Journal of Autonomous Intelligence (2024) Volume 7 Issue 1
doi: 10.32629/jai.v7i1.735

ORIGINAL RESEARCH ARTICLE

High-capacity reversible data hiding by recursive look-ahead in encrypted images adaptive MSB prediction for the cloud computing

Selwyn Paul J*, R. Suchithra

Department of Computer Science, Chirashreee Institute of Research & Development (CIRD), Bengaluru 560010, India

*** Corresponding author:** Selwyn Paul J, Selwyn.paul@sju.edu.in

ABSTRACT

The present research proposes a high-capacity reversible data embedding technique (RDH-EI) for applications utilizing cloud storage that use a recursive look ahead adaptive MSB prediction approach. The paper also proposes a three-level image encryption approach where on the first level, the image is subdivided into 4 × 4 blocks and the blocks are permuted. In the second level, two different neighbourhoods namely the middle neighbourhood and peripheral neighbourhood are estimated, from which the elements of the middle neighbourhood are permuted within the respective neighbourhood. In the third level, the elements of the peripheral neighbourhood without including the elements of the middle neighbourhood are permuted within the respective neighbourhood. The image that was encrypted by the image owner was uploaded to the cloud. Out of five different neighbourhoods estimated on the 4 × 4 blocks, the best neighbourhood is then estimated from the encrypted image which provides a maximum embedding capacity. The data is then embedded in the best neighbourhood using the adaptive MSB prediction algorithm. Once the embedding iteration is completed, the marked encrypted image obtained on the previous iteration is again evaluated for the best neighbourhood. This recursive process is repeated till the embedding capacity of the best neighbourhood is less than the threshold. The standard test images from the datasets BOSS base, UCID and BOWS-2 were used to evaluate the proposed approach using measures like embedding rate, SSIM and PSNR. The suggested method offers an average embedding rate that is greater than previous recent RDH-EI approaches, at 3.714 bpp, 3.4826 bpp and 2.9412 bpp for the datasets BOSS base, BOWS-2 and UCID, respectively.

Keywords: permutation; encrypted images; a recursive process; adaptive MSB predictor; reversible data hiding

ARTICLE INFO

Received: 15 June 2023
Accepted: 25 July 2023
Available online: 7 October 2023

1. Introduction

Numerous users upload their private data to cloud storage as a result of the cloud technology's rapid growth. Integrity, authentication, and secrecy are a few of the threats to cloud storage, though. To protect the privacy of content, encryption is a solution. Reversible data concealment is supported by a number of approaches, such as prediction error[1], integer transforms[2], histogram shifting[3], and difference expansion[4]. Before being uploaded to the cloud, the original image is often encrypted in order to preserve its privacy-related information. With the integer transform method, the embedding capacity is larger. After embedding the data, the histogram shifting-based method produces an image with good visual quality. The capacity is considerably enhanced by the fact that the data is placed on peak points of the histogram. The difference value computed between two adjacent pixels is enlarged to embed the data in difference expansion schemes. A significant degree of distortion and poor embedding capacity are produced by the lossless compression-based method. There is a risk of privacy leaking in a cloud that is only

partially trustworthy. Based on RDH-EI, there are two methods for protecting image privacy: (i) reserving the room before encryption; and (ii) vacating the room after encryption.

After encryption, the user leaves the room and uploads the image to the cloud, where the cloud embeds the extra data needed to manage the encrypted image. Zhang[5] performs an exclusive-OR operation, which embeds the extra information by flipping three LSBs of the pixel in the encrypted image. Data is also embedded in various LSBs using multi-layer wet paper coding[6]. A parity check matrix compresses the LSB plane to embed the data[7]. Qian and Zhang[8] employed a low-density parity check to create a spare room. Chen et al.[9] presented lightweight cryptography with multi-secret sharing, where no secret key is shared between the data hider and receiver.

Before encryption, while reserving a room, the plain image of the room is reserved. There is enough data included in the encrypted image that was uploaded to the cloud to handle it. Zhang et al.[10] employed prediction error and the histogram of prediction error to incorporate the data. The LSBs of a few pixels are integrated into other pixels to save space Ma et al.[11]. Patch-level representation was employed by Cao et al.[12] to increase embedding capacity. This room reservation before encryption offers a huge payload.

The following is the paper's contribution. The study suggests an approach for embedding adaptive MSB predictor data on encrypted images that is recursive and looks ahead. The research also suggests data encryption based on three-level permutations to encrypt the photos. The best neighbourhood with the highest embedding capacity is found by the look-ahead adaptive data embedding algorithm, which then embeds the data there. Finally, the PSNR, SSIM and embedding rate metrics were used to evaluate the suggested approach.

The remaining sections are organised as shown below. In Section 2, a few of the connected works are covered. The suggested recursive look-ahead adaptive MSB predictor data embedding and extraction method is shown in Section 3. Additionally, it demonstrates how the three-level permutation-based Image encryption/decryption system functions. The experimental outcomes of the suggested data embedding approach are shown in Section 4. Section 5 brings the process to a close.

This research work contributes a high-capacity reversible data embedding technique (RDH-EI) for cloud storage applications, featuring a three-level image encryption process and a recursive look ahead adaptive MSB prediction approach. It achieves superior embedding rates compared to previous methods, offering potential advancements in secure and efficient data hiding within cloud-based environments.

2. Related works

The image, which resembles a chess board, was divided into two groups by the authors Wu and Sun[13]: the qualifying set and the banned set. The qualifying set is the only set that is used to embed the data; the banned set is not utilised. The qualified set is also broken into smaller segments by the authors Nguyen et al.[14], who used smooth areas to insert the extra data. The most important bit was changed by author Mansoor Mohammadi et al.[15] in order to incorporate the data. A prediction approach is employed during the extraction phase to rebuild the original image. Block permutation and a particular stream encryption technique were employed by Huang et al.[16] to encrypt data. To embed the data, the plain image is divided into blocks, and the positions of the blocks are changed using a process known as scrambling. The data is concealed in these components using a histogram shifting method based on a bit-plane parameter according to Di et al.[17] classification of the bit-planes received from the encrypted image into two components.

When embedding the data, pixel value ordering[18] is employed. The plain image is broken into 22 blocks, and each block is then block permuted to produce the encrypted image. Block-based encryption is employed by Tang et al.[19] to communicate spatial correlation between the adjacent pixels, where the compression is carried out among the encrypted pixel difference to vacate the room. In addition, Huffman coding[20] MSB bit-

plane and adaptive block encoding are utilized, with positive outcomes. Yi and Zhou[21] suggested a binary tree labelling method that embeds the data on the encrypted image. Puteaux and Puech[22] embed the data on MSB instead of LSB, flipping the MSB of the plain picture pixel to obtain its inverse. Estimates are made of the absolute difference between the inverse values. Similar to that, an estimate is also made of the absolute difference between the original pixel value and the predicted value. If the difference between the inverse value and prediction value is greater than the difference between the original pixel value and prediction value, one hidden bit may be encoded. If the requirement is not met, the data cannot be incorporated. With this method, the embedding rate is about 1 bpp.

The number of subsequent MSB with the same value was computed by Yin et al.[23] and the difference between the prediction value and the original pixel value was used to embed the data. The additional data is compressed using Huffman coding. The pixels whose value is closer to the prediction values are not used, despite the fact that this strategy offers a high embedding rate. The authors Wu et al.[24] suggested a recursive method from MSB to LSB using a combination of encryption, reversible adaptation, error prediction, and embedding. Every bit plane is rebuilt during the extraction phase from the LSB to the MSB plane Puteaux and Puech[25] proposed a smart RDH-EI that estimates the prediction error between the reference pixel and the last remaining pixel of each block. The estimated prediction error value determines the potential payload for each pixel. The RDH-EI that the authors Wang and He[26] presented groups the image into various blocks. The encryption is then carried out block-by-block. The top-left pixel in a 22 block is utilised to anticipate the other pixels during the embedding step.

Data is concealed in two phases via a hierarchical embedding, as employed by Yu et al.[27]. Using a prediction approach, a hierarchical label map is created from the bit-plane of the plain text picture in the first stage. After compression, a label map is embedded in the second stage. Three different prediction errors of various sizes, including large, medium and tiny are estimated; the prediction errors of large and small sizes are employed to carry secret data. By using vector quantization compression to identify the pixels that need to leave the room before encryption, Xu et al.[28] offer vector quantization prediction. In the embedding process, adaptive block selection is utilised, where different data sizes are embedded on various blocks dependent on their type.

