Hindawi
Security and Communication Networks
Volume 2018, Article ID 9103418, 13 pages
https://doi.org/10.1155/2018/9103418

*Research Article*

# Reversible Data Hiding with Pixel Prediction and Additive Homomorphism for Encrypted Image

**Chunqiang Yu**,[1,2] **Xianquan Zhang**,[1,2,3] **Zhenjun Tang**,[3] **Yan Chen**,[1,2] **and Jingyu Huang**[4]

[1]*Network Information Center, Guangxi Normal University, Guilin 541004, China*
[2]*Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems, Guilin University of Electronic Technology, Guilin 541004, China*
[3]*Guangxi Key Lab of Multi-Source Information Mining & Security, Guangxi Normal University, Guilin 541004, China*
[4]*Fangchenggang College, Guangxi University of Finance and Economic, Fangchenggang 538000, China*

Correspondence should be addressed to Xianquan Zhang; zxq6622@163.com and Zhenjun Tang; tangzj230@163.com

Received 1 May 2018; Revised 27 June 2018; Accepted 17 July 2018; Published 4 September 2018

Academic Editor: David Megias



Data hiding in encrypted image is a recent popular topic of data security. In this paper, we propose a reversible data hiding algorithm with pixel prediction and additive homomorphism for encrypted image. Specifically, the proposed algorithm applies pixel prediction to the input image for generating a cover image for data embedding, referred to as the preprocessed image. The preprocessed image is then encrypted by additive homomorphism. Secret data is finally embedded into the encrypted image via modular 256 addition. During secret data extraction and image recovery, addition homomorphism and pixel prediction are jointly used. Experimental results demonstrate that the proposed algorithm can accurately recover original image and reach high embedding capacity and good visual quality. Comparisons show that the proposed algorithm outperforms some recent algorithms in embedding capacity and visual quality.

## 1. Introduction

Data hiding is an important technology for embedding secret data into a meaningful cover medium (such as an image or a video) to generate a stego-medium with a small distortion [1, 2]. Reversible data hiding (RDH) is a branch of data hiding, which can restore the original image from the stego-image after extraction of the embedded data. This restoration property of RDH plays an important role in those data-sensitive applications, such as medical imagery, military imagery, and law forensics, in which the cover image must be accurately restored.

In the past years, various RDH algorithms have been proposed. Generally, these RDH algorithms can be classified into three categories: lossless compression based algorithms, difference expansion (DE) based algorithms, and histogram shifting (HS) based algorithms. Lossless compression based algorithms vacate space for embedding secret message by losslessly compressing the least significant bit (LSB) planes or quantization residuals [3, 4]. They can be applied to image authentication and watermarking, but their embedding capacities are limited. DE based algorithms usually shift the difference of neighboring pixels for creating a vacant least significant bit (LSB) and append one secret bit to the vacated LSB [5, 6]. The HS based algorithms firstly shift the bins of histogram of gray values [7] or the predicted errors [8–12] for generating vacated space and then embed secret data into the vacated space. This kind of algorithms can provide a good trade-off between embedding capacity and visual quality.

Image encryption is a useful technique for protecting image content [13, 14]. It can convert an original image, known as plaintext image, into a meaningless image called the encrypted image. Since it cannot observe any useful information from the encrypted image, image content security is achieved. In some application scenarios, such as cloud storage, there are many encrypted images and people would

like to embed secret message into encrypted images for privacy protection. These applications require efficient RDH algorithms for encrypted images.

Recently, many researchers have developed various RDH algorithms in encrypted images since the encrypted images are widely generated and stored in cyberspace. For example, Zhang [15] proposed a famous RDH algorithm in encrypted image. This RDH algorithm divides the encrypted image into several blocks and embeds a secret bit into a block by flipping 3 LSBs of a half of pixels in the block. This algorithm extracts secret data by exploiting a fluctuation function in terms of spatial correlation in natural images. Zhang' algorithm is the preliminary research of RDH algorithms in encrypted images, but this algorithm will cause bit errors in the recovered image and the extracted data for some cases. Hong et al. [16] exploited the spatial correlation using a different estimation equation and side match technique to achieve a low error rate. To reduce error rate, Liao and Shu [17] designed a useful metric for measuring block complexity by considering neighbor pixels in terms of the locations. In another work, Qin and Zhang [18] presented an RDH scheme with capability of image content protection. This scheme only alters three LSBs of selected pixels of encrypted image for secret bit embedding. The above-mentioned algorithms rely on spatial correlation of original image when extracting data, and their procedures of image recovery and data extraction are similar. For the encrypted JPEG images, Qian et al. [19] exploited the coding/decoding principle of the JPEG image to design a reversible data hiding scheme.

To separate data extraction from image decryption, Zhang [20] proposed to compress the encrypted images to vacate room for data hiding. In [21], Qian and Zhang exploited distributed source coding to improve the embedding capacity of the RDH algorithm [20]. However, their encryption disorganizes spatial correlation of pixels and thus it is difficult to vacate space for data hiding. In another study, Yi et al. [22] exploited block permutation conduct image encryption for preserving spatial redundancy and used adaptive block-level based prediction-error expansion to conceal secret bits in the encrypted image blocks. Tang et al. [23] proposed a reversible data hiding algorithm in encrypted domain by exploiting alpha channel of portable network graphics (PNG) image. In this RDH algorithm, secret data is divided into some segments. For each segment, one bit is embedded into the LSB of encrypted pixel and other bits are hidden in the corresponding element of the alpha channel. A common feature of the above RDH algorithms is that they all embed secret data after image encryption.

Some researchers [24–27] have proposed the idea of preprocessing image before encryption for data embedding. For example, Ma et al. [24] provided an RDH algorithm in encrypted images by reserving room before encryption. This algorithm first empties out room by embedding LSBs of some pixels into other pixels with a traditional RDH method, then encrypts the image, and finally uses the positions of these LSBs in the encrypted image to conceal secret data. Xu et al. [25] designed a specific encryption mode to encrypt interpolation-error of nonsample pixels and embedded secret data into interpolation-error using a modified version of histogram shifting and difference expansion technique. In another work, Nguyen et al. [26] used half of the pixels to classify the rest of the pixels into smooth and complex regions to provide room for embedding additional data. Agrawal and Kumar [27] divided original image into several consecutive sets, calculated mean of each set, stored it at the first element of its set before encryption, and finally embedded a secret bit into the encrypted pixels of one set except for the pixel holding the mean value in the set.

Recently, homomorphic encryption is introduced to RDH algorithms in encrypted images [28–32]. In [28], the exclusive-or values of two neighboring pixels are reserved after image encryption so as to carry the additional data to be embedded. This is done by using the same pseudo-random bits to encrypt two neighboring pixels. Actually, the homomorphic property is equipped in the scheme [28]. Chen et al. [29] used Paillier encryption [33] to encrypt each pixel and embedded a secret bit into a pair of adjacent encrypted pixels. Shiu et al. [30] proceed a pair of adjacent original pixels using difference expansion to obtain a pair of odd or even pixels and then used Paillier encryption to encrypt the proceeded pixels. Similar to [29], this algorithm also embeds a secret bit into a pair of adjacent encrypted pixels. A common weakness of the use of Paillier encryption in [29, 30] is data expansion. In another study, Li et al. [31] divided image into several crosses, adopted the same key to encrypt pixels in each cross, and accommodated for data hiding by using the difference histogram of the encrypted image. Xiao et al. [32] divided image into several blocks and conducted data hiding in each block via additive homomorphism and pixel value ordering strategy. Zhang et al. [34] proposed a lossless and reversible data hiding algorithm for ciphertext images encrypted by public key cryptography with probabilistic and homomorphic properties. This algorithm shows better performances than some previous algorithms [15, 16, 20] in terms of PSNR in directly decrypted image.

Although some useful homomorphism encryption-based RDH algorithms [29–31] for encrypted image have been reported. Their embedding capacity and visual quality are not desirable yet. For example, Chen's method [29] and Shiu's method [30] can only reach 0.5 bpp, and Li' scheme [31] cannot reach 0.8 bpp for most test images. Aiming at these problems, we propose an error-free RDH algorithm in encrypted domain. Our algorithm not only perfectly recovers original images but also reaches high embedding capacity and good visual quality. Many experiments are conducted to validate our efficiency and the results demonstrate that our algorithm outperforms some recent algorithms. The remainder of this paper is organized as follows. Section 2 explains our algorithm and Section 3 discusses experimental results. Conclusions are finally drawn in Section 4.

