including RDHEI with MSB prediction, or with homomorphic encryption, the computational complexity is relatively high. Therefore, RDHEI schemes based on secret image sharing is proposed in this paper. With the growth of cloud storage and computing, the transmission security of data is threatened. The encrypted image on the cloud server might undergo attacking from the third party, and the original image is unable to be recovered losslessly for receiver. In current state-of-the-art methods, sharing image via Shamir secret sharing and embed additional data into image shares via DE, HS and homomorphic addition are developed. However, due to the restriction of the algorithm, pixels values lager than 250 cannot be encrypted. In this paper, the main feature is to provide the higher embedded rate and to assure that all different pixel value can be encrypted.

Related work

Over the past two decades, reversible data hiding (RDH) techniques have gained significant popularity in the field of data hiding. However, in certain data hiding methods, the image that contains embedded information, known as the marked image, undergoes distortion during the embedding process, making it impossible to restore the image to its original form after data extraction. As a result, experts have been increasingly motivated to address this issue, leading to the emergence of reversible data hiding (RDH).

RDH can be classified into three main categories: lossless compression [3], difference expansion [4], and histogram shifting [5]. Nevertheless, due to the growing emphasis on privacy protection in encrypted images, traditional reversible data hiding techniques have become less suitable and have been replaced by a more advanced approach known as reversible data hiding in encrypted images (RDHEI). In this method, the process is divided into two sides: the data hider side and the receiver side. On the data hider side, the desired information is embedded, while on the receiver side, the data can be extracted using a data hiding key.

Solutions or techniques by the paper

The whole procedure is represented as follow:

1. Plain image is encrypted with image scrambling method (key-based and zigzag.)
2. Shamir's secret sharing is applied to the encrypted image to generate multiple shares.
3. Data hider takes charge of embedding the additional data into different shares.
4. In our implementation, we realize the transmission of encrypted image and key via socket programming with python. Content owner can transmit the any shares and keys to the receiver.
5. Receiver collects the shares and keys. In our program, once the receiver collects the enough shares and keys, the encryption and recovery process are conducted automatically.

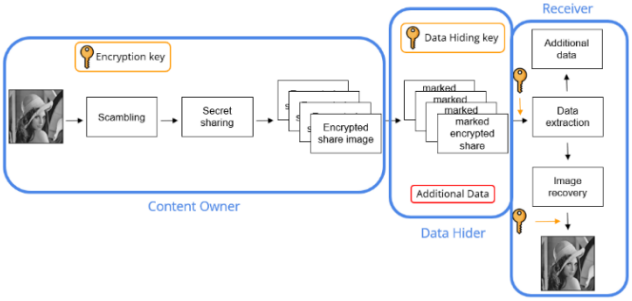

Fig. 1. Our proposed flowchart

Image Encryption

In our research, we adopted two different methods of image encryption to compare the security and functionality of the whole scheme.

Method 1: Key-based scrambling

Before Shamir's secret sharing, we make the image unreadable and break the correlation between the neighboring pixels first. Most image encryption can be divided into two phases: confusion and diffusion. In the confusion phase, only the order of pixels is changed, not the pixel values, but in the diffusion phase, the pixel values will be recalculated. The technique here is image scrambling, and we tried two different ways to achieve this. One of the methods is Key-based image scrambling and the other one is Zigzag scrambling. Key-based scrambling only does image confusion. Zigzag scrambling not only does image confusion but also does image diffusion.

Key-based scrambling is based on row and column swapping of pixel arrangement. The flow diagram of key-based image scrambling is shown in Fig. 2. First, we divide the original image into several non-overlapping blocks and each size of block is 2x2, thus, the number of blocks is K = ⌊MN/4⌋ blocks. Second, using two encryption keys $K_{ey}^{(1)}$ and $K_{ey}^{(2)}$ as the new order of the blocks and the pixels in each block. We use a pseudo-random number generator to generate two keys. Here, $K_{ey}^{(2)}$ includes K keys because each block needs to have a different key to each other. After the image is scrambled, both the sender and receiver can get these two keys.

