Reversible data hiding for encrypted image based on adaptive prediction error coding

Zhenjun Tang | Mingyuan Pang | Chunqiang Yu | Guijin Fan | Xianquan Zhang

Abstract
Reversible data hiding (RDH) is a useful technique of data security. Embedding capacity is one of the most important performance of RDH for encrypted image. Many existing RDH algorithms for encrypted image do not reach desirable embedding capacity yet. To address this problem, a new RDH algorithm is proposed for encrypted image based on adaptive prediction error coding. The proposed RDH algorithm uses a block-based encryption scheme to preserve spatial correlation of original image in the encrypted domain and exploits a novel technique called adaptive prediction error coding to vacate room for data embedding. A key contribution of the proposed RDH algorithm is the adaptive prediction error coding. It can efficiently vacate room from encrypted image block by adaptively coding prediction errors according to block content and thus contributes to a large embedding capacity. Many experiments on benchmark image databases are done to validate performance of the proposed RDH algorithm. The results show that the average embedding rates on the open databases of UCID, BOSSBase and BOWS-2 are 1.7081, 2.4437 and 2.3083 bpp, respectively. Comparison results illustrate that the proposed RDH algorithm outperforms some state-of-the-art RDH algorithms in embedding capacity.

1 | INTRODUCTION
With the rapid development of information technology, communication via the Internet has become more convenient. Meanwhile, there are security risks during information transmission, such as information monitoring, information tampering and information forgery. Therefore, data security has attracted much attention in the past years. Reversible data hiding (RDH) [1–4] is a useful technique of data security. It can embed secret information into multimedia carrier, e.g. image, video and audio, and does not significantly change perceptual characteristic of the carrier. Moreover, it can exactly recover the carrier after data extraction. These advantages make RDH techniques suitable for many applications [5–8], such as judicial imaging, privacy protection, medical application and military communication. As digital images are widely used in our lives, researchers have designed many image-based RDH algorithms. According to the meanings of visual contents of digital images, these RDH algorithms can be divided into two kinds: RDH algorithms for plaintext image [9–11] and RDH algorithms for encrypted image [12–15]. In this work, we study a novel RDH algorithm for encrypted image.

The RDH algorithms for encrypted image are proposed for the scenario of cloud storage. In this application scenario, content-owners would like to encrypt their images before uploading them to the cloud for privacy protection. As there are many encrypted images in the cloud, data-hiders can exploit them to conduct data hiding. In the past years, researchers have developed various RDH algorithms for encrypted image. These existing RDH algorithms can be divided into two kinds: Reserving room before encryption (RRBE) [16–25] and Reserving room after encryption (RRAE) [26–34]. Typical techniques of each category are briefly reviewed as follows.

(1) RDH algorithms based on RRBE: An early work of this kind of algorithms was proposed by Ma et al. [16]. In this work, original image is divided into coarse area and smooth area, the traditional RDH method [3] is exploited to embed several least significant bit (LSB) planes of the coarse area into the smooth area, and the vacated LSB planes are used for data embedding. Motivated by [16], Mathew and Wilscy [17] presented an
improved algorithm by dividing image into coarse blocks and smooth blocks. The results show that the improved algorithm is better than the algorithm [16] in embedding capacity. Zhang et al. [18] proposed to estimate some pixels before encryption and vacate room for embedding additional information by shifting the estimated error histogram. This RDH algorithm is error-free in image recovery and data extraction. To make a high embedding capacity, Yi and Zhou [19] improved Zhang et al.’s RDH algorithm [18] by using half of image pixels for estimation. In another work, Cao et al. [20] introduced an RDH algorithm by using the patch-level sparse representation. This algorithm can reach high capacity. Wu et al. [21] used pixel redundancy of natural images to make room for embedding by vacating two lowest bits of pixels. Puteaux and Puech [22] proposed a new most significant bit (MSB) prediction method for RDH algorithm in encrypted image, in which MSB values are used to embed secret message instead of LSB values. Motivated by [22], Puyang et al. [23] designed a two-MSB prediction to explore correlation between local pixels for improving embedding capacity. The average payload of [23] is about 1.5 times as larger as the average payload of [22]. Recently, Wu et al. [24] used an edge detector to predict pixels, and exploited pixel differences between original pixels and their predicted pixels to make room for embedding before image encryption. Mohammadi et al. [25] proposed to divide original image into blocks and used a local difference predictor to reserve room. Both the schemes [24, 25] make full use of spatial correlation among original pixels, which contributes to a high embedding capacity.

(2) RDH algorithms based on RRAE: A famous RRAE based algorithm was designed by Zhang in 2011 [26]. This algorithm divides the encrypted image into blocks and embeds one bit into a block by flipping three LSBs of half block pixels. It extracts secret data by using a fluctuation function and cannot avoid extraction errors theoretically. Inspired by [26], Hong et al. [27] designed a new effective fluctuation function to measure block smoothness. The results illustrate that the number of the extracted error bits of Hong et al.’s algorithm [27] is smaller than that of Zhang’s algorithm [26]. Zhou et al. [28] presented a secure RDH algorithm by embedding secret data using a public key modulation mechanism. This algorithm can conceal more than one bit in an encrypted block. Yu et al. [29] exploited two-dimensional permutation and the exploiting modification direction (EMD) to design a new RDH scheme. In this scheme, a histogram of adjacent encrypted pixel errors is constructed, and secret data is embedded into the histogram via EMD. To preserve spatial correlation in the encrypted domain, Tang et al. [30] encrypted image by additive homomorphism and then embedded secret data by shifting block histogram of pixel differences. In another work, Tang et al. [31] used a block-based encryption scheme to conduct image encryption and proposed a novel technique called differential compression to vacate room from encrypted image for data embedding. Both the methods [30, 31] reach high embedding rate. Zheng et al. [32] used logistic map to encrypt image and compressed LSBs of encrypted image for vacating room by lossless compression. Yin et al. [33] divided original image into blocks, encrypted image through a multi granularity encryption method, and exploited the histogram shifting to embed secret information. Recently, Yi and Zhou [34] proposed a novel technique called parametric binary tree labelling for reserving room in encrypted domain by classifying encrypted pixels. This scheme has a high embedding rate.

