Reversible Data Hiding in Encrypted Binary Images Based on Improved Prediction Methods

Fang Ren[1,2], Meng-Meng Zeng[1,2], Zi-Fan Wang[1], and Xuan Shi[1]
(Corresponding author: Fang Ren)
School of Cyberspace Security, Xi'an University of Posts and Telecommunications[1]
National Engineering Research Center for Secured Wireless[2]
Xi'an, 710121, China
Email: renfang_81@163.com
(Received May 29, 2024; Revised and Accepted Jan. 24, 2025; First Online Mar. 14, 2025)

Abstract

With the increasing application of encrypted image reversible data hiding in cloud storage, military, medical, and other fields, its research has received increasing attention. A reversible data-hiding algorithm for encrypted binary images based on an improved prediction method is proposed in this paper. The image owner divides the image into non-overlapping uniform and non-uniform blocks and encrypts these blocks. The data hider encrypts the secret data and embeds it in different blocks. The receiver implements extraction or recovery functions based on different keys, where the non-uniform blocks use a combination of cross-prediction and T-shaped prediction to achieve data extraction. Simulation experiments show that the algorithm is completely reversible at low embedding rates and can restore images with minimal distortion at high embedding rates. In addition, the algorithm can achieve complete image recovery by correcting the prediction error while maintaining a high embedding rate.

Keywords: Binary Images; Encrypted Images; Image Blocking; Pixel Prediction; Reversible Data Hiding

1 Introduction

With the development of society and the progress of technology, people have paid more and more attention to data security. Data hiding technology can be selected to achieve the confidentiality of digital media [1,2,8,9,15,17]. Early algorithms tended to compromise the integrity of the original image during the process of data embedding. However, in certain special scenarios where the integrity of the original image is paramount and the carrier data cannot be tampered with, researchers proposed the concept of Reversible Data Hiding (RDH). This technology not only allows for the complete extraction of embedded data, but also the lossless recovery of the original image. Among the various methods used in reversible data hiding for gray images, lossless compression, differential expansion, and histogram shifting are the three most commonly employed approaches [3, 7, 22].

Yin proposed a RDH scheme for binary images [18], which embed data by flipping pairs of patterns with opposite central pixels. Feng [4] explored a secure binary image steganography technique based on minimizing texture distortion and enhanced the security and robustness of the steganography. Xuan [16] constructed a histogram by analyzing the lengths of consecutive pixels with the same value in an image, and then modified the histogram to achieve data embedding. Ho [5] explored a high-capacity reversible data hiding method for binary images based on pattern replacement. This method utilized the characteristics of binary images and achieved data embedding by replacing pattern blocks in the image.

In certain applications, where maintaining the confidentiality of the original image is crucial to prevent unauthorized access or tampering, users may encrypt their images before storing them in the cloud. This process, where reversible data hiding techniques are applied to the encrypted image, is known as Reversible Data Hiding in Encrypted Images (RDHEI) [6,10–12]. When dealing specifically with binary images, it is referred to as Reversible Data Hiding in Encrypted Binary Images (RDHEBI).

In general, RDHEI consists of three parts: the content owner, the data hider, and the receiver. The content owner encrypts the original image using encryption key and sends it to the data hider. The data hider embeds data into the encrypted image using data embedding key and transmits it to the receiver. Upon receiving the image, the receiver can choose to either restore the image or extract the data, using decryption key and data extraction key.

Existing RDHEI approaches can be classified into two categories: Reserving Room Before Encryption (RRBE) and Vacating Room After Encryption (VRAE). RRBE involves preprocessing the original image to create space for data embedding before encryption, leveraging the inherent redundancy in the image. VRAE, on the other

hand, directly embeds data into the encrypted image.

Zhang [20] presented the first RDHEI scheme, while the original image can only be approximately recovered. Subsequently, Zhang [21] proposed a novel separable RDHEI algorithm that enables the receiver to restore the image and extract the data separately. Yin [19] proposed a novel algorithm that utilizes pixel prediction and multi-MSB plane rearrangement to achieve the purpose of hiding data. Wu [14] explored a technique based on adaptive prediction error that securely embeds additional data into encrypted images while maintaining data reversibility.

The latest achievement in the field of RDHEBI is proposed by Ren [13], which allows for completely independent image restoration and secret data extraction. However, this method not only exhibits relatively large errors but also poses certain security concerns.

This paper proposes an improved reversible data hiding method for encrypted binary images. By dividing the image into uniform blocks (UBs) and non-uniform blocks (NUBs) and employing different strategies for data embedding, the overall embedding rate can be effectively increased. For the restoration of NUBs, a novel prediction method is utilized, which achieves more precise prediction results and improves the similarity between the restored image and the original image. Additionally, this paper employs image encryption and data embedding keys, effectively enhancing the security of the encrypted image and secret data.

2 Related Works

This section will introduce the main work of the literature [13], which consists of three parts: the rearrangement of the original image, data embedding, extraction and image restoration.

2.1 Original Image Rearranged

First, the $N * N$ size image is divided into $m * m$ size non-overlapping blocks, and got num size blocks:

$$num = \frac{N * N}{m * m} \tag{1}$$

Then, classify the blocks: if all values in a block are either completely black or completely white, this block is classified as a uniform block (UB) and labeled as 1. If there are both black and white values in the block, it is classified as a non-uniform block (NUB) and labeled as 0. Using an array of size num called Tag to store the types of all blocks sequentially, if the number of UBs is S_n, then the number of NUBs, S_{un}, is calculated by Formula (2).

$$S_{un} = num - S_n \tag{2}$$

Finally, all UBs are placed in front sequentially, followed by all NUBs in order. When restoring the image, the blocks are rearranged according to their respective types based on the Tag values.

2.2 Data Embedding, Extraction and Image Recovery for UBs

For UBs, a pixel point at the bottom right corner is selected to record the state of the block. If the block is completely black, a value of 0 is stored in the bottom right corner, and if the block is completely white, a value of 1 is stored. The remaining positions are used to embed data. Since Tag values is necessary to restore the image, the last S_y blocks among the UBs are selected to store the Tag values, where S_y is calculated using Formula (3).

$$S_y = \left\lceil \frac{num}{(m * m - 1)} \right\rceil \tag{3}$$

The data embedding capacity of UBs is MS_n, which is calculated by Formula (4).

$$MS_n = (m * m - 1) * (S_n - S_y) \tag{4}$$

During data embedding, since the first S_n blocks of the image are all UBs, data can be embedded in all but the bottom right pixel of the first $S_n - S_y$ blocks.