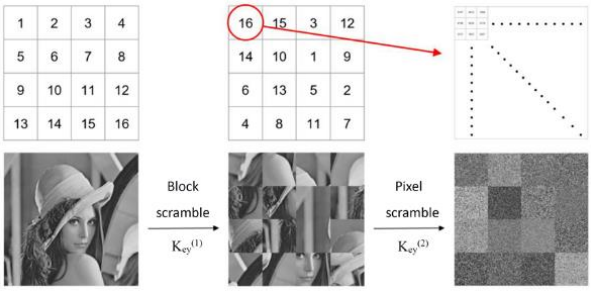

Fig. 2. Key-based scrambling

Method 2: Zigzag pattern scrambling

In [2], a new image scrambling method is proposed. The feature of zigzag pattern scrambling is that the correlation between different pixels is relatively low. It has a higher security level. The whole method can be separated into four parts: image splitting, image scrambling (confusion), key generation and diffusion.

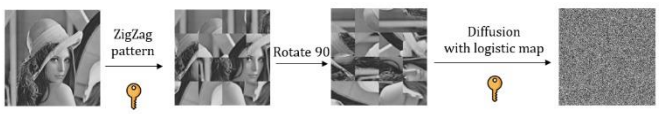

Fig. 3. Zigzag scrambling

1. Image splitting:
   The plain image is divided into several non-overlapping blocks of the same size which can be selected by user. Afterward, each block can either be subdivided into equal-sized sub-blocks or kept intact without any further splitting. The decision on selecting sub-blocks within each block depends on a randomly generated number specific to that block.

2. Image Scrambling:
   The scrambling algorithm is based on zigzag pattern. The blocks are scrambled based on the order of zigzag pattern. The block order is saved as the first key. Next, all the blocks are rotated by 90°

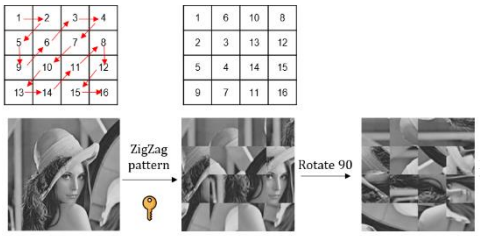

Fig. 4. Image scramblinng by zigzag pattern and rotate 90°

3. Key generation
   The key used in the diffusion process is generated from a logistic map, which is defined as:

$$Y_{n+1} = aY_n(1 - Y_n)$$

Control parameter is defined as a, with range $0 < a \leq 4$. $a \in [3.57, 4]$ is the most chaotic. Y_n is the sequence with $0 < Y_n < 1$.

a. Calculate initial value $Y_0 = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} P(i,j)}{M*N*255}$ where M and N is the number of rows and columns in the plain image respectively. P is the plain image.

b. Iterate the logistic map for $N_0 + MN$ times, skip the first N_0 elements to get the new sequence S. We set N_0 as 10000

c. Use sequence S to calculate the key

$$K(i) = mod(floor(S(i) * 10^{14}), 256)$$

i = 1: MN

Therefore, the second key is generated to diffuse the scrambling image.

4. Diffusion
   Bit-wise XOR operation between the key K and the zigzag scramble image to get the encrypted image.

$$E = X \oplus K$$

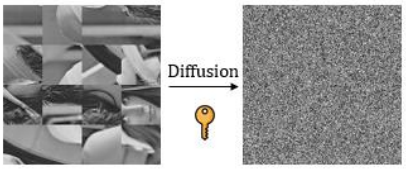

Fig. 5. Diffusion of zigzag method

The algorithm of zigzag scrambling is as follow:

Algorithm 1 The Proposed Algorithm for Medical Image Encryption

Input: the plain image P with size $M \times N$, parameter a of the logistic map and N_0.