## 2. Proposed Reversible Data Hiding

Our proposed algorithm consists of four components: image encryption with homomorphism, data embedding with additive homomorphism, data extraction, and image recovery.

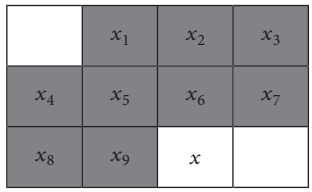



Figure 1: The current $x$ and its neighboring pixels.

Details of these components will be explained in the following sections.

*2.1. Image Encryption with Homomorphism.* This component can be divided into two parts: the preprocessed image generation with prediction and homomorphism encryption. In our RDH algorithm, secret data will be embedded into the encrypted image by using additive homomorphism.

*2.1.1. Preprocessed Image Generation with Prediction.* Predictors [35–38] are widely used in RDH algorithms for plaintext image. In this paper, we select an accurate gradient selective prediction (AGSP) predictor [38] to determine the original pixel predictions, which are used to construct preprocessed pixels. The reason of our selection of AGSP predictor is that it can reach a good balance between predictor performance and computational cost. The AGSP predictor operates on the nine neighboring pixels (i.e., $x_1$, $x_2$, $x_3$, $x_4$, $x_5$, $x_6$, $x_7$, $x_8$, and $x_9$) of the current pixel $x$, as shown in Figure 1. The estimated gradients of four directions (i.e., horizontal, vertical, 45 degrees, and -45 degrees) are denoted by $D_1$, $D_2$, $D_3$, and $D_4$. The corresponding pixels of these gradients $D_1$, $D_2$, $D_3$, and $D_4$ are $x_6$, $x_9$, $x_7$, and $x_5$, respectively. Suppose that $D_{\min1}$ and $D_{\min2}$ are two smallest gradients among $D_1$, $D_2$, $D_3$, and $D_4$ and their corresponding pixels are $C_{\min1}$ and $C_{\min2}$. The prediction of $x$ can be calculated as follows:

$$x' = \frac{D_{\min1} \times C_{\min2} + D_{\min2} \times C_{\min1}}{D_{\min1} + D_{\min2}} \quad (1)$$

Let $\mathbf{I}$ be an 8-bit grayscale uncompressed original image with $H \times W$ size and $I(i,j)$ be the pixel value in the $i$-th row and $j$-th column of $\mathbf{I}$, where $1 \le i \le H$, $1 \le j \le W$. As shown in Figure 2, the original image $\mathbf{I}$ is divided into four regions denoted as $\mathbf{\Omega_1}$, $\mathbf{\Omega}_2$, $\mathbf{\Omega}_3$, and $\mathbf{\Omega}_4$, where $\mathbf{\Omega}_1$ consists of all pixels used for embedding auxiliary information, $\mathbf{\Omega}_2$ consists of unused pixels, $\mathbf{\Omega}_3$ consists of pixels used for prediction, and $\mathbf{\Omega}_4$ consists of pixels used for embedding secret data. The sizes of $\mathbf{\Omega}_1$, $\mathbf{\Omega}_2$, $\mathbf{\Omega}_3$, and $\mathbf{\Omega}_4$ will be described in Section 2.2. Those pixels of $\mathbf{\Omega}_1$, $\mathbf{\Omega}_2$, and $\mathbf{\Omega}_3$ are kept unchanged in this stage. The pixels of $\mathbf{\Omega}_4$ are scanned from left to right and top to down, as shown in Figure 2. For the pixel $I(i,j)$ of $\mathbf{\Omega}_4$, if it is located in a texture region, it may have a poor prediction. Here the local smoothness estimator is defined as the invariance denoted by $Cp(i,j)$, which is used to determine whether $I(i,j)$ is located in a smooth region or not. Consequently, $Cp(i,j)$ can be determined as follows.

$$Cp(i,j) = V_{\max} - V_{\min} \quad (2)$$

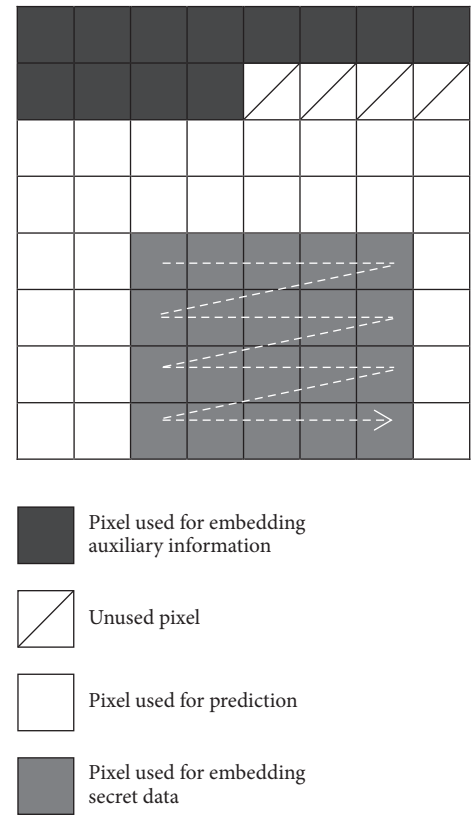



Figure 2: Different regions in the image.

where $V_{\max}$ and $V_{\min}$ are the maximum value and minimum value of the nine neighboring pixels of $I(i,j)$, respectively. If $Cp(i,j)$ is not bigger than a predefined threshold $T \in [0, 255]$, $I(i,j)$ and its neighboring pixels are highly correlative and it is considered as the pixel in smooth region. Otherwise, $I(i,j)$ is viewed as the pixel in textural region. Note that $T$ should be determined in terms of image content. In general, a small $T$ value is good for smooth image and a big $T$ value is suitable for textural image.

If $I(i,j)$ is located in a textural region, $I(i,j)$ is kept unchanged. If $I(i,j)$ is located in a smooth region, we calculate the prediction of $I(i,j)$ using AGSP predictor and denote it as $I(i,j)'$. Therefore, the prediction-error can be calculated as $D(i,j)=I(i,j) - I(i,j)'$. Since $I(i,j)$ is close to its predictive value $I(i,j)'$, $D(i,j)$ is small with a high probability. Consequently, the preprocessed pixel $I_p(i,j)$ can be calculated as follows:

$$I_p(i,j) = I(i,j) + (2^r - 1) \times D(i,j) \quad (3)$$

where $r$ ($r \ge 1$) implies how many bits can be embedded in the pixel and $r$ will be introduced in Section 2.2. Further, (3) can be rewritten as

$$I_p(i,j) = 2^r \times I(i,j) - (2^r - 1) \times I(i,j)'. \quad (4)$$

Note that $I_p(i,j)$ may cause underflow or overflow. To avoid this, a location map $L_1$ with $H \times W$ size is exploited to record the underflow or overflow pixels. Specifically, if $I_p(i,j) > 255$ or $I_p(i,j) < 0$, $L_1(i,j)=1$ and $I(i,j)$ is kept unchanged. Otherwise,

$L_1(i,j)$=0 and we use (3) to calculate $I_p(i,j)$. Let **F** be the preprocessed image and $F(i,j)$ $(1 \le i \le H, 1 \le j \le W)$ be the pixel value in the $i$-th row and $j$-th column of **F**. Thus, $F(i,j)$ can be obtained as follows.

$$F(i,j) = \begin{cases} I(i,j) & Cp > T \text{ or } L_1(i,j) = 1 \text{ or } (i,j) \notin \Omega_4 \\ 2^r \times I(i,j) - (2^r - 1) \times I(i,j)' & \text{Otherwise} \end{cases} \tag{5}$$

After the above calculation, the preprocessed image **F** is generated.