Most RDH algorithms for encrypted image have good performance in their applications, but they do not reach desirable embedding capacity yet. To address this problem, we propose a new RDH algorithm for encrypted image based on adaptive prediction error coding. A key contribution is the novel technique called adaptive prediction error coding. It can efficiently vacate room from encrypted image block by adaptively coding prediction errors according to block content and thus contributes to a desirable embedding capacity. Experiments on benchmark image databases are done to validate performance of the proposed RDH algorithm. The comparison results illustrate that the proposed RDH algorithm outperforms some state-of-the-art RDH algorithms for encrypted image.

The rest of this paper is organized as follows. Section 2 explains the proposed RDH algorithm in detail. Section 3 presents experimental results and Section 4 makes conclusions of this paper.

2 | PROPOSED REVERSIBLE DATA HIDING ALGORITHM

Framework of the proposed RDH algorithm consists of three parts, as shown in Figure 1. In the first part, content owner encrypts original image under the control of encryption key. Output of the first part is the encrypted image. In the second part, data hider applies the proposed adaptive prediction error coding to vacate room from the encrypted image and then embeds secret data into the reserved room with data-hiding key. Note that the secret data mentioned in this work is a bit sequence, which can be generated from any data file, such as an image, a text file or a PDF file. In the third part, receiver can conduct different operations on the marked encrypted image according to the available knowledge of keys. If the receiver only has the data-hiding key, he/she can exactly extract secret data from the marked encrypted image. If the receiver only has the encryption key, he/she can recover image from the marked encrypted image. If both keys are available, he/she can conduct data extraction and image recovery from the marked encrypted image. Detailed steps of the proposed RDH algorithm are explained as follows.

2.1 | Image encryption

To preserve spatial correlation of original image in the encrypted domain, a block-based scheme is used to conduct image encryption. To do so, an original image is firstly divided into non-overlapping blocks sized $r \times s$. For simplicity, suppose that there are $M$ image blocks in total and the blocks are numbered from left to right and top to bottom ($i = 1, 2, ..., M$). Next, a well-known chaotic map called skew tent map is used to
calculate a chaotic sequence with \( M \) random numbers, which are exploited to encrypt blocks by XOR operation. The definition of the used skew tent map [35] is as follows.

\[
y_{i+1} = \begin{cases} 
\frac{y_i}{\mu} & y_i \in [0, \mu] \\
\frac{(1 - y_i)}{(1 - \mu)} & y_i \in (\mu, 1] 
\end{cases}
\]  

where \( \mu \in (0, 1) \) represents the control parameter. In this work, the control parameter \( \mu \) and the initial value \( y_0 \in [0, 1] \) are used as the keys for controlling the random sequence \( e \) by iteratively repeating the Equation (1). It is reported that the total combinations of \( \mu \) and \( y_0 \) for the skew tent map is \( 2^{104} \) [36], which is a large key space against the brute-force attacks [37, 38]. Since the elements of the random sequence \( e \) are real numbers, they are converted to integers in the range of \([0, 255]\) by the below equation.

\[
d_i \leftarrow \text{mod} \left( \lceil e_i \times 2^{48} \rceil, 256 \right) \tag{2}
\]

where \( \text{mod}(\cdot, \cdot) \) represents the modulus operation, and \( e_i \) and \( d_i \) represent the \( i \)-th elements of the sequences \( e \) and \( d \) \((1 \leq i \leq M)\), respectively.

After the random integer sequence \( d \) is obtained, the \( i \)-th image block is encrypted by bit-wise XOR operation as follows.

\[
p_{i,j} = b_{i,j} \oplus d_i \tag{3}
\]

where \( b_{i,j} \) is the \( j \)-th pixel of the \( i \)-th block, \( p_{i,j} \) is the corresponding encrypted pixel, and the symbol \( \oplus \) represents the bitwise XOR operation. After all image blocks are processed, the encrypted blocks are scrambled to make the final encrypted image \( E \) under the control of secret key. The key space of our block scrambling is \( M! \). Therefore, the key space of our encryption scheme is \( 2^{104 \times M!} \).

### 2.2 Data embedding with adaptive prediction error coding

Prediction errors of the encrypted image are firstly calculated. To do so, the encrypted image is divided into non-overlapping image blocks sized \( s \times s \). Let \( B_j \) be the \( i \)-th block \((1 \leq i \leq M)_j \) and \( p_{i,j} \) be the \( j \)-th pixel of \( B_j \). Assume that \( p_{i,j} \) is the smallest pixel of \( B_j \). Thus, the prediction error of \( p_{i,j} \) can be calculated by the below equation.

\[
e_{i,j} = p_{i,j} - p_{i,t} \quad (j \neq t) \tag{4}
\]

Clearly, \( e_{i,j} \geq 0 \). As spatial correlation of original image is well preserved in the encrypted domain, pixels in encrypted block are still similar. This property makes small prediction errors, which is helpful to adaptive coding. After prediction error calculation, the processed encrypted block \( B'_j \) is obtained as follows.

\[
l'_i,j = \begin{cases} 
p_{i,j} & \text{If } j = t \\
\hat{e}_{i,j} & \text{Otherwise} 
\end{cases} \tag{5}
\]

where \( l'_i,j \) represents the \( j \)-th pixel of \( B'_j \). After all image blocks are processed, a predicted encrypted image for adaptive coding is available.

