An Improved Reversible Data Hiding in Encrypted Image Method Based on Block Selection

YAOMIN WANG, GANGQIANG XIONG, AND WENGUANG HE
School of Biomedical Engineering, Guangdong Medical University, Zhanjiang 524023, China
Corresponding author: Wenguang He (56207403@qq.com)

ABSTRACT For reversible data hiding in encrypted image (RDHEI), the most important problem is the conflict between reversibility and embedding capacity. Reversibility has been realized by existing methods, however, the embedding capacity is still too low for practical RDHEI applications. To settle this problem, an improved RDHEI method is proposed in this article by presenting a block selection mechanism. Since extraction error occurs on blocks from which the fluctuation function fails to identifies the flipped part, we propose to distinguish these exceptional blocks from extraction-correct-block (ECB) (i.e., pixel block from which the hidden data can be exactly extracted) and embed additional data into only ECBs. In this way the minimum block size can be adopted to pursue high capacity. To guarantee reversibility, a new self-embedding method is proposed to embed the auxiliary information into the cover image. Finally, a new fluctuation function is also proposed to provide more ECBs and thus further improve the embedding capacity. Experimental results show that the proposed method not only achieves significant improvement in capacity but also guarantees lossless data extraction and perfect image recovery.

INDEX TERMS Image encryption, reversible data hiding, real reversibility, multimedia security.

I. INTRODUCTION
Reversible data hiding (RDH) is a technique that aims to reversibly embed additional data into distortion-unacceptable medium so that the cover medium can be restored after data extraction [1]–[3]. Due to the reversibility, RDH has been applied to many sensitive fields such as medical, military and law imagery [4], [5]. So far, a large number of RDH methods have been proposed [1]–[4], [6]–[15] and they are mainly classified into three categories [16], i.e., difference expansion based methods [7]–[9] embed additional data by expanding the difference between adjacent pixels, histogram shifting based methods [10]–[13] modify the gray value of pixels using the histogram shift mechanism, lossless compression based methods [6], [14], [15] create the embedding space through compressing a feature set of the cover image.

More and more images are stored on a cloud storage server nowadays with the rapid development of cloud computation [17]. For privacy protection, the image owner usually encrypts the images before transmission. At the cloud storage server, additional message such as the information of the image owner or other notation information may be embedded for the needs of management, even though the original content is unknown [18]. As the entropy of the image is greatly improved after encryption, traditional RDH methods cannot be applied [19], [20] and reversible data hiding in encrypted images (RDHEI) becomes desirable. Existing RDHEI methods can be classified into joint RDHEI [21]–[26] and separable RDHEI [2]–[5], [27]–[33]. For the former, the recipient who wants to extract the hidden data must have both the decryption key and the data-hiding key. In other words, the cover image is recovered with the hidden data extracted. For the latter, the hidden data can be extracted with only the data-hiding key and a directly-decrypted image which is highly similar to the cover image can also be obtained with only the decryption key. That is to say, data extraction and image decryption are separable.

RDHEI was firstly proposed by Zhang [21] with the idea of exploiting the spatial correlation of pixels to realize data embedding and extraction. After encrypted image partition, additional bit is embedded by flipping the three least-significant-bits (LSBs) of half of pixels within block. In this
way, image recovery and data extraction can be realized with a fluctuation function which identifies the flipped part. Following Zhang’s work, a new fluctuation function and side match technique which considers neighboring blocks were proposed to reduce the error rate of data extraction [22]. In [23] block smoothness is calculated by applying specific approaches for various pixels including vertex pixel, edge pixel and internal pixel. Moreover, data embedding ratio is also considered so as to further reduce the error rate. [24] similarly presents a new fluctuation function for the reduction of the error rate. To improve the embedding capacity, Bhardwaj and Aggarwal [25] proposed a new RDHEI method by segmenting each non-overlapping pixel block into $n$ sub-blocks such that $n \times n$ bits can be embedded.

For the aforementioned methods [21]–[25], the block size must be large enough to guarantee reversibility for the reason that all blocks are used to carry additional data. In this way the embedding capacity benefits little from the improvement of fluctuation function. In this article, to achieve high capacity we propose to embed data into only extraction-correct-block fluctuation function. In this article, to achieve high capacity we propose to embed data into only extraction-correct-block fluctuation function. In this article, to achieve high capacity we propose to embed data into only extraction-correct-block fluctuation function.

The contributions of this article can be summarized as follows:

- A new block selection mechanism. Since the ECBs can be exactly distinguished at encoder or decoder, real reversibility is guaranteed by embedding data into only ECBs and the minimum block size (i.e., $2 \times 2$) can be adopted to pursue high capacity.
- A new fluctuation function. With the proposed fluctuation function, the fluctuation value of pixel block can be better evaluated such that more ECBs are obtained. In this way the embedding capacity is further improved.
- A new self-embedding mechanism. The three LSBs are flipped for additional data embedding while the five most-significant-bits (MSBs) are used to carry the position file which is required for distinguishing ECBs at decoder. With the proposed self-embedding mechanism, the position file can be exactly extracted and the MSBs can be perfectly recovered.

The remainder of this article is organized as follows. Section II briefly reviews the related works. Section III describes the detailed procedures of the proposed method. The experimental results and the comparison with the prior arts are shown in Section IV. Finally, Section V concludes this article.