*2.1.2. Homomorphism Encryption.* Homomorphic encryption is an efficient encryption technology with the useful property that computation on ciphertext is equivalent to the computation on its plaintext. Here, a homomorphic encryption method [39] by addition modulo 256 is used. Suppose that $m$ and $c$ represent the plaintext and ciphertext. Thus, the homomorphic encryption and homomorphic decryption are described in (6) and (7), respectively,

$$c = E(m,k) = (m + k) \bmod 256 \tag{6}$$

$$m = D(c,k) = (c - k) \bmod 256 \tag{7}$$

where $k$ denotes encryption key and $E(\cdot,\cdot)$ and $D(\cdot,\cdot)$ are the encryption and decryption operations, respectively. An important property of the above encryption is the addition homomorphism in the plaintext domain. Let $m_1$ and $m_2$ be two different plain values whose corresponding random encryption keys are $k_1$ and $k_2$. Thus, we have the following equations:

$$E(m_1,k_1) \oplus E(m_2,k_2) = E(m_1 + m_2, k_1 + k_2) \bmod 256 \tag{8}$$

$$D(E(m_1,k_1) \oplus E(m_2,k_2), k_1 + k_2) = (m_1 + m_2) \bmod 256 \tag{9}$$

where $\oplus$ is modular 256 addition. Especially, if $k_2$ is equal to zero, $m_2$ is considered as a given signal. Therefore, (8) and (9) can be rewritten as

$$E(m_1,k_1) \oplus m_2 = E(m_1 + m_2, k_1) \bmod 256 \tag{10}$$

$$D(E(m_1,k_1) \oplus m_2, k_1) = (m_1 + m_2) \bmod 256 \tag{11}$$

From (10) and (11), it can be found that addition operation can be done directly on the encrypted data without decryption.

In this work, we exploit the RC4 [40] controlled by an encryption key *seed* to generate a key matrix $k(i,j)$ $(1 \le i \le H, 1 \le j \le W)$, encrypt $F(i,j)$ with the corresponding key $k(i,j)$ by (6), and obtain the encrypted pixel $C(i,j)$. After all pixels are processed, the encrypted image **C** is then available.

*2.2. Data Embedding with Additive Homomorphism.* Since the encrypted image **C** is generated using homomorphic encryption, we can embed secret data according to the property of additive homomorphism. Firstly, we convert a binary secret message into a sequence of secret digits in $2^r$-ary notational system. Then, secret digits can be embedded into those pixels of the encrypted image **C** in $\boldsymbol{\Omega}_4$ except the marked pixels by $L_1$. Let a secret digit be $s$ $(0 \le s \le 2^r - 1)$. For a pixel $C(i,j)$, if $Cp(i,j) > T$, $C(i,j)$ is kept unchanged; i.e., $C(i,j)' = C(i,j)$. Otherwise, we use the key whose value is 0 to encrypt $s$ and embed $s$ into $C(i,j)$ as follows.

$$C(i,j)' = C(i,j) \oplus s = E(F(i,j) + s, k(i,j)) \bmod 256 \tag{12}$$

Note that if $(F(i,j) + s) \ge 256$, the decrypted result of (10) should be $(F(i,j)+s) \bmod 256$. To accurately recover the original image and extract secret data, those pixels satisfying $(F(i,j) + s) \ge 256$ must be recorded. To do so, another location map $L_2$ is used here. Since $0 \le s \le 2^r - 1$, the location map $L_2$ records the pixels whose values are larger than or equal to $256 - s = 256 - (2^r - 1)$. Therefore, if $F(i,j) \ge 256 - (2^r - 1)$, $L_2(i,j)$=1 and $C(i,j)' = C(i,j)$. Otherwise, $L_2(i,j)$=0 and $s$ is embedded according to (12). Moreover, secret bits are not embedded into those pixels marked by $L_1$ and $L_2$. Note that the location map $L_1$ is generated in the process of image encryption, while the location map $L_2$ is generated during data hiding. Both location maps should be embedded into the cover image. To reduce the size of data embedded, lossless compression with arithmetic coding is exploited to compress the location maps.

To exactly extract the secret data and recover the original image, some auxiliary information must be embedded into the encrypted image, including the size of secret digits, capacity parameter $r$, the threshold $T$, the sizes of two compressed location maps, and the two compressed location maps. As the size of secret digits is less than or equal to the number of pixels in the original image, $\log_2 HW$ bits are enough to record it, where $H$ and $W$ are the height and width of the original image, respectively. Similarly, $2\log_2 HW$ bits are enough to save the sizes of two compressed location maps. Although a larger $r$ will bring higher embedding capacity, it will also degrade image quality according to (3). To balance the embedding capacity and image quality, the $r$ value should be less than or equal to 7. Therefore, 3 bits can represent the $r$ value. As the range of $T$ is [0, 255], 8 bits can represent the $T$ value. Let the size of compressed version of $L_1$ be $l_1$ and the size of compressed version of $L_2$ be $l_2$. Thus, the size of auxiliary information can be determined as follows.

$$t_{au} = l_1 + l_2 + 11 + 3\log_2 HW \tag{13}$$

Table 1 presents the auxiliary information and their data sizes.

Table 1: Auxiliary information and its sizes.

| Auxiliary information | Size (Bit) |
|---|---|
| Size of secret digits | $\log_2 HW$ |
| Capacity parameter $r$ | 3 |
| Threshold $T$ | 8 |
| Size of compressed version of $L_1$ | $\log_2 HW$ |
| Size of compressed version of $L_2$ | $\log_2 HW$ |
| Compressed version of $L_1$ | $l_1$ |
| Compressed version of $L_2$ | $l_2$ |

Here the auxiliary information is embedded into the 3 LSBs of the encrypted pixels in $\mathbf{\Omega}_1$ by using LSB replacement technique. Note that secret data and the 3 LSBs of the encrypted pixels in $\mathbf{\Omega}_1$ are firstly concatenated to construct the data for embedding into $\mathbf{\Omega}_4$ in our algorithm. As the 3 LSBs of the used encrypted pixels in $\mathbf{\Omega}_1$ are preserved in advance, they can be accurately recovered during image recovery.

Clearly, the number of pixels in $\mathbf{\Omega}_1$ is $t_1 = \lceil t_{au}/3 \rceil$, where $\lceil \cdot \rceil$ is the rounding function. The number of pixels in $\Omega_2$ is

$$t_2 = \begin{cases} 0 & \left\lceil \frac{t_{au}}{3} \right\rceil \bmod W = 0 \\ W - \left\lceil \frac{t_{au}}{3} \right\rceil \bmod W & \text{Otherwise.} \end{cases} \tag{14}$$

The number of pixels in $\Omega_3$ is $t_3=2\times W+3\times(H-\lceil \lceil t_{au}/3 \rceil / W \rceil - 2)$. Let the total number of the pixels marked by $L_1$ and $L_2$ in $\mathbf{\Omega}_4$ be $t_4$. Thus, the maximum pure embedding capacity (EC) of our RDH algorithm can be calculated as follows.

$$EC = r \times \left(H \times W - t_1 - t_2 - t_3 - t_4\right) - t_{au} \tag{15}$$

Note that the data hiding keys of our algorithm are the $r$ value and the $T$ value. In addition, the encrypted image containing secret data is called the marked encrypted image in the following sections.

*2.3. Data Extraction and Image Recovery.* When the receiver obtains a marked encrypted image, he/she can conduct different operations according to the knowledge of keys. If only the encryption key is available, he/she can decrypt the marked encrypted image to directly retrieve the decrypted image, which is approximate to the original image. If both the encryption key and data hiding key are known, he/she can extract secret data exactly and recover the original image without errors. In this case, auxiliary information is firstly extracted from the 3 LSBs of pixels in $\mathbf{\Omega}_1$. Then, the two compressed location maps are decompressed to retrieve the location maps $L_1$ and $L_2$. Next, we extract secret digits and recover the original pixels as follows.

If $C(i,j)' \in \mathbf{\Omega}_2$ or $C(i,j)' \in \mathbf{\Omega}_3$ or $C(i,j)' \in \mathbf{\Omega}_4$ and $L_1(i,j)=1$, we directly decrypt $C(i,j)'$ with $k(i,j)$ to recover the original pixel $I(i,j)$ according to (7).