As prediction error is a non-negative small value, it can be coded with fixed-length bits for reserving room. Suppose that \( \alpha_i \) bits are used to represent the prediction errors of the block \( B'_j \), where \( 1 \leq \alpha_i \leq 7 \). Thus, the reserved room of an encrypted pixel is \( 8 - \alpha_i \) bits. Clearly, the smaller the \( \alpha_i \) value, the larger the reserved room and thus the higher the embedding capacity.

Note that there are \( 2^7 - 1 \) prediction errors in each block, and \( \alpha_i \) bits can only represent \( 2^\alpha_i \) integers ranging from 0 to \( 2^\alpha_i - 1 \). If the selected \( \alpha_i \) is too small, some prediction errors cannot be coded by the fixed-length bits and thus they must be marked with sign. In this paper, a binary sequence with \( m \) ‘0’ (1 \( \leq m \leq 7 \)) is selected as the sign for marking those prediction errors which cannot be coded by \( \alpha_i \) bits. If the used \( \alpha_i \) is too big, the reserved room is limited. Therefore, a proper \( \alpha_i \) selection can make a good coding performance.

Figure 2 presents structure of prediction error after adaptive coding. Figure 2(a) is the structure of prediction error coded with \( \alpha_i \) bits. Clearly, the first \( \alpha_i \) bits are the corresponding binary code of the prediction error, and the rest is the reserved room sized \((8 - \alpha_i)\) bits, which is used to conceal data. Figure 2(b) is the structure of prediction error marked sign, where the first \( m \) bits are all ‘0’ (the sign) and the remained \((8 - m)\) bits are the original data of the prediction error. Note that the original first \( m \) bits of the prediction error are picked out in advance before
inserting the sign and will be embedded into the block for image recovery.

Since the elements of the sign are all ‘0’, those fixed-length binary codes with \( m \times 0 \)'s as the prefix cannot be used for adaptive coding. According to the relation of \( \alpha_i \) and \( m \), two code books can be generated as shown in Tables 1 and 2. Table 1 presents the code book for \( \alpha_i \leq m \), where only one code with \( m \times 0 \)'s is discarded and other codes are all used. In this case, the binary code of the prediction error not bigger than \( 2^{\alpha_i} - 2 \) can be found in the one-to-one mapping table. Table 2 presents the code book for \( \alpha_i > m \), where \( 2^{\alpha_i - m} \) codes are not used since their prefixes contain \( m \times 0 \)'s. In this case, the binary code of the prediction error not bigger than \( 2^{\alpha_i} - 2^{\alpha_i - m} - 1 \) can be found in the code book. Therefore, the prediction error to be coded should satisfy the below rule.

\[
\begin{cases}
|b'_{ij}| \leq 2^{\alpha_i} - 2 & \alpha_i \leq m \\
|b'_{ij}| \leq 2^{\alpha_i} - 2^{\alpha_i - m} - 1 & \alpha_i > m
\end{cases}
\]  

(6)

Suppose that there are \( n_i \) pixels whose prediction errors satisfying the rule in Equation (6). Therefore, the practical capacity of the \( i \)-th block can be determined as below equation.

\[
C_i = (8 - \alpha_i) n_i - m \left( 2^2 - 1 - n_i \right)
\]  

(7)

In the right part of the Equation (7), the first term is the size of reserved room and the second term is the cost of preserving the prefixes of those prediction errors marked with sign. Consequently, for a given \( m \), the optimal \( \alpha_i \) can be determined by solving an optimization problem as below.

\[
\alpha_i = \arg \max_{\alpha_i} C_i = \arg \max_{\alpha_i} \left[ (8 - \alpha_i) n_i - m \left( 2^2 - 1 - n_i \right) \right]
\]  

(8)

where \( \alpha_i \in \{1, 2, \ldots, 7\} \). Note that \( \alpha_i \) denotes the number of the used bits of a pixel for prediction error coding. Therefore, \( \alpha_i = 8 \) cannot be selected because there is no bit to conceal data. Clearly, different blocks have different optimal \( \alpha_i \) values. To ensure exact data extraction, the used optimal \( \alpha_i \) value is embedded into its block. Since its maximum value is 7, 3 bits are enough for storing the \( \alpha_i \) value. Except the optimal \( \alpha_i \) value, other auxiliary information should be also preserved. Note that the smallest pixel is used during prediction error calculation, which is also required for data extraction. However, the position of the smallest pixel in each block is unknown. To solve this problem, the position of the smallest pixel is first recorded and embedded into the block, and then the smallest pixel and the last pixel of the block are swapped. During data extraction, the smallest pixel can be easily picked out and restored under the guidance of the position information. As the block size is \( s \times s \), the required bits for storing position of the smallest pixel is \( \lceil \log_2 s^2 \rceil = \lceil 2 \log_2 s \rceil \), where \( \lceil \cdot \rceil \) is the upward rounding. Note that range of the pixel position is \( \{0, 2^2 - 1\} \). Specifically, position of the first pixel is represented by 0, position of the second pixel is represented by 1, and so on. Therefore, the size of auxiliary information of each block is \( 2 \log_2 s + 3 \) bits. Consequently, the usable room of the \( i \)-th block is determined by the below equation.