1. Divide P into an equal number of blocks, with a block size $h = 2^n$ where n takes a value from 4, 5, 6
2. Generate a random number for each block depending on the block size 2^l where l takes a random value from $2, 3, \ldots n$, put the generated random numbers in a vector R_i, $i = 1: (MN/h^2)$.
3. For i = 1: MN/h^2 do
    4. Divide a block into sub-blocks with the same size or keep without dividing based on R_i.
5. End for
6. Perform a Zigzag pattern and rotation with angle 90, respectively, to both undivided blocks and sub-blocks.
7. Generate a random vector r with size MN/h^2.
8. Random permutation of image blocks based on r is performed to get the scrambled image X.
9. Generate the initial condition of the logistic map using (eq.2)
10. Iterate the chaotic map (eq.1) $N_0 + MN$ times, and then discard first N_0 elements to get a new sequence S with size MN.
11. For i = 1: MN do
    12. $K(i) = mod\ (floor\ (S(i) \times 10^{14}), 256)$
13. End for
14. Convert the matrix X into the 1D image pixel vector X'
16. $E = X' \oplus K$
17. Convert E into a 2D matrix C.

Output: the encrypted image C

Shamir's secret sharing

Shamir's Secret Sharing is a cryptographic technique that allows a secret to be divided into multiple shares. These shares are distributed among different parties, where a specified number of shares is required to reconstruct the original secret. This scheme ensures that no single party possesses the complete secret. It provides a robust and secure method for distributing sensitive information, as long as the threshold for reconstructing the secret is not exceeded.

It is a (t, n) threshold scheme based on polynomial interpolation over finite fields. For a polynomial of degree t-1, we can define this polynomial with t points.

$$f(x) = (s + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1})$$

Let Shamir's secret sharing implement apply over Galois fields $GF(2^8)$, the polynomial f(x) and the irreducible polynomial p(x) are defined as:

$$f(x) = (s \oplus a_1 x \oplus a_2 x^2 \oplus \ldots \oplus a_{t-1} x^{t-1})\ mod\ p(x)$$

$$p(x) = x^n + x + 1$$

In $GF(2^8)$, the addition or subtraction operator represents XOR.

Because $GF(p)$ cannot perform the secret sharing operation when the pixel values are greater than 250. If p(x) is an irreducible polynomial, it's more efficient for modulo $p(x) = x^n + x + 1$ in a finite field. In our scheme with $GF(2^8)$, an irreducible polynomial $p(x) = x^8 + x^4 + x^3 + x + 1$ is used.

The values of a and x are generated pseudo irregularly and each pixel in block can transformed into secret pixels by the equation. If n=3, it means that after calculation it will produce 3 encrypted shares.

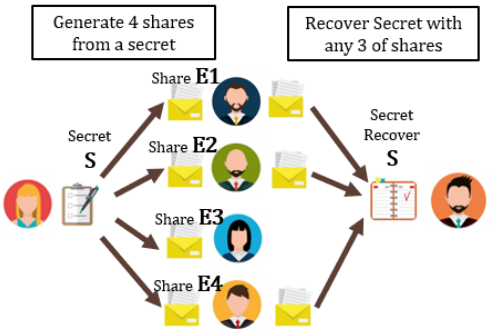

Fig. 6. Shamir's secret sharing

In our scheme, take four one block as an example:

Let the degree of polynomial t-1 = 2, n = 4

Apply $GF(2^8)$ and the irreducible polynomial p(x)

$$f(x) = (s \oplus a_1 x \oplus a_2 x^2)\ mod\ p(x)$$

$$p(x) = x^8 + x^4 + x^3 + x + 1$$

For K blocks, pixels $\{y_{j1},\ y_{j2},\ y_{j3},\ y_{j4}\}$ = s, $1 < j \leq K$ can be transformed into