If $C(i,j)' \in \mathbf{\Omega}_4$ and $L_1(i,j)=0$, we can obtain a decrypted pixel $F(i,j)'$ by $k(i,j)$. Since the original pixels which belong to $\mathbf{\Omega}_2$ or $\mathbf{\Omega}_3$ have been recovered, $Cp(i,j)$ can be calculated by AGSP predictor. If $Cp(i,j) > T$, $I(i,j)=F(i,j)'$. Otherwise, $Cp(i,j) \le T$ and secret digits are embedded by additive homomorphism. Clearly, if $L_2(i,j)=0$, $F(i,j)'=F(i,j)+s$. Otherwise $F(i,j)'=F(i,j)$. Clearly, $I(i,j)=(F(i,j)+(2^r-1)\times I(i,j)')/2^r$ by (5), where $I(i,j)'$ is the prediction which can be calculated by AGSP predictor. For $L_2(i,j)=1$, its original pixel can be determined by $I(i,j) = (F(i,j)+(2^r-1)\times I(i,j)')/2^r = \lfloor (F(i,j)' + (2^r-1)\times I(i,j)')/2^r \rfloor$, where $\lfloor \cdot \rfloor$ is the floor rounding operation. For $L_2(i,j)=0$, since $F(i,j)'=F(i,j)+s$, its original pixel $I(i,j)=(F(i,j)'-s+(2^r-1)\times I(i,j)')/2^r = \lfloor (F(i,j)' + (2^r-1)\times I(i,j)')/2^r \rfloor - \lfloor s/2^r \rfloor$. As $0 \le s \le 2^r-1$, $\lfloor s/2^r \rfloor$ is 0. Therefore, $I(i,j)=\lfloor (F(i,j)' + (2^r-1)\times I(i,j)')/2^r \rfloor$. Consequently, the original pixel can be uniformly rewritten as

$$I(i,j) = \left\lfloor \frac{F(i,j)' + (2^r-1)\times I(i,j)'}{2^r} \right\rfloor \tag{16}$$

Meanwhile, if $L_2(i,j)=0$, $F(i,j)'=F(i,j)+s=2^r\times I(i,j) - (2^r-1)\times I(i,j)'+s$. It is clear that $F(i,j)'-I(i,j)'=2^r\times(I(i,j)-I(i,j)')+s$. Then the secret digit $s$ can be extracted as follows.

$$s = \left(F(i,j)' - I(i,j)'\right) \bmod 2^r \tag{17}$$

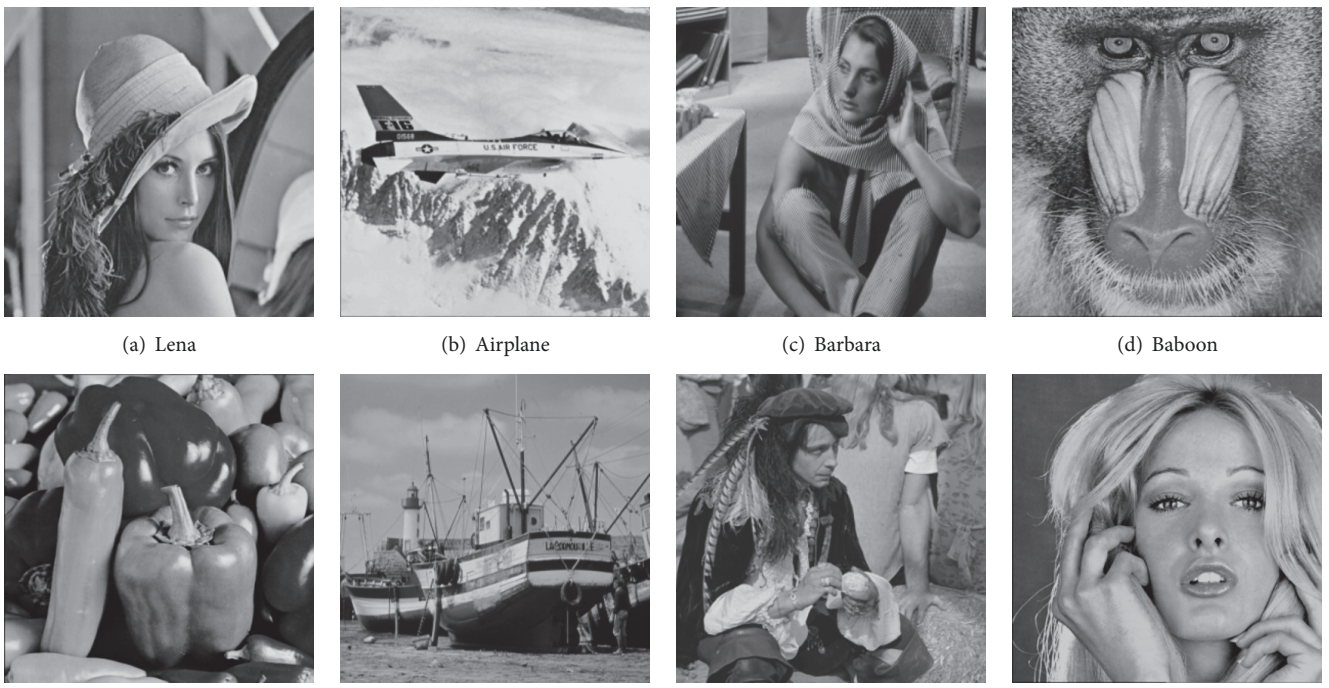
The recovered pixel $I(i,j)$ is used to recover the next pixel and extract the next secret digit. Consequently, the entire original pixels in $\mathbf{\Omega}_4$ are recovered and all secret digits are extracted according to the above scheme. Then, the secret digits are converted into a sequence of binary data. Note that the binary data is constructed by secret bits and the 3 LSBs of the used pixels in $\mathbf{\Omega}_1$. Consequently, the binary data is divided into two parts and the preserved 3 LSBs of the used pixels are used to replace the 3 LSBs of the corresponding pixels in $\mathbf{\Omega}_1$. After that, the encrypted pixels in $\mathbf{\Omega}_1$ are recovered and then they are decrypted to restore the original pixels. Finally, the original image is restored by the recovered pixels of the four regions.

## 3. Experimental Results

Many experiments are carried out to validate our performances. In the experiments, for a given embedding capacity, the threshold $T$ is varied from 0 to 255 until the embedding capacity is satisfied. Section 3.1 presents the performance of encryption, Section 3.2 discusses our embedding capacity and visual quality, and Section 3.3 analyzes comparisons with other RDH algorithms.

*3.1. Encryption Performance.* Eight images sized 512×512 as shown in Figure 3 are used to test the perceptual security and the statistical security of the proposed encryption scheme. For space limitation, we encrypt eight images with $r$=1 to obtain the encrypted versions, as shown in Figure 4. Clearly, the encrypted images are meaningless images, and we cannot observe any useful details of the original images from Figure 5. This verifies that the proposed encryption scheme is effective.

To quantitatively measure the performance of the proposed encryption scheme, the objective metrics including entropy and correlation coefficient are used. The information



(a) Lena (b) Airplane (c) Barbara (d) Baboon

(e) Peppers (f) Boats (g) Man (h) Tiffany

Figure 3: Eight test images.

entropy is an important measure of randomness, which is defined as follows.

$$H(E) = -\sum_{i=0}^{L-1} P(e_i) \log_2 P(e_2) \tag{18}$$

where $E=\{e_0, e_1, \ldots, e_{L-1}\}$ and $P(e_i)$ is the possibility of occurrence of $e_i$. In general, the bigger the entropy, the more secure the algorithm. For grayscale images, $L=256$ and the theoretical maximum value of entropy is 8. Table 2 presents the entropies of the encrypted images. It is found that all values are close to 8. This means that our encryption scheme is secure.