II. RELATED WORKS

Fig. 1 provides the general sketch of the previous methods [21]–[25]. As it is shown there are three participants. First, the cover image is encrypted by the content owner with the encryption key. Then, the data hider embeds additional data into the encrypted image with the data-hiding key even knowing nothing about the original content. Finally, the authorized recipient who has both the decryption key and the data-hiding key decrypts the image and then extracts the hidden data and restores the cover image.

The previous methods [21]–[25] share the same encryption and embedding procedures. A stream cipher encryption method is used to encrypt the cover image. Then, the encrypted image is segmented into a lot of non-overlapping blocks. For each $s \times s$ sized block, pixels are pseudo-randomly divided into two equal sets denoted by $S_0$ and $S_1$ according to the data-hiding key. If the to-be-embedded bit is 0, flip the three LSBs of pixels in $S_0$. Otherwise, flip those in $S_1$. During data extraction and image recovery, two new blocks denoted by $H_0$ and $H_1$ are firstly obtained by flipping the three LSBs of pixels in $S_0$ or those in $S_1$. Then, to determine either $H_0$ or $H_1$ is the original block and the hidden bit is 0 or 1, a fluctuation function is designed in [21] as

$$f = \sum_{u=2}^{s-1} \sum_{v=2}^{s-1} p_{u,v} - \frac{p_{u,v+1} + p_{u,v-1} + p_{u+1,v} + p_{u-1,v}}{4}$$

where $u, v$ indicate the pixel position. Taking into account that one of $H_0$ and $H_1$ must be the original block while the other one is more seriously interfered, the fluctuation values of $H_0$ and $H_1$, $f_0$ and $f_1$, are compared. If $f_0 < f_1$, $H_0$ is taken as the original block and the extracted bit is 0. Otherwise, $H_1$ is taken as the original block and the extracted bit is 1.

Zhang’s fluctuation function considers only non-boundary pixels. As shown in Fig. 2, for a $4 \times 4$ sized block only four pixels are employed while the other twelve ones are ignored. This may lead to precision loss of the fluctuation function.
By summing the absolute differences between two consecutive pixels, an improved fluctuation function is proposed in [22] as

\[ f = \sum_{u=1}^{s} \sum_{v=1}^{s-1} |p_{u,v} - p_{u,v+1}| + \sum_{u=1}^{s-1} \sum_{v=1}^{s} |p_{u,v} - p_{u+1,v}|. \] (2)

With the idea of multiple neighboring pixels, another fluctuation function is proposed in [23] by dividing pixels into vertex pixel, edge pixel and internal pixel. For these three types of pixels, the sum of absolute mean differences is respectively calculated as

\[ f_1 = \left| p_{1,1} - \frac{p_{1,2} + p_{2,1}}{2} \right| + \left| p_{1,2} - \frac{p_{1,1} + p_{2,2}}{2} \right| \]
\[ + \left| p_{2,1} - \frac{p_{3-1,1} + p_{3,2}}{2} \right| + \left| p_{3,2} - \frac{p_{3-1,2} + p_{3,3-1}}{2} \right|, \] (3)

\[ f_2 = \sum_{v=2}^{s-1} (p_{1,v} - p_{1,v-1} + p_{1,v+1} + p_{2,v}) \]
\[ + (p_{v,v-1} - p_{v,v+1} + p_{v,4}) \]
\[ + \sum_{u=2}^{s-1} (p_{u,1} - p_{u-1,1} + p_{u+1,1} + p_{u,2}) \]
\[ + (p_{u,u-1} - p_{u,u+1} + p_{u,3-1})], \] (4)

\[ f_3 = \sum_{u=2}^{s-1} \sum_{v=2}^{s-1} (p_{u,v} - p_{u-1,v}) + (p_{u,u+1} + p_{u+1,u} + p_{u-1,v} + p_{u,v-1}) \] \[ + (p_{u,v} - p_{u,v-1}) + (p_{u,v} - p_{u,v+1})]. \] (5)

Overall, the whole summation is calculated as

\[ F = f_1 + f_2 + f_3. \] (6)

This fluctuation function considers only non-boundary pixels just like Zhang’s, which is also adopted in [25]. On this basis, Bhardwaj et al. proposed to segment pixel block into \( n (n = 2, 3, 4, \ldots) \) sub-blocks each of which carries one additional bit.

Table 1 presents the embedding capacity comparison of methods [21]–[25], where ten 512 × 512 sized standard test images from [34] are selected. One can see that the embedding capacity of [21] is generally quite limited, especially on the images Baboon, House, Stream and F16. Although slight improvement has been achieved by improved methods [22]–[25], it is still far from sufficient for practical applications.

| Image     | [21] | [22] | [23] | [24] | [25] |
|-----------|------|------|------|------|------|
| Lena      | 1024 | 1024 | 1764 | 1024 | 1875 |
| Baboon    | 196  | 324  | 289  | 324  | 300  |
| Peppers   | 1296 | 2601 | 1296 | 1296 | 1875 |
| House     | 100  | 100  | 100  | 100  | 432  |
| Boat      | 1024 | 1764 | 1296 | 1204 | 1875 |
| Splash    | 1024 | 1024 | 1024 | 1024 | 1875 |
| Stream    | 81   | 121  | 144  | 81   | 162  |
| Tank      | 1764 | 1764 | 1764 | 1764 | 2352 |
| Lake      | 784  | 1024 | 784  | 1024 | 1323 |
| F16       | 196  | 196  | 196  | 196  | 1250 |

III. PRELIMINARY

In this section, the principle of the proposed RDHEI method is first presented, followed by the details of ECB selection. Next, the self-embedding method and the new fluctuation function are sequentially presented.