$\{f_{yj1}(x_i),\ f_{yj2}(x_i), f_{yj3}(x_i), f_{yj4}(x_i)\},\ 1 < i \leq n$

with $\{a_{j1} \ldots\ a_{j(t-1)}\}$ and $\{x_{j1} \ldots\ x_{jn}\}$

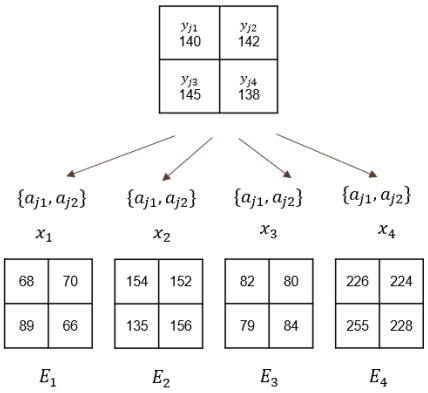

Fig. 7. Example of our scheme

Data hider: Data Embedding

After image encryption, the original image is encrypted into n shares with Shamir's secret sharing and the encrypted shares are sent to data hider for data embedding. Reversible data hiding techniques are highly valued for their ability to conceal messages within an image discreetly. The sender possesses the skill to embed messages without raising any suspicion from third parties, while the receiver can extract and restore the embedded information using a dedicated extraction and restoration algorithm. These techniques are quite commonly used in daily life, particularly in fields like military intelligence and medical imaging, where even the slightest compromise in image quality is deemed unacceptable.

The widespread adoption of reversible information hiding techniques can be attributed to their dual advantage of ensuring the security of embedded information and facilitating the seamless restoration of the image to its original, unaltered state.

The algorithm we used to achieve reversible data hiding is shown in Algorithm. First, define threshold(ε) and the secret data. Second, we need to classify all blocks into two sets: Embedded set (ES) and Non-embedded set (NS). The classification standard is the following formula, where the functions bin2dec and dec2bin indicate binary-to-decimal and decimal-to-binary operation. If the block satisfies with formula, then it is classified into ES, thus, the block is NS. Third, divide each pixel in each block into Reference pixel (RP) and Embedding pixel (EP), the first pixel in each block is called RP and the rest of the pixels are called EP. Step4 to Step 6 are explained with the example in Fig. 8. In Fig. 8, we can see that block 1 is represented into NS and block 2 is represented into ES. In Step 4, we redefine each RP's MSB. If ES, then the MSB of RP is '0', thus, RP is '1'. Next, replaced MSB of each RP are combined with the additional data as payload. Finally, according to the threshold(ε) and the relationship between ε an u1, u2 is shown in Table I. We know that u1 is 3 and u2 is 5, u1 means the result of XOR, u2 indicates that how many bits we need to embed in the payload. Notice that the reason we use the result of XOR to calculate the difference between RP and EP is because the entire image processing process is based on Galois field $GF(2^8)$

$$\text{bin2dec}(\text{dec2bin}(\text{fyjl}(\text{xi})) \oplus \text{dec2bin}(\text{fyjz}(\text{xi}))) \leqq \varepsilon,$$

$$\{\text{fyjl}(\text{xi}),\ j = 1,2,\ldots,K,\ l=1,2,3,4,\ 1 \leqq i \leqq n,\ \forall\ Z \in \{2,3,4\}\}$$

Algorithm 2 Adaptive Data Embedding Algorithm Over $GF(2^8)$

Input: Encrypted share, secret data, threshold ε'

1: Classify all blocks into two categories of ES and NS according to (20).
2: Divide all pixels in each block into RP and EPs.
3: Embed payload bits into EPs.
4: **for** each block in ES and NS **do**
5: Replace the MSB of RP with "0" for ES or the MSB of the first pixel with "1" for NS.
6: **end for**
7: **for** each pixel in EP **do**
8: Replace the u_1 bits according to Table II with the binary representation of D_{jz}.
9: Replace the remaining u_2 bits with payload.
10: **end for**