During data extraction, data is sequentially extracted from all but the bottom right pixel of the first $S_n - S_y$ blocks. To restore the image, only the bottom right pixel values of the first S_n blocks need to be read, and the other pixel values are restored based on those values.

2.3 Data Embedding, Extraction and Image Recovery for NUBs

When embedding data in NUBs, fixed positions are chosen to embed the data. During data extraction, the data is retrieved from those fixed positions in the same order.

During image restoration, the pixel values in NUBs are recovered using a T-shaped prediction method, which employs Formula (5) to predict the central pixel P using adjacent pixels R, S and L, as depicted in Figure 1.

$$P = \begin{cases} 0 & if \quad R + S + L = 0, 1 \\ 1 & if \quad R + S + L = 2, 3 \end{cases} \tag{5}$$

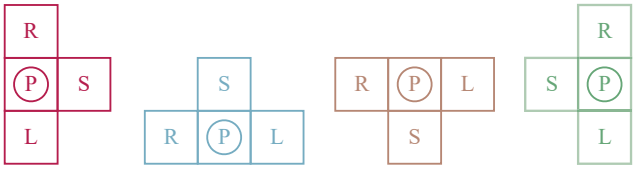

Figure 1: Four types of T-shaped models

The literature [13] and others have demonstrated that the embedding capacity can be maximized when an image is divided into 4*4-sized blocks. Each block is further divided into four parts as shown in Figure 2, and four fixed P points are used to embed data. The embedding capacity MS_{un} for all NUBs is calculated by Formula(6).

$$MS_{un} = 4 * S_{un} \tag{6}$$

The total embedding capacity MA for the entire image is calculated by Formula (7).

$$MA = MS_n + MS_{un} \tag{7}$$

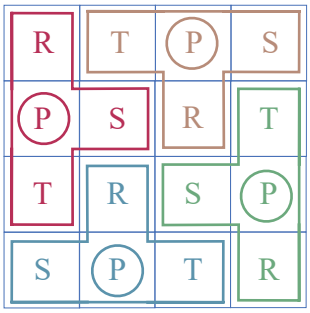

Figure 2: T-shaped prediction embedding position

3 Proposed Algorithm

This section presents a novel reversible data hiding algorithm for encrypted binary images. The algorithm consists of four components: image preprocessing, image encryption, data embedding, image restoration and data extraction, as illustrated in Figure 3.

During the image preprocessing and image encryption stages, the image owner first rearranges the UBs and NUBs, and then performs preprocessing on the UBs. When encrypting the image, the UBs and NUBs are padded and permuted separately, and then the image is encrypted using a stream cipher.

During the data embedding stage, the data hider encrypts the secret message before embedding it into the UBs and NUBs.

During the image restoration and data extraction stages, the receiver restores the image and extracts the data based on different keys separately.

This section will describe the algorithm details in six parts: (1) Image preprocessing; (2) Image encryption; (3) Data embedding, extraction, and image restoration for UBs; (4) Data embedding, extraction, and image restoration for NUBs; (5) Enhancing the security of data embedding and extraction; (6) Overall algorithm workflow.

3.1 Image Preprocessing

The image preprocessing consists of two parts: image rearrangement and UBs processing.

3.1.1 Image Rearrangement

Taking Figure 4 as an example to illustrate the image rearrangement, the image is first divided into 4*4 blocks, with UBs labeled as 1 and NUBs as 0, resulting in an image block type map. In this case, the value of Tag is 001110010, which is stored in the last S_y blocks of the UBs. Then, all the blocks labeled 1 are placed in front in order, and all the blocks labeled 0 are placed behind in order, resulting in a rearranged image where all UBs are in front and NUBs are at the back.

During image restoration, the image is divided into blocks and the receiver reads the value of Tag from the last S_y blocks of the UBs. Then the blocks are restored based on the Tag value.

3.1.2 UBs Processing

We use a key k_a to determine the prediction pixel position within a UB where the block's value is recorded. Figure 5 illustrates the order of each pixel in a 4*4 block, and a 0-1 random binary sequence r generated by k_a determines the prediction pixel position.

The sequence r is divided into groups of $\{r_i\}$, where r_i is four bits binary and the value ranges from 0 to 15.

$$r = r_1 r_2 r_3 \ldots\ldots r_n \tag{8}$$

The prediction pixel position for the i-th UB is $ri + 1$, the value of this position remains unchanged and the remaining positions can be used to embed data.

The last S_y blocks of the UBs are selected to embed the Tag value sequentially. When restoring the image, the prediction pixel positions of the last S_y UBs can be determined by k_a. The remaining bit in these blocks is then extracted sequentially to obtain the Tag value.

3.2 Image Encryption

Image encryption consists of two stages: padding and permutation, stream cipher encryption.

3.2.1 Padding and Permutation

A random sequence of 0-1 values generated by the seed key k_{e1} is used to pad in the first $S_n - S_y$ UBs (excluding the prediction pixel positions).

To enhance the security, block permutation is applied to the NUBs and the rules are as follows:

1) Let G be the set of all possible permutations on $Z_m = \{1, 2, 3, ..., m\}$, where $m = S_{un}$.
2) Taking $\pi \in G$.
3) Let the sequence of NUBs be:$a_1, a_2, a_3, ..., a_m$, where a_i is the i-th NUB.
4) Permuting all the NUBs by Formula (9):

$$e_\pi(a_1, a_2, a_3, \ldots, a_m) = \left(a_{\pi(1)}, a_{\pi(2)}, a_{\pi(3)}, \ldots, a_{\pi(m)}\right) \tag{9}$$

The original NUBs can be obtained by Formula (10), where b_i is the i-th NUB after permutation.

$$d_{\pi^{-1}}(b_1, b_2, b_3, \ldots, b_m) = \left(b_{\pi^{-1}(1)}, b_{\pi^{-1}(2)}, b_{\pi^{-1}(3)}, \ldots, b_{\pi^{-1}(m)}\right) \tag{10}$$

where π and π^{-1} are mutually inverse permutations on Z_m.