\[
r_i = C_i - \lceil 2 \log_2 s + 3 \rceil = (8 - \alpha_i) n_i - m \left( 2^2 - 1 - n_i \right) - \lceil 2 \log_2 s \rceil - 3
\]  

(9)

If \( r_i \geq 1 \), the \( i \)-th block is a usable block and can be exploited to conceal secret data. Otherwise, the block is unusable. Let \( r_i \) be the embedding capacity of the \( i \)-th block. Thus, it can be
determined by the following rule.

\[
v_i = \begin{cases} 
    v_i & \text{if } v_i \geq 1 \\
    0 & \text{Otherwise} 
\end{cases} 
\]

(10)

Since there are \( M \) blocks in the encrypted image, a binary map \( S \) with \( M \) elements can be used to represent block signs. The element of the map \( S \) can be defined as follows.

\[
S(i) = \begin{cases} 
    1 & \text{if } v_i \geq 1 \\
    0 & \text{Otherwise} 
\end{cases} 
\]

(11)

Consequently, the pure capacity of an encrypted image can be calculated by the following equation.

\[
V' = \sum_{i=1}^{M} v_i - M 
\]

(12)

To make easy operations of data embedding and extraction, the block sign (i.e. \( S(i) \)), the optimal \( \alpha \) value, and the position of the smallest pixel are all embedded in the block. Specifically, the block sign is inserted into the eighth bit of the last pixel of the block. If the block sign \( S(i) = 1 \), the three bits of the \( \alpha \) value are hidden in the seventh to fifth bits of the last pixel and then the \([2\log_2 s] \) bits of pixel position are inserted into the first \([2\log_2 s] \) bits of the reserved room. Note that the eighth to fifth bits of the last pixel are also picked out in advance before data insertion and will be embedded into the encrypted block for image recovery.

During data embedding, the eighth to fifth bits of the last pixels of the usable blocks (suppose that the number of the usable blocks is \( q \)) are connected with the encrypted version of the secret data to construct the data to be embedded. The encrypted version of secret data is calculated by the bitwise XOR operation between the secret data and a pseudorandom bit sequence generated by the famous RC4 algorithm [39] under the control of the data-hiding key. Note that, for each usable block, the first \([2\log_2 s] \) bits have been used for storing pixel position, some vacated bits are used to store the preserved bits (i.e. those bits of the pixels replaced by \( m \) ‘0’s), and only the remained room can be used for data embedding. Therefore, we embed data into the remained room block by block. To better illustrate our embedding, Figure 3 presents the structures of the room and the embedded data used in our algorithm. Figure 3(a) is the structure of the reserved room of a block, where the first \([2\log_2 s] \) bits represent the position of the smallest pixel of the block, the second part is the preserved bits and the remained room can be used to embed data. Figure 3(b) is the structure of the embedded data, where the first \( M \) bits are the 8th bits of the last pixels of all blocks, the \( 3q \) bits are the seventh to fifth bits of the last pixels of the usable blocks, and the rest data is the encrypted secret data.

Figure 4 illustrates an example of our data embedding with prediction error coding. Suppose that the parameter selection is \( m = 2 \). Figure 4(a) is a 2 × 2 image block, (b) is the result of prediction errors and (c) is the binary codes of (b). By solving the Equation (8), it can be determined that the optimal value is \( \alpha = 2 \). Thus, a code book can be generated as listed in Table 3.

According to the code book, the prediction errors 1 and 2 are represented by ‘10’ and ‘11’ and the prediction error 17 cannot be coded. Therefore, the coding results of errors 1 and 2 are ‘10**********’ and ‘11************’, where ‘*’ is the reserved room. Moreover, the eighth to seventh bits of the prediction error 17 are replaced with the sign ‘00’ (its original bits ‘00’ are picked out in advance and will be preserved in the reserved room). Figure 4(d) is the results of error coding. Next, the positions of the smallest pixel 54 and the last pixel are swapped. As the position of the smallest pixel 54 is 2 (it is numbered from left to right and top to down), its binary code is ‘10’. Clearly, the block sign is 1. As \( \alpha = 2 \), its binary code with 3 bits is ‘010’. Therefore, the eighth bit of the last pixel is replaced with ‘1’ and its seventh to fifth bits are replaced with ‘010’. The position of the smallest pixel ‘10’ is inserted into the reserved room of the first prediction error, and the first two bits of the prediction error 17 (i.e. ‘00’) are preserved in the next two bits of the first prediction error. Figure 4(e) presents our coding with auxiliary information, where ‘*’ is the reserved room for data embedding.

### 2.3 Data extraction and image recovery

When the marked encrypted image is obtained, the receiver can extract auxiliary information for data extraction and image recovery. The marked encrypted image is firstly divided into blocks and the eighth bits of the last pixels of all blocks are picked out. Once the block signs are available, the number of
the usable blocks (i.e. $q$) can be calculated. According to the seventh to fifth bits of the last pixels of each block, the used $\alpha_i$ of the block can be determined. Consequently, the code book of prediction error coding can be generated with the guide of the parameter $m$. Next, the receiver can extract data hidden in each block by jointly using the code book and the prefix with $m$ '0's. As the structures of the room of block and the embedded data are known, the receiver can pick out different data according to the available knowledge of keys. There are three cases as follows.

1. Only the data-hiding key is available. In this case, the receiver only picks out the encrypted secret data from the extracted data. He/she can correctly decrypt the encrypted secret data by bitwise XOR operation with the pseudorandom bit sequence generated by the RC4 algorithm under the control of the data-hiding key.