Output: Marked share

RELATIONSHIP BETWEEN ε', D_{jz}, u_1, AND u_2

ε'	D_{jz}	u_1	u_2
0	$D_{jz} = 0$	0	8
1	$D_{jz} = 1$	1	7
3	$D_{jz} \leq 3$	2	6
7	$D_{jz} \leq 7$	3	5
15	$D_{jz} \leq 15$	4	4
31	$D_{jz} \leq 31$	5	3
63	$D_{jz} \leq 63$	6	2
127	$D_{jz} \leq 127$	7	1

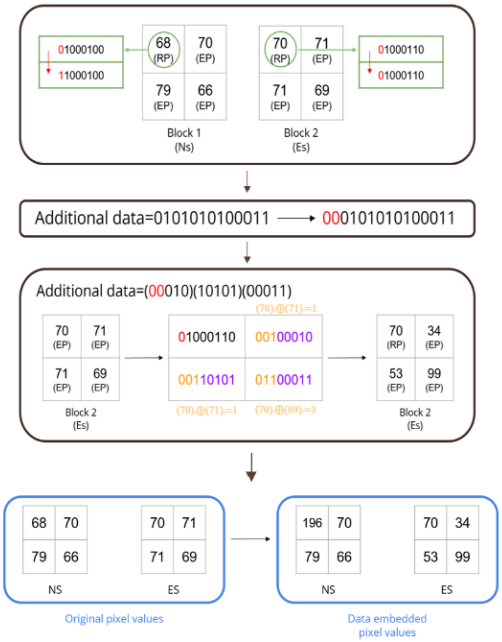

Fig. 8. Example of data embedding over $GF(2^8)$

Receiver: Data Extraction

After the receiver receives a set of images containing embedded data, referred to as marked encrypted shares and a data hiding key, the additional data can be extracted. To begin with, the marked share is divided into non-overlapping blocks of size 2x2. By analyzing the most significant bit (MSB) of a reference pixel (RP), it is possible to determine whether a block is an Embedded Set (ES) block or a Non-embedded set (NS).

The data hiding key provides information about a threshold, allowing us to determine the values of u_1 and u_2. The last u_2 bits in each block represent specific parts of the additional data, enabling the extraction of the entire payload. Once the payload is obtained, the first K bits are utilized to recover the pixels of each reference pixel (RP). Moreover, the value of u_1 helps determine the difference between RPs and Embedded Pixels (EPs). The recovered By following these steps, the RPs within each ES block can be successfully recovered. To aid in comprehending the aforementioned procedure, an example is provided in the Fig. 9.

$$f_{yjz}(x_i) = \text{bin2dec}(\text{dec2bin}(f_{yjl}(x_i)) \oplus \text{dec2bin}(D_{jz})) \quad (2)$$

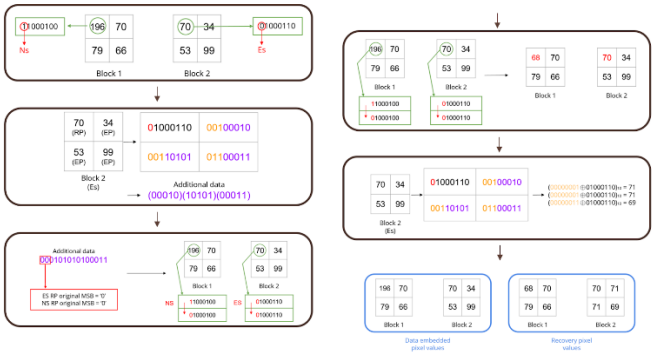

Fig. 9. Example of data extraction over GF(2^8)

Receiver: Shamir's secret sharing recovery

According to Shamir's secret sharing, when any t or more shares the known x are collected, the coefficients *a* of *f(x)* and the secret message s can be reconstructed by using a Lagrange interpolation method

$$f(x) = \sum_{q=1}^{t} \left(f(x_q) \prod_{\substack{1 \le w \le t \\ w \ne q}} \frac{x - x_w}{x_q - x_w} \right)$$

$$s = f(0) = \sum_{q=1}^{t} \left(f(x_q) \prod_{\substack{1 \le w \le t \\ w \ne q}} \frac{-x_w}{x_q - x_w} \right)$$