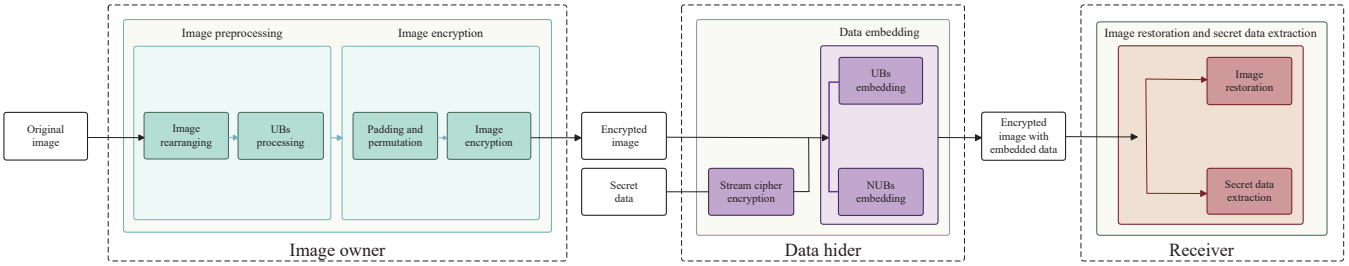

Figure 3: Algorithm framework

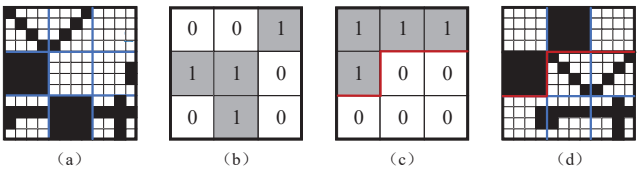

Figure 4: Image rearrangement: (a) Image block; (b) Block type; (c) Block type rearrangement; (d) The rearranged image.

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

Figure 5: The order of a 4*4 block

3.2.2 Stream Cipher Encryption

Formula (11)) employs the seed key k_{e1} to generate a random sequence $\{k_i\}$ and performs an XOR operation with the image $\{X_i\}$ to yield the encrypted image $\{E_i\}$.

$$E_i = k_i \oplus X_i \tag{11}$$

Here, $\oplus$ stands for the XOR operation. The image can be restored by Formula (12).

$$X_i = k_i \oplus E_i \tag{12}$$

3.3 Data Embedding, Extraction and Block Recovery for UBs

When embedding data, for the first $S_n - S_y$ UBs, apart from the prediction pixel bit, the remaining 15 positions can be used to sequentially embed the data.

During data extraction, for the first $S_n - S_y$ UBs, the prediction pixel bit is determined using k_a, and then the remaining bits are extracted sequentially.

When restoring UBs, the image is first decrypted. Then, using k_a, the prediction pixel bit's value is determined. After that, the remaining 15 bits of the block are restored to the same value as the prediction pixel bit.

3.4 Data Embedding, Extraction and Image Recovery for NUBs

For NUBs, 4 bits or 6 bits of data can be chosen for embedding.

3.4.1 Embedding 4-bit Data

Four fixed positions are selected for sequential data embedding, and the data are extracted from those four positions in the same order when extracting.

When restoring NUBs, the image is first decrypted and the locations of the NUBs are recovered using Formula (10). Finally, the complete NUBs are restored using the prediction algorithm proposed in this subsection.

The prediction methods are classified into two types: cross prediction and T-shape prediction. Cross prediction, as illustrated in Figure 6, utilizes the Formula (13) to predict the central pixel P based on U, A, D and B:

$$P = \begin{cases} 0 & if \quad U + A + D + B = 0, 1, 2 \\ 1 & if \quad U + A + D + B = 3, 4 \end{cases} \tag{13}$$

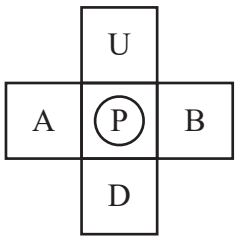

Figure 6: Cross prediction

Cross prediction is used for the two positions labeled 6 and 11, as illustrated in Figure 7.

T-shape prediction is conducted based on Formula (5). Unlike the embedding positions in reference [13], the proposed algorithm selects any two of the four positions 3, 8, 9 and 14 for data embedding, as shown in Figure 8. The number of prediction errors in these four positions may vary slightly. The two positions with the smallest number of errors can be selected for embedding.

By combining the two prediction methods, we get the final predictive method. We illustrate one such combination. If the T-shape prediction selects positions 8 and 9, we integrate it with the cross prediction and embed data in four positions 6, 8, 9 and 11, as depicted in Figure 9.

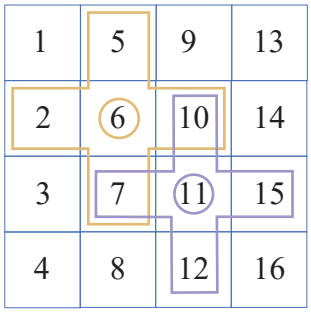

Figure 7: Cross prediction embedding position

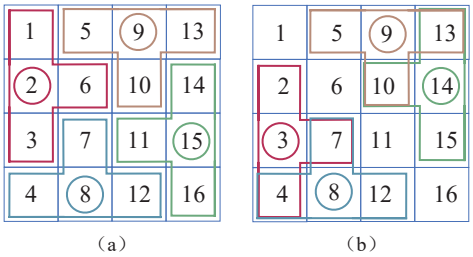

Figure 8: Comparison of embedded positions (a) Reference [13]; (b) Proposed Scheme

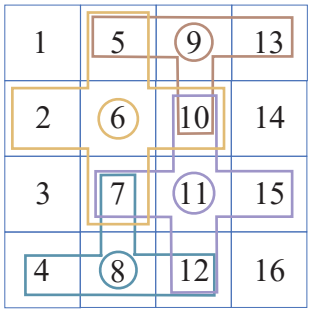

Figure 9: Embedding positions for cross prediction and T-shaped prediction

During the recovery of this NUB, the values of positions 8 and 9 are restored by Formula (5), while the values of positions 6 and 11 are recovered by Formula (13).