A. THE PRINCIPLE OF THE PROPOSED RDHEI METHOD

The proposed method is inspired by the association among embedding capacity, block size and fluctuation function. Firstly, the fluctuation function determines the extraction performance. With specific fluctuation function, the extraction error rate simply depends on block size. Fig. 4 shows the comparison of extraction error rate on the test image “Man”. It is seen that when block size is set as \( 4 \times 4 \) the extraction error rate is 16.1%, 8.7%, 9.1%, 13.1% and 29.3% for methods [21]–[25] respectively. As the block size increases, the extraction error rate generally decreases. Therefore, to achieve real reversibility with zero extraction error rate, there are always minimum block size and corresponding maximum embedding capacity for every image.

To better illustrate the extraction error rate, Fig. 5 shows the recovery result using the block size of \( 4 \times 4 \), where blocks except ECBs are marked by white. It is seen that there is quite small white area, which corresponds to high percentage of ECBs in all blocks. Table 2 presents the numbers of ECBs.
FIGURE 4. The comparison of extraction error rate using the test image "Man."

derived from methods [21]–[24] using various block sizes. Since the number of ECBs is mainly determined by the fluctuation function, [25] is ignored here due to the same fluctuation function as [21]. One can see that the number of ECBs significantly increases with the decrease of block size. On the image “Lena”, the minimum block size guaranteeing real reversibility is 16 × 16 for [21], [22] and [24], and 12 × 12 for [23]. Although all blocks are ECB when using the minimum block size, however, the derived ECBs are much fewer than that derived from smaller block size. For example, the number of ECBs reaches up to 13264 on the image “Lena” using the block size of 4 × 4 in [21]. Therefore, besides the improvement of fluctuation function, how to distinguish ECBs and embed additional data into them only is quite worthy of investigating.

TABLE 2. Comparison of the numbers of ECBs derived from methods [21]–[24] using various block sizes.

| Image  | method | Block size |
|--------|--------|------------|
|        |        | 4          | 8          | 12         | 16          | 20          | 24          | 28          |
| Lena   | [21]   | 13264      | 4034       | 1760       | 1024       | 625         | 441         | 324         |
|        | [22]   | 15100      | 4079       | 1763       | 1024       | 625         | 441         | 324         |
|        | [23]   | 14982      | 4071       | 1764       | 1024       | 625         | 441         | 324         |
|        | [24]   | 14287      | 4048       | 1762       | 1024       | 625         | 441         | 324         |
| Baboon | [21]   | 10704      | 3414       | 1623       | 969        | 606         | 435         | 323         |
|        | [22]   | 12133      | 3688       | 1684       | 993        | 617         | 440         | 324         |
|        | [23]   | 11985      | 3574       | 1640       | 982        | 606         | 438         | 324         |
|        | [24]   | 11640      | 3562       | 1654       | 994        | 616         | 439         | 324         |
| Stream | [21]   | 10185      | 3303       | 1559       | 938        | 590         | 424         | 316         |
|        | [22]   | 11282      | 3404       | 1564       | 933        | 580         | 422         | 313         |
|        | [23]   | 11912      | 3491       | 1588       | 957        | 599         | 430         | 316         |
|        | [24]   | 11028      | 3298       | 1539       | 127        | 575         | 416         | 310         |
| Splash | [21]   | 13179      | 3991       | 1744       | 1024       | 625         | 441         | 324         |
|        | [22]   | 15166      | 4060       | 1756       | 1024       | 625         | 441         | 324         |
|        | [23]   | 14956      | 4004       | 1744       | 1024       | 625         | 441         | 324         |
|        | [24]   | 14470      | 4019       | 1749       | 1024       | 625         | 441         | 324         |

B. ECB SELECTION AND IMAGE RECONSTRUCTION

After dividing the $M \times N$ sized cover image into a lot of non-overlapping blocks with the size of $s_1 \times s_2$, the fluctuation value is first calculated for each pixel block. Then, all pixels are flipped to calculate another fluctuation value.

FIGURE 5. Comparison of recovery results: (a), (f), (k),(p) are of [21]; (b),(g),(l),(q) are of [22]; (c),(h),(m),(r) are of [23]; (d)(i)(n)(s) are of [24]; (e)(j)(o)(t) are of [25].
difference between it and the original one. Let $F_{i,j}$ denote the difference, pixel blocks with $F_{i,j} < 0$ will be selected as ECB.

To distinguish ECBs from others after data hiding, a $[M/s_1] \times [N/s_2]$ sized binary matrix is generated as a position file (PF) where a bit 1 represents that the block is ECB while a bit 0 represents that the block is not. After that, all pixel blocks are collected in raster-scan order to achieve rearrangement such that ECBs are always in front of others. As shown in Fig. 6, zone ECB denote the collection of ECBs while zone NECB denote the collection of other pixel blocks.

C. SELF-EMBEDDING

The generated PF should be embedded into the reconstructed image. As shown in Fig. 7, by decomposing each pixel into the so-called spare bits (i.e., five MSBs) and loading bits (i.e., three LSBs), the reconstructed image is decomposed into two sub-images denoted by $I_M$ and $I_L$, i.e., $I_M$ collects the spare bits while $I_L$ collects the loading bits. In this way, pixel value falls in $[0,31]$ in $I_M$ while pixel value falls in $[0,7]$ in $I_L$.