A double linear regression predictor with additional auxiliary rooms is employed by Li et al.[29] for secret data embedding. For data embedding, Yin et al.[30] combined pixel prediction with a multi-MSB plane. The median edge detector first estimates the prediction value, and one bit-plane is represented by the sign of the prediction error while the other bit-planes are represented by the absolute values of the prediction error. To embed the data adaptively, the bit-planes are then divided into uniform and non-uniform blocks. A data embedding approach using different time Arnold transform, sub-block position disordering, and bit-plane disordering was proposed by the authors Bas et al.[31]. A 2D-logistic modified sign map is utilised as encryption to increase security. The research suggests iteratively embedding the data using an adaptive look-ahead embedding method. The block with the highest embedding capacity is estimated prior to embedding the data in it. The suggested data embedding method is displayed in the next section.

3. Proposed method

Block diagram of a novel data embedding and extraction procedure is shown in **Figure 1**.

The block diagram of the suggested data embedding and extraction technique is shown in **Figure 1**. Three-level block permute image encryption, recursive look ahead adaptive MSB predictor data embedding, recursive look ahead adaptive MSB predictor data extraction, and three-level block permute image decryption are the four operations that make up the proposed reversible data hiding approach. Let G_0 represent the user's uncomplicated cloud-uploadable image. The picture owner will encrypt the image using the three-level

permutation method to produce the encrypted image G_3 in order to protect the privacy content of the image as displayed in **Figure 2**.

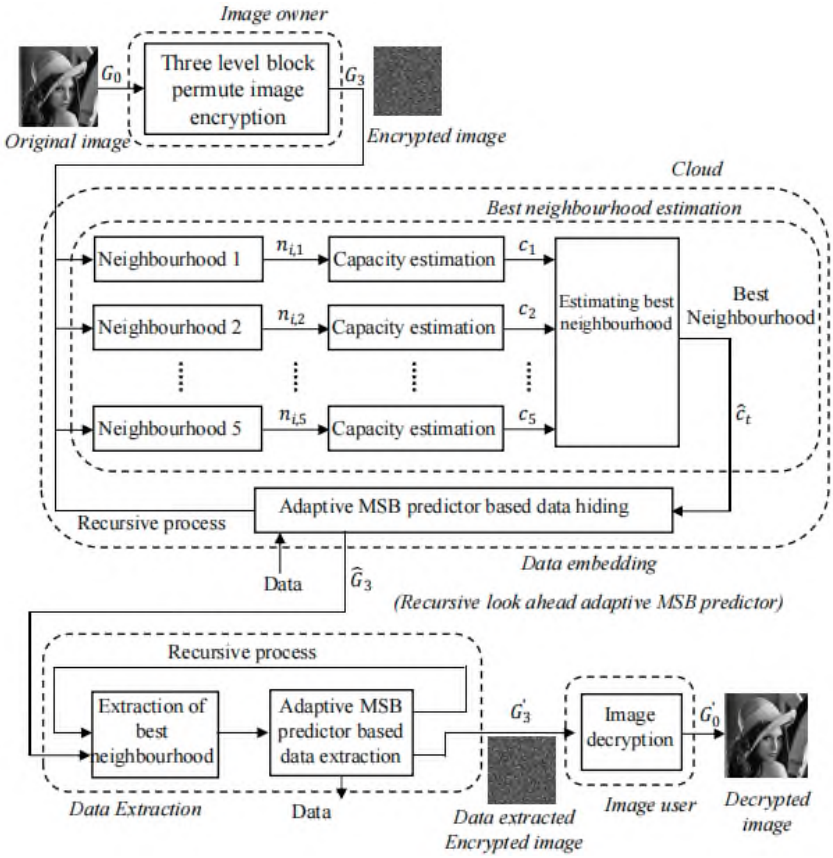

Figure 1. Proposed data embedding/extraction procedure.

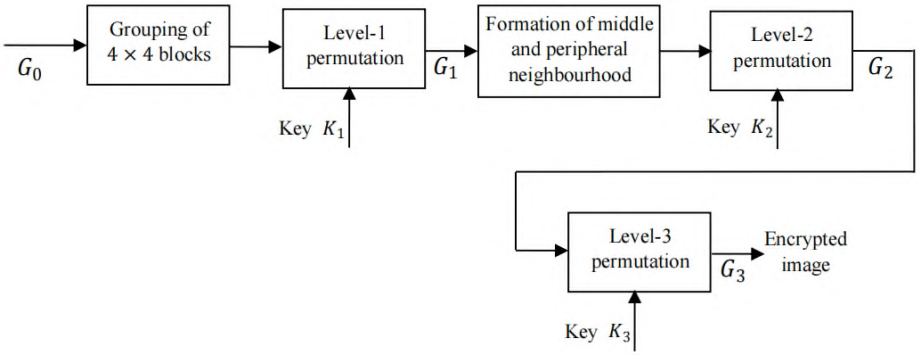

Figure 2. Three-level permutation-based image encryption.

3.1. Three-level block permute image encryption

Let M × N stand in for the G_0 image's size when it comes to encryption. Three keys are used in the three-level block permute image encryption, $K = \{K_1, K_2, K_3\}$. The original image is initially separated into 44 blocks that don't overlap. The integer ranges for the random permutation sequences produced using the key K_1 is between 1 and L_1. Let K_1 serve as the key and let S_1 serve as the permutation sequence.

$$L_1 = \frac{MN}{16} \tag{1}$$

The first level block permuted image G_1 is created by scrambling the positions of each 4 × 4 block using the permutation sequence S_1. A sample image before encryption is shown in **Figure 3a** and a sample image after first-level block permute encryption is shown in **Figure 3b**. For every 4 × 4 block in the first level

encrypted image G_1, a 2 × 2 neighbourhood is built in the second level. In a similar manner, four additional outer neighbourhoods are built, each measuring 2 × 2. The representation of middle and peripheral blocks is shown in **Figure 4**.

211	91	91	135	219	89	171	127
81	174	115	180	222	163	94	142
87	211	187	164	143	225	207	188
156	222	87	96	170	214	197	197
221	90	100	196	67	74	112	118
115	101	219	200	53	85	108	121
119	212	230	88	105	99	92	98
198	173	167	213	110	111	95	106

(a)

67	74	112	118	221	90	100	196
53	85	108	121	115	101	219	200
105	99	92	98	119	212	230	88
110	111	95	106	198	173	167	213
211	91	91	135	219	89	171	127
81	174	115	180	222	163	94	142
87	211	187	164	143	225	207	188
156	222	87	96	170	214	197	197

(b)

Figure 3. Illustrates a first-level block permute process. (**a**) before encryption; (**b**) after encryption.

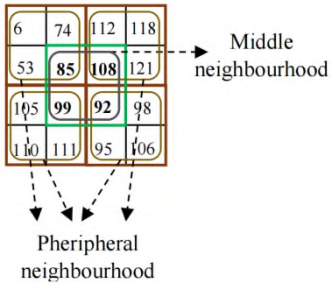

Figure 4. Representation of middle and peripheral blocks.

Then for each 4 × 4 block, L_2 the number of random permutation sequences is generated whose integer range is between 1 to 4, such that each consecutive four numbers in the sequence should contain the range of random integers between 1 to 4. Let the sequence be represented as S_2. Using the sequence S_2, each 2 × 2 middle neighbourhood is permuted, such that based on the permutation of middle elements the remaining elements of the peripheral neighborhood are also permuted to obtain the second level block permuted image G_2.

$$L_2 = \frac{MN}{4} = 4L_1 \tag{2}$$

In the third level, using the key K_3, L_3 number of random permutation sequences is generated whose integer range is between 1 to 3, such that each consecutive four numbers in the sequence should contain the range of random integers between 1 to 3. Let the sequence be represented as S_3. Using the sequence S_3, the three elements in the peripheral neighbourhood (elements that are not included in the middle neighbourhood) are permuted, to obtain the third level block permuted image G_3.

$$L_3 = \frac{3MN}{4} = 12L_1 = 3L_2 \tag{3}$$

An example of second and third-level permute is provided in **Figure 5**.

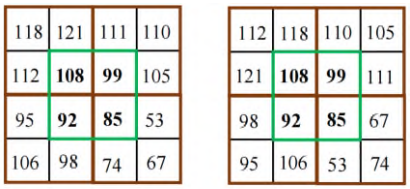

Figure 5. Representation of second and third-level block permute image encryption; **(a)** second-level block permute; **(b)** third-level block permute.

Algorithm 1: Three-level permute image encryption.

Input: Plain image G_0, key K
Output: Encrypted image G_3
Step 1: Using the key K, generate three sub keys K_1, K_2 and K_3.
Step 2: Estimate the length of the permutation sequence L_1, L_2 and L_3 using equations (1), (2) and (3) respectively.
Step 3: Subdivide the image G_0 into 4 × 4 non-overlapping blocks.
Step 4: Generate the permutation sequence S_1 using the key K_1 with length L_1
Step 5: Obtain the level-1 permuted image G_1 using the sequence S_1.
Step 6: On the image G_1, obtain the peripheral neighbourhood and the middle neighbourhood
Step 7: Generate the permutation sequence S_2 using the key K_2 with length L_2.
Step 8: Obtain the level-2 permuted image G_2 using the sequence S_2.
Step 9: Generate the permutation sequence S_3 using the key K_3 with length L_3.
Step 10: Obtain the level-3 permuted image (Encrypted image) G_3 using the sequence S_3.

3.2. Recursive look ahead adaptive MSB predictor data embedding

The recursive look ahead adaptive MSB predictor-based data embedding has two modules namely best neighbourhood estimation followed by adaptive MSB predictor-based data embedding. The embedding is done in a recursive process. The encrypted image G_3 is subdivided into 4 × 4 blocks. Let the 4 × 4 block be represented as

$$H_j = [a_{11}\ a_{12}\ a_{21}\ a_{22}\ a_{13}\ a_{14}\ a_{23}\ a_{24}\ a_{31}\ a_{32}\ a_{41}\ a_{42}\ a_{33}\ a_{34}\ a_{43}\ a_{44}] \quad (4)$$

From each 4 × 4 block, five different neighbourhoods are estimated. Let the five neighbourhoods be represented as n_1, n_2, n_3, n_4 and n_5. Where $n_1 = [a_{11}\ a_{12}\ a_{21}\ a_{22}]$, $n_2 = [a_{13}\ a_{14}\ a_{23}\ a_{24}]$, $n_3 = [a_{31}\ a_{32}\ a_{41}\ a_{42}]$, $n_4 = [a_{33}\ a_{34}\ a_{43}\ a_{44}]$, $n_5 = [a_{22}\ a_{23}\ a_{32}\ a_{33}]$. The data is initially embedded in the neighbourhood n_1, followed by n_2, n_3, n_4 and n_5 without finding the best neighbourhood. Once the data is embedded in these neighbourhoods, the recursive data embedding is performed. Let $n_{i,1}$, $n_{i,2}$, $n_{i,3}$, $n_{i,4}$ and $n_{i,5}$ represent the five neighbourhoods of the i-th 4 × 4 block, where $i = 1, 2, \ldots, L_1$. Let $C_{i,1}$, $C_{i,2}$, $C_{i,3}$, $C_{i,4}$ and $C_{i,5}$ represents the capacity of $n_{i,1}$, $n_{i,2}$, $n_{i,3}$, $n_{i,4}$ and $n_{i,5}$ respectively. The total capacity of L_1 number of blocks whose neighbourhood position n_1 is

$$c_1 = C_{1,1} + C_{2,1} + \cdots + C_{L_1,1} = \sum_{i=1}^{L_1} C_{i,1} \quad (5)$$

Similarly, the total capacity of the remaining neighbourhoods can be estimated by the relations

$$c_2 = \sum_{i=1}^{L_1} C_{i,2},\ c_3 = \sum_{i=1}^{L_1} C_{i,3},\ c_4 = \sum_{i=1}^{L_1} C_{i,4},\ c_5 = \sum_{i=1}^{L_1} C_{i,5} \quad (6)$$

From the capacity c_1, c_2, c_3, c_4 and c_5, the best neighbourhood can be estimated as

$$\hat{c}_t = max\{c_1, c_2, c_3, c_4, c_4\} \quad (7)$$

The neighbourhood that provides the maximum capacity represents the best neighbourhood.

$$\hat{n}_t = n_j \text{ where } \hat{c}_t = c_j \quad (8)$$

Once the best neighbourhood is estimated, the data is then embedded in the respective neighbourhood using the adaptive MSB predictor-based data embedding approach. Let $\widehat{H}_i^{(t)}$ represent the neighbourhood of i-th block after embedding the data on i-th recursive stage. For embedding the data in a 2 × 2 neighbourhood, two different types of pixels namely primary pixel and secondary pixel are chosen. **Figure 6** depicts the representation of primary and secondary pixels in the neighbourhood n_1, n_2, n_3, n_4, and n_5.

$$\underset{n_1}{\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}} \quad \underset{n_2}{\begin{bmatrix} a_{13} & a_{14} \\ a_{23} & a_{24} \end{bmatrix}} \quad \underset{n_3}{\begin{bmatrix} a_{31} & a_{32} \\ a_{41} & a_{42} \end{bmatrix}} \quad \underset{n_4}{\begin{bmatrix} a_{33} & a_{34} \\ a_{43} & a_{44} \end{bmatrix}} \quad \underset{n_5}{\begin{bmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{bmatrix}}$$

Figure 6. Representation of primary and secondary pixels in the neighbourhood n_1, n_2, n_3, n_4 and n_5.

3.3. Adaptive MSB predictor based embedding

Let the 2 × 2 neighbourhood with primary and secondary pixels be represented as

$$F = [p\ s_1\ s_2\ s_3] \tag{9}$$

here p represent the primary pixel, s_1, s_2, s_3 represents the remaining three secondary pixels taken in clockwise order of the 2 × 2 block. For example, the neighbourhood n_1, $F = [a_{22}\ a_{21}\ a_{11}\ a_{12}]$. Initially, the primary and secondary pixels in F are converted to 8-bit binary form. The number of common bits in p, s_1, s_2 and s_3 are estimated which can be mathematically expressed by the function

$$c = C(F) = C(p, s_1, s_2, s_3) - 1 \tag{10}$$

where the function $C(.)$ will find the number of consecutive common bits from MSB. The data cannot be embedded in the block F, if $c = -1$. There are bits that can be inserted in block F in a maximum of

$$N_b = 21 - 3 \times c \tag{11}$$

The sequence after embedding in F can be represented as

$$\hat{F} = \{\hat{p}, \hat{s}_{1,c+1}, \hat{s}_{1,c+2}, \dots, \hat{s}_{1,8}, \hat{s}_{2,c+1}, \hat{s}_{2,c+2}, \dots, \hat{s}_{2,8}, \hat{s}_{3,c+1}, \hat{s}_{3,c+2}, \dots, \hat{s}_{3,8}, D_1, D_2, \dots, D_{N_b}, \hat{c}_1, \hat{c}_2, \hat{c}_3\} \tag{12}$$

where $\hat{p}$ is the form of 8-bit binary, of p, $\hat{s}_{1,j}$, $\hat{s}_{2,j}$ and $\hat{s}_{3,j}$ be the j-th binary bit from MSB in 8-bit binary form of s_1, s_2 and s_3 respectively. $\hat{c}_1, \hat{c}_2, \hat{c}_3$ represent the 3-bit binary representation of c from MSB to LSB. $D_1, D_2, \dots, D_{N_b}$ represent the N_b number of data that is to be embedded. The sequence $\hat{F}$ contains a totally of 32 bits. To get the marked encrypted block, the 32 bits are divided into 8 bits and translated to decimal.

$$F' = \{p, s_1', s_2', s_3'\} \tag{13}$$

Examples of the data embedding procedure are shown in **Figure 7** and an example of blocks before and after the data embedding in the blocks n_1 and n_2 is shown in **Figure 8**.

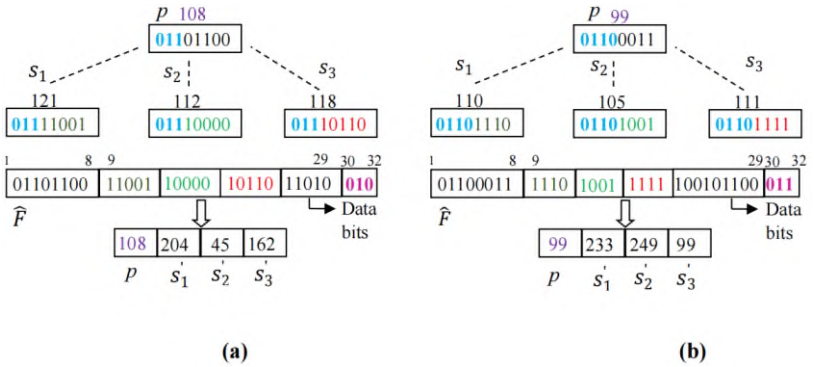

Figure 7. Data embedding: **(a)** procedure a; **(b)** procedure b.

112	118	110	105
121	**108**	**99**	111
98	**92**	**85**	67
95	106	53	74

(a)

45	162	223	249
204	**108**	**99**	99
98	**92**	**85**	67
95	106	53	74

(b)

Figure 8. Example of blocks before and after embedding on the blocks n_1. and n_2; **(a)** before embedding; **(b)** after embedding.

Algorithm 2: Data embedding

Input: Plain image G_o, key K, number of recursive processes R, data for embedding
Output: Marked encrypted image
Step 1: Encrypt the image G_o using the three-level permutation approach provided in algorithm 1.
Step 2: Sub-divide the image onto 4×4 blocks and obtain the five different neighbourhoods of each block $n_{i,1}$, $n_{i,2}$, $n_{i,3}$, $n_{i,4}$, and $n_{i,5}$.
Step 3: Embed the data and compressed location map on the blocks $n_{i,1}$ followed by $n_{i,2}$, $n_{i,3}$, $n_{i,4}$, and finally $n_{i,5}$. Thus the number of recursive processes reaches $t = 5$.
Step 4: After completing the recursive process $t = 5$ estimate the best neighbourhood.
Step 5: Embed the data and the compressed location map information of the non-embeddable blocks on the best neighbourhood using the adaptive MSB neighbourhood predictor process. Increment the value of t by 1. ($t = t + 1$)
Step 6: Repeat step 4 to 5 till $t = R$.

3.4. Recursive look ahead adaptive MSB predictor data extraction

Initially, the marked encrypted pixels of the block F' is represented in binary, so that Equation (12) can be reconstructed. From the 32-bit sequence, the first 8-bit sequence is used to reconstruct the primary pixel p, the last 3-bit is used to reconstruct the value of c. The Equation (11) can be used to calculate the number of bits embedded in the block F' from the value of c.

The data can be extracted from the sequence F' by

$$\widehat{D} = \{F'_{29-N_b+1}, F'_{29-N_b+2}, \dots, F'_{29}\} \tag{14}$$

The number of common bits between the primary pixel p and the remaining secondary pixels can be estimated as

$$C = c + 1 \tag{15}$$

The secondary pixel in binary form can be recovered as

$$\begin{aligned} s'_1 &= \{F'_1, F'_2, \dots, F'_C, F'_9, F'_{10}, \dots, F'_{16-C}\} \\ s'_2 &= \{F'_1, F'_2, \dots, F'_C, F'_{17-C}, F'_{17-C+1}, \dots, F'_{25-2C}\} \\ s'_3 &= \{F'_1, F'_2, \dots, F'_C, F'_{26-C}, F'_{26-C+1}, \dots, F'_{28}\} \end{aligned} \tag{16}$$

The extracted secondary pixel bits of s'_1, s'_2, s'_3 are converted to 8-bit decimal numbers to obtain the reconstructed secondary pixels s_1, s_2, s_3. From the primary pixel p, and secondary pixels s_1, s_2, s_3 the extracted pixel block F can be reconstructed using Equation (9).

Algorithm 3: Data extraction

Input: Marked encrypted image, key K, number of recursive processes R
Output: Extracted data, decrypted image G'_o
Step 1: Sub-divide the image into 4×4 sub-blocks and obtain the neighbourhoods $n_{i,1}$, $n_{i,2}$, $n_{i,3}$, $n_{i,4}$, and $n_{i,5}$ of each block.
Step 2: Extract the location map information and extract the data that was embedded in recursive stage $t = R$ using the adaptive MSB predictor data extraction algorithm. After extracting the data, decrement count value t by 1($t = t + 1$).
Step 3: Repeat step 3 till $t = 5$ is reached
Step 4: Further, extract the data on the blocks $n_{i,5}$ followed by $n_{i,4}$, $n_{i,3}$, $n_{i,2}$, and $n_{i,1}$.
Step 5: Decrypt the image G'_3 using the three-level permutation approach provided in algorithm 4 using the key K.

3.5. Three-level block permute image decryption

As seen in **Figure 9**, there are three levels to the decryption process. The marked encrypted image will be represented by G_3'. The initial block subdivision of the designated encrypted image is 44 non-overlapping blocks. Using the key K, three keys are estimated which are represented as K_1, K_2 and K_3. From the 4 × 4 sub-blocks, the middle neighbourhood and peripheral neighbourhood are estimated. Using the key K_3, totally L_3 number of random permutation sequences are generated whose integer range is between 1 to 3, such that each consecutive three numbers in the sequence should contain the range of random integers between 1 to 3. Let the sequence be represented as S_3. Using the position as S_3, the permuted position is rearranged to obtain the level three decrypted image G_2'.

In the second level of decryption L_2 number of random permutation sequences are generated whose integer range is between 1 to 4, such that each consecutive four numbers in the sequence should contain the range of random integers between 1 to 4. Let the sequence be represented as S_2. Using the sequence S_2 as permuted sequence, each 2 × 2 middle neighbourhood is arranged, such that based on the shuffling of middle elements the remaining elements of the corresponding peripheral neighbourhood are also arranged to obtain the second level block permuted image G_1'. In the third level of decryption, using the key K_1, totally L_1 number of random permutation sequences are generated whose integer range is between 1 to L_1. Let S_1 stand for the permutation sequence produced by key K_1. The permutation sequence S_1 is used to arrange the positions of each 4 × 4 block in order to produce the level-3 decrypted image G_0'.

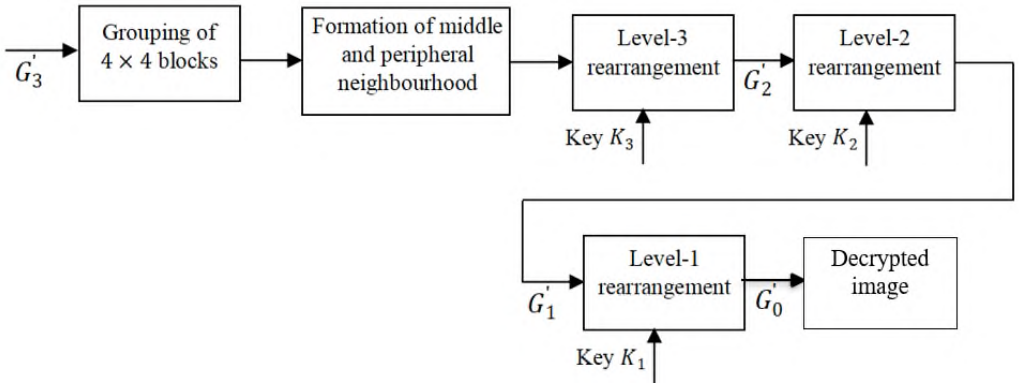

Figure 9. Block diagram of three-level permutation-based decryption.

Algorithm 4: Three-level permute image decryption

Input: Marked encrypted image G_3', key K
Output: Decrypted image G_0'
Step 1: Create the three sub-keys K_1, K_2, and K_3 using the key K.
Step 2: Estimate the length of the permutation sequence L_1, L_2 and L_3 using equation (1), (2) and (3)
Step 3: Subdivide the image G_3' into 4 × 4 non-overlapping blocks.
Step 4: On the image G_3', obtain the peripheral neighbourhood and the middle neighbourhood
Step 5: Generate the permutation sequence S_1 using the key K_1 with length L_1
Step 6: Obtain the level-1 rearranged image pixels G_2' using the sequence s_1.
Step 7: Generate the permutation sequence S_2 using the key K_2 with length L_2.
Step 8: Obtain the level-2 rearranged image G_1' using the sequence S_2.
Step 9: Generate the permutation sequence S_3 using the key K_3 with length L_3.
Step 10: Obtain the level-3 rearranged image (Decrypted image) G_0' using the sequence S_3.

4. Experimentation outcomes

Utilising the benchmark test pictures from the BOSSbase[32], BOWS-2[33] and UCID[34] datasets, the proposed recursive data embedding approach was assessed using the metrics embedding rate, structural similarity index (SSIM), and peak signal to noise ratio (PSNR). The relations (17), (18) and (20) can be used to compute the embedding rate, SSIM and PSNR.

$$ER = \frac{capacity}{2 \times M \times N} \tag{17}$$

$$PSNR = 10\frac{255^2}{\varepsilon} dB \tag{18}$$

where ε is the mean square error can be estimated as

$$\varepsilon = \frac{1}{M \times N}\sum_{x=1}^{M}\sum_{y=1}^{N}\left(G_3(x,y) - \hat{G}_3(x,y)\right)^2 \tag{19}$$

The structural similarity index (SSIM) can be expressed as

$$SIM = Lu\left(G_3, \hat{G}_3\right)Co\left(G_3, \hat{G}_3\right)Sa(G_3, \hat{G}_3) \tag{20}$$

where $Sa(G_3, \hat{G}_3)$ is the saturation comparison function, $Co\left(G_3, \hat{G}_3\right)$ is the comparison function for luminance, while $Lu\left(G_3, \hat{G}_3\right)$ is the comparison function for contrast.

The three-level permutation was initially used to encrypt the plain image. The three-level permutation-based encrypted graphics with hidden privacy material are shown in **Figure 10**. As the permutations from level 1 to level 3 were carried out. Stronger encryption is achieved.

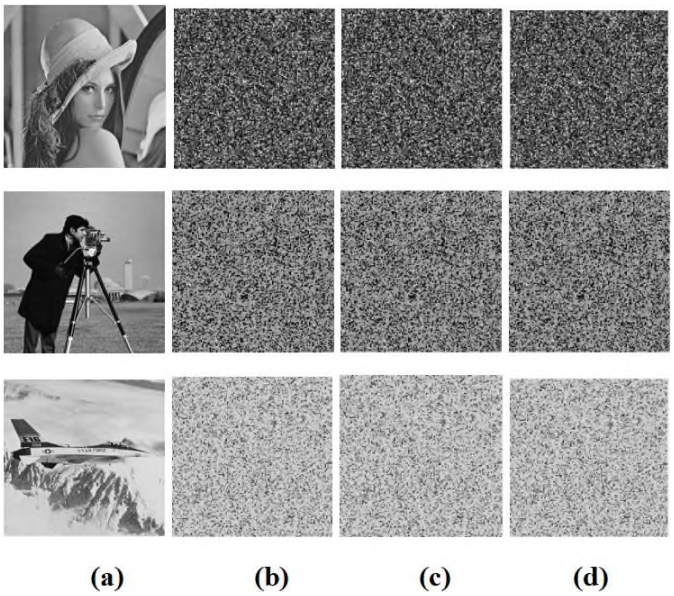

Figure 10. Three-level permutation-based encryption results **(a)** plain image; **(b)** first level permutation result; **(c)** second level permutation result; **(d)** third level permutation result.

The image that was encrypted was embedded with data (shown in **Figure 11)** recursively with a different number of recursive processes $R = 1$ to $R = 15$. The embedding rate is high, for the recursive process $R = 1$ to $R = 5$.

Figure 11. Binary 'android image' for secret data.

Figure 12 shows the binary image which represents the position of embedded and non-embedded pixels (1.'s represents the position of data embedded pixels and 0.'s represents the position of non-embedded pixels). As the number of recursive processes increases, the number of 1.'s gets increases. This indicates that as the number of recursive embedding processes increases, the embedding rate also increases.

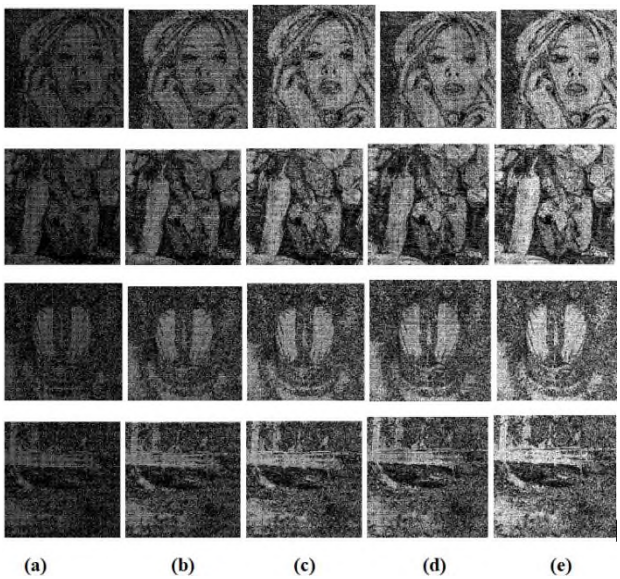

Figure 12. Binary image representing the embedded pixel locations **(a)** recursive process $R = 1$; **(b)** recursive process $R = 2$; **(c)** recursive process $R = 3$; **(d)** recursive process $R = 4$; **(e)** recursive process $R = 5$.

Using the datasets BOSSbase, BOWS-2 and UCID datasets, the average embedding rate (ER) of the proposed approach is compared with the ER of other traditional schemes, including Puteaux and Puech[22], Ren et al.[34], Wu et al.[24], Yin et al.[23], Yin et al.[30], Wang and He[26]. The UCID dataset has 1338 colour images, whereas the BOSSbase and BOWS-2 datasets each have 10,000 grayscale images Bas et al.[31], Bas and Furon[32]. The UCID algorithm first converts the colour images to grayscale before analysing them. The comparison of ER for various techniques for the test images displayed is presented in **Table 1**. The proposed method has an average ER of 2.79 bpp with $R = 10$, which is 0.1756 bpp and 0.2295 bpp. The highest and minimum ER for the provided test images are 2.105 bpp and 3.289 bpp, respectively.

Table 1. Embedding rate comparison for different approaches with different test images.

Test image	Puteaux and Puech[22]	Ren et al.[34]	Wu et al.[24]	Yin et al.[23]	Yin et al.[30]	Wang and He[26]	Yu et al.[27]	Proposed
Lena	0.977	1.712	2.645	2.583	2.87	2.83	3.019	3.124
Mandril	0.838	0.456	0.969	1.006	1.321	1.452	1.4596	2.162
Jetplane	0.983	2.097	2.673	3.03	3.232	3.154	3.237	3.289
Cameraman	0.981	1.577	2.479	2.349	2.49	2.629	2.651	2.714
Blonde	0.993	2.158	2.652	2.824	2.943	3.017	3.091	3.105
House	0.846	1.457	2.861	3.08	2.532	2.896	3.149	3.197
Pepper	0.871	1.824	1.963	2.348	1.581	2.347	2.087	2.438
Bridge	0.957	2.024	2.406	1.054	2.54	2.768	2.851	2.915
Living room	0.825	2.11	1.087	2.251	2.557	2.617	2.672	2.849
Lake	0.954	2.054	2.546	2.705	1.51	1.893	1.923	2.105

Figures 13 and **14** show the variance in embedding rate for various recursive process counts. As the value of R goes from 1 to 5, the embedding rate rises quickly. The embedding capacity gradually increases as the R-value is raised. This is because as R approaches 5, all neighbourhoods are embedded with data, reducing the number of embeddable blocks in the marked neighbourhood that results. According to **Table 2**, the average embedding rates with R = 10 for the BOSSbase, BOWS-2 and UCID datasets are respectively 3.714 bpp, 3.4826 bpp, and 2.9412 bpp.

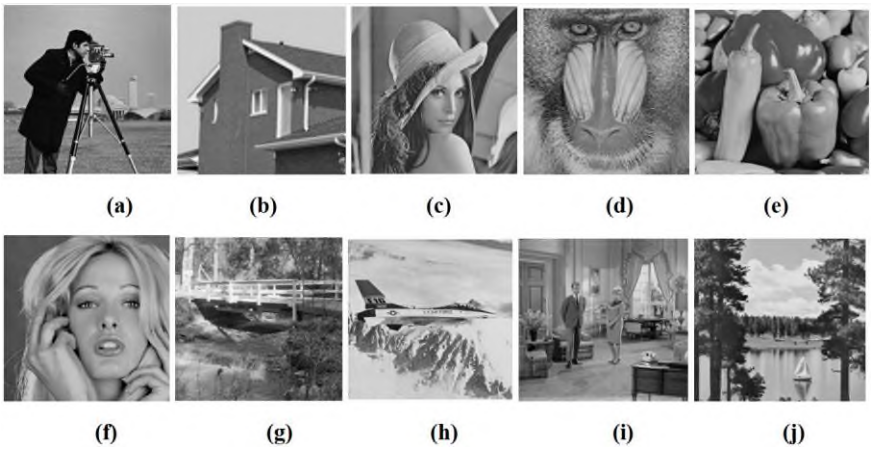

Figure 13. Standard test images **(a)** cameraman; **(b)** house; **(c)** lena; **(d)** mandril; **(e)** pepper; **(f)** blonde; **(g)** bridge; **(h)** jetplane; **(i)** living room; **(j)** lake.

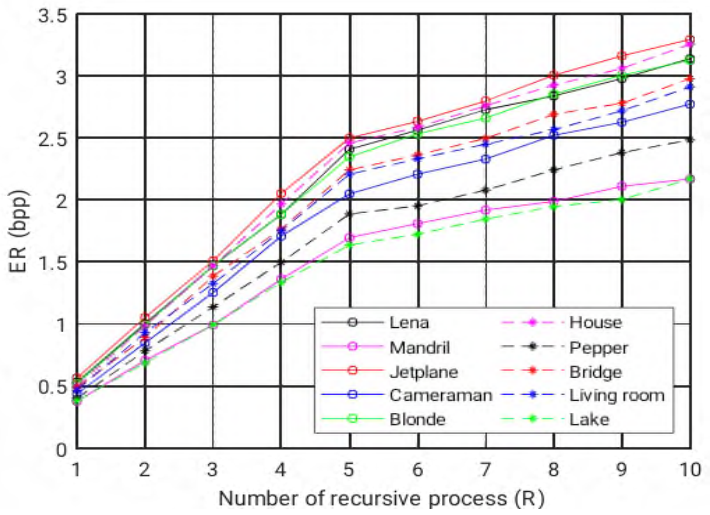

Figure 14. Variation of embedding rate for different number of recursive processes.

Table 2. Comparison of average embedding rate on BOSSbase, BOWS-2, UCID dataset.

Schemes	BOSSbase	BOWS-2	UCID
Puteaux and Puech[22]	0.966	0.968	0.893
Ren et al.[34]	2.561	2.519	2.226
Wu et al.[24]	3.361	3.246	2.688
Yin et al.[23]	3.498	3.393	2.797
Yin et al.[30]	3.517	3.421	2.815
Wang and He[26]	3.587	3.4428	2.857
Zhang et al.[27]	3.6823	3.4568	2.883
Proposed	3.714	3.4826	2.9412

Figure 15 depicts the graphical comparison of ER for the various methods. In comparison to the Zhang et al. technique, the suggested method offers a higher average embedding rate for the datasets BOSSbase, BOWS-2, and UCID.

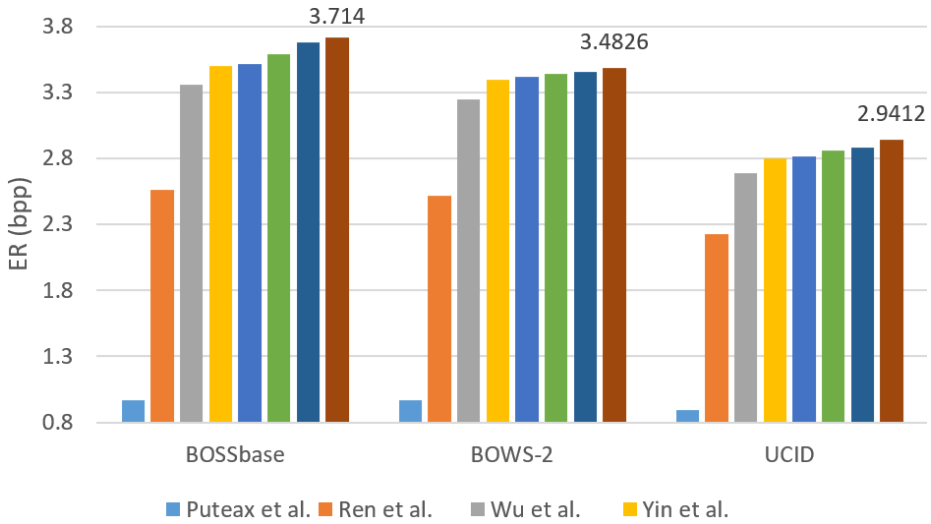

Figure 15. A visual comparison of the ER for the datasets from BOSSbase, BOWS-2, and UCID.

Table 3. PSNR for various recursive process numbers between the encrypted picture and the annotated encrypted image.

Test images	$R = 1$		$R = 2$		$R = 3$		$R = 4$		$R = 5$	
	PSNR (dB)	**SSIM**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**	**PSNR (dB)**	**SSIM**
Lena	15.19	0.6340	12.06	0.4101	10.41	0.2589	9.06	0.1341	8.29	0.0510
Mandril	17.55	0.6270	14.65	0.4240	12.90	0.2837	11.60	0.1757	10.58	0.0975
Jetplane	14.89	0.4994	11.80	0.3163	10.12	0.2118	8.77	0.1270	8.15	0.0842
Cameraman	16.13	0.6238	13.09	0.4051	11.39	0.2573	10.08	0.1392	9.19	0.0594
Blonde	16.77	0.5786	13.83	0.3752	11.98	0.2364	10.78	0.1431	9.85	0.0783
House	16.44	0.5916	13.41	0.3872	11.69	0.2531	10.39	0.1493	9.51	0.0814
Peppers	15.64	0.6104	12.55	0.3813	10.87	0.2293	9.57	0.1079	8.70	0.0296
Bridge	15.24	0.5813	12.14	0.3692	10.47	0.2333	9.13	0.1230	8.38	0.0549
Livingroom	17.17	0.6161	14.10	0.4033	12.34	0.2594	11.10	0.1551	10.11	0.0847
Lake	16.02	0.5905	12.98	0.3782	11.25	0.2380	9.97	0.1308	9.11	0.0611

The variation of PSNR and SSIM for different number of recursive processes is provided in **Table 3**. As the number of recursive processes R increases the PSNR and SSIM get reduced. For the test images provided in **Figure 15.** The average PSNR for $R = 1, 2, 3, 4, and\ 5$ is 16.1 dB, 13.06 dB, 11.34 dB, 10.04 dB and 9.19 dB respectively. The SSIM for $R = 1, 2, 3, 4, and\ 5$ is estimated as 0.5953, 0.3850, 0.2461, 0.1385 and 0.0682 respectively.

To determine the computational complexity, the embedding/extraction algorithm's computational complexity is examined. **Table 4** provides the computation times for 3-level permutation-based encryption and decryption.

Table 4. Computational time in encryption and decryption.

	Level-1	**Level-2**	**Level-3**	**Total**
Encryption time (s)	0.236	0.384	0.391	1.011
Decryption time (s)	0.217	0.376	0.383	0.976

The proposed work's average encryption and decryption times are 1.011 s and 0.976 s, respectively. On a computer with an Intel Core i5 CPU running at 3.00 GHz, Windows 10, 8 GB of RAM, and a 64-bit operating

system, the time complexity is assessed using MATLAB 2018 a. Comparing level-3 permutations to level-1 and level-2, respectively, requires additional processing time.

Let t_b, t_{em} represent the computation time for best neighbourhood search, and data embedding without including the best neighbourhood search process respectively. Let T_{em} and T_{ex} represent the time of embedding and extraction. The time of embedding can be expressed as:

$$T_{em} = t_b + t_{em} \tag{21}$$

As the number of the recursive process increases, the computational time linearly increases as illustrated in **Figure 16**.

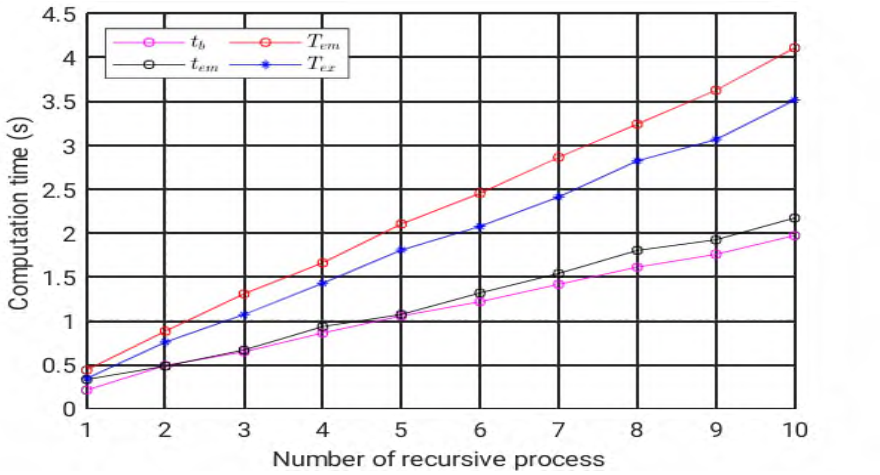

Figure 16. Computation time in data embedding and extraction for different number of recursive processes.

Figure 17 shows the histogram of images before and after data embedding. As the data is embedded recursively from $R = 1$ to $R = 5$, the histogram gets changed.

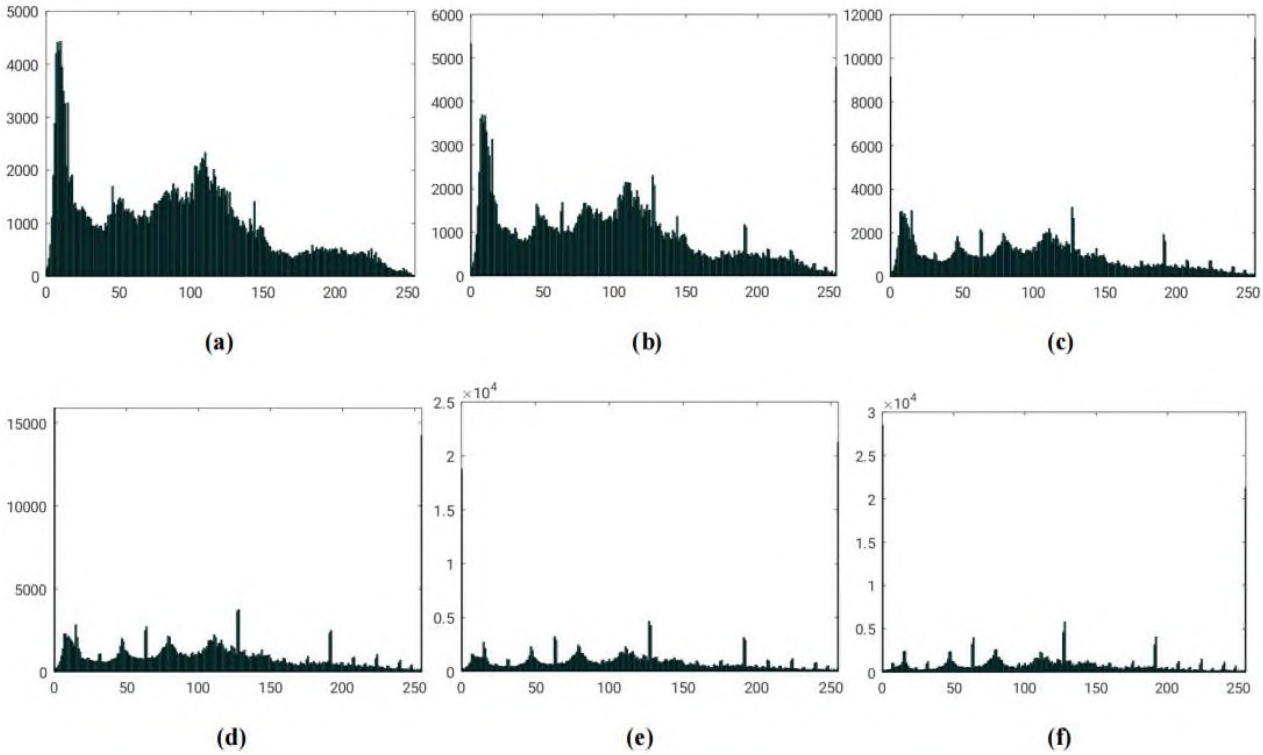

Figure 17. Before and after data embedding histogram of the image **(a)** marked encrypted image; **(b)** marked encrypted image with *R* = 1; **(c)** *R* = 2; **(d)** *R* = 3; **(e)** *R* = 4; **(f)** *R* = 5.

Figure 18 compares the embedding rates on 100 randomly chosen photos from the BOWS-2 image dataset, where the embedding rates range from 2 to 4 bits per picture (bpp). The paper's conclusion is presented in the following section.

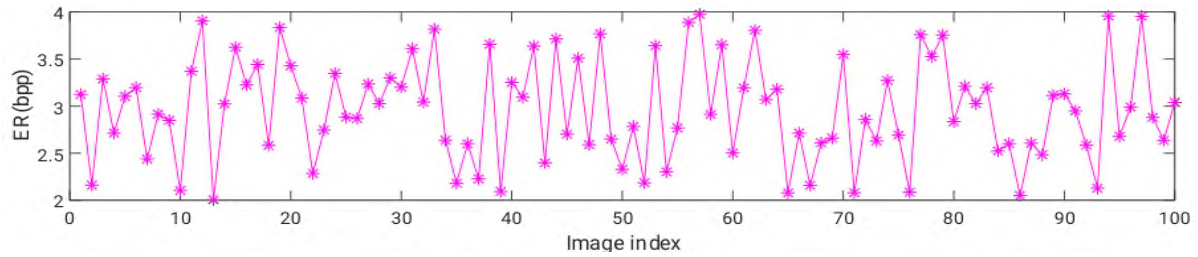

Figure 18. Embedding rate comparison on 100 randomly selected images of BOWS-2 image database.

To enhance the research work, deep learning can be incorporated for image encryption and data embedding. A convolutional neural network (CNN) can be trained to learn the optimal image encryption patterns for the three-level encryption process, improving security. Additionally, a deep embedding network can be developed to dynamically estimate the best neighbourhoods for data embedding, optimizing capacity. By leveraging deep learning models, the research can achieve more robust and adaptive encryption and data hiding techniques, ensuring better performance and security in cloud storage applications.

5. Conclusion

This study suggested an RDH-EI technique that makes use of an adaptive MSB predictor with a recursive look-ahead. The research also suggests a three-level permutation-based technique for image encryption and decryption. In order to estimate the middle neighbourhood and the periphery, the computer first divides the image into 4 × 4 blocks. Blocks are permuted in the first level of encryption, whereas in the second level, the position of the middle neighbourhood pixels and its matching peripheral neighbourhood are permuted while keeping the same middle neighbourhood components. The third stage involves permuting the pixels within their peripheral neighbourhood without affecting the components of the middle neighbourhood. A recursive look ahead adaptive MSB predictor approach is used in the data embedding strategy, where the algorithm predicts the ideal neighbourhood that offers the most embedding capacity before actual embedding. The adaptive MSB prediction technique embeds the data in the neighbourhood with the highest embedding capacity. The standard test images obtained from the datasets were used to evaluate the technique using the metrics embedding capacity (embedding rate), SSIM, and PSNR. The average embedding rate for the datasets BOSSbase, BOWS-2, and UCID provided by the suggested approach is 3.714 bpp, 3.4826 bpp, and 2.9412 bpp, respectively, with the number of recursive processes R = 10, which is greater than that of previous comparable approaches.

The research presents a high-capacity reversible data embedding technique (RDH-EI) for cloud storage applications, incorporating a three-level image encryption process. While achieving superior embedding rates compared to previous RDH-EI methods, the approach faces challenges. The complexity of the three-level encryption and recursive embedding may lead to higher computational overhead, hindering real-time and large-scale data embedding scenarios. Moreover, the lack of a comprehensive security analysis raises concerns about potential vulnerabilities. Although the proposed approach outperforms previous methods, the data hiding capacity remains limited for applications requiring larger payloads. Additionally, the focus on image data raises questions about generalization to other data types. To address these limitations, future work should focus on algorithm optimization, thorough security evaluations, improved capacity, cross-domain application feasibility, and adaptability to evolving cloud services. Implementing these enhancements can make the RDH-EI approach more versatile, secure, and relevant for a wider range of data hiding and cloud storage applications.

Author contributions

Conceptualization, RS; methodology, SPJ; software, SPJ; validation, SPJ; formal analysis, SPJ; investigations, SPJ; resources, SPJ; data curation, SPJ; writing—original draft preparation, RS; writing—review and editing, RS; visualization, SPJ; supervision, RS; project administration, RS. All authors have read and agreed to the published version of the manuscript.

Conflict of interest

The authors declare no conflict of interest.

References

1. He W, Cai, Z. Reversible data hiding based on dual pairwise prediction-error expansion. *IEEE Transactions on Image Processing* 2021; 30: 5045–5055. doi: 10.1109/TIP.2021.3078088
2. Wang X, Chang CC, Liu CC, Chang CC. Privacy-preserving reversible data hiding based on quad-tree block encoding and integer wavelet transform. *Journal of Visual Communication and Image Representation* 2021; 79: 103203. doi: 10.1016/j.jvcir.2021.103203
3. Faragallah OS, Elaskily MA, Alenezi AF, et al. Quadruple histogram shifting-based reversible information hiding approach for digital images. *Multimedia Tools and Applications* 2021; 80: 26297–26317. doi: 10.1007/s11042-021-10956-3
4. Mandal PC, Mukherjee I, Chatterji BN. High capacity reversible and secured data hiding in images using interpolation and difference expansion technique. *Multimedia Tools and Applications* 2021; 80: 3623–3644. doi: 10.1007/s11042-020-09341-3
5. Zhang X. Reversible data hiding in encrypted image. *IEEE Signal Process Letters* 2011; 18(4): 255–258. doi: 10.1109/LSP.2011.2114651
6. Zhang X, Long J, Wang Z, Cheng H. Lossless and reversible data hiding in encrypted images with public key cryptography. *IEEE Transactions on Circuits and Systems for Video Technology* 2016; 26(9): 1622–1631. doi: 10.1109/TCSVT.2015.2433194
7. Zhang X. Separable reversible data hiding in encrypted image. *IEEE Transactions on Information Forensics and Security* 2012; 7(2): 826–832. doi: 10.1109/TIFS.2011.2176120
8. Qian Z, Zhang X. Reversible data hiding in encrypted image with distributed source encoding. *IEEE Transactions on Circuits and Systems for Video Technology* 2016; 26(4): 636–646. doi: 10.1109/TCSVT.2015.2418611
9. Chen YC, Hung TH, Hsieh SH, Shiu CW. A new reversible data hiding in encrypted image based on multi-secret sharing and lightweight cryptographic algorithms. *IEEE Transactions on Information Forensics and Security* 2019; 14(12): 3332–3343. doi: 10.1109/TIFS.2019.2914557
10. Zhang W, Ma K, Yu N. Reversibility improved data hiding in encrypted images. *Signal Processing* 2014; 94: 118–127. doi: 10.1016/j.sigpro.2013.06.023
11. Ma K, Zhang W, Zhao X, et al. Reversible data hiding in encrypted images by reserving room before encryption. *IEEE Transactions on Information Forensics and Security* 2013; 8(73): 553–562. doi: 10.1109/TIFS.2013.2248725
12. Cao X, Du L, Wei X, et al. High capacity reversible data hiding in encrypted images by patch-level sparse representation. *IEEE Transactions on Cybernetics* 2016; 46(5): 1132–1143. doi: 10.1109/TCYB.2015.2423678
13. Wu X, Sun W. High-capacity reversible data hiding in encrypted images by prediction error. *Signal Processing* 2014; 104(6): 387–400. doi: 10.1016/j.sigpro.2014.04.032
14. Nguyen TS, Chang CC, Chang WC. High capacity reversible data hiding scheme for encrypted images. *Signal Processing: Image Communication* 2016; 44: 84–91. doi: 10.1016/j.image.2016.03.010
15. Mohammadi A, Nakhkash M, Akhaee MA. A high-capacity reversible data hiding in encrypted images employing local difference predictor. *IEEE Transactions on Circuits and Systems for Video Technology* 2020; 30(8): 2366–2376. doi: 10.1109/TCSVT.2020.2990952
16. Huang F, Huang J, Shi YQ. New framework for reversible data hiding in encrypted domain. *IEEE Transactions on Information Forensics and Security* 2016; 11(12): 2777–2789. doi: 10.1109/TIFS.2016.2598528
17. Di F, Huang F, Zhang M, et al. Reversible data hiding in encrypted images with high capacity by bitplane operations and adaptive embedding. *Multimedia Tools and Applications* 2018; 77: 20917–20935. doi: 10.1007/s11042-017-5498-8
18. Xiao D, Xiang Y, Zheng H, Wang Y. Separable reversible data hiding in encrypted image based on pixel value ordering and additive homomorphism. *Journal of Visual Communication and Image Representation* 45: 1–10. doi: 10.1016/j.jvcir.2017.02.001
19. Tang Z, Xu S, Yao H, et al. Reversible data hiding with differential compression in encrypted image. *Multimedia Tools and Applications* 2019; 78: 9691–9715. doi: 10.1007/s11042-018-6567-3

20. Fu Y, Kong P, Yao H, et al. Effective reversible data hiding in encrypted image with adaptive encoding strategy. *Information Sciences* 2019; 494: 21–36. doi: 10.1016/j.ins.2019.04.043
21. Yi S, Zhou Y. Separable and reversible data hiding in encrypted images using parametric binary tree labeling. *IEEE Transactions on Multimedia* 2019; 21(1): 51–64. doi: 10.1109/TMM.2018.2844679
22. Puteaux P, Puech W. An efficient MSB prediction-based method for high-capacity reversible data hiding in encrypted images. *IEEE Transactions on Information Forensics and Security* 2018; 13(7): 1670–1681. doi: 10.1109/TIFS.2018.2799381
23. Yin Z, Xiang Y, Zhang X. Reversible data hiding in encrypted images based on multi-MSB prediction and Huffman coding. *IEEE Transactions on Multimedia* 2020; 22(4): 874–884. doi: 10.1109/TMM.2019.2936314
24. Wu Y, Xiang Y, Guo Y, et al. An improved reversible data hiding in encrypted images using parametric binary tree labeling. *IEEE Transactions on Multimedia* 2020; 22(8): 1929–1938. doi: 10.1109/TMM.2019.2952979
25. Puteaux P, Puech W. A recursive reversible data hiding in encrypted images method with a very high payload. *IEEE Transactions on Multimedia* 2020; 23: 636–650. doi: 10.1109/TMM.2020.2985537
26. Wang Y, He W. High capacity reversible data hiding in encrypted image based on adaptive MSB prediction. *IEEE Transactions on Multimedia* 2022; 24: 1288–1298. doi: 10.1109/TMM.2021.3062699
27. Yu C, Zhang X, Zhang X, et al. Reversible data hiding with hierarchical embedding for encrypted images. *IEEE Transactions on Circuits and Systems for Video Technology* 2022; 32(2): 451–466. doi: 10.1109/TCSVT.2021.3062947
28. Xu S, Chang CC, Liu Y. A high-capacity reversible data hiding scheme for encrypted images employing vector quantization prediction. *Multimedia Tools and Applications* 2021; 80: 20307–20325. doi: 10.1007/s11042-021-10698-2
29. Li F, Zhu H, Yu J, Qin C. Double linear regression prediction based reversible data hiding in encrypted images. *Multimedia Tools and Applications* 2021; 80: 2141–2159. doi: 10.1007/s11042-020-09805-6
30. Yin Z, She X, Tang J, Luo B. Reversible data hiding in encrypted images based on pixel prediction and multi-MSB planes rearrangement. *Signal Processing* 2021; 187: 108146. doi: 10.1016/j.sigpro.2021.108146
31. Bas P, Filler T, Pevný T. "Break our steganographic system": The ins and outs of organizing boss. In: Filler T, Pevný T, Craver S, et al. (editors). *Information Hiding*, Proceedings of the 13th International Conference on Information Hiding; 18–20 May 2011; Prague, Czech Republic. Springer-Verlag; 2011. pp. 59–70.
32. Bas P, Furon T. Image database of BOWS-2. Available online: https://data.mendeley.com/datasets/kb3ngxfmjw/1 (accessed on 18 September 2023).
33. Schaefer G, Stich M. UCID: An uncompressed color image database. In: Proceedings of the Storage and Retrieval Methods and Applications for Multimedia 2004; 20 January 2004; San Jose, CA, USA. pp. 472–580.
34. Ren H, Lu W, Chen B. Reversible data hiding in encrypted binary images by pixel prediction. *Signal Process* 2019; 165: 268–277. doi: 10.1016/j.sigpro.2019.07.020