3.4.2 Embedding 6-bit Data

If all four positions of the T-shape prediction are chosen for data embedding and combined with the cross prediction, each NUB can embed 6 bits of data. The steps for embedding, extraction, and restoration are identical to those described in Section 3.4.1. Formula (14) is used to calculate the embedding capacity MS'_{un} of the NUBs.

$$MS'_{un} = 6 * S_{un} \tag{14}$$

Compared to embedding 4 bits of data in each NUB, the increased capacity of all NUBs can be calculated using Formula (15).

$$MS'_{un} - MS_{un} = 2 * S_{un} \tag{15}$$

3.5 Enhance Data Security with Keys

The secret data to be embedded is first encrypted using stream cipher, where a seed key k_{e2} is required to generate the stream cipher. To further enhance security, seed keys k_b and k_c are introduced to control the number of bits to be embedded during each embedding process and the interval between embedded data. Specifically:

Using the keys k_b and k_c as the seed keys for a chaotic system, a random sequence is generated. A typical example of a chaotic system is the Logistic Map:

$$Q_{n+1} = r * Q_n * (1 - Q_n) \tag{16}$$

where the parameter r is typically set to a fixed value that enables the system to produce chaotic sequences. The seed keys k_b and k_c are converted into the initial values of the sequence separately through a hash function or modulo operation. The chaotic system is then iterated repeatedly to generate the chaotic sequence. By applying a binarization process using Formula (17), the chaotic sequence can be converted into a random sequence of 0s and 1s.

$$S_i = \begin{cases} 0 & if \quad Q_i \geq 0.5 \\ 1 & if \quad Q_i < 0.5 \end{cases} \tag{17}$$

The resulting random sequence is segmtioned into decimal, denoted as $\{S_{bi}\}$ and $\{S_{ci}\}$, respectively, to determine the embedding position of the secret data. For example, if the segment length is 4, the value of each bit in the sequence ranges from 0 to 15, and the S_{ci} bit data is embedded after each interval of S_{bi} bits

3.6 Complete Algorithm Flow

In this section, the complete algorithm flow is described from three perspectives: the image owner, the data hider, and the receiver.

The image owner preprocesses and encrypts the image, as illustrated in Figure 10. Firstly, the original image

is rearranged, and the prediction pixel positions in the UBs are determined based on k_a. Then, a Tag value is embedded in the last S_y UBs. During the encryption process, for the first $S_n - S_y$ UBs, k_{e1} is used to fill in the positions other than the prediction pixels. For NUBs, π is applied to scramble their positions. Finally, the en-tire image is encrypted using k_{e1} and sent to the data hider.

The data hider performs data embedding, as depicted in Figure 11. Firstly, the data is encrypted using k_{e2}. Then, the encrypted data is embedded into both UBs and NUBs. For the first $S_n - S_y$ UBs, data are embedded using k_b and k_c outside the prediction pixel positions. For NUBs, data embedding is done directly using k_b and k_c. The encrypted image with embedded data is sent to the receiver.

The receiver performs data extraction and image restoration, as shown in Figure 12. The receiver first calculates the values of S_{un} and S_y based on S_n to determine the number of UBs and NUBs and get the Tag value.

During the image restoration process, the receiver first decrypts the image using k_{e1} and then restores the UBs and NUBs separately. For the UBs, the prediction pixel positions are identified based on k_a, the remaining positions are restored according to the prediction pixel values. For NUBs, the scrambled positions are first restored using π, then the values of the NUBs are recovered based on the prediction method. Finally, the UBs and NUBs are rearranged using Tag value, resulting in the restored image.

When extracting data, for the first $S_n - S_y$ UBs, the receiver first identifies the prediction pixel positions using k_a and then extracts the remaining valid bits of data using k_b and k_c. For NUBs, the valid bits of data are extracted using k_b and k_c. Finally, the extracted data is decrypted using k_{e2} to obtain the embedded data.

The recovery of the image and extraction of data in this algorithm are separable, meaning that they can be performed simultaneously. Only the keys k_a, k_{e1} and π are required to restore the image. To extract the data, only the keys k_a, k_b, k_c and k_{e2} are needed. Having all the keys enables both image restoration and data extraction.

4 Experimental Results and Discussion

This section evaluates the performance of the algorithm and discusses the results. The test images are binary images of 300*300 pixels, which are divided into blocks of 4*4 pixels. Data embedding is performed according to the maximum embedding capacity. The test images consist of six types of binary images, namely “CAD”, “Cartoon”, “QR Code”, “Mask”, “Texture” and “Text” with ten images for each type. Figure 13 shows one example image from each type of the test images.

4.1 Reversible Data Hiding Performance

In this section, three types of experimental results are presented: (1) Comparison of the number of errors between original images and recovered images when embedding 4-bit data into NUBs, using the method in reference [13] and this paper. (2) Comparison of the image embedding capacity when embedding 4-bit data and 6-bit data, respectively, into NUBs. (3) Performance analysis of fully reversible data hiding.

4.1.1 Experimental Results of Embedding 4-bit

Figure 14 presents the comparison results between the test images and the recovered images. (a) shows one original test image for each type. (b) displays the images recovered using the algorithm proposed in this paper. (c) is the error comparison diagram be-tween (a) and (b). It is evident that most differences exist in the contours.

Let P_E denote the number of errors caused by the algorithm in this paper, and O_E denote the number of errors caused by the algorithm in [13]. The reduced error rate RE is obtained by Formula (18).

$$R_E = \frac{(O_E - P_E)}{O_E} \tag{18}$$

Table 1 lists the maximum R_E for each image type and the corresponding number of errors.

Table 1: **Comparison of maximum error rate**

	Error number in [13]	Error number in proposed algorithm	Maximum R_E
CAD	630	341	45%
Cartoon	1461	1253	14%
QR Code	40	31	22%
Mask	117	89	23%
Texture	568	501	11%
Text	1137	961	15%

Table 1 shows that the maximum reduction in error achieved by the proposed algorithm can be up to 45%. Furthermore, based on the average values of errors generated by the algorithm in [13] and the proposed algorithm, the average error reduction rate for each image type is calculated and presented in Table 2.