1) POSITION FILE EMBEDDING

In this article, the block median preservation method [35] is adopted to embed PF into $I_M$. For each pixel block, all pixels are sorted in ascending order to obtain $p_1(i,j) \leq p_2(i,j) \leq \ldots \leq p_{s_1 \times s_2}(i,j)$. Then, the median $p_m(i,j)$ is obtained as

\[ p_m(i,j) = p\left[ \frac{s_1 \times s_2 + 1}{2} \right](i,j). \tag{8} \]

Next, the differences between $p_m(i,j)$ and the other $s_1 \times s_2 - 1$ pixels are calculated as

\[ d_k(i,j) = p_k(i,j) - p_m(i,j) \tag{9} \]

where $1 \leq k \leq s_1 \times s_2$ and $k \neq m$. By collecting $s_1 \times s_2 - 1$ differences of each pixel block, the difference histogram is thus constructed and PF will be embedded by histogram shifting. Specifically, the bins in the range of $[−2EL−1, −EL−1]$ and $[EL+1, 2EL+1]$ are first emptied as

\[ d'_k(i,j) = \begin{cases} d_k(i,j) + EL + 1 & \text{if } d_k(i,j) > EL \\ d_k(i,j) - EL - 1 & \text{if } d_k(i,j) < -EL \\ d_k(i,j) & \text{otherwise} \end{cases} \tag{10} \]

After that, PF is embedded into bins in the range of $[−EL, EL]$. For example, bins in the range of $[−EL, 0]$ and $(0, EL]$ are shifted as

\[ d'_k(i,j) = \begin{cases} 2 \times d_k(i,j) + W & \text{if } d_k(i,j) \in (0, EL] \\ 2 \times d_k(i,j) - W & \text{if } d_k(i,j) \in [−EL, 0). \end{cases} \tag{11} \]

where $W$ is a to-be-embedded bit. Accordingly, the pixel is modified as

\[ p'_k(i,j) = d'_k(i,j) + p_m(i,j). \tag{12} \]

2) OVERFLOW/UNDERFLOW PREVENTION

Overflow/underflow would occur if $p'_k(i,j)$ out of the range of $[0,31]$. Therefore, a $[M/s_1] \times [N/s_2]$ sized location map is needed to exclude exceptional blocks. Next, the generated location map is losslessly compressed using arithmetic coding and embedded into the first two rows by LSB replacement. Finally, the original LSBs of selected pixels in the first two rows and PF are embedded into $I_M$ except the first two rows. During data extraction, the same pixels are selected from the first two rows using the encryption key to retrieve the compressed location map. After lossless decompression, the original location map can be obtained to guide the extraction of the original LSBs and PF.

Table 3 presents the comparison of embedding capacity with various ELs when the block size is set as $4 \times 4$. As the length of PF is 65536 bits at most for $512 \times 512$ sized image, Table 3 shows that PF can be generally embedded due to sufficient embedding capacity.

| Image   | EL=0 | EL=1 | EL=2 | EL=3 | EL=4 |
|---------|------|------|------|------|------|
| Baboon  | 108932 | 168872 | 194667 | 205183 | 208112 |
| Barbara | 104501 | 192954 | 210162 | 207781 | 175434 |
| Boat    | 104575 | 199830 | 214594 | 210322 | 189164 |
| Fish    | 93464 | 219103 | 229649 | 231423 | 231316 |
| House   | 95329 | 202477 | 216339 | 223022 | 205469 |
| Lake    | 104011 | 195053 | 217419 | 215300 | 172001 |
| Lena    | 100658 | 214194 | 202532 | 229284 | 232376 |
| Man     | 101738 | 88529 | 81613 | 66086 | 51314 |
| Peppers | 95155 | 209445 | 207554 | 197192 | 190223 |
| Splash  | 94085 | 228030 | 214851 | 197025 | 176385 |
| Stream  | 94085 | 167566 | 184765 | 175789 | 149575 |
| Tank    | 97630 | 209968 | 227387 | 237129 | 237629 |
D. A NEW FLUCTUATION FUNCTION

To better evaluate the fluctuation of a block, a new fluctuation function is proposed in this article with the idea of summing the absolute mean differences between each pixel and its neighbors. As shown in Fig. 8, pixels within block are classified into pixels at four vertexes, pixels on the down and up borders except vertexes, pixels on the right and left borders except vertexes, and pixels inside block. In the proposed method, the minimum block size is $2 \times 2$. The distribution of pixels within $2 \times 2$ sized block is shown in Fig. 9. As it is shown, all pixels are at vertexes and all of them are the neighbors of the other pixels. Notice that the grid marked by red denote the target pixel, the grid marked by other colors denote its neighbor. Specific function is designed for each kind of pixel to calculate its fluctuation.