Here, the well-known correlation coefficient is exploited to measure correlation among an image pair. It is defined as

$$\rho_{XY} = \frac{\mathrm{Cov}(X, Y)}{\sqrt{D(X)}\sqrt{D(Y)}} \tag{19}$$

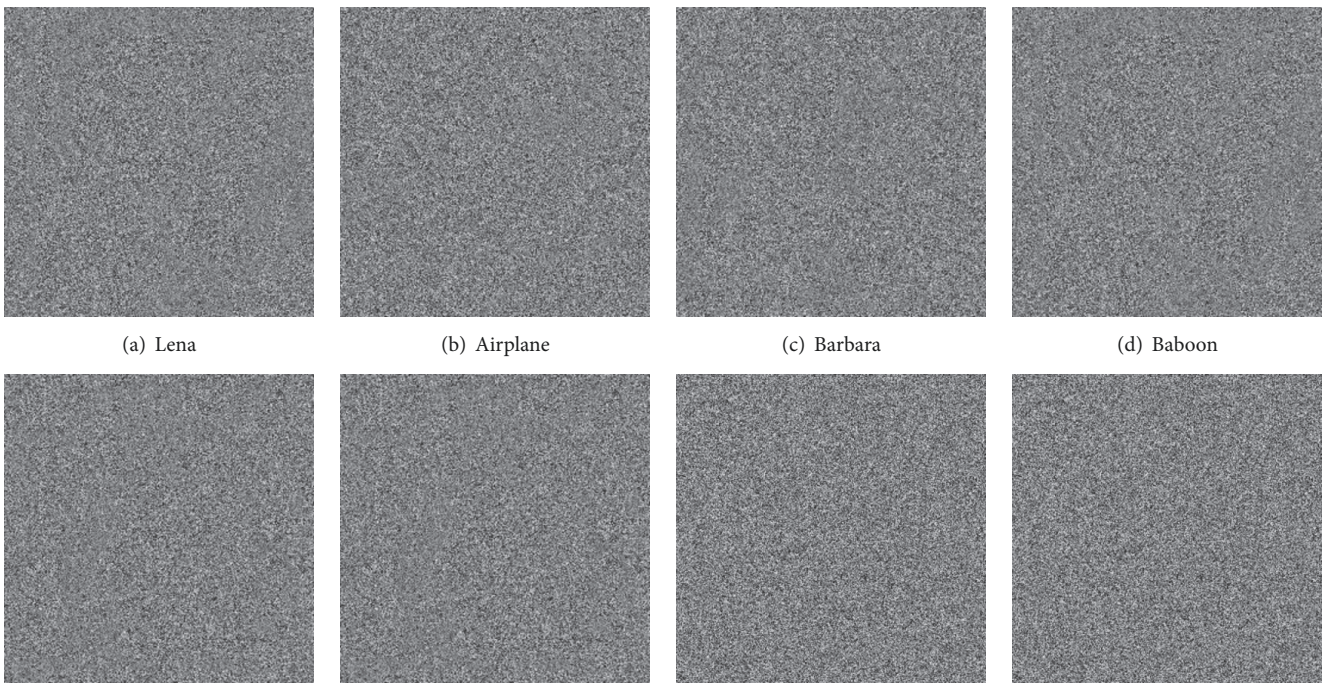
where $X$ and $Y$ denote the matrices of the original image and the encrypted image, respectively, Cov means the covariance, and $D$ means the variance. Table 2 illustrates the correlation coefficients between the original images and the encrypted images. It can be seen that the correlation coefficients are all almost 0. This means that there is almost no correlation between the encrypted image and the original image, indicating security of our encryption scheme.

*3.2. Embedding Capacity and Visual Quality.* When $T=255$, all pixels except those marked pixels by $L_1$ and $L_2$ in $\mathbf{\Omega_4}$ of the encrypted image can be used for data hiding.

Table 2: Results of entropy and correlation coefficient.

| Image | Entropy | Correlation coefficient |
|---|---|---|
| Lena | 7.9992 | −0.0003 |
| Airplane | 7.9993 | 0.0022 |
| Barbara | 7.9994 | 0.0005 |
| Baboon | 7.9993 | −0.0026 |
| Peppers | 7.9993 | 0.0007 |
| Boats | 7.9994 | −0.0002 |
| Man | 7.9993 | 0.0002 |
| Tiffany | 7.9993 | 0.0009 |

Obviously, the maximum pure embedding capacity of each image is determined by the parameter $r$, as shown in (12). Tables 3–6 list the pure embedding rate and PSNR of the directly decrypted images with different $r$ values. For space limitation, the directly decrypted images with $r=1$ are shown in Figures 5(a)–5(h). It is found that their visual qualities are satisfactory and their PSNR values are 30.6759, 32.3934, 28.458, 23.2063, 32.1131, 32.3676, 30.2753, and 30.7629. It can be also seen that there exists a maximum embedding rate for each image. In Tables 3–6, the texts in bold are the maximum embedding rates of the test images. For Lena, Airplane, Barbara, Baboon, Peppers, Boats, Man, and Tiffany, their maximum pure embedding rates are 1.947, 1.8433, 1.452, 1.002, 1.5292, 1.8122, 1.6755, and 1.7007 bpp when $r=3$, $r=3$, $r=2$, $r=2$, $r=3$, $r=3$, $r=2$, and $r=3$, respectively. Figure 6 presents the embedding rate of different test images under different $r$ values. As $r$ increases, the embedding rate also increases and



(a) Lena (b) Airplane (c) Barbara (d) Baboon

(e) Peppers (f) Boats (g) Man (h) Tiffany

Figure 4: The encrypted images with $r$=1.

(a) Lena (b) Airplane (c) Barbara (d) Baboon

(e) Peppers (f) Boats (g) Man (h) Tiffany

Figure 5: The directly decrypted images with $r$=1.

reaches maximum value when $r$=2 or $r$=3. A bigger $r$ means more pixels marked by $L_1$ and $L_2$ and then more auxiliary information should be recorded, which will decrease the pure embedding rate. In addition, the embedding rate and visual quality are both closely related to image contents. It is observed that smooth images will generally have bigger embedding rate and better visual quality than the textural images. That is the reason why the EC and PSNR of

Table 3: Embedding rates and PSNR of the directly decrypted images with $r$=1.

| Image | | Lena | Airplane | Barbara | Baboon | Peppers | Boats | Man | Tiffany |
|---|---|---|---|---|---|---|---|---|---|
| $t_{au}$ | | 10410 | 10042 | 9587 | 27066 | 20310 | 10721 | 11374 | 23430 |
| $t_1$ | | 3470 | 3347 | 3196 | 9022 | 6770 | 3574 | 3791 | 7810 |
| $t_2$ | | 114 | 237 | 388 | 194 | 398 | 10 | 305 | 382 |
| $t_3$ | | 2533 | 2533 | 2536 | 2500 | 2515 | 2533 | 2533 | 2509 |
| $t_4$ | | 1093 | 1099 | 1308 | 3041 | 2209 | 1099 | 1162 | 2615 |
| EC | bit | 244524 | 254928 | 254716 | 247387 | 250252 | 254928 | 254353 | 248828 |
| | bpp | 0.9725 | 0.9724 | 0.9716 | 0.9437 | 0.9546 | 0.9724 | 0.9702 | 0.9492 |
| PSNR(dB) | | 30.6759 | 32.3934 | 28.458 | 23.2063 | 32.1131 | 32.3676 | 30.2753 | 30.7629 |

Table 4: Embedding rates and PSNR of the directly decrypted images with $r$=2.

| Image | | Lena | Airplane | Barbara | Baboon | Peppers | Boats | Man | Tiffany |
|---|---|---|---|---|---|---|---|---|---|
| $t_{au}$ | | 37311 | 46145 | 68543 | 114244 | 62886 | 38897 | 38894 | 52038 |
| $t_1$ | | 12437 | 15382 | 22848 | 38081 | 20962 | 12966 | 12965 | 17346 |
| $t_2$ | | 363 | 490 | 192 | 319 | 30 | 346 | 347 | 62 |
| $t_3$ | | 2482 | 2464 | 2419 | 2332 | 2431 | 2479 | 2479 | 2452 |
| $t_4$ | | 4728 | 5897 | 12091 | 32900 | 10396 | 5232 | 7294 | 7774 |
| EC | bit | 446957 | 429677 | 380645 | 262780 | 393764 | 443345 | 439224 | 416982 |
| | bpp | 1.705 | 1.6391 | **1.452** | **1.002** | 1.5021 | 1.6912 | **1.6755** | 1.5907 |
| PSNR(dB) | | 23.78 | 23.4564 | 20.4752 | 16.9237 | 23.1429 | 23.3794 | 21.8245 | 21.6842 |

Table 5: Embedding rates and PSNR of the directly decrypted images with $r$=3.