The data in Table 2 indicate that the errors for each type of image have been reduced.

4.1.2 Experimental Results of Embedding 6-bit

The algorithm proposed in this paper can embed up to 6-bit data into NUBs, and the embedding capacity MA' can be obtained using Formula (19).

$$MA^{'} = MS_n + 6 * S_{un} \tag{19}$$

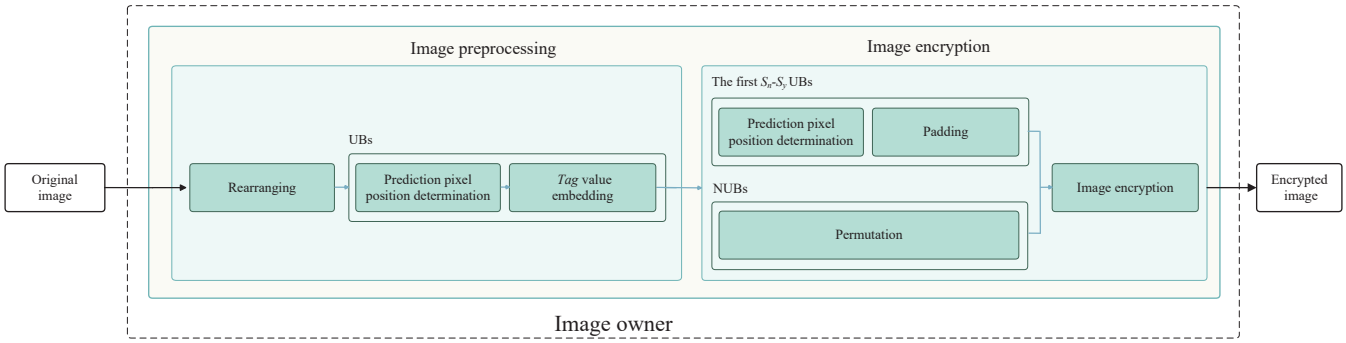

Figure 10: The Process Flow Diagram by Image owner

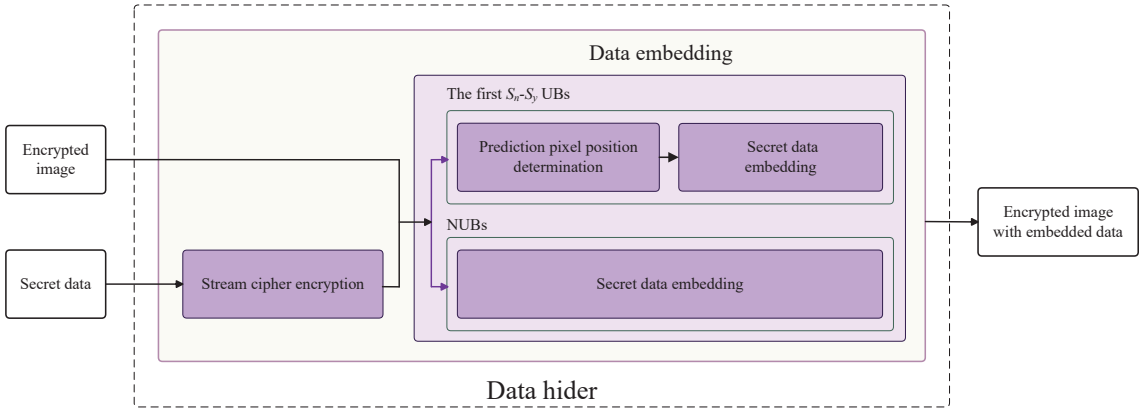

Figure 11: The Process Flow Diagram by Data hider

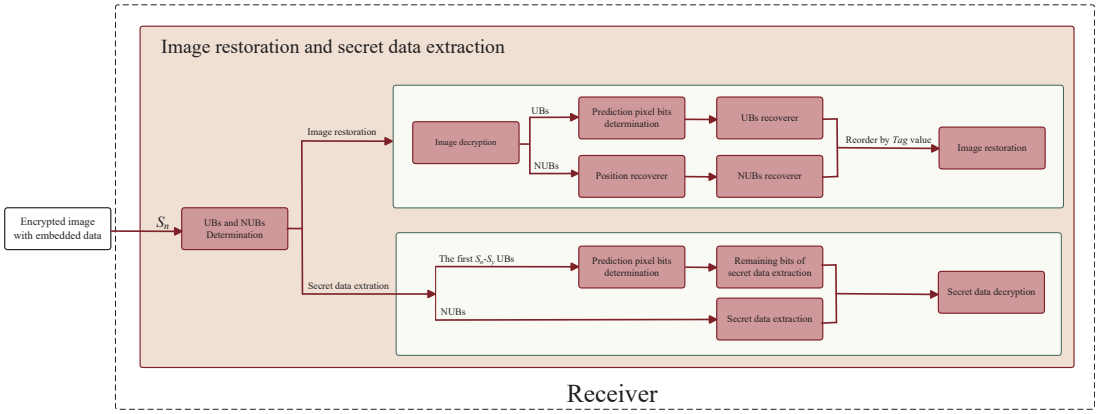

Figure 12: The Process Flow Diagram by Receiver

Table 2: **Comparison of average error rate**

	Average error number in [13]	Average error number in proposed algorithm	Average R_E
CAD	663	584	11%
Cartoon	830	799	3%
QR Code	63	58	7%
Mask	124	122	1 %
Texture	1219	1190	2%
Text	1031	923	10%

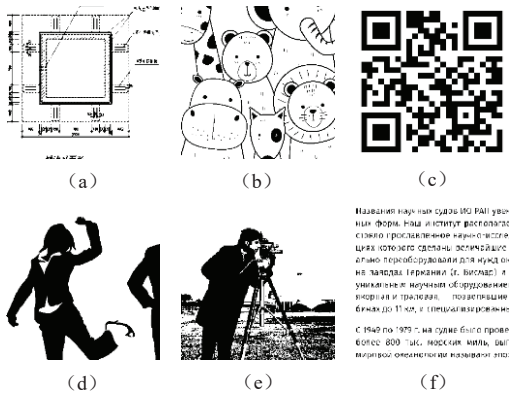

Figure 13: 6 kinds of test images: (a) CAD; (b) Cartoon; (c) QR Code; (d) Mask; (e) Texture; (f) Text

To compare the degree of improvement of embedding capacity, the embedding capacity improvement rate R_{EC} is obtained using Formula (20).

$$R_{EC} = \frac{\left(MA^{'} - MA\right)}{MA} \tag{20}$$

Using Formula (7) and Formula (19), we calculate the capacity sizes of the test images when embedding 4-bit and 6-bit data, respectively. Then, we determine the average capacity values for each image type when embedding 4-bit and 6-bit data. Finally, we compute the R_{EC} value for each type and summarize them in Table 3. It can be observed that the average embedding capacity of all six types has increased. Among them, the embedding capacity enhancement of texture and text types is the most significant.