For pixels at four vertexes, the fluctuation is calculated as

$$ f_1 = 2 \times (|p_{1,1} - p_{1,2}| + |p_{1,1} - p_{2,1}| + |p_{1,3} - p_{1,4}| + |p_{2,3} - p_{3,4}|) + 2 \times |p_{1,1} - p_{2,1}|.$$  \hspace{1cm} (13)

For pixels on the right and left borders except vertexes, the fluctuation is calculated as

$$ f_2 = \sum_{v=2}^{s-1} (|p_{1,v} - p_{1,v-1}| + |p_{1,v} - p_{2,v}| + |p_{v,2} - p_{v,1}| + |p_{v,3} - p_{v,4}| + 2 \times |p_{1,v} - p_{2,v}|).$$  \hspace{1cm} (14)

For pixels inside block, the fluctuation is calculated as

$$ f_3 = \sum_{u=2}^{s-1} (|p_{u,1} - p_{u-1,1}| + |p_{u,1} - p_{u+1,1}| + 2 \times |p_{u,1} - p_{u,2}| + |p_{u,s} - p_{u-1,s}| + |p_{u,s} - p_{u+1,s}| + 2 \times |p_{u,s} - p_{u,s-1}|).$$  \hspace{1cm} (15)

At last, the summation of each block’s fluctuation value is calculated as

$$ F = f_1 + f_2 + f_3 + f_4.$$  \hspace{1cm} (16)

IV. METHODOLOGY

The sketch of the proposed method is given in Fig. 10. As we can see, there are three participants including content owner, data hider and receiver. With the encryption key, ECB selection and image reconstruction are first conducted by the content owner to obtain PF and the reconstructed image. Next, PF is embedded into the reconstructed image through self-embedding. Finally, the reconstructed image containing PF is encrypted using the encryption key and a classical symmetric encryption method such as RC4 or SEAL.

After transmission, some additional data is embedded into ECBs by the data hider using the data-hiding key. Then, the authorized receiver who has both the decryption key and the data-hiding key can losslessly extract the hidden data and perfectly recover the cover image. The details of the proposed method are given in the following aspects: 1) image encryption; 2) data hiding; 3) data extraction and image recovery.

A. IMAGE ENCRYPTION

For simplicity, the cover image is assumed to be uncompressed gray-scale image and pixel value falls into $[0, 255]$. Each pixel is first decomposed into 8 bits using Eq. 18, where $p_{i,j}$ is the pixel value located at $(i, j)$ and $\lfloor . \rfloor$ is the flooring function.

$$ b_{i,j,k} = \left\lfloor \frac{p_{i,j}}{2^k} \right\rfloor \mod 2 \quad k = 0, 1, 2, \ldots, 7 $$  \hspace{1cm} (18)

After that, standard stream encryption method such as RC4 or SEAL is used to generate pseudo-random bits using the encryption key. Then, each pixel is encrypted by Eq. 19, where $e_{i,j,h}$ represents the cipher-text, $r_{i,j,k}$ is the key stream and $\oplus$ stands for exclusive-or.

$$ e_{i,j,k} = b_{i,j,k} \oplus r_{i,j,k} \quad k = 0, 1, 2, \ldots, 7 $$  \hspace{1cm} (19)

The encrypted pixel is calculated by Eq. 20. After that, the encrypted image and the encrypted PF are transmitted.

$$ E_{i,j} = \sum_{k=0}^{7} e_{i,j,k} \times 2^k $$  \hspace{1cm} (20)
B. DATA HIDING

After receiving the encrypted image containing PF, some additional data is embedded into the encrypted image by the data hider. The details of embedding are given as follows.

Step 1: Additional data encryption.
To prevent additional data from being read illegally, the additional data is first encrypted using the data-hiding key.

Step 2: Image segmentation.
The encrypted image is segmented into a lot of non-overlapping pixel blocks with the size of \( s_1 \times s_2 \).

Step 3: Data hiding.
Pixels of ECB are pseudo-randomly divided into two equal-size sets \( N_0 \) and \( N_1 \) using the data-hiding key. If the to-be-embedded bit is “0”, flip the three LSBs of each pixel in \( N_0 \) as

\[
e'_{i,j,k} = e_{i,j,k} \quad (i, j) \in N_0 \quad k = 0, 1, 2 \tag{21}
\]

Otherwise, flip the three LSBs of each pixel in \( N_1 \) as

\[
e'_{i,j,k} = e_{i,j,k} \quad (i, j) \in N_1 \quad k = 0, 1, 2 \tag{22}
\]

By repeating step 3, the additional data is embedded into the encrypted image. After that, an end mark is also embedded.

C. DATA EXTRACTION AND IMAGE RECOVERY

Data extraction and image recovery can be completed by an authorized recipient. The details of data extraction and image recovery are given as follows.

Step 1: Image decryption.
An pseudo-random matrix that has the same size as the received image is first generated using the decryption key. Then, the received image is decrypted by calculating the exclusive-or value of the received images and the pseudo-random matrix.

Step 2: PF extraction.
PF is extracted from the decrypted image such that pixel blocks can be rearranged into the original state.

Step 3: Data extraction and image recovery.
Pixels of ECB are pseudo-randomly divided into two equal-size sets \( N_0 \) and \( N_1 \) using the data-hiding key. Then, the three LSBs of pixels in \( N_0 \) (\( N_1 \)) are flipped to obtain \( N'_0 \) (\( N'_1 \)). Next, the fluctuation values of \( N'_0 \) and \( N'_1 \), denoted by \( F_0 \) and \( F_1 \), are calculated using the proposed fluctuation function. If \( F_0 > F_1 \), the hidden bit is “1” and \( N'_1 \) is the original pixel block. Otherwise, the hidden bit is “0” and \( N'_0 \) is the original pixel block.