In our scheme, the degree of polynomial t-1 = 2, n = 4

With $GF(2^8)$ and the irreducible polynomial p(x)

$$f(x) = (s \oplus a_1 x \oplus a_2 x^2) \bmod p(x)$$

$$p(x) = x^8 + x^4 + x^3 + x + 1$$

We only need any 3 shares to recover the secret

$$f(x) = \sum_{q=1}^{t} \left(f(x_q) \prod_{\substack{1 \le w \le t \\ w \ne q}} \frac{x - x_w}{x_q - x_w} \right) \bmod p$$

$$s = f(0) = \sum_{q=1}^{t} \left(f(x_q) \prod_{\substack{1 \le w \le t \\ w \ne q}} \frac{-x_w}{x_q - x_w} \right)$$

Receiver: Image Recovery

In our research, two different methods of image encryption are implemented to compare the security and functionality of the whole scheme. For image recovery, we also divide into two method to recover image respectively.

Method 1: Key-based scrambling

The key-based scramble method involves the receiver obtaining an encryption key that plays a crucial role in the decryption process. This encryption key consists of two sub-keys: the block scramble order key and the pixel scramble order key. The pixel scramble order key indicates the specific order in which the pixels within each block are sorted, while the block scramble order key determines the shuffled sequence of K blocks.

By utilizing these keys, it becomes a straightforward task to restore the scrambled image back to its original form. It is important to note that this scrambling process solely introduces confusion without altering the actual pixel values. Thus, the resulting decrypted image maintains the same pixel values as the original image.

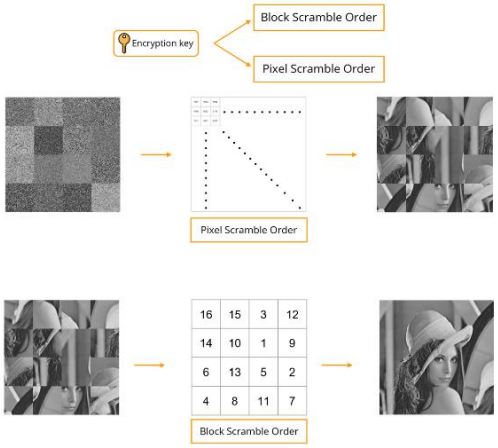

Fig. 10. Recovery of key-based scrambling

Method 2: Zigzag pattern scrambling

To recover the plain image, the decryption process involves reversing the encryption stages using the encryption key consisting of two sub-keys: the zigzag pattern key and the logistic map key. The steps for decryption are as follows:

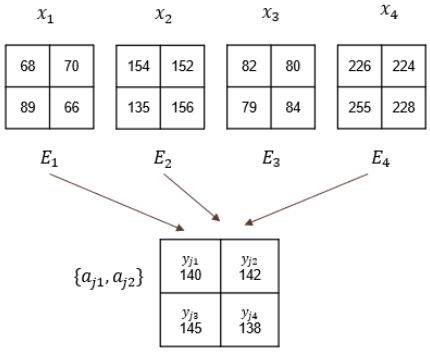

Fig. 11. Zigzag pattern decryption

1. In image scrambling process, XOR operation between the logistic map key K and the zigzag scramble image to get the encrypted image. $E = X \oplus K$. Therefore, with the logistic map key K, we can recover the scrambling image with $X = E \oplus K$

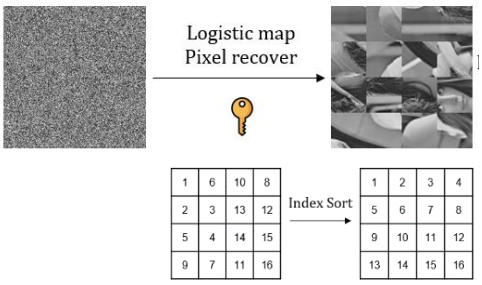

Fig. 12. Zigzag pattern recovery by using logistic map

2. With the key of zigzag pattern, we can rearrange each block to the original position. Reverse the rotation and zigzag pattern operations, applying them in the

inverse order to both undivided blocks and sub-blocks.