| Image | | Lena | Airplane | Barbara | Baboon | Peppers | Boats | Man | Tiffany |
|---|---|---|---|---|---|---|---|---|---|
| $t_{au}$ | | 105911 | 115441 | 151522 | 246246 | 145015 | 119466 | 133242 | 129437 |
| $t_1$ | | 35304 | 38480 | 50507 | 82082 | 48338 | 39822 | 44414 | 43146 |
| $t_2$ | | 24 | 432 | 181 | 350 | 302 | 114 | 130 | 374 |
| $t_3$ | | 2347 | 2329 | 2257 | 2074 | 2272 | 2320 | 2293 | 2302 |
| $t_4$ | | 19058 | 21356 | 41230 | 87028 | 29270 | 21717 | 26726 | 24569 |
| EC | bit | 510394 | 483200 | 352385 | 25584 | 400871 | 475047 | 432504 | 445822 |
| | bpp | **1.947** | **1.8433** | 1.3442 | 0.0975 | **1.5292** | **1.8122** | 1.6499 | **1.7007** |
| PSNR(dB) | | 18.1015 | 18.5971 | 17.0775 | 14.771 | 17.3412 | 18.2071 | 17.295 | 16.6696 |

Table 6: Embedding rates and PSNR of the directly decrypted images with $r$=4.

| Image | | Lena | Airplane | Barbara | Baboon | Peppers | Boats | Man | Tiffany |
|---|---|---|---|---|---|---|---|---|---|
| $t_{au}$ | | 222009 | 249026 | - | - | 274612 | 224970 | - | 267648 |
| $t_1$ | | 74003 | 83009 | - | - | 91537 | 74990 | - | 89216 |
| $t_2$ | | 237 | 447 | - | - | 111 | 274 | - | 384 |
| $t_3$ | | 2119 | 2068 | - | - | 2017 | 2116 | - | 2032 |
| $t_4$ | | 55427 | 64158 | - | - | 73374 | 59165 | - | 72876 |
| EC | bit | 299423 | 200822 | - | - | 105808 | 277426 | - | 122896 |
| | bpp | 1.1422 | 0.766 | - | - | 0.404 | 1.0583 | - | 0.469 |
| PSNR(dB) | | 14.0578 | 14.0551 | - | - | 13.638 | 15.1664 | - | 13.8975 |

Barbara and Baboon are smaller than those of other test images.

Table 7 lists the PSNRs of the eight decrypted images containing embedded data under different embedding rates when $r$=1. Clearly, a high embedding rate will lead to a low PSNR. This is because more secret bits embedded will introduce more distortions on the encrypted images. Moreover, the PSNRs of the textural image Baboon are smaller than those of other test images. The reason is that the used prediction in our RDH algorithm is more effective in smooth images than textural images.

Table 7: PSNRs under different embedding rates when $r$=1 (dB).

| Image | Embedding rate (bpp) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0.005 | 0.01 | 0.05 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
| Lena | 63.55 | 60.55 | 58.03 | 50.38 | 46.88 | 44.61 | 42.73 | 40.74 | 37.96 | 35.14 | 33.98 | 31.56 |
| Airplane | 68.39 | 63.24 | 59.90 | 55.56 | 51.54 | 48.99 | 46.47 | 45.21 | 43.07 | 40.92 | 38.18 | 34.73 |
| Barbara | 63.82 | 60.82 | 53.58 | 50.26 | 46.88 | 44.61 | 42.64 | 40.74 | 37.55 | 34.60 | 32.01 | 29.81 |
| Baboon | 58.10 | 54.46 | 46.11 | 42.87 | 38.32 | 35.39 | 32.70 | 30.08 | 28.14 | 26.22 | 24.80 | 23.63 |
| Peppers | 65.12 | 61.66 | 51.96 | 48.22 | 44.20 | 41.76 | 40.30 | 38.59 | 37.50 | 36.39 | 35.08 | 33.58 |
| Boats | 65.18 | 62.25 | 54.98 | 51.69 | 48.48 | 46.44 | 44.74 | 42.98 | 40.69 | 38.53 | 36.16 | 34.12 |
| Man | 68.89 | 65.29 | 55.76 | 50.89 | 46.94 | 43.54 | 41.40 | 39.38 | 37.54 | 35.66 | 33.88 | 31.96 |
| Tiffany | 63.64 | 61.11 | 50.40 | 48.17 | 43.46 | 41.44 | 39.75 | 38.43 | 37.05 | 35.59 | 33.92 | 32.06 |
| Average | 64.59 | 61.17 | 53.84 | 49.76 | 46.33 | 43.35 | 41.34 | 39.52 | 37.44 | 35.38 | 33.50 | 31.43 |

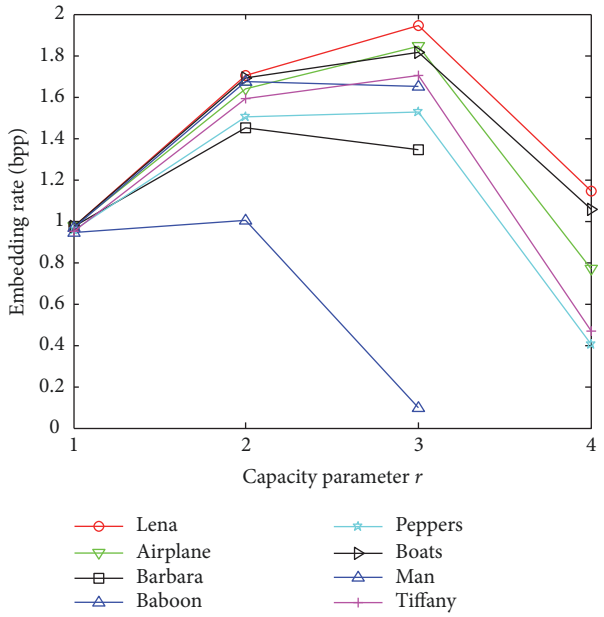



Figure 6: Embedding rates of different test images under different $r$ values.

*3.3. Performance Comparisons.* In this section, we compare the proposed algorithm with some recent popular RDH algorithms [22, 26, 27, 30, 31]. Figure 7 is the PSNR comparison among the assessed algorithms under different test images when $r$=1. It is observed that the PSNRs of the proposed algorithm are bigger than those of the compared algorithms [26, 27, 30] under any embedding rate for all test images. The RDH algorithm reported in [27] has lower performance than other algorithms due to the fact that it embeds several bits into hundreds of pixels. When the embedding rate is small, the PSNRs of the proposed algorithm are not always bigger than those of the compared algorithms [22, 31]. However, when the embedding rate is big, such as 0.8 and 0.9, the proposed algorithm will reach good performance. Specifically, for Lena, Peppers, and Tiffany, the PSNRs of the proposed algorithm are bigger than those of [22] when the embedding rate is bigger than 0.4 bpp. For Barbara, Baboon, Boats, and Man, the PSNRs of the proposed algorithm are bigger than those of [22] when the embedding rate exceeds 0.2 bpp. Moreover, the PSNRs of the proposed algorithm are bigger than those of [31] for all test images except Lena and Baboon.

Comparisons of the average PSNR of the eight test images among different algorithms are shown in Figure 8. Clearly, our average PSNRs are all bigger than those of the compared algorithms [26, 27, 30, 31] under different embedding rates. Our average PSNRs are all bigger than those of the compared algorithm [22] when the embedding rate is bigger than 0.2 bpp. Our proposed algorithm has better performance than the compared algorithms in terms of PSNR. This is contributed by our efficient strategies as follows. Firstly, during preprocessing image generation, image pixels are classified into four regions, i.e., $\mathbf{\Omega_1}$, $\mathbf{\Omega_2}$, $\mathbf{\Omega_3}$, and $\mathbf{\Omega_4}$, and only those pixels in $\mathbf{\Omega_1}$ and $\mathbf{\Omega_4}$ will be changed and the pixels in $\mathbf{\Omega_2}$ and $\mathbf{\Omega_3}$ are kept unchanged. It is clear that fewer pixels changed will lead to better visual quality. Secondly, for those pixels in $\mathbf{\Omega_4}$, data embedding is conducted by additive homomorphism, i.e., adding a number $s$ smaller than $2^r$ to pixel, which will not greatly change the pixel value. For example, if $r$=1, $s$ is 0 or 1. If $r$=2, $s$ may be 0, 1, 2, or 3. Thirdly, the location maps are compressed by arithmetic coding. This operation can reduce the amount of auxiliary information. Finally, auxiliary information is embedded into those pixels in $\mathbf{\Omega_1}$ by the LSB replacement technique, which introduces slight distortion on the image.