Table 3: **Comparison of embedding capacity**

	Embedded 4-bit average capacity	Embedded 6-bit average capacit	R_{EC}
CAD	73441	76201	4%
Cartoon	71128	74274	4%
QR Code	74288	76907	4%
Mask	85647	86569	1%
Texture	65312	69427	6%
Text	65728	69774	6%

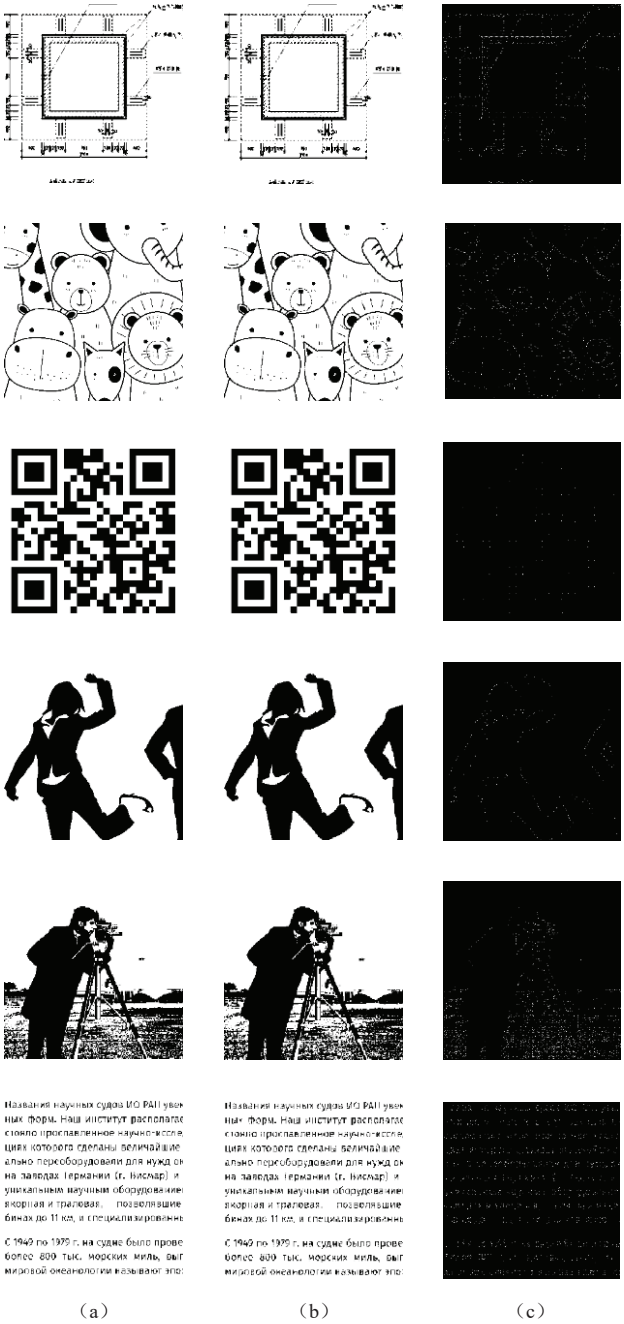

Figure 14: Comparison between original images and restored images: (a) The original image; (b) The recovered image; (c) The difference between these images.

4.1.3 Fully Reversible Data Hiding Analysis

The distortion generated by this scheme originates from the prediction error of P in NUBs. This section provides an analysis of the fully reversible data hiding.

When there is less secret information, it is optional to embed the information only in UBs. After obtaining the size of the embedded data in all UBs, MS_n, through Formula (4), the maximum embedding rate (EMS_n) for UBs can be calculated by Formula (21).

$$EMS_n = \frac{MS_n}{N * N} \tag{21}$$

Table 4 presents the average maximum embedding rates for different types of UBs. When the required embedding rate is less than the maximum embedding rate of the UBs, embedding data only in the UBs allows for complete reversible recovery of the image.

When there is a large amount of secret information, the correction error sequence of NUBs can be embedded into the UBs. Based on the prediction errors in NUBs, 0 and 1 are used to represent whether the prediction results are correct or not, generating a correction error sequence W. This sequence is then compressed to obtain W', which is stored in the UBs. When restoring the image, W' is used to correct the prediction errors in NUBs, achieving complete reversibility. Assuming that $l_{W'}$ represents the length of W', under the condition of embedding 4-bit data, a new embedding capacity MA_c can be obtained using Formula (22).

$$MA_c = MA - l_{w'} \tag{22}$$

Table 4: **Maximum embedding rate for UBs**

	Maximum embedding rate of UBs
CAD	65%
Cartoon	61%
QR Code	66%
Mask	82%
Texture	53%
Text	54%
Average value	64 %

According to Table 2, the number of prediction errors in this paper is less than that in [13], therefore, under the condition of complete reversibility, the embedding capacity of this paper is greater than that in [13].

Furthermore, the average restoration degree R of each type of image is calculated by Formula (23) and the results are shown in Table 5.

$$R = \frac{N * N - O_E}{N * N} \tag{23}$$

As can be seen from Table 5, even if complete reversibility is not required, the distortion of the proposed scheme in this paper is extremely small, so a choice can be made between reversibility and complexity according to the requirements.