V. EXPERIMENTAL RESULT

In this section, we conduct several experiments to evaluate the proposed method, including recovery result comparison and analysis, embedding capacity analysis using different block sizes, fluctuation function comparison and analysis, comparison with existing methods. The experiment PC configuration is: Intel E6-1620 v3, CPU 3.50GHz, 16GB RAM. All program is encoded by MATLAB. All test images are sized 512 \( \times \) 512 and can be obtained from [34].

A. RECOVERY RESULT COMPARISON AND ANALYSIS

To verify the advantages of the proposed method, standard test images “Lena” and “Baboon” are used to test the recovery result of the proposed method and related methods. When 4096 bits are embedded, Fig. 11 and Fig. 12 show the changes...
TABLE 4. Embedding capacity (bit) comparison for all test images under different block sizes.

| Image | 2 x 2 | 4 x 4 | 6 x 8 | 8 x 10 | 12 x 16 | 18 x 18 | 20 x 20 | 22 x 22 | 24 x 24 | 26 x 26 | 28 x 28 | 30 x 30 | 32 x 32 |
|-------|-------|-------|-------|-------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| Lena  | 39414 | 15184 | 7059  | 4067  | 2599    | 1764    | 1296    | 1024    | 784     | 625     | 529     | 441     | 361     | 312     | 289     | 256     |
| Baboon| 38084 | 12468 | 6075  | 3642  | 2393    | 1671    | 1239    | 992     | 770     | 617     | 527     | 439     | 357     | 324     | 289     | 256     |
| Peppers| 42013 | 15165 | 7080  | 4073  | 2600    | 1764    | 1296    | 1024    | 784     | 625     | 529     | 441     | 361     | 324     | 289     | 256     |
| House | 33927 | 13346 | 6437  | 3772  | 2437    | 1676    | 1243    | 989     | 760     | 610     | 523     | 431     | 355     | 322     | 283     | 254     |
| Stream| 29409 | 11479 | 5609  | 3347  | 2214    | 1541    | 1149    | 924     | 728     | 580     | 489     | 418     | 342     | 310     | 274     | 243     |
| Splash| 35989 | 15187 | 7024  | 4040  | 2586    | 1758    | 1294    | 1024    | 784     | 625     | 529     | 441     | 361     | 324     | 289     | 256     |
| Boat  | 41686 | 14679 | 7009  | 4049  | 2592    | 1763    | 1295    | 1024    | 784     | 625     | 529     | 441     | 361     | 324     | 289     | 256     |
| Tank  | 42857 | 15082 | 7084  | 4074  | 2599    | 1764    | 1296    | 1024    | 784     | 625     | 529     | 441     | 361     | 324     | 289     | 256     |
| F16   | 32400 | 14191 | 6784  | 3952  | 2532    | 1740    | 1281    | 1010    | 781     | 621     | 528     | 437     | 359     | 322     | 288     | 256     |

When 4096 bits are embedded, the recovery results of related methods are shown in Fig. 13 and Fig. 14, where incorrectly recovered blocks are marked by white. As we can see, there are far more incorrectly recovered blocks on the rough image “Baboon”. This indicates that embedding capacity is highly close to image complexity for related methods. Moreover, all related methods fail to embed 4096 bits into the images “Lena” and “Baboon”. By contrast, the proposed method has much higher embedding capacity.

B. EMBEDDING CAPACITY ANALYSIS USING DIFFERENT BLOCK SIZES

To further verify the performance of the proposed method, we compute the embedding capacity on all test images under different block sizes. As shown in Table 4, the embedding capacity generally increases with the decrease of block size. With the block size of 2 x 2, the proposed method achieves the highest capacity of 42857 bits on the image “Tank”, and the lowest capacity of 29409 bits on the image “Stream”. Therefore, it can be concluded that the proposed method can always vacate enough room for data embedding such that it is a practical RDHEI method.
C. FLUCTUATION FUNCTION COMPARISON AND ANALYSIS

For further improvement, a new fluctuation function is introduced in the proposed method. Compared with those of [21] and [24], the new fluctuation function additionally considers pixels on the edge of the pixel block. Moreover, more neighboring pixels are considered when compared with [22]. Compared with those of [21] and [23], the new fluctuation function evaluates the real fluctuation of pixels, especially for pixels shown in Fig. 3.
Table 5 presents the comparison of the number of ECBs when various fluctuation functions are adopted with the block size of $4 \times 4$. As it is shown, the proposed fluctuation function generally provides the most ECBs. This confirms the significance of new fluctuation function to capacity improvement.

D. COMPARISON WITH EXISTING METHODS
To better demonstrate the advantages of the proposed method, the extraction error rate and PSNR of the proposed method are compared with those of previous methods under the payload size of 256, 625, 1024, 1764, 2601, 4096, 7225 and 16384 bits. As shown in Fig.15, the proposed method always...
TABLE 5. Comparison of the number of ECBs between the proposed fluctuation function and previous ones using the block size of $4 \times 4$.