Fig. 13. Rotate 270° to recover each block

By following these steps, the encrypted image can be decrypted and restored to its original form.

Comparison with existing approaches

Security

In the context of an original image with dimensions MxN, we can observe that it is divided into K non-overlapping blocks, where K is calculated as ⌊MN/4⌋. Each block encompasses a size of 2x2 pixels. It is important to highlight that both the blocks themselves and the individual pixels within each block undergo a scrambling process. Consequently, the number of potential situations exponentially increases, with ⌊MN/4⌋! variations arising from block scrambling and (4!) ⌊MN/4⌋ possibilities originating from pixel scrambling. Therefore, when considering the overall key space for block and pixel scrambling, it can be calculated as the product of ⌊MN/4⌋! and (4!) ⌊MN/4⌋.

It is essential to emphasize the magnitude of the numbers obtained from the aforementioned formula. Due to the substantial size of the key space, establishing a direct, one-to-one mapping between the scrambled image and the original image without the encryption key is virtually impossible. Consequently, in our work, we encounter an incredibly vast number of potential situations, estimated to be 65536! x (4!)65536.

Performance

In order to assess the performance of our reversible data hiding technique, we employ two widely-used metrics, namely Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM). The selection of these metrics stems from our objective to measure the visual quality of the recovered image in comparison to the original image.

To conduct the performance analysis, we utilize MATLAB. By employing MATLAB, we are able to calculate the PSNR and SSIM values for our results. And the results are shown in Table 1. Upon reviewing the obtained results, we find that the PSNR value is infinite, indicating an extremely high fidelity between the recovered image and the original image. Furthermore, the SSIM value is 1, indicating an almost perfect similarity between the recovered and original images.

Based on these evaluation metrics, it can be concluded that the encryption image is not the same as the original image, to the extent that it is nearly impossible to distinguish them visually. This provides evidence of the successful implementation of our reversible data hiding technique, as it achieves the preservation of the visual quality of the image while effectively concealing the hidden data.

	τ	Our method	[6]	[7]
PSNR	τ=0.05bpp	Inf	44.21	50.26
	τ=0.1bpp	Inf	41.93	47.15
	τ=0.2bpp	Inf	38.94	43.02
SSIM	τ=0.05bpp	1	0.9860	0.9906
	τ=0.1bpp	1	0.9708	0.9839
	τ=0.2bpp	1	0.9436	0.9716

Table. 1. Performance comparison

Image Encryption

In our implementation, we utilized two methods to evaluate performance by analyzing the histograms of encrypted images. The key-based method, which preserves the pixel values, ensures that the original image retains its histogram characteristics, allowing us to assess the preservation of data distribution and evaluate the effectiveness of encryption algorithms.

On the other hand, the zigzag method, employing the logistic map to scramble pixel values, results in a flattened histogram and introduces a higher level of randomness, reducing the correlation between adjacent pixels. This decrease in pixel correlation enhances the security of the encrypted image, making it more challenging to extract meaningful information or patterns. By analyzing histograms, we gain valuable insights into the benefits of image encryption, identifying potential vulnerabilities, and enhancing the confidentiality of sensitive information and secret message by reducing pixel interdependence.