Table 5: **Average recovery degree**

	Embedding 4-bit average restoration	Embedding 6-bit average restoration
CAD	99.35%	98.97 %
Cartoon	99.11%	98.64%
QR Code	99.94%	99.90%
Mask	99.86%	99.82%
Texture	98.68%	97.98%
Text	98.97%	98.38%

4.2 Security Analysis

This section analyzes the security of the algorithm from the key space and evaluates the visual quality of the restored image compared to the original image.

4.2.1 Key Space Analysis

This algorithm utilizes six keys: k_a, k_b, k_c, k_{e1}, k_{e2} and π. Figure 15 demonstrates the function of each key. Extracting secret data alone requires k_a, k_b, k_c and k_{e2}. Restoring the image alone requires k_a, k_{e1} and π. Extracting secret data and restoring the image simultaneously requires all six keys.

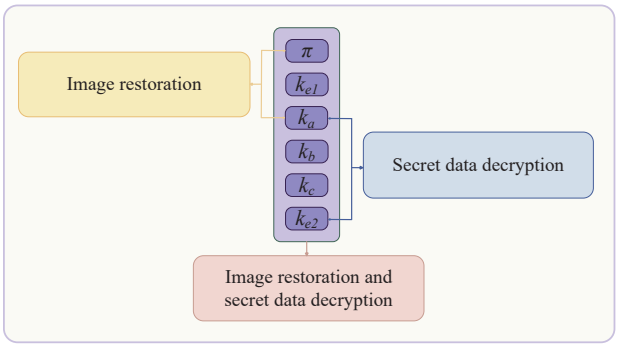

Figure 15: Key function diagram

The lengths of the keys k_a, k_b, k_c, k_{e1} and k_{e2} are generally not less than 128 bits, meaning that each key has at least 2^{128} possible values, and the key spaces of these five keys are mutually independent. The key space required for extracting secret data is 2^{512}, while the key space needed for restoring the image is 2^{256}. The key space required for simultaneously extracting secret data and restoring the image is 2^{640}. Additionally, the size of π is related to the number of NUBs. When the number of NUBs is S_{un}, there are $S_{un}!$ possible values for π. Experimental results show that the typical value of S_{un} is generally greater than 1000, making it nearly impossible to crack via brute force. Additionally, as these six keys are randomly generated and not correlated with each other, they are completely independent and diverse. This means that even if one key is cracked via brute force, the other keys remain secure.

Furthermore, the keys k_b and k_c are used to generate random sequences $\{S_{bi}\}$ and $\{S_{ci}\}$ utilizing chaotic sys-

tems. Since chaotic systems are extremely sensitive to initial conditions (i.e., the key seeds k_b and k_c), this means that even minor changes in the keys k_b and k_c will result in completely different $\{S_{bi}\}$ and $\{S_{ci}\}$, further increasing the difficulty of cracking.

4.2.2 Visual Quality Analysis

Figure 16 presents a comparison between the original images and the encrypted image after embedding data. Specifically, (a) and (c) are the original images, while (b) and (d) are the images after image encryption and secret data embedding using the proposed algorithm.

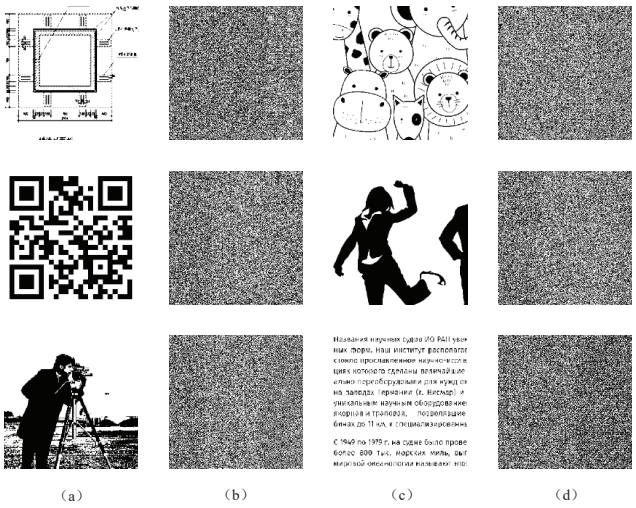

Figure 16: Original image and encrypted image after embedding data. (a) The original image; (b) Encrypted image after embedding data; (c) The original image; (d) Encrypted image after embedding data.

The PSNR, SSIM values for each pair of images in the test set were calculated, and the average PSNR, SSIM values for each type of image and the overall average PSNR,SSIM value are presented in Table 6. It can be observed that the PSNR values of the test images fluctuate around 3.4 and the SSIM values of the test images fluctuates around 0.0030, indicating a minimal correlation. Therefore, it is highly unlikely for an attacker to analyze and extract the original image features directly from the processed images.

Table 6: **PSNR and SSIM values**

	PSNR	SSIM
CAD	3.4	0.00375
Cartoon	3.4	0.00352
QR Code	3.0	0.00407
Mask	3.2	0.00224
Texture	3.0	0.00302
Text	3.6	0.00391
Average value	3.2	0.00342

5 Conclusions

This paper proposes an improved reversible data hiding algorithm for encrypting binary images. The image blocks are classified into UBs and NUBs, and different embedding strategies are applied in different blocks. A novel combined method of cross prediction and T-shaped prediction is designed for the restoration of NUBs, which reduces the distortion of the restored image under the same embedding capacity.

The algorithm comprehensively utilizes multiple encryption methods, which not only enhances the concealment of the original image but also improves the security of the embedded data. Moreover, the algorithm proposes a method to increase the embedding capacity, allowing the embedding of 6-bit information in NUBs. Experimental results show that under the condition of equal block division, the embedding capacity is significantly improved. Furthermore, a prediction error correction method is presented, which enables the complete reversible recovery of the original image.

Acknowledgments

This paper was supported by the National Nature Science Foundation of China (Program No. 62202377), the Natural Science Basic Research Plan of Shaanxi Province of China (Program No. 2024JC-YBMS-524, 2022JM-353), the Key Research and Development Program of Shaanxi (Program No. 2023-YBGY-015), the scientific Research Program Funded by Shaanxi Provincial Education Department (No. 22JK0560). The authors gratefully acknowledge the anonymous reviewers for their valuable comments.

References

[1] Ambika and Virupakshappa, "Applicable techniques for image steganography: A surveyn," in *2023 International Conference on Computer Communication and Informatics (ICCCI)*, pp. 1–7, Dalian, China, 2023.