| Image  | [21] | [22] | [23] | [24] | Proposed |
|--------|------|------|------|------|----------|
| Lena   | 13264| 14917| 14982| 14287| 15184    |
| Baboon | 10704| 12262| 11985| 11640| 12468    |
| Peppers| 12367| 14497| 13661| 14232| 15165    |
| House  | 12367| 12891| 12773| 12796| 13346    |
| Boat   | 12363| 14249| 13243| 13447| 14679    |
| Splash | 13719| 14696| 14956| 14470| 15187    |
| Stream | 10185| 11282| 11912| 11028| 11474    |
| Tank   | 13419| 14992| 13703| 14076| 15082    |
| Lake   | 13495| 13299| 12739| 12929| 13981    |
| F16    | 13065| 13495| 13431| 13394| 14191    |

has an extraction error rate of zero. This implies that real reversibility is guaranteed by the proposed method for given payloads. For previous methods, by contrast, the extraction error rate of zero can only be achieved when the payload is fairly low. For example, Fig.15(c) shows that with a payload of 256 bits the extraction error rate is not zero for all previous methods except [25]. Moreover, all previous methods achieve a extraction error rate larger than zero on the extremely complex image “Stream”, as shown in Fig.15 (e). That is to say, reversibility cannot by guaranteed by previous methods if cover image with high complexity is given.

The PSNR comparison of the proposed method and previous methods is shown in Fig.16. As we can see, for the given payloads and test images, the PSNR of the proposed method is always $+\infty$. This means that the cover images can be perfectly recovered. On the contrary, the PSNR of previous methods can reach $+\infty$ only with very low payload and low-complexity image. As shown in Fig.16 (e), on the image “Stream” the PSNR of previous methods cannot reach $+\infty$ even though the payload size is 256 bits. However, it is seen from Fig.16 (e) that the PSNR of the proposed method is always $+\infty$. This indicates that reversibility is guaranteed by the proposed method despite the high complexity of cover image. Table 1 presents the maximum capacity of previous methods. By comparing Table 1 and Table 4, one can see that the embedding capacity is significantly improved by the proposed method.

Based on the above analysis, we can conclude that the performance of the proposed method is much better than previous methods. For practical RDHEI applications, the proposed method is undoubtedly more suitable.

VI. CONCLUSION

For traditional RDHEI methods which fulfill data embedding by flipping the three LSBs of half of pixels within block, it is not guaranteed that the fluctuation function always identifies the flipped part whatever the block size is. To ensure real reversibility, an efficient block selection mechanism based on precise extraction-correct-block (ECB) distinguishment is proposed in this article. By embedding additional data into only ECBs, the dependence of reversibility on block size can be completely removed and one can seek sufficient embedding capacity with flexible block size.

In a word, on the issue of real reversibility and embedding capacity, the advantage of the proposed ECB based solution is incomparable to existing improvements in fluctuation function. Moreover, a better fluctuation function is also proposed to provide more ECBs and thus higher embedding capacity. One deficiency of the proposed method is that the embedding room is vacated before image encryption. This increases the computational burden of the owner. In the future we plan to study embedding room vacating at the cloud sever.

REFERENCES

[1] H.-T. Wu, W. Mai, S. Meng, Y.-M. Cheung, and S. Tang, “Reversible data hiding with image contrast enhancement based on two-dimensional histogram modification,” IEEE Access, vol. 7, pp. 83332–83342, 2019.
[2] G.-D. Su, C.-C. Chang, and C.-C. Lin, “A high capacity reversible data hiding in encrypted AMBTC-compressed images,” IEEE Access, vol. 8, pp. 26984–27000, 2020.
[3] Y. Liu, H. Zhao, S. Liu, C. Feng, and S. Liu, “A robust and improved visual quality data hiding method for HEVC,” IEEE Access, vol. 6, pp. 53984–53997, 2018.
[4] H. Yao, X. Liu, Z. Tang, Y.-C. Hu, and C. Qin, “An improved image camouflage technique using color difference channel transformation and optimal prediction-error expansion,” IEEE Access, vol. 6, pp. 40569–40584, 2018.
[5] P.-F. Shiu, W.-L. Tai, J.-K. Jan, C.-C. Chang, and C.-C. Lin, “An interpolative AMBTC-based high-payload RDH scheme for encrypted images,” Signal Process., Images Commun., vol. 74, pp. 64–77, May 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0923596518300419
[6] R. Abbasi, L. Xu, F. Amin, and B. Luo, “Efficient lossless compression based reversible data hiding using multilayered n-Bit localization,” Secur. Commun. Netw., vol. 2019, pp. 1–13, Jan. 2019.
[7] Y. Hu, H.-K. Lee, K. Chen, and J. Li, “Difference expansion based reversible data hiding using two embedding directions,” IEEE Trans. Multimedia, vol. 10, no. 8, pp. 1500–1512, Dec. 2008.
[8] D. Ioan-Catalin and C. Dinu, “Local-prediction-based difference expansion reversible watermarking,” IEEE Trans. Image Process., vol. 23, no. 4, pp. 1779–1790, Apr. 2014.
[9] W. Wang, J. Ye, T. Wang, and W. Wang, “Reversible data hiding scheme based on significant-bit-difference expansion,” IET Image Process., vol. 11, no. 11, pp. 1018–1024, 2017.
[10] W.-L. Tai, C.-M. Yeh, and C.-C. Chang, “Reversible data hiding based on histogram modification of pixel differences,” IEEE Trans. Circuits Syst. Video Technol., vol. 19, no. 6, pp. 906–910, Jun. 2009.
[11] L. R. Mathews and V. A. C. Haran, “Histogram shifting based reversible data hiding using block division and pixel differences,” in Proc. Int. Conf. Control, Instrum., Commun. Comput. Technol. (ICCICCT), Jul. 2014, pp. 937–940.
[12] Z. Yin, A. Abel, X. Zhang, and B. Luo, “Reversible data hiding in encrypted image based on block histogram shifting,” in Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP), Mar. 2016, pp. 2129–2133.
[13] V. R. VijayKumar and V. S. Babu, “Histogram shifting based reversible data hiding with controlled contrast enhancement,” in Proc. 3rd Int. Conf. Biosignals, Images Instrum. (ICBSII), Mar. 2017, pp. 1–6.
[14] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, “Lossless generalized-LSB data embedding,” IEEE Trans. Image Process., vol. 14, no. 2, pp. 253–266, Feb. 2005.
[15] X. Cao, L. Du, X. Wei, D. Meng, and X. Guo, “High capacity reversible data hiding in encrypted images by patch-level sparse representation,” IEEE Trans. Cybern., vol. 46, no. 5, pp. 1132–1143, May 2016.
[16] Y.-Q. Shi, X. Li, X. Zhang, H.-T. Wu, and B. Ma, “Reversible data hiding: Advantages in the past two decades,” IEEE Access, vol. 4, pp. 3210–3237, 2016.
[17] Y.-C. Chen, C.-W. Shiu, and G. Horng, “Encrypted signal-based reversible data hiding with public key cryptosystem,” J. Vis. Commun. Image Represent., vol. 25, no. 5, pp. 1164–1170, Jul. 2014.