Moreover, the RDH algorithm [22] is not secure enough. In this algorithm, the users include image provider, data-hider, and receiver. The image provider encrypts input image by block permutation and stream encipher, where the block permutation only scrambles pixel positions and does not change pixel value. This means that the histogram of the original image is preserved after block permutation. During data embedding, data-hider must firstly decrypt the encrypted image by stream decipher (a reverse process of stream encipher), then conducts data embedding, and finally performs stream encipher again. In practice, image provider and data-hider are different persons, and the image provider will generally expect that anyone including the data-hider cannot observe any information of the encrypted image. However, as data-hider can decrypt encrypted image by stream decipher, he/she can obtain a scrambled image whose histogram is the

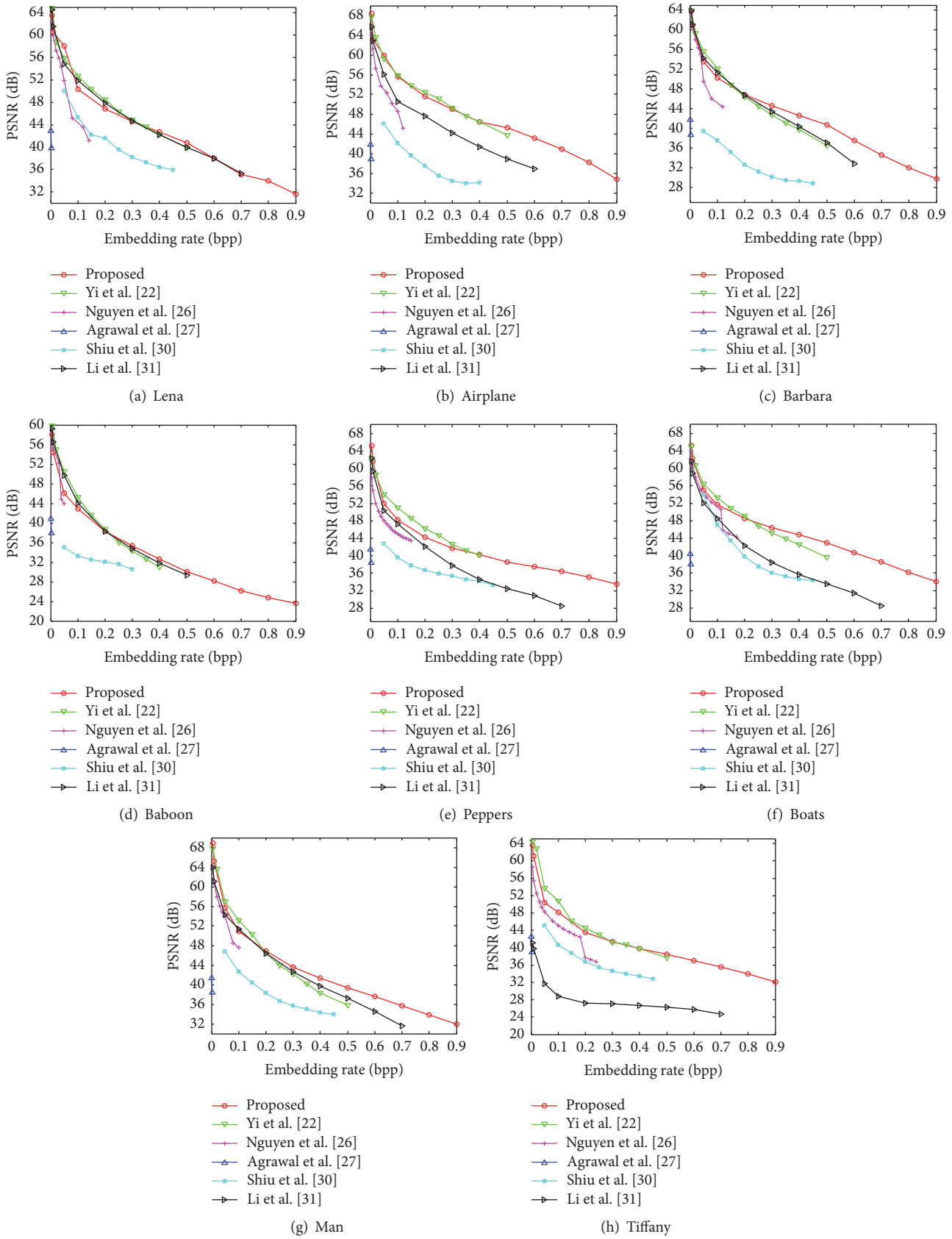



Figure 7: PSNR comparison under different test images.

Table 8: Running time of the assessed algorithms (unit: second).

| Algorithm | Lena | Airplane | Barbara | Baboon | Peppers | Boats | Man | Tiffany |
|---|---|---|---|---|---|---|---|---|
| [22] | 7.2546 | 6.1648 | 6.5787 | 8.9654 | 8.5644 | 6.2354 | 9.5578 | 7.2021 |
| [26] | 5.1344 | 4.2354 | 4.5441 | 5.6214 | 3.9241 | 5.6346 | 4.2364 | 5.3214 |
| [27] | 2.3547 | 2.1245 | 2.6544 | 2.4030 | 2.6456 | 2.2211 | 2.7895 | 2.5642 |
| [30] | 663.1231 | 664.145 | 665.1244 | 663.8789 | 664.564 | 665.2365 | 663.987 | 663.658 |
| [31] | 10.1244 | 9.2678 | 9.8641 | 10.9874 | 9.43241 | 10.2445 | 10.89741 | 9.3356 |
| Proposed | 5.3245 | 5.2345 | 5.5644 | 4.8645 | 4.6878 | 4.2156 | 4.3265 | 5.2364 |

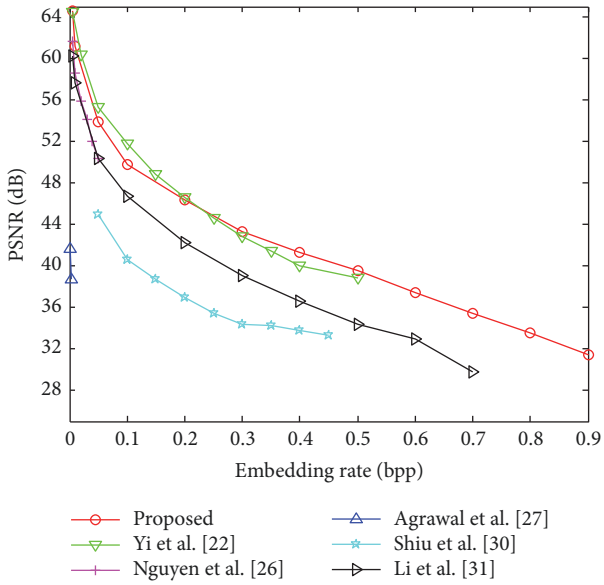



Figure 8: Average PSNR comparison under different embedding rates.

same as the original image. Consequently, he/she can further exploit those histogram-based image hashing algorithms [41, 42] to retrieve the original image of the scrambled image. This is a security loophole. For the proposed algorithm and other compared algorithms, image decryption is not needed during data embedding, and therefore there is no such security issue.

Computational time of the assessed algorithms is also compared. All RDH algorithms are coded in MATLAB R2011a and run on a computer with an Intel i5 CPU with 3.40 GHz, 4.00 GB memory, and Windows 10 operating system. Table 8 lists the running time of different algorithms under the different images. Note that the running time consists of image encryption, data embedding, image decryption, and secret extraction and image recovery for all the algorithms. From Table 8, it is found that the algorithm reported in [27] has the fastest speed and the algorithm reported in [30] has the lowest speed. The algorithm reported in [30] is time-consuming. This is due to the high complexity of Paillier encryption adopted in the algorithm [30]. The proposed algorithm and the algorithm in [26] have similar running time performances, which are both better than those of [22, 31].

## 4. Conclusions

In this paper, a reversible data hiding with pixel prediction and additive homomorphism for encrypted image has been proposed. The proposed algorithm constructs a preprocessed image from the original image by using pixel prediction before encryption, then encrypts the preprocessed image by additive homomorphism, and embeds secret data into the encrypted image via modular 256 addition. The proposed algorithm uses homomorphic encryption, but it does not lead to data expansion. Experimental results have demonstrated that the proposed algorithm can accurately recover original image and reach high embedding capacity and good visual quality. Comparisons have shown that the proposed algorithm outperforms some recent RDH algorithms in embedding capacity and visual quality.