[2] M. Anna and E. Oleg, "Image data hiding schemes based on metaheuristic optimization: a review," *Artificial Intelligence Review*, vol. 56, no. 12, pp. 15375–15447, 2023.

[3] M. Duevedi and S.K. Muttoo, "An improved separable and reversible steganography in encrypted grayscale images," *International Journal of Information Security and Privacy (IJISP)*, vol. 15, no. 2, pp. 1–28, 2021.

[4] B. W. Feng, W. Lu, and W. Sun, "Secure binary image steganography based on minimizing the distortion on the texture," *IEEE transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 243–255, 2014.

[5] Y. A. Ho, Y. K. Chan, H. C. Wu, and Y. P. Chu, "High-capacity reversible data hiding in binary images using pattern substitution," *Computer Standards & Interfaces*, vol. 31, no. 4, pp. 787–794, 2009.

[6] I. Jafar, K. A. Darabkh, and F. Jubair, "Separable high capacity reversible data hiding algorithm for encrypted images.," *Int. Arab J. Inf. Technol.*, vol. 19, no. 5, pp. 812–821, 2022.

[7] S. A. Jebur, A. K. Nawar, L. E. Kadhim, and M. M. Jahefer, "Hiding information in digital images using lsb steganography technique.," *International Journal of Interactive Mobile Technologies*, vol. 17, no. 7, 2023.

[8] X. Liu, Q. M. Wu, Z. Zuo, Z. L. Yang, H. Y. Zhang, L. C.Dai, Y. Liao, X. W. Li, and C. Y. Zhang, "3d image steganography using cellular automata transform and depth estimation network," *Optics Communications*, vol. 550, p. 129936, 2024.

[9] N. X. Mao, H. J. He, F. Chen, P. Bellavista, and Y. L. Yang, "Reversible data hiding of jpeg images based on block sorting and segmented embedding," *Biomedical Signal Processing and Control*, vol. 87, p. 105555, 2024.

[10] K. L. Qi, M. Q. Zhang, and F. Q. Diand T. J. Kong, "High capacity reversible data hiding in encrypted images based on adaptive quadtree partitioning and msb prediction," *Frontiers of Information Technology & Electronic Engineering*, vol. 24, no. 8, pp. 1156–1168, 2023.

[11] A. Rai, H. Om, and S. Chand, "High capacity reversible data hiding in encrypted images using prediction error encoding," *Multimedia tools and applications*, pp. 25883–25898, 2023.

[12] F. Ren, Z. Y. Wu, Y. Q. Xue, and Y. L. Hao, "Reversible data hiding in encrypted image based on bit-plane redundancy of prediction error," *Mathematics*, vol. 11, no. 11, p. 2537, 2023.

[13] H. L. Ren, W. Lu, and B. Chen, "Reversible data hiding in encrypted binary images by pixel prediction," *Signal Processing*, vol. 165, pp. 268–277, 2019.

[14] X. S. Wu, T. Qiao, M. Xu, and N. Zheng, "Secure reversible data hiding in encrypted images based on adaptive prediction-error labeling," *Signal Processing*, vol. 188, p. 108200, 2021.

[15] X. G. Xiong, S. Y. Zhong, and Y. Lu, "Separable and reversible data hiding scheme for medical images using modified logistic and interpolation," *Biomedical Signal Processing and Control*, vol. 87, p. 105521, 2024.

[16] G. R. Xuan, Y. Q. Shi, P. Q. Chai, X. F. Tong, J. Z. Teng, and J. Li, "Reversible binary image data hiding by run-length histogram modification," in *2008 19th International Conference on Pattern Recognition*, pp. 1–4. IEEE, 2008.

[17] T. F. Yang, Z. Q. Liu, J. J. Guo, Y. Yu, F. Ren, and T. Wang, "Image analysis by fractional-order weighted spherical bessel-fourier moments," *Pattern Recognition*, vol. 157, p. 110872, 2025.

[18] X. L. Yin, W. Lu, W.T. Liu, and J. H. Zhang, "Reversible data hiding in binary images by symmetrical flipping degree histogram modification," in *Security with Intelligent Computing and Big-data Services: Proceedings of the Second International Conference on Security with Intelligent Computing and Big Data Services (SICBS-2018) 2*, pp. 891–903. Springer, 2020.

[19] Z. X. Yin, N. Xu, F. Wang, L. L., and B. Luo, "Separable reversible data hiding based on integer mapping and multi-msb prediction for encrypted 3d mesh models," in *Pattern Recognition and Computer Vision: 4th Chinese Conference, PRCV 2021, Beijing, China, October 29–November 1, 2021, Proceedings, Part II 4*, pp. 336–348. Springer, 2021.

[20] X. P. Zhang, "Reversible data hiding in encrypted image," *IEEE signal processing letters*, vol. 18, no. 4, pp. 255–258, 2011.

[21] X. P. Zhang, "Separable reversible data hiding in encrypted image," *IEEE transactions on information forensics and security*, vol. 7, no. 2, pp. 826–832, 2011.

[22] L. N. Zhou, Z. G. Lu, W. K. You, and X. F. Fang, "Reversible data hiding using a transformer predictor and an adaptive embedding strategy," *Frontiers of Information Technology & Electronic Engineering*, vol. 24, no. 8, pp. 1143–1155, 2023.

Biography

Fang Ren received the Ph.D. degree in cryptography from Xidian University in 2012. Now he is an associate professor of Xi'an University of Posts and Telecommunications. His current research interests include reversible data hiding and information security.

Mengmeng Zeng received the B.S. degree from Xi'an University of Posts and Telecommunications in 2022. She is currently going in for the M.S. degree in information security with Xi'an University of Posts and Telecommunications. Her research interests concentrate on reversible data hiding and image security.

Zifan Wang is currently studying at Xi'an University of Posts and Telecommunications for his undergraduate degree. Her research interests are focused on the fields of network and information security.

Xuan Shi received the B.S. degree from Xi'an University of Posts and Telecommunications in 2022. She is currently going in for the M.S. degree in cyberspace security with Xi'an University of Posts and Telecommunications. Her research interests concentrate on data hiding and cryptography.