2. Only the encryption keys are available. The receiver firstly picks out the $M$ bits and $3q$ bits from the extracted data. The $M$ bits are used to replace the eighth bits of the last pixels of all blocks, and the $3q$ bits are used to replace the seventh to fifth bits of the last pixels of the usable blocks. For each usable block, the receiver picks out the bits of pixel position and the preserved bits. With the position information, the original last pixel and the original smallest pixel can be correctly swapped. In addition, those prediction errors marked with signs can be reconstructed by using the preserved bits. Next, the encrypted pixels can be restored by adding the smallest pixel to their prediction errors. Encrypted blocks are then re-scrambled under the control of secret key. A random integer sequence generated by the skew tent map is calculated under the control of secret keys, as illustrated in Section 2.1. Finally, the original image can be reconstructed from the encrypted image by bitwise XOR operation between the random integer sequence and the image blocks.

3. Both the data-hiding key and the encryption keys are available. The receiver can exactly extract secret data by using the operations described in the first case. Moreover, he/she can also correctly decrypt the encrypted image by the operations described in the second case.

3 | EXPERIMENTAL RESULTS

Extensive experiments on benchmark images are done to verify the performance of the proposed RDH algorithm. In the following experiments, our parameter settings are as follows. The block size is $4 \times 4$, and the parameter $m$ is 5. Section 3.1 introduces our data-hiding results, and Section 3.2 discusses security of our encrypted images. Section 3.3 presents parameter discussion. Section 3.4 compares the proposed RDH algorithm with some state-of-the-art RDH algorithms.

3.1 | Data-hiding results

Six benchmark grey images of size $512 \times 512$ from the open image database [40] are used as test images. Figure 5 illustrates these benchmark images. To validate effectiveness of the proposed RDH algorithm, these benchmark images are taken as input images and then encrypted by encryption keys. Next, different secret data is generated and embedded into the encrypted images under the control of data-hiding key. Finally, data extraction and image recovery are done on the marked encrypted images. It is found that the secret data can be correctly extracted and the reconstructed images are exactly the same as their original images. For page limitation, a visual example of the standard Lena image is presented here. Figure 6(a) is the encrypted version of Lena, (b) is the marked encrypted version after data embedding, and (c) is the reconstructed version of Lena decrypted from the marked encrypted image. It can be seen from Figures 5(b) and 6(c) that there is no difference between the original Lena and the reconstructed image. The
correct extraction of secret data and the extract image recovery verify the effectiveness of the proposed RDH algorithm.

### 3.2 Security analysis

Security performances of our encrypted images are evaluated by using some well-known metrics, including number of pixel change rate (NPCR), unified average changing intensity (UACI), entropy and histogram. NPCR and UACI are generally exploited to test the performance of key sensitivity. The definition of NPCR [31] is defined as follows.

\[
\text{NPCR} = \frac{1}{W \times Y} \sum_{i=1}^{W} \sum_{j=1}^{Y} F(i, j) \times 100\% 
\]  

where \(W\) and \(Y\) are the image row number and image column number, and \(F(i, j)\) is determined by the below equation.

\[
F(i, j) = \begin{cases} 
0, & \text{If } I(i, j) = I'(i, j) \\
1, & \text{Otherwise} 
\end{cases} 
\]  

where \(I(i, j)\) and \(I'(i, j)\) represent the pixels in the \(i\)-th row and \(j\)-th column of two encrypted images, respectively. The definition of UACI [41] is as follows.

\[
\text{UACI} = \frac{1}{W \times Y} \sum_{i=1}^{W} \sum_{j=1}^{Y} \frac{|I(i, j) - I'(i, j)|}{255} \times 100\% 
\]  

For digital images with 256 grey levels, the expected theoretical values of NPCR and UACI are 100\% and 33.33\% [31, 41], respectively. If the calculated NPCRs/UACIs are close to their theoretical values, it can be concluded that the encrypted images are key-sensitive and secure.

In the experiments, six benchmark grey images are also selected as test images. For every image, different keys are used to conduct image encryption and the NPCRs/UACIs between every pairs of encrypted images are calculated. It is observed that the obtained NPCRs/UACIs are close to the theoretical values. For page limitation, typical results of the Baboon image are demonstrated here. Figure 7 presents four encrypted images of Baboon under the control of four slightly different encryption keys \(T_0, T_1, T_2,\) and \(T_3\). Table 4 lists the detailed NPCRs and UACIs. It can be seen that the calculated results of NPCR and UACI are all close to their theoretical values. This illustrates that our encrypted images are secure.

Information entropy is an effective measure for evaluating security of encrypted images. The entropy of a grey image with 256 levels can be calculated by the below equation.

\[
H(I) = - \sum_{i=0}^{255} P(x_i) \log_2(P(x_i)) 
\]  

where \(I\) is the grey image with 256 levels and \(P(x_i)\) is the probability of the grey level \(x_i\) (0 \(\leq i \leq 255\)). In general, a bigger entropy means a more secure encrypted image. The maximum theoretical entropy of a grey image with 256 levels is 8. Table 5 presents the calculated entropy values of six benchmark images. It can be seen that the entropy values of all encrypted images are very close to 8. This also validates security of our encrypted images.

Histogram is an important image feature. In general, an effective image encryption scheme is expected to produce the encrypted image with a uniform histogram. In this case, it is almost impossible to successfully predict the original image of a given encrypted image by searching images with histogram-based techniques. Figure 8 demonstrates histograms of the encrypted versions of six benchmark images. It is found that these histograms are nearly uniform distributions. So the encrypted images of our algorithm are also secure in terms of histogram.

