applied sciences

MDPI

Article

High-Capacity Reversible Data Hiding in Encrypted Images Based on Pixel Prediