Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

High capacity reversible data hiding in encrypted images based on adaptive quadtree partitioning and MSB prediction*

Kaili QI†‡, Minqing ZHANG, Fuqiang DI, Yongjun KONG

Key Laboratory of Network & Information Security under Chinese People Armed Police Force, Engineering University of PAP, Xi'an 710086, China

†E-mail: 1804480181@qq.com

Received Oct. 22, 2022; Revision accepted Feb. 14, 2023; Crosschecked July 26, 2023

Abstract: To improve the embedding capacity of reversible data hiding in encrypted images (RDH-EI), a new RDH-EI scheme is proposed based on adaptive quadtree partitioning and most significant bit (MSB) prediction. First, according to the smoothness of the image, the image is partitioned into blocks based on adaptive quadtree partitioning, and then blocks of different sizes are encrypted and scrambled at the block level to resist the analysis of the encrypted images. In the data embedding stage, the adaptive MSB prediction method proposed by Wang and He (2022) is improved by taking the upper-left pixel in the block as the target pixel, to predict other pixels to free up more embedding space. To the best of our knowledge, quadtree partitioning is first applied to RDH-EI. Simulation results show that the proposed method is reversible and separable, and that its average embedding capacity is improved. For gray images with a size of 512×512, the average embedding capacity is increased by 25 565 bits. For all smooth images with improved embedding capacity, the average embedding capacity is increased by about 35 530 bits.

Key words: Adaptive quadtree partitioning; Adaptive most significant bit (MSB) prediction; Reversible data hiding in encrypted images (RDH-EI); High embedding capacity

https://doi.org/10.1631/FITEE.2200501 **CLC number:** TP309.2

1 Introduction

With the rapid development of cloud computing, an increasing number of users choose to store data (such as images) in the cloud. However, while users enjoy convenience, their privacy security is under unprecedented threat. To protect the data privacy of image owners while performing necessary operations, information processing on encrypted images has attracted increasing attention (Zhang et al., 2014; Ke et al., 2020; Du et al., 2022; Yu et al., 2022). In many applications, for example, in cloud storage, to protect the images, the owners of images encrypt the images before sending them to the cloud server. After receiving the encrypted images, the cloud server embeds the secret data and thus can protect the privacy of the image owner. Moreover, information processing on encrypted images is conducted for image storage management, labeling, etc. Reversible data hiding in encrypted images (RDH-EI) has thus received much attention (Puech et al., 2008; Zhang, 2011; Liao and Shu, 2015). At present, RDH-EI schemes can be roughly divided into two categories: vacating room after encryption (VRAE) (Zhang, 2011; Zhou et al., 2016) and reserving room before encryption (RRBE) (Yi and Zhou, 2017; Puteaux and Puech, 2018).

In recent years, VRAE methods have included the original image encryption scheme using homomorphic encryption proposed by Xiong and Dong

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 62272478, 61872384, and 62102451) and the Basic Frontier Research Foundation of Engineering University of PAP, China (Nos. WJY202012 and WJY202112)

ORCID: Kaili QI, https://orcid.org/0000-0002-0578-9819

(2019) and Ke et al. (2020), and a general RDH-EI method based on most significant bit (MSB) prediction and error embedding proposed by Gao et al. (2023). Xiong and Dong (2019) used homomorphic encryption to encrypt the original image. The data hider embeds the secret data using prediction error expansion based on the redundancy of pixels in each block in the encrypted image. It was shown that the proposed method can recover the original image well only when the embedding rate was low; when the embedding rate was high, the quality of the reconstructed image was not satisfactory. The embedding rate was basically no more than 0.45 bits per pixel (bpp). Ke et al. (2020) used homomorphic encryption to encrypt the original image and designed a least significant bit (LSB) data hiding method based on key exchange, by which data could be directly extracted from the encrypted image without a private key. It was shown that the embedding rate of the VRAE method was relatively high, but it was also no more than 0.5 bpp. Gao et al. (2023) proposed an RDH-EI method based on MSB prediction and error embedding. Flags, errors, and message blocks were used to mark the location of prediction errors and data that could be embedded. This method can reconstruct the original image better and improve the embedding rate, but the embedding rate is below 1 bpp. The above reflects an important problem in current VRAE methods: data are hidden after image encryption, and the embedding capacity is limited because the redundancy between pixels is not fully considered; meanwhile, it is difficult to achieve lossless image reconstruction.

To ensure a high embedding rate and good image quality after data extraction and image restoration, Wang and He (2022) used the correlation between pixels and proposed a reversible information hiding scheme with a large capacity. To further improve the embedding capacity, in this paper we propose an improved scheme based on Wang and He (2022)'s method, and adopt adaptive quadtree partitioning according to the smoothness of the image to adaptively generate a number of blocks with different sizes. For blocks greater than 2×2, we use the upper-left pixels to predict other pixels, so that it can free up more space for embedding. As far as we know, this is the first time that adaptive quadtree partitioning has been applied to RDH-EI. The proposed algorithm combines adaptive MSB prediction with adaptive quadtree partitioning, and can implement adaptive quadtree partitioning according to the image texture and free up the space of embedded information. The proposed algorithm can increase the average embedding capacity based on improved adaptive MSB prediction, significantly improving the embedding rate of smooth images. For gray images with a size of 512×512, the average embedding capacity is increased by 25565 bits. For all smooth images with improved embedding capacity, the average embedding capacity is increased by about 35530 bits. What is more, the embedding capacity can be adjusted according to the threshold T, which enables our algorithm to have good practicability. Also, the proposed algorithm has reversibility and separability.

2 Related works

2.1 Analysis of adaptive MSB prediction

After the data hider receives the encrypted image, by adaptive MSB prediction the upper-left pixel is used as the target pixel for each pixel block, to predict the MSB shared among the remaining pixels, so that the remaining pixels can adaptively retain the remaining unshared bits to free up the embedding space.

In the encryption process of Wang and He (2022)'s method, after 2×2 block partitioning, block-level encryption and scrambling are conducted to resist the analysis of encrypted images. Fig. 1 shows the structure of each pixel block. Each pixel block is composed of four pixels P, C_1, C_2, and C_3. Pixel P is used to predict the three other pixels, and here it is the target pixel. For each pixel block, all the four pixels are first decomposed into binary 8 bits, and then the three variables D_1, D_2, and D_3 are obtained using the following formula:

$$D_i=\mathrm{dif}\left(P,C_i\right),\quad i=1,2,3, \tag{1}$$

where dif(a, b) returns the maximum number of MSBs that a and b share. For example, because 127=$(01111111)_2$, 98=$(01100010)_2$, we can obtain dif(127, 98)=3. This means that the number of MSB values shared by 127 and 98's binary 8 bits is at most three; that is, the first three bits are the same. According to variables D_1, D_2, and D_3, the minimum bit value

of MSB shared by P with C_1, C_2, and C_3 can be calculated, and the variable md can be calculated as follows:

$$\mathrm{md} = \min\left(D_1, D_2, D_3\right). \tag{2}$$

It can be seen from Fig. 1 that after pixel block reconstruction, P is first arranged, followed by the minimum number of shared MSBs (MD). Then, arrange the remaining binary value after removing the minimum number of shared MSBs of other pixels. Finally, the remaining bits can be used to embed additional information, which we define as nc. nc represents (3md−2) bits in quantity, indicating that each sub-block can free up (3md−2) bits.

Fig. 1 Structure demonstration and pixel bit reconstruction of a 2×2 block

Similarly, when the block is a sub-block larger than 2×2, the pixel structure demonstration and pixel bit reconstruction of a 4×4 block can be expressed as Fig. 2, using the improved prediction method proposed by Wang and He (2022).

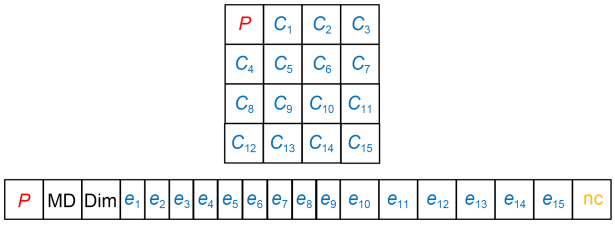


Fig. 2 Structure demonstration and pixel bit reconstruction of a 4×4 block

Pixel P is used to predict the other 15 pixels, which are the target pixels. For each reconstructed pixel block, pixel P is first arranged, followed by the size of the pixel block Dim, which is represented by two bits. Then, the minimum number of shared MSBs is arranged. Since the minimum number of shared MSBs md is within [1, 8], there are a total of eight possibilities. To express md uniformly with three bits, md−1 (MD) is used to represent the minimum number of shared MSBs, and the value range of MD here is [0, 7]. Next, the remaining binary values of the remaining pixels are arranged after removing the shared MSB values, and finally the binary values used to embed the data are arranged.

2.2 Quadtree partitioning

Quadtree is a typical tree-like data structure. The quadtree structure is designed to provide an efficient data organization structure. The quadtree block is implemented based on the quadtree structure. Because of its high efficiency in multiresolution partitioning of spatial data, quadtree partitioning has been widely used in image processing and other fields. The basic idea of quadtree segmentation is to iteratively and adaptively divide the square image (if the carrier image is not square, it can be extended by supplementing the background) according to the preset rules until all the image sub-blocks meet the rules. The whole quadtree structure can be represented by a binary group (I, rule). In each round, the current image block is divided into four sub-blocks: upper left, upper right, lower left, and lower right. Each sub-block corresponds to a node of the quadtree structure. Sub-blocks that do not meet the preset rule will be further divided into the next level's nodes, and sub-blocks that meet the rule will not be further divided and become leaf nodes. The quadtree partitioning rules can be set according to the actual situation (Di et al., 2019).

In the following we take a 16×16 image block as an example to briefly introduce the basic quadtree partitioning principle. Suppose that the rule is set as follows: the difference between the maximum and minimum pixel values of the image block is less than a threshold T. If the rule is met, the current node is set as a leaf node, represented by binary bit 1. If the rule is not met, the current node is divided into four child nodes in the quadtree, represented by $0b_1b_2b_3b_4$, where b_1,b_2,b_3, and b_4 correspond to the northwest, northeast, southwest, and southeast sub-blocks, respectively. Then, determine whether the child nodes meet the rule. If the rule is not met, continue to divide the child nodes (Di et al., 2019).

3 The proposed method

The overall structure of our algorithm is shown in Fig. 3, divided mainly into three stages: adaptive

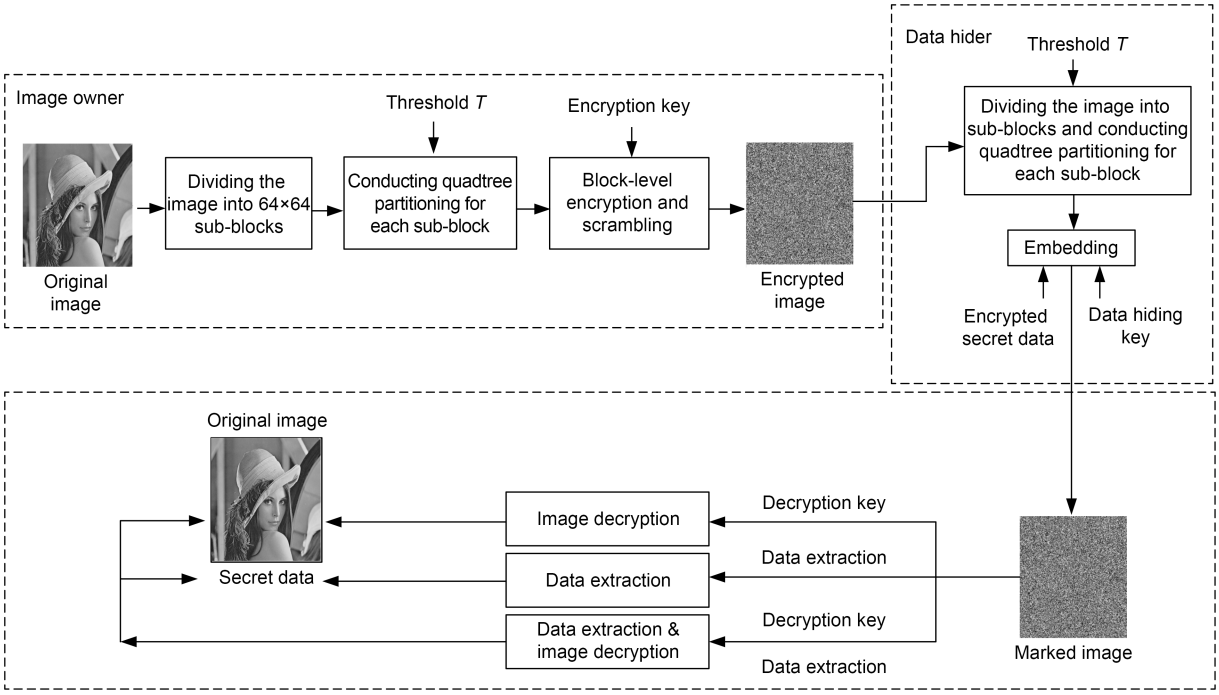


Fig. 3 Algorithm framework and process

quadtree partitioning and encryption, data hiding, and data extraction and image recovery. In the partitioning and encryption stage, first the original image is divided into several 64×64 sub-blocks, and then each sub-block is adaptively divided into blocks of different sizes using the quadtree partitioning method according to threshold *T*. The image owner uses the encryption key to encrypt the image and then sends it to the cloud server. In the data hiding stage, adaptive quadtree partitioning is used to divide the image, and for each block, improved adaptive MSB prediction is used to free up more space for the embedded data. Then, the encrypted secret data are embedded in the image, and the image is transmitted to the receiver. In the stage of data extraction and image recovery, the receiver uses the data hiding key to extract data directly and the decryption key to recover the original image.

3.1 Adaptive quadtree partitioning

The core problem of the quadtree coding algorithm is to determine the preset rule of quadtree partitioning and related parameters, that is, to determine when the image block meets the conditions for the next-level partitioning. For a specific *T*, the quadtree partitioning threshold is defined as the standard value which the minimum number of MSBs shared by the upper-left corner pixels and other pixels should be no less than. The owner adopts adaptive quadtree partitioning to the original image processing before image encryption. Initially, the $M\times N$ image is divided into several 64×64 non-overlapping sub-blocks. For each 64×64 sub-block, the owner should judge whether it meets the preset rule. If it meets the condition of partitioning, it will no longer be divided into four average blocks until a block of size 2×2 can no longer be partitioned. As shown in Fig. 4b, when *T* is set to 3, the 8×8 block at the upper-left corner of the Lena image shown in Fig. 4a has been partitioned into quadtree blocks.

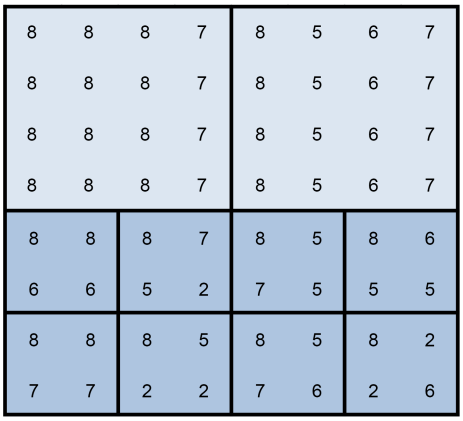
As shown in Fig. 4b, the image can be divided into sub-blocks of different sizes according to threshold *T*, which is the minimum value of md. Through such a threshold setting, blocks of different sizes can be divided according to the different smoothness of the image. Compared with the case in which the image is simply divided into 2×2 sub-blocks in Wang and He (2022), more embedding space can be freed up according to the different texture distributions of each image.

3.2 Image encryption

To protect the privacy of the image owner, the owner needs to encrypt the image before the original

162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
162	162	162	161	162	157	163	161
164	164	158	155	161	159	159	160
160	160	163	158	160	162	159	156
159	159	155	157	158	159	156	157

(a)



(b)

Fig. 4 Pixel values of the selected block (a) and the partitioning result (b)

image transmission. For the first encryption, all the pixels in the block are decomposed into eight binary pixels. Then, encryption keys and a classic stream cipher are used to perform block-level encryption, thus maintaining the relevance between the block pixels to some extent. The encryption formula of the k^{th} bit $b_m^k(i,j)$ of the m^{th} original pixel in sub-block (i,j) is encrypted into $e_m^k(i,j)$ as follows:

$$e_m^k(i,j) = b_m^k(i,j) \oplus r^k(i,j),\ k = 0, 1, \cdots, 7, \quad (3)$$

where $r^k(i,j)$ represents the k^{th} value of the element at (i,j) in the pseudorandom matrix.

For image encryption, a number of 64×64 sub-blocks are first scrambled. Next, in each stage of quadtree partitioning scrambling, four same-level blocks of each 64×64 sub-block are scrambled according to the key of scrambling, to resist the analysis of the encrypted image, improve the security of the encrypted image, and satisfy the image owner's demand on protecting the privacy of the image.

3.3 Adaptive quadtree MSB prediction

After receiving the encrypted image, the data hider adopts first adaptive MSB prediction (Wang and He, 2022) to free up a large amount of embedding space for data hiding and then adaptive quadtree partitioning. When the block size $a\times a$ is greater than 2×2, pixels of the block are reconstructed, first to arrange pixel P with eight bits, using the first three bits to arrange the minimum number of shared MSBs (MD), and then two bits are used to represent the size of the block (Dim). For each 64×64 block, after quadtree partitioning by setting T, the blocks have four sizes of 16×16, 8×8, 4×4, and 2×2, which are represented by 00, 01, 10, and 11, respectively. Next, arrange the binary values of the remaining pixels after removing the shared MSB value. Finally, arrange the binary value nc of embedding data, which is $\text{md}(a^2-1)-5$.

As shown in Fig. 5, when the pixel bits of 4×4 blocks are reconstructed to P=11111111, the minimum number of shared MSBs is md=6 (that is, MD=5=$(101)_2$), the type of block size is 4×4 (that is, Dim=10), then arrange the corresponding remaining binary values of other pixels, which are e_1=$(11)_2$, e_2=$(11)_2$, e_3=$(11)_2$, e_4=$(10)_2$, e_5=$(10)_2$, e_6=$(11)_2$, e_7=$(11)_2$, e_8=$(11)_2$, e_9=$(01)_2$, e_{10}=$(11)_2$, e_{11}=$(10)_2$, e_{12}=$(11)_2$, e_{13}=$(11)_2$, e_{14}=$(11)_2$, and e_{15}=$(01)_2$, and finally arrange the secret data, represented by nc. The secret data that can be embedded in this example are 85 bits.

255	255	255	255
254	254	255	255
255	253	255	254
255	255	255	253

Fig. 5 Pixel structure demonstration and pixel bit reconstruction of a 4×4 block

When the block size is 2×2, the block cannot be further divided, so there will be a kind of block that cannot be embedded in the data. When the block is available, the pixel bit reconstruction is as shown in Fig. 6a, first to arrange the top-left corner pixel P, then to arrange 11 to represent the size of the block, followed by "1" indicating an available block. Then, MD and

the remaining binary values of the three other pixels are arranged, and finally the secret data to be embedded are arranged.

When the block is unavailable, the reconstruction of pixels is as shown in Fig. 6b, first to arrange the top-left corner pixel P, then to arrange pixel block types Dim for 11, followed by 0 indicating that the block is unavailable. Next, MD and the remaining binary value of the three other pixels are arranged. Finally, the embedding secret data are arranged.

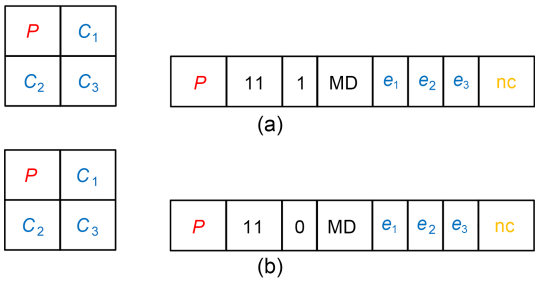


Fig. 6 Demonstration and pixel bit reconstruction of a 2×2 available (a) or unavailable (b) block

Since 2×2 blocks cannot be further divided, according to the characteristics of quadtree blocks, 2×2 blocks always appear as four consecutive blocks, so when representing 2×2 blocks, we need only to arrange the size of the block (Dim) in the pixel bit reconstruction of the first 2×2 blocks in the four consecutive blocks. For the remaining three blocks, we need only to directly arrange the value indicating an available or unavailable block after pixel P in the upper-left corner and then the remaining binary value of other pixels and secret data (nc) to be embedded. Each 2×2 block can thus be embedded with (3md−4.5) bits of data on average.

3.4 Data hiding

The data hider first divides the received encrypted image into several 64×64 sub-blocks and then performs adaptive quadtree partitioning for each sub-block according to T. In the image encryption stage, a number of 64×64 scrambled sub-blocks are first decrypted. After scrambling, four same-level blocks of each 64×64 sub-block are scrambled according to the key of encryption in each stage of quadtree partitioning scrambling. Therefore, in the data hiding stage, when the data hider performs adaptive quadtree partitioning for each 64×64 sub-block according to T, the partitioning result obtained by the data hider is the same as that sent to the data hider by the image owner after scrambling.

After adaptive quadtree partitioning, the data hider performs adaptive MSB prediction for each size of the block to free up space for embedding the secret data.

To prevent malicious attacks from reading the embedded data, classical symmetric cryptographic algorithms such as RC4 and AES are used to encrypt the secret data using the data hiding key.

Finally, the data hider embeds the encrypted secret data into each available block and side information such as the secret key, threshold T, and hidden key into the end of the image. The labeled encrypted image with embedded secret data is sent to the receiver.

3.5 Data extraction and image restoration

In the scheme designed in this study, data extraction and image restoration can be truly separated. There are three types of receivers based on the type of key they hold, and receivers with different types of keys will recover different contents.

When the receiver has only the decryption key, the original image can be reversibly restored. First, the labeled encrypted image is divided into several non-overlapping sub-blocks, and the pixels in each block are decomposed to obtain the corresponding bit sequence. Then, additional information is retrieved from the bit sequence of the last few blocks until the end tag is extracted. Then, adaptive MSB prediction is used for retrieval. First, P and Dim are retrieved from the bit sequence, followed by recognition of three bits to retrieve MD, and then e_i and hidden data. If Dim is retrieved as 00, 01, and 10, C_i is restored as

$$C_i = \mathrm{Trunc}(p, \mathrm{MD} + 1) + e_i, i = 1, 2, 3, \quad (4)$$

where Trunc(a, b) indicates that b MSBs in a are truncated, and "+" indicates that left join operation is performed. For example, in Fig. 4b C_1 is identified as C_1=(11111111, 6)+11=11111111, and the blocks in the first part are processed successively until Dim is identified as 11, indicating that the retrieved blocks and the next three blocks in a row are all 2×2 ones. At this time, one bit is further retrieved backwards. If the bit is 1, it indicates that the blocks are available,

and then MD is identified. If the bit is 0, it indicates that the block is unavailable, and then the search for this block is stopped but the search for the next block is continued until the end tag is recognized. After the pixel values of all blocks are identified, the decryption key is used for block-level decryption, and the order of all blocks is recovered; that is, the original image is obtained.

When the receiver has only the data hiding key, the original encrypted image and the encrypted secret data are obtained similarly, as described in the previous case. Next, the embedded secret data are decrypted using the data hiding key, and eventually only the secret data are reversibly recovered.

When the receiver possesses not only the decryption key but also the data hiding key, the receiver can perfectly recover the original image while obtaining the secret data.

4 Simulations

Compared with Wang and He (2022)'s method, the algorithm proposed in this study does not simply divide the image into blocks, but uses the adaptive quadtree partitioning algorithm to divide the image into blocks of different sizes according to the smoothness of the image, and then improves the adaptive MSB algorithm by adding tagged bits to predict the pixels. The improved algorithm fully considers the redundancy between pixels and can release more embedding space in most cases. At the same time, the algorithm has the characteristic of separability and can reversibly extract data and restore the image.

To compare the performances of our proposed method and Wang and He (2022)'s method, simulations are carried out for embedding rate comparison and embedding capacity comparison. The simulations are conducted in a computer with an Intel i7-10875H 2.30 GHz CPU, 32 GB RAM, and MATLAB R2018a. To more accurately analyze the effectiveness of the proposed scheme in improving the embedding capacity, eight standard test gray images with different texture features, obtained from the USC-SIPI image database (http://sipi.usc.edu/database/), are used as the test images.

4.1 Embedding capacity

The embedding performance about which data hiders are concerned is the number of data bits that can be embedded in the received encrypted images. Compared with Wang and He (2022)'s method, our method is more effective in improving the embedding capacity. Since MSB prediction is improved using adaptive quadtree partitioning, the redundancy between pixels is more fully considered than simply 2×2 blocks, and more space can be released for embedding.

As shown in Fig. 7, for the eight standard test images, the embedding capacity of the proposed method is greatly improved compared with that of Wang and He (2022)'s method.

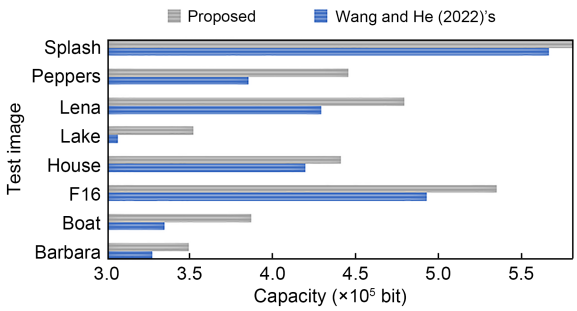


Fig. 7 Comparison of the embedding capacity between the two schemes

To better verify the embedding ability of our method, 100 grayscale images are randomly selected from the BOW-2 database (http://dde.binghamton.edu/download/), which has 10 000 images with strong statistical characteristics. It can be seen from Fig. 8 that the maximum embedding rate can reach about 4.30 bpp, and that the embedding rates of most images are above 1 bpp. Generally half of the images have an embedding rate higher than 2 bpp.

To better illustrate the embedding performance, the two schemes are compared on 100 images randomly selected from the BOW-2 database, and the results are shown in Table 1. The average embedding capacity of our method is increased by 25 565 bits, and the average embedding rate is increased by 0.0975 bpp. This proves the effectiveness of our proposed method in improving the embedding performance.

To obtain a comprehensive understanding of the embedding performance of our algorithm, the average embedding capacity and average embedding rate

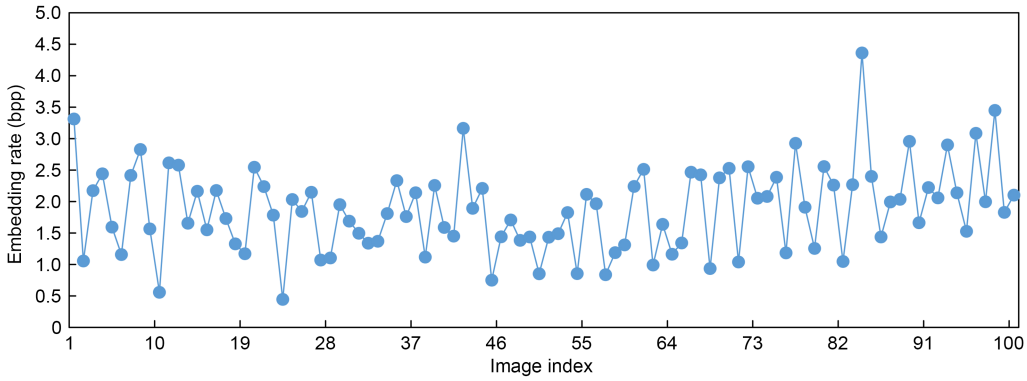


Fig. 8 Embedding rate of the proposed method on 100 images randomly selected from the BOW-2 image database

Table 1 Comparison of the embedded performance between the two schemes on 100 images randomly selected from the BOW-2 database

	Embedding capacity (bit)	Embedding rate (bpp)
Highest	1143691	4.3628
Lowest	116566	0.4447
Average	492111	1.8773
Highest improvement	148288	0.5656
Average improvement	25565	0.0975

Table 2 Comparison of the embedding performance on 100 images randomly seclected from the BOW-2 database between the two schemes

	EC_1 (bit)	ER_1 (bpp)	EC_2 (bit)	ER_2 (bpp)
Average	518062	2.1118	303796	1.1220
Average improvement	35530	0.1355	−9682	−0.0369

of 10 000 images are calculated. Simulation results show that for 10 000 images in the BOW-2 database, the average embedding capacity is 512 863 bits, and the average embedding rate is 1.9564 bpp, indicating that our algorithm has good embedding performance.

Among the 100 images randomly selected, the embedding capacity of 76 images with smooth texture has been improved. As shown in Table 2, EC_1 and ER_1 represent the embedding capacity and embedding rate of 76 images with improved embedding performance, respectively, and EC_2 and ER_2 represent the embedding capacity and embedding rate of 24 images without improved embedding performance, respectively. Among all images with improved embedding capacity, the average embedding capacity reaches 518 062 bits, and the average embedding rate reaches 2.1118 bpp. The average embedding capacity of our method is increased by 35 530 bits, and the average embedding rate is increased by 0.1355 bpp. However, for the remaining 24 images with complex texture, the average embedding capacity is decreased by 9682 bits, and the average embedding rate is decreased by 0.0369 bpp. Thus, the proposed method can adaptively partition the image according to the image smoothness, so that smooth texture images can have a great increase in embedding capacity. However, for the images with complex texture, more side information is required for quadtree partitioning; therefore, the embedding capacity is not improved.

For blocks greater than 2×2, although more than two tagged bits are used for each block, compared to Wang and He (2022)'s method, in the proposed method one block spares more space than two bits. Moreover, the proposed method does not need a location map. Therefore, for an image with smooth texture, the proposed method can obtain more blocks greater than 2×2, and the embedding capacity is larger than that of Wang and He (2022)'s method. For the 2×2 block, however, the number of tagged bits of each block is 1.5 more than that in Wang and He (2022)'s method. Although their method needs side information, through simulation discovery, for an image with more complex texture, there are more 2×2 blocks. Therefore, for images with complex texture, the embedding capacity of our method is lower. For the 2×2 blocks, using our method the increased number of tagged bits leads to less embedding space.

To further verify the effectiveness of the proposed method in increasing the embedding capacity, the proposed method is compared with the existing

methods (Khanam et al., 2016; Nguyen et al., 2016; Puteaux and Puech, 2018; Di et al., 2019; Bhardwaj and Aggarwal, 2020; Wang and He, 2022). As shown in Table 3, the embedding capacity of the proposed method is higher than those of the existing methods.

To better show that the embedding performance of our method is superior to that of the latest reversible information hiding method after encryption using MSB prediction, six standard test images with a size of 512×512, Airplane, Lena, Peppers, Baboon, Man, and Crowd, are tested using our method and Gao et al. (2023)'s method. Simulations show that the average embedding rate of Gao et al. (2023)'s method for six standard test images is 0.9385 bpp, while that of our method is 1.5776 bpp. Moreover, there have been some other recent VRAE methods, such as those proposed by Xiong and Dong (2019) and Ke et al. (2020). In Xiong and Dong (2019), the embedding capacity is almost no more than 0.45 bpp, and if the embedding capacity is high, the quality of the reconstructed image is not satisfactory. In Ke et al. (2020), the embedding rate is relatively high, but it is also no more than 0.5 bpp, and the average peak signal-to-noise ratio (PSNR) of the reconstructed image can reach 50 dB. However, the average embedding rate of our method reaches 1.8773 bpp and the average embedding rate for all smooth images reaches 2.1118 bpp, higher than those of the methods proposed by Ke et al. (2020) and Gao et al. (2023). Meanwhile, our method can achieve lossless image reconstruction.

4.2 Selection of parameter *T*

When the quadtree is partitioned, if the minimum number of shared MSBs of pixels in a block is smaller than *T*, the block is partitioned; otherwise, improved adaptive MSB prediction is performed. The proposed method enables adaptive quadtree partitioning according to the smoothness of the image by adjusting *T*. We can adjust *T* to make full use of the smoothness of the image, or choose a proper intensity to perform adaptive quadtree partitioning. In Table 4, $T=t$ (t=1, 2, ⋯, 8) indicates that when the maximum number of shared MSBs of pixels in a block is smaller than *t*, adaptive quadtree partitioning is performed. For the eight standard test images, adjusting the value of *T* will produce different embedding capacities, and each image has an optimal value, which enables the best use of the smoothness of the image to adaptively partition the quadtree to maximize the embedding capacity. For example, for Lena, when T=3, the embedding capacity reaches a maximum of 478 891 bits.

To analyze the relationship between the optimal *T* and the image texture in more detail, we perform a statistical analysis of the optimal *T* corresponding to the 100 images randomly selected from the BOW-2 database. As shown in Fig. 9, the optimal *T* is distributed mainly in three values 3, 4, and 5. Among them, T=3 accounts for 57%, and T=4 and T=5 account for 23% and 20%, respectively. By combination with image texture analysis, the image texture is smooth when *T* is 4 or 5, and most of the selected *T*'s are 3 when the image texture is complex.

4.3 Reversibility

Fig. 10 shows the line charts of embedding rate vs. PSNR of the existing methods (Cao et al., 2016; Nguyen et al., 2016; Chen and Chang, 2019; Xiong and Dong, 2019; Ke et al., 2020) and the proposed method on the Lena image. For different embedding rates, the PSNR of previous methods is no more than

Table 3 Comparison of embedding capacity among our method and state-of-the-art methods

Image	Embedding capacity (bit)						
	Khanam et al. (2016)	Bhardwaj and Aggarwal (2020)	Di et al. (2019)	Nguyen et al. (2016)	Puteaux and Puech (2018)	Wang and He (2022)	This paper
Barbara	225	436	25 180	34 342	253 576	327 263	349 297
Boat	1024	1875	29 561	27 262	259 864	334 343	386 905
F16	196	1250	48 722	73 139	260 208	492 674	534 581
House	81	162	26 085	44 009	261 024	419 121	440 927
Lake	1024	1323	25 137	29 750	259 896	306 395	352 139
Lena	1296	1875	38 588	41 157	258 760	428 661	478 891
Peppers	1296	1875	46 686	41 419	260 304	384 993	445 527

Table 4 Comparison of image embedding capacity with different *T*'s

Image	Embedding capacity (bit)							
	T=1	*T*=2	*T*=3	*T*=4	*T*=5	*T*=6	*T*=7	*T*=8
Barbara	295 755	338 287	**349 297**	320 520	280 133	269 592	269 511	269 511
Boat	360 908	**386 905**	368 788	303 909	279 938	274 863	274 599	274 599
F16	360 098	529 727	**534 581**	527 150	484 394	449 465	435 443	432 546
House	304 619	421 256	**440 927**	433 633	406 845	395 419	373 527	364 647
Lake	265 284	**352 139**	349 582	297 865	261 254	255 926	254 619	254 619
Lena	397 827	454 480	**478 891**	427 973	379 118	365 741	365 122	365 122
Peppers	370 312	**445 527**	429 639	364 670	325 083	321 105	321 105	321 105
Splash	440 894	582 370	**642 560**	585 376	520 215	498 406	491 571	489 200

For each image the maximum embedding capacity is in bold

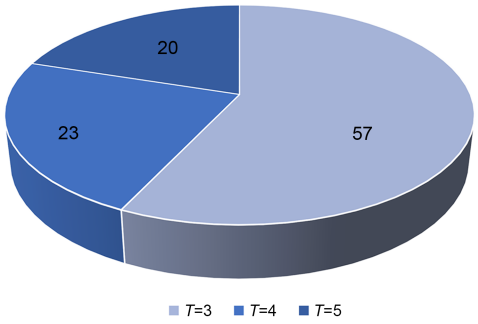


Fig. 9 Distribution diagram of the optimal parameter *T* in 100 randomly selected images

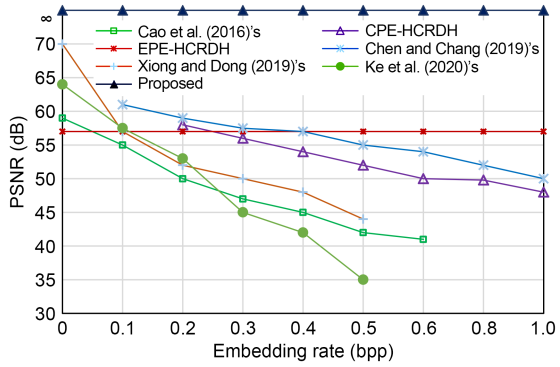


Fig. 10 Embedding rate vs. PSNR for our method and previous methods on image Lena

70 dB; in contrast, the PSNR of the proposed method is approaching $+\infty$ for any given embedding rate. In other words, the proposed method achieves true reversibility to restore the original image with a given embedding rate.

4.4 Security

To ensure the security of the encrypted images, the proposed method first adopts block-level encryption in the encryption stage. Fig. 11b shows the results of image complexity analysis after block-level encryption. It can be seen that the image can be revealed by complexity analysis to a certain extent. Fig. 11c shows the result of complexity analysis of the encrypted image after scrambling. The image cannot be revealed; that is, the proposed encryption method is resistant to the analysis of the encrypted image and ensures the privacy of the image owner.

To achieve more accurate and objective analysis, taking Lena and F16 as examples, we obtain the scatter diagrams of the original images, the encrypted images obtained by our method, and the marked encrypted images. As we can see from Figs. 12a and 12b, pixels are unevenly distributed, the information of the images can be easily analyzed, and the correlation between pixels is strong. As can be easily seen in Figs. 12c–12f,

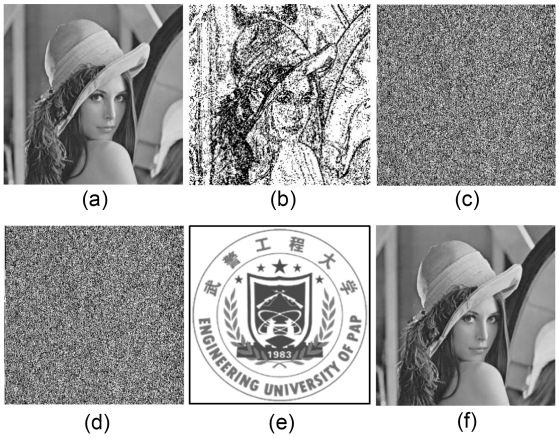


Fig. 11 Image demonstration of several key stages of the proposed method: (a) original image; (b) results of complexity analysis of the block-level encrypted image; (c) final encrypted image complexity analysis results; (d) marked encrypted image; (e) extracted secret data; (f) recovered image

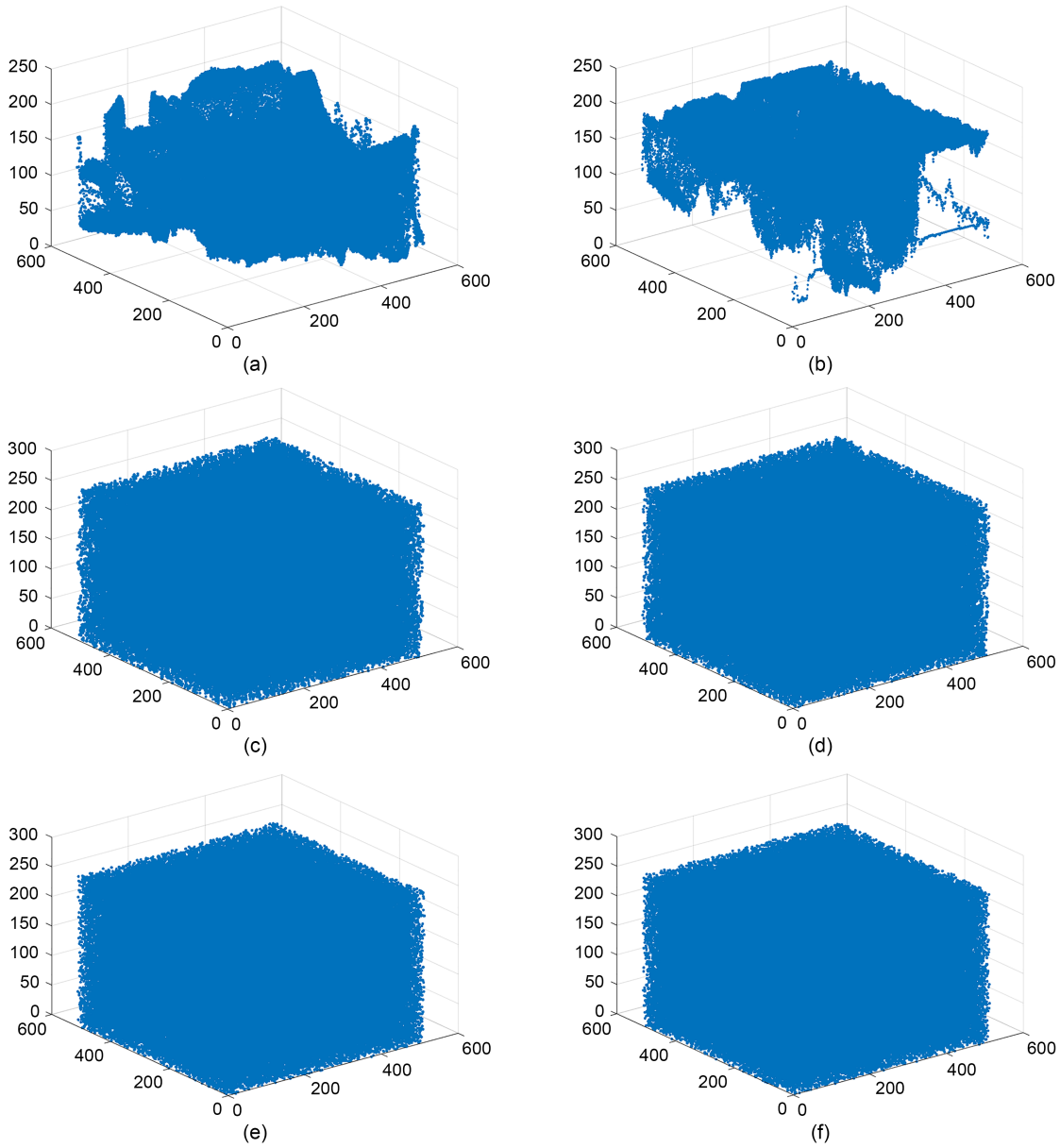


Fig. 12 Scatter maps of different images: (a) original image of Lena; (b) original image of F16; (c) encrypted image of Lena; (d) encrypted image of F16; (e) marked encrypted image of Lena; (f) marked encrypted image of F16

it is difficult to analyze the related information of the images, the correlation between pixels is weak, and the security of the image is strong. It can be concluded that the pixel correlation between the encrypted image obtained by the proposed method and the marked encrypted image is weak; that is, the proposed scheme can resist statistical analysis attacks.

4.5 Separability

To verify the separability of our method, we select the Lena image for image restoration and data extraction. When the receiver holds only the decryption key, as shown in Fig. 11f, it fully recovers the original image. When the receiver holds only the data hiding key, as shown in Fig. 11e, the embedding secret data are fully restored. When the receiver has both the decryption key and data hiding key, as shown in Figs. 11e and 11f, the receiver can fully recover the original image and the embedded secret data. That is, our method has the ability to extract the secret data and restore the original images.

5 Conclusions

We have proposed high capacity reversible data hiding in encrypted images based on adaptive quadtree partitioning and MSB prediction. This method is developed based on Wang and He (2022)'s method. It adopts adaptive quadtree partitioning according to the threshold *T* and the smoothness of the image while ensuring reversibility to restore the original image. It can adaptively change the embedding capacity by adjusting threshold *T*. For images with smooth textures, the embedding capacity of our method is greatly improved compared with that of Wang and He (2022)'s method. In addition, our method has separability and reversibility, and thus is more practical than previous methods.

In the future, we will further improve the universality and practicability of our method in increasing the image embedding capacity.

Contributors

Kaili QI designed the research. Kaili QI and Fuqiang DI processed the data. Kaili QI drafted the paper. Yongjun KONG helped organize the paper. Kaili QI and Minqing ZHANG revised and finalized the paper.

Compliance with ethics guidelines

Kaili QI, Minqing ZHANG, Fuqiang DI, and Yongjun KONG declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Bhardwaj R, Aggarwal A, 2020. An improved block based joint reversible data hiding in encrypted images by symmetric cryptosystem. *Patt Recogn Lett*, 139:60-68. https://doi.org/10.1016/j.patrec.2018.01.014

Cao XC, Du L, Wei XX, et al., 2016. High capacity reversible data hiding in encrypted images by patch-level sparse representation. *IEEE Trans Cybern*, 46(5):1132-1143. https://doi.org/10.1109/TCYB.2015.2423678

Chen KM, Chang CC, 2019. High-capacity reversible data hiding in encrypted images based on extended run-length coding and block-based MSB plane rearrangement. *J Vis Commun Image Represent*, 58:334-344. https://doi.org/10.1016/j.jvcir.2018.12.023

Di FQ, Zhang MQ, Liao X, et al., 2019. High-fidelity reversible data hiding by quadtree-based pixel value ordering. *Multim Tools Appl*, 78(6):7125-7141. https://doi.org/10.1007/s11042-018-6469-4

Du Y, Yin ZX, Zhang XP, 2022. High capacity lossless data hiding in JPEG bitstream based on general VLC mapping. *IEEE Trans Depend Sec Comput*, 19(2):1420-1433. https://doi.org/10.1109/TDSC.2020.3013326

Gao GY, Tong SK, Xia ZH, et al., 2023. A universal reversible data hiding method in encrypted image based on MSB prediction and error embedding. *IEEE Trans Cloud Comput*, 11(2):1692-1706. https://doi.org/10.1109/TCC.2022.3155744

Ke Y, Zhang MQ, Liu J, et al., 2020. Fully homomorphic encryption encapsulated difference expansion for reversible data hiding in encrypted domain. *IEEE Trans Circ Syst Video Technol*, 30(8):2353-2365. https://doi.org/10.1109/TCSVT.2019.2963393

Khanam FTZ, Song KY, Kim S, 2016. A modified reversible data hiding in encrypted image using enhanced measurement functions. Proc 8th Int Conf on Ubiquitous & Future Networks, p.869-872. https://doi.org/10.1109/ICUFN.2016.7537160

Liao X, Shu CW, 2015. Reversible data hiding in encrypted images based on absolute mean difference of multiple neighboring pixels. *J Vis Commun Image Represent*, 28: 21-27. https://doi.org/10.1016/j.jvcir.2014.12.007

Nguyen TS, Chang CC, Chang WC, 2016. High capacity reversible data hiding scheme for encrypted images. *Signal Process Image Commun*, 44:84-91. https://doi.org/10.1016/j.image.2016.03.010

Puech W, Chaumont M, Strauss O, 2008. A reversible data hiding method for encrypted images. *SPIE*, 6819:68191E. https://doi.org/10.1117/12.766754

Puteaux P, Puech W, 2018. An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images. *IEEE Trans Inform Forens Secur*, 13(7): 1670-1681. https://doi.org/10.1109/TIFS.2018.2799381

Wang YM, He WG, 2022. High capacity reversible data hiding in encrypted image based on adaptive MSB prediction. *IEEE Trans Multim*, 24:1288-1298. https://doi.org/10.1109/TMM.2021.3062699

Xiong LZ, Dong DP, 2019. Reversible data hiding in encrypted images with somewhat homomorphic encryption based on sorting block-level prediction-error expansion. *J Inform Secur Appl*, 47:78-85. https://doi.org/10.1016/j.jisa.2019.04.005

Yi S, Zhou YC, 2017. Binary-block embedding for reversible data hiding in encrypted images. *Signal Process*, 133:40-51. https://doi.org/10.1016/j.sigpro.2016.10.017

Yu CQ, Zhang XQ, Zhang XP, et al., 2022. Reversible data hiding with hierarchical embedding for encrypted images. *IEEE Trans Circ Syst Video Technol*, 32(2):451-466.

https://doi.org/10.1109/TCSVT.2021.3062947

Zhang XP, 2011. Reversible data hiding in encrypted image. *IEEE Signal Process Lett*, 18(4):255-258. https://doi.org/10.1109/LSP.2011.2114651

Zhang XP, Ren YL, Shen LQ, et al., 2014. Compressing encrypted images with auxiliary information. *IEEE Trans Multim*, 16(5):1327-1336. https://doi.org/10.1109/TMM.2014.2315974

Zhou JT, Sun WW, Dong L, et al., 2016. Secure reversible image data hiding over encrypted domain via key modulation. *IEEE Trans Circ Syst Video Technol*, 26(3):441-452. https://doi.org/10.1109/TCSVT.2015.2416591