In addition, visual security is also tested by using the metrics of mean square error (MSE), Pearson correlation coefficient (PCC), peak signal to noise ratio (PSNR) and structural similarity (SSIM) [30]. For MSE, a bigger value means higher visual security. For PCC, PSNR and SSIM, a smaller value implies higher visual security. Experiments on six benchmark images demonstrate that our algorithm has good results on these metrics. For space limitation, typical results of the Airplane are listed in Table 6. It can be seen that all MSE values are big and all PCC, PSNR and SSIM values are small. These results illustrate good visual security of our algorithm.
FIGURE 7  Encrypted results of Baboon under different keys. (a) $T_0$, (b) $T_1$, (c) $T_2$, (d) $T_3$

FIGURE 8  Histograms of six encrypted images. (a) Baboon, (b) Lena, (c) Jetplane, (d) Sailboat, (e) Tiffany, (f) Airplane

3.3  Parameter discussion

The main parameters of the proposed RDH algorithms are the block size $s \times s$ and the parameter $m$. To determine optimal parameter selection, different parameter settings are exploited to conduct data embedding. The used block sizes are $2 \times 2, 3 \times 3, 4 \times 4, 5 \times 5$ and $6 \times 6$, i.e. $s = 2, 3, 4, 5$ and $s = 6$. For each block size, the used $m$ settings are $m = 2, 3, 4, 5, 6$ and $m = 7$. The embedding rate used for measuring data-hiding performance is the bit per pixel (bpp), which is defined as the ratio between the number of embedded bits and

| Input images | MSE   | PCC    | PSNR   | SSIM  |
|--------------|-------|--------|--------|-------|
| Original airplane and encrypted airplane | 8320.58 | −0.0020 | 8.9293 | 0.0221 |
| Original airplane and marked encrypted airplane | 8071.23 | −0.0075 | 9.0614 | 0.0149 |
| Encrypted airplane and marked encrypted airplane | 9575.68 | 0.1178  | 8.3191 | 0.1055 |
the total pixel number. A bigger embedding rate means a better data-hiding performance.

Tables 7–11 present the embedding rates of the six benchmark images under different \( m \) settings when the block size \( s \) varied from \( s = 2 \) to \( s = 6 \). In these tables, the numbers in bold are the maximum embedding rates (the best results). It can be seen that when \( m = 5 \), the embedding rates of all benchmark images reach the maximum values under different block sizes. Therefore, \( m = 5 \) is the optimal parameter selection. Figure 9 presents the trend curve of average embedding rate under different block sizes when \( m = 5 \). It can be found that when the block size is small, the average embedding rate is also small. As the block size increases, the average embedding rate also increases and reaches the maximum value when block size \( s = 4 \). After it reaches the maximum value, it decreases when the block size increases. This can be understood as follows. When the block size is too small, the number of image blocks increases and thus there is much more auxiliary information needed for self-embedding, which limits the embedding rate. If the block size is too large, spatial correlation in the block will be weakened and thus some prediction errors will become large numbers, which also restrict the improvement of embedding rate.

### 3.4 Performance comparison

To verify advantages of the proposed RDH algorithms, performance comparisons with four state-of-the-art RDH algorithms are presented in Tables 7–11.
FIGURE 10  Average embedding rate comparisons

are conducted. The selected algorithms include Wu et al.’s algorithm [21], Puteaux and Puech’s algorithm [22], Puyang et al.’s algorithm [23] and Yi and Zhou’s algorithm [34]. All these compared algorithms are designed for encrypted image and recently published in famous conference or journals. The used block size of Wu et al.’s algorithm is 16 × 16. The parameter settings of Yi and Zhou’s algorithm are as follows: the block size is 3 × 3, α = 5 and β = 2.

Three open image databases are exploited to test the performances of these algorithms. The used databases include UCID [42], BOSSBase [43] and BOWS-2 [44]. The UCID contains 1338 colour images, which are converted to grey images with 256 levels in the experiments. The BOSSBase and BOWS-2 both contain 10,000 standard grey images. All these evaluated RDH algorithms are applied to the grey images of these databases. The average embedding rate of grey images in these databases are calculated and the results are demonstrated in Figure 10, where the x-axis is the used database and the y-axis is the average embedding rate. It can be seen that the average embedding rates of most images are bigger than those of the compared algorithms.

Comparing with Wu et al.’s algorithm, Puteaux and Puech’s algorithm, Puyang et al.’s algorithm and Yi and Zhou’s algorithm, our average improvements are 1.3179, 1.211, 0.8292 and 0.3448 bpp on the three databases, respectively. This illustrates that the proposed RDH algorithm outperforms the compared RDH algorithm in embedding rate.

To view the detailed embedding capacities of the proposed algorithm and the compared algorithms on every test image of these databases, the results of embedding capacities are all plotted in the same figure. It is found that our embedding capacities of most images are bigger than those of the compared algorithms. For page limitation, only the comparisons between the proposed algorithm and Wu et al.’s algorithm are presented here. Figures 11–13 demonstrate the embedding capacity comparison on the databases of UCID, BossBase and BOWS-2, respectively, where the x-coordinate represents the image index, and the y-coordinate represents the embedding capacity. It is clear that the proposed algorithm outperforms Wu et al.’s algorithm in the embedding capacities of most test images. This also verifies advantage of the proposed algorithm in embedding performance.

Computational time of these RDH algorithms is also compared. The experimental environments are as follows. The coding perform is MATLAB R2016. The configurations of the used computer include the Intel Core i5-6500 CPU with 3.20 GHz and 8 GB RAM. The operating system is Windows 10 Education. All images in the BOWS-2 are used to conduct data embedding. The results are presented in Table 13. The first column is the evaluated algorithm, the second column is the average time of embedding data into an image, the third column is the average capacity of an image, and the final column is the average time of embedding one bit (T/C).

### TABLE 12  Improvement of average embedding rate on different databases (unit: bpp)

| Database | Proposed vs Wu et al. [21] | Proposed vs Puteaux and Puech [22] | Proposed vs Puyang et al. [23] | Proposed vs Yi and Zhou [34] |
|----------|----------------------------|-----------------------------------|-------------------------------|-----------------------------|
| UCID     | 0.5690                     | 0.8150                            | 0.5287                        | 0.1220                      |
| BOSSBase | 1.6116                     | 1.4778                            | 0.9963                        | 0.4868                      |
| BOWS-2   | 1.7732                     | 1.3402                            | 0.9627                        | 0.4257                      |
| Average  | 1.3179                     | 1.2110                            | 0.8292                        | 0.3448                      |

### TABLE 13  Computational time comparison

| Algorithm    | Average time of an image (T) | Average capacity of an image (C) | Average time of embedding one bit (T/C) |
|--------------|------------------------------|----------------------------------|----------------------------------------|
| Wu et al. [21] | 3.175                        | 140,662                          | 0.00002257                             |
| Puyang et al. [23] | 2.651                        | 352,740                          | 0.00000752                             |
| Puteaux and Puech [22] | 2.432                        | 253,782                          | 0.00000958                             |
| Yi and Zhou [34] | 2.017                        | 493,041                          | 0.00000409                             |
| Proposed     | 2.565                        | 605,153                          | 0.00000424                             |
FIGURE 11  Embedding capacity comparison on the database of UCID

FIGURE 12  Embedding capacity comparison on the database of BossBase

FIGURE 13  Embedding capacity comparison on the database of BOWS-2
time of embedding one bit into an image. It is found that the proposed algorithm is faster than Wu et al.’s algorithm, Puyang et al.’s algorithm and Puteaux and Puech’s algorithm, but it is slightly slower than Yi and Zhou’s algorithm.

4 CONCLUSIONS

In this paper, a novel RDH algorithm for encrypted images has been proposed. The proposed RDH algorithm uses a block-based encryption scheme to preserve spatial correlation of original image in the encrypted domain. An important contribution is the adaptive prediction error coding. This technique can efficiently vacate room from encrypted image block by adaptively coding prediction errors according to block content, and thus contributes to a large embedding capacity. Many experiments on benchmark image databases have been done to validate performance of the proposed RDH algorithm. The comparison results have illustrated that the proposed RDH algorithm outperforms some state-of-the-art RDH algorithms in embedding capacity. Computational time results have also shown that the proposed RDH algorithm is better than most compared algorithms. In future work, we will investigate other RDH algorithms for encrypted image, such as RDH algorithms via steganography coding, colour image based RDH algorithms using channel correlation, and multi-image based RDH algorithms.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. Many thanks to Dr. Fengyong Li for sharing their code [21]. This work was partially supported by the National Natural Science Foundation of China (61962008, 62062013, 61762017, 61562007), Guangxi “Bagui Scholar” Team for Innovation and Research, the Guangxi Talent Highland Project of Big Data Intelligence and Application, and Guangxi Collaborative Innovation Center of Multi-source Information Integration and Intelligent Processing.

ORCID
Zhengjun Tang https://orcid.org/0000-0003-3664-1363

REFERENCES

1. Shah, M., et al.: Prediction error expansion-based reversible data hiding in encrypted images with public key cryptosystem. IET Image Proc. 13(10), 1708–1713 (2019)
2. Wang, W., et al.: Reversible data hiding scheme based on significant-bit difference expansion. IET Image Proc. 11(11), 1002–1014 (2017)
3. Ni, Z., et al.: Reversible data hiding. IEEE Trans. Circuits Syst. Video Technol. 16(3), 354–362 (2006)
4. Zhang, X.P.: Reversible data hiding with optimal value transfer. IEEE Trans. Multimedia 15(2), 316–325 (2013)
5. Tang, Z., et al.: Color image reversible data hiding with double-layer embedding. IEEE Access 8, 6915–6926 (2020)
6. Fridrich, J., Goljan, M., Du, R.: Lossless data embedding: New paradigm in digital watermarking. EURASIP J. Appl. Signal Process. 2002(2), 185–196 (2020)
7. Li, S., Zhang, X.P.: Toward construction-based data hiding: From secrets to fingerprint images. IEEE Trans. Image Process. 28(3), 1482–1497 (2019)
8. Xiang, Y., Xiao, D., Zhang, R.: Cryptanalysis and improvement of a reversible data-hiding scheme in encrypted images by redundant space transform. Inf. Sci. 545, 188–206 (2021)
9. Yao, H., et al.: High-fidelity dual-image reversible data hiding via prediction-error shift. Signal Process. 170, 107447 (2020)
10. Sachnev, V., et al.: Reversible watermarking algorithm using sorting and prediction. IEEE Trans. Circuits Syst. Video Technol. 19(7), 989–999 (2009)
11. Alattar, A.M.: Reversible watermark using the difference expansion of a generalized integer transform. IEEE Trans. Image Process. 13(8), 1147–1156 (2004)
12. Wu, H., et al.: A high-capacity reversible data hiding method for homomorphic encrypted images. J. Visual Commun. Image Represent. 62, 87–96 (2019)
13. Yu, C., et al.: Reversible data hiding with pixel prediction and additive homomorphism for encrypted image. Secur. Commun. Netw. 2018, 1–13 (2018)
14. Tang, Z., et al.: Error-free reversible data hiding with high capacity in encrypted image. Optik - Int. J. Light Electron Opt. 157, 750–760 (2018)
15. Yu, C., et al.: Separable and error-free reversible data hiding in encrypted image based on two-layer pixel errors. IEEE Access 6, 76956–76969 (2018)
16. Ma, K., et al.: Reversible data hiding in encrypted images by reusing room before encryption. IEEE Trans. Inf. Forensics Secur. 8(3), 553–562 (2013)
17. Mathew, T., Wilsey, M.: Reversible data hiding in encrypted images by active block exchange and room reservation. In: 2014 International Conference on Contemporary Computing and Informatics. Mysore, India, pp. 839–844 (2014)
18. Zhang, W., Ma, K., Yu, N.: Reversibility improved data hiding in encrypted images. Signal Process. 94(1), 118–127 (2014)
19. Yi, S., Zhou, Y.: An improved reversible data hiding in encrypted images. In: 2015 IEEE China Summit and International Conference on Signal and Information Processing. Chengdu, China, pp. 225–229 (2015)
20. Cao, X., et al.: High capacity reversible data hiding in encrypted images by patch-level sparse representation. IEEE Trans. Cybern. 46(5), 1132–1143 (2016)
21. Wu, H., et al.: Separable reversible data hiding in encrypted images based on scalable blocks. Multimedia Tools Appl. 78(18), 25349–25372 (2019)
22. Puteaux, P., Puech, W.: An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images. IEEE Trans. Inf. Forensics Secur. 13(7), 1670–1681 (2018)
23. Puyang, Y., Yin, Z., Qian, Z.: Reversible data hiding in encrypted images with two-MSB prediction. In: 2018 IEEE International Workshop on Information Forensics and Security. Hong Kong, pp.1–7 (2018)
24. Wu, Y., et al.: An improved reversible data hiding in encrypted images using parametric binary tree labeling. IEEE Trans. Multimedia 22(8), 1929–1938 (2020)
25. Mohammad, A., Nakhash, M., Alkhaee, M.A.: A high-capacity reversible data hiding in encrypted images employing local difference predictor. IEEE Trans. Circuits Syst. Video Technol. 30(6), 2366–2376 (2020)
26. Zhang, X.P.: Reversible data hiding in encrypted image. IEEE Signal Process. Lett. 18(4), 255–258 (2011)
27. Hong, W., Chen, T., Wu, H.: An improved reversible data hiding in encrypted images using side match. IEEE Signal Process. Lett. 19(4), 199–202 (2012)
28. Zhou, J., et al.: Secure reversible image data hiding over encrypted domain via key modulation. IEEE Trans. Circuits Syst. Video Technol. 26(3), 441–452 (2016)
29. Yu, C., et al.: Separable reversible data hiding in encrypted image based on two-dimensional permutation and exploiting modification direction. Mathematics 7(10), 976 (2019)
30. Tang, Z., et al.: Real-time reversible data hiding with shifting block histogram of pixel differences in encrypted image. J. Real-Time Image Process. 16(3), 709–724 (2019)
31. Tang, Z., et al.: Reversible data hiding with differential compression in encrypted image. Multimedia Tools Appl. 78(8), 9691–9715 (2019)
32. Zheng, S., et al.: Lossless data hiding algorithm for encrypted images with high capacity. Multimedia Tools Appl. 75(21), 13765–13778 (2016)
33. Yin, Z., Luo, B., Hong, W.: Separable and error-free reversible data hiding in encrypted image with a high payload. Sci. World J. 2014, 1–8 (2014)
34. Yi, S., Zhou, Y.: Separable and reversible data hiding in encrypted images using parametric binary tree labeling. IEEE Trans. Multimedia 21(1), 51–64 (2019)
35. Tang, Z., Zhang, X.Q., Lan, W.: Efficient image encryption with block shuffling and chaotic map. Multimedia Tools Appl. 74(15), 5429–5448 (2015)
36. Zhang, G., Liu, Q: A novel image encryption method based on total shuffling scheme. Opt. Commun. 284(12), 2775–2780 (2011)
37. Sun, F., Lu, Z., Liu, S.: A new cryptosystem based on spatial chaotic system. Opt. Commun. 283(10), 2066–2073 (2010)
38. Wang, Q., Guo, Q., Zhou, J.: Double image encryption based on linear blend operation and random phase encoding in fractional Fourier domain. Opt. Commun. 284(12), 2775–2780 (2011)
39. Rivest, R.L.: The RC4 Encryption Algorithm. RSA Data Security Inc., Bedford, MA (1992)
40. USC-SIPI Image Database. http://sipi.usc.edu/database/ (2020). Accessed 10 Sept 2020

41. Tang, Z., Wang, F., Zhang, X.Q: Image encryption based on random projection partition and chaotic system. Multimedia Tools Appl. 76(6), 8257–8283 (2017)
42. Schaefer, G., Stich, M.: UCID - An uncompressed colour image database. In: Proceedings of SPIE, Storage and Retrieval Methods and Applications for Multimedia. vol. 5307, pp. 472–480.SPIE, Bellingham, WA (2004)
43. BOSSBase. http://dde.binghamton.edu/download/ (2020). Accessed 15 Oct 2020
44. Bows-2. http://bows2.ec-lille.fr/ (2020). Accessed 20 Oct 2020

How to cite this article: Tang Z, Pang M, Yu C, Fan G, Zhang X. Reversible data hiding for encrypted image based on adaptive prediction error coding. IET Image Process. 2021;15:2643–2655. https://doi.org/10.1049/ipr2.12252