## Data Availability

The images used to support the findings of this study are included within the article.

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

This work was supported by the National Natural Science Foundation of China (61562007, 61762017, 61363034, and 81701780), Guangxi "Bagui Scholar" Teams for Innovation and Research, the Guangxi Natural Science Foundation (2017GXNSFAA198222, 2015GXNSFDA139040, and 2017GXNSFBA198221), the Project of Guangxi Science and Technology (GuiKeAD17195062), the Project of the Guangxi Key Lab of Multi-Source Information Mining & Security (16-A-02-02, 15-A-02-02), and Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems (15202).

## References

[1] X. Zhang and S. Wang, "Efficient steganographic embedding by exploiting modification direction," *IEEE Communications Letters*, vol. 10, no. 11, pp. 781–783, 2006.

[2] X. Zhang, Z. Sun, Z. Tang, C. Yu, and X. Wang, "High capacity data hiding based on interpolated image," *Multimedia Tools and Applications*, vol. 76, no. 7, pp. 9195–9218, 2017.

[3] J. Fridrich, M. Goljan, and R. Du, "Lossless data embedding: new paradigm in digital watermarking," *EURASIP Journal on Applied Signal Processing*, vol. 2002, pp. 185–196, 2002.

[4] M. U. Celik, G. Sharma, A. M. Tekalp, and E. Saber, "Lossless generalized-LSB data embedding," *IEEE Transactions on Image Processing*, vol. 14, no. 2, pp. 253–266, 2005.

[5] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 890–896, 2003.

[6] A. M. Alattar, "Reversible watermark using the difference expansion of a generalized integer transform," *IEEE Transactions on Image Processing*, vol. 13, no. 8, pp. 1147–1156, 2004.

[7] Z. Ni, Y.-Q. Shi, N. Ansari, and W. Su, "Reversible data hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 3, pp. 354–361, 2006.

[8] X. Ma, Z. Pan, S. Hu, and L. Wang, "High-fidelity reversible data hiding scheme based on multi-predictor sorting and selecting mechanism," *Journal of Visual Communication and Image Representation*, vol. 28, pp. 71–82, 2015.

[9] H.-T. Wu, S. Tang, J. Huang, and Y.-Q. Shi, "A novel reversible data hiding method with image contrast enhancement," *Signal Processing: Image Communication*, vol. 62, pp. 64–73, 2018.

[10] L. Luo, Z. Chen, M. Chen, X. Zeng, and Z. Xiong, "Reversible image watermarking using interpolation technique," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 1, pp. 187–193, 2010.

[11] S. Weng, G. Zhang, J.-S. Pan, and Z. Zhou, "Optimal PPVO-based reversible data hiding," *Journal of Visual Communication and Image Representation*, vol. 48, pp. 317–328, 2017.

[12] W. He, K. Zhou, J. Cai, L. Wang, and G. Xiong, "Reversible data hiding using multi-pass pixel value ordering and prediction-error expansion," *Journal of Visual Communication and Image Representation*, vol. 49, pp. 351–360, 2017.

[13] Z. Tang, F. Wang, and X. Zhang, "Image encryption based on random projection partition and chaotic system," *Multimedia Tools and Applications*, vol. 76, no. 6, pp. 8257–8283, 2017.

[14] Z. Tang, J. Song, X. Zhang, and R. Sun, "Multiple-image encryption with bit-plane decomposition and chaotic maps," *Optics and Lasers in Engineering*, vol. 80, pp. 1–11, 2016.

[15] X. P. Zhang, "Reversible data hiding in encrypted image," *IEEE Signal Processing Letters*, vol. 18, no. 4, pp. 255–258, 2011.

[16] W. Hong, T.-S. Chen, and H.-Y. Wu, "An improved reversible data hiding in encrypted images using side match," *IEEE Signal Processing Letters*, vol. 19, no. 4, pp. 199–202, 2012.

[17] X. Liao and C. Shu, "Reversible data hiding in encrypted images based on absolute mean difference of multiple neighboring pixels," *Journal of Visual Communication and Image Representation*, vol. 28, pp. 21–27, 2015.

[18] C. Qin and X. Zhang, "Effective reversible data hiding in encrypted image with privacy protection for image content," *Journal of Visual Communication and Image Representation*, vol. 31, pp. 154–164, 2015.

[19] Z. Qian, X. Zhang, and S. Wang, "Reversible data hiding in encrypted JPEG bitstream," *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1486–1491, 2014.

[20] X. P. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 826–832, 2012.

[21] Z. Qian and X. Zhang, "Reversible data hiding in encrypted images with distributed source encoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 4, pp. 636–646, 2016.

[22] S. Yi, Y. Zhou, and Z. Hua, "Reversible data hiding in encrypted images using adaptive block-level prediction-error expansion," *Signal Processing: Image Communication*, vol. 64, pp. 78–88, 2018.

[23] Z. Tang, Q. Lu, H. Lao, C. Yu, and X. Zhang, "Error-free reversible data hiding with high capacity in encrypted image," *Optik - International Journal for Light and Electron Optics*, vol. 157, pp. 750–760, 2018.

[24] K. Ma, W. Zhang, X. Zhao, N. Yu, and F. Li, "Reversible data hiding in encrypted images by reserving room before encryption," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 3, pp. 553–562, 2013.

[25] D. Xu and R. Wang, "Separable and error-free reversible data hiding in encrypted images," *Signal Processing*, vol. 123, pp. 9–21, 2016.

[26] T.-S. Nguyen, C.-C. Chang, and W.-C. Chang, "High capacity reversible data hiding scheme for encrypted images," *Signal Processing: Image Communication*, vol. 44, pp. 84–91, 2016.

[27] S. Agrawal and M. Kumar, "Mean value based reversible data hiding in encrypted images," *Optik - International Journal for Light and Electron Optics*, vol. 130, pp. 922–934, 2017.

[28] X. Zhang, "Commutative reversible data hiding and encryption," *Security and Communication Networks*, vol. 6, no. 11, pp. 1396–1403, 2013.

[29] Y.-C. Chen, C.-W. Shiu, and G. Horng, "Encrypted signal-based reversible data hiding with public key cryptosystem," *Journal of Visual Communication and Image Representation*, vol. 25, no. 5, pp. 1164–1170, 2014.

[30] C.-W. Shiu, Y.-C. Chen, and W. Hong, "Encrypted image-based reversible data hiding with public key cryptography from difference expansion," *Signal Processing: Image Communication*, vol. 39, pp. 226–233, 2015.

[31] M. Li, D. Xiao, Y. Zhang, and H. Nan, "Reversible data hiding in encrypted images using cross division and additive homomorphism," *Signal Processing: Image Communication*, vol. 39, pp. 234–248, 2015.

[32] D. Xiao, Y. Xiang, H. Zheng, and Y. Wang, "Separable reversible data hiding in encrypted image based on pixel value ordering and additive homomorphism," *Journal of Visual Communication and Image Representation*, vol. 45, pp. 1–10, 2017.

[33] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 223–238, 1999.

[34] X. Zhang, J. Long, Z. Wang, and H. Cheng, "Lossless and Reversible Data Hiding in Encrypted Images with Public-Key Cryptography," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 9, pp. 1622–1631, 2016.

[35] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309–1324, 2000.

[36] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," *IEEE Transactions on Communications*, vol. 45, no. 4, pp. 437–444, 1997.

[37] J. Jiang, B. Guo, and S. Y. Yang, "Revisiting the JPEG-LS prediction scheme," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 147, pp. 575–580, 2000.

[38] H. Tang and S.-I. Kamata, "A gradient based predictive coding for lossless image compression," *IEICE Transaction on Information and Systems*, vol. 89, no. 7, pp. 2250–2256, 2006.

[39] A. V. Subramanyam, S. Emmanuel, and M. S. Kankanhalli, "Robust watermarking of compressed and encrypted JPEG2000 images," *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 703–716, 2012.

[40] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code*, John Wiley and Sons, Indianapolis, IN, USA, 2007.

[41] Z. Tang, X. Zhang, X. Li, and S. Zhang, "Robust image hashing with ring partition and invariant vector distance," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 200–214, 2016.

[42] Z. Tang, X. Zhang, and S. Zhang, "Robust perceptual image hashing based on ring partition and NMF," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 3, pp. 711–724, 2014.