[18] S. Yi and Y. Zhou, “Separable and reversible data hiding in encrypted images using parametric binary tree labeling,” IEEE Trans. Multimedia, vol. 21, no. 1, pp. 51–64, Jan. 2019.

[19] Y. Wang and K. N. Plataniotis, “An analysis of random projection for changeable and privacy-preserving biometric verification,” IEEE Trans. Syst., Man, Cybern., B (Cybern.), vol. 40, no. 5, pp. 1280–1293, Oct. 2010.

[20] Y. Liu, S. Jia, M. Hu, Z. Jia, C. Liang, and H. Zhao, “A robust reversible data hiding scheme for H.264 based on secret sharing,” in Proc. Int. Conf. Intell. Comput., 2014, pp. 553–559.

[21] X. Zhang, “Reversible data hiding in encrypted image,” IEEE Signal Process. Lett., vol. 18, no. 4, pp. 255–258, Apr. 2011.

[22] W. Hong, T.-S. Chen, and H.-Y. Wu, “An improved reversible data hiding in encrypted images using side match,” IEEE Signal Process. Lett., vol. 19, no. 4, pp. 199–202, Apr. 2012.

[23] X. Liao and C. Shu, “Reversible data hiding in encrypted images based on absolute mean difference of multiple neighboring pixels,” J. Vis. Commun. Image Represent., vol. 28, pp. 21–27, Apr. 2015.

[24] F.-T.-Z. Khanam, K.-Y. Song, and S. Kim, “A modified reversible data hiding in encrypted image using enhanced measurement functions,” in Proc. 8th Int. Conf. Ubiquitous Future Netw. (ICUFN), Jul. 2016, pp. 869–872.

[25] R. Bhardwaj and A. Aggarwal, “An improved block based joint reversible data hiding in encrypted images by symmetric cryptosystem,” Pattern Recognit. Lett., vol. 139, pp. 60–68, Nov. 2020.

[26] Z. Qian, S. Dai, F. Jiang, and X. Zhang, “Improved joint reversible data hiding in encrypted images,” J. Vis. Commun. Image Represent., vol. 40, pp. 732–738, Oct. 2016.

[27] 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 Security, vol. 8, no. 3, pp. 553–562, Mar. 2013.

[28] Z. Yin, A. Abel, J. Tang, X. Zhang, and B. Luo, “Reversible data hiding in encrypted images based on multi-level encryption and block histogram modification,” Multimedia Tools Appl., vol. 76, no. 3, pp. 3899–3920, Feb. 2017.

[29] Z. Yin, B. Luo, and W. Hong, “Separable and error-free reversible data hiding in encrypted image with high payload,” Sci. World J., vol. 2014, Apr. 2014, Art. no. 604876.

[30] S. Yi and Y. Zhou, “Binary-block embedding for reversible data hiding in encrypted images,” Signal Process., vol. 133, pp. 40–51, Apr. 2017.

[31] L. Xin, K. Li, and J. Yin, “Separable data hiding in encrypted image based on compressive sensing and discrete Fourier transform,” Multimedia Tools Appl., vol. 76, no. 20, pp. 20739–20753, Oct. 2017.

[32] C. Qin and X. Zhang, “Effective reversible data hiding in encrypted image with privacy protection for image content,” J. Vis. Commun. Image Represent., vol. 31, pp. 154–164, Aug. 2015.

[33] D. Xiao and S. Chen, “Separable data hiding in encrypted image based on compressive sensing,” Electron. Lett., vol. 50, no. 8, pp. 598–600, Apr. 2014.

[34] I. Database. Accessed: Jul. 2018. [Online]. Available: http://sipi.usc.edu/database/

[35] H. Luo, F.-X. Yu, H. Chen, Z.-L. Huang, H. Li, and P.-H. Wang, “Reversible data hiding based on block median preservation,” Inf. Sci., vol. 181, no. 2, pp. 308–328, Jan. 2011.