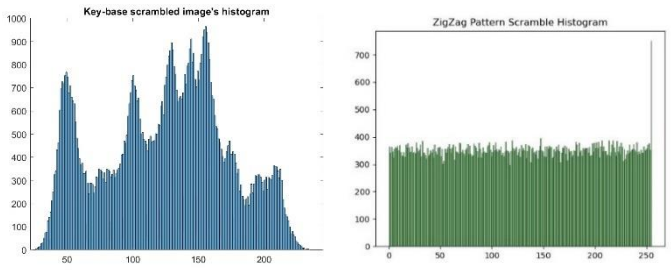

Fig. 14. Histogram comparison of Key-based scrambling method and Zigzag pattern scrambling method

Limitation of the data embedding method

In the unique data embedding method proposed in this paper, it presents a higher data embedding rate compared to conventional techniques. However, this method is constrained by the available embedding space (ES), which makes it difficult to determine the precise amount of embedded information in advance. In our implementation, encountering challenges in accurately

calculating the length of the embedded data. It should be noted that this approach may not be universally applicable to all types of images and encryption methods, as the effectiveness of the embedding process depends on various factors such as image size and complexity. In particular, in our implementation, we observed that the resulting zigzag encrypted image did not provide a sufficiently large embedded set to accommodate all of the payload information. This limitation necessitates further exploration and refinement to ensure compatibility with a broader range of image data and encryption techniques, while maintaining a satisfactory embedding capacity.

Conclusions

The RDHIE (Reversible Data Hiding for Image Encryption) scheme proposed in this study offers a versatile solution applicable across various fields. When comparing different data encryption methods, the zigzag technique demonstrates higher efficiency and lower pixel correlation, enhancing the overall security of the encrypted data.

Incorporating Shamir's secret sharing mechanism ensures that even in scenarios where shares are compromised by a third party, the original data remains protected against unauthorized recovery. This robust protection adds an additional layer of security to the RDHIE scheme.

However, a significant challenge encountered in this research is the difficulty in accurately determining the length of embedded data. This limitation affects the generality of the proposed method, as it may not be easily applicable to all types of images or encryption techniques. Further research is needed to address this issue and refine the embedding process.

By analyzing image quality metrics such as Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM), we can ascertain that the extracted data is lossless and the recovered image maintains its original integrity. This observation highlights the successful extraction of complete data while preserving the image's visual fidelity.

In conclusion, the RDHIE scheme presented in this paper offers a comprehensive solution, and the proposed method demonstrates innovation. However, the lack of general applicability across all image types remains a limitation that requires further exploration and development.

References

1. C. Qin, C. Jiang, Q. Mo, H. Yao and C. -C. Chang, "Reversible Data Hiding in Encrypted Image via Secret Sharing Based on GF(p) and GF(2^8)," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 32, no. 4, pp. 1928-1941, April 2022
2. S. T. Kamal, K. M. Hosny, T. M. Elgindy, M. M. Darwish and M. M. Fouda, "A New Image Encryption Algorithm for Grey and Color Medical Images," in IEEE Access, vol. 9, pp. 37855-37865, 2021
3. M. U. Celik, G. Sharma, A. M. Tekalp and E. Saber, "Lossless generalized-LSB data embedding," in *IEEE Transactions on Image Processing*, vol. 14, no. 2, pp. 253-266, Feb. 2005
4. Wenjia Ding, Huyin Zhang, Ralf Reulke, and Yulin Wang. 2022. Reversible image data hiding based on scalable difference expansion. Pattern Recogn. Lett. 159, C (Jul 2022), 116–124.
5. Huang, D., Wang, J. Efficient reversible data hiding based on the histogram modification of differences of pixel differences. Multimed Tools Appl 79, 20881–20896 (2020).
6. X. Zhang, "Reversible Data Hiding in Encrypted Image," in IEEE Signal Processing Letters, vol. 18, no. 4, pp. 255-258, April 2011, doi: 10.1109/LSP.2011.2114651.
7. Xin Liao and Changwen Shu. 2015. Reversible data hiding in encrypted images based on absolute mean difference of multiple neighboring pixels. J. Vis. Comun. Image Represent. 28, C (April 2015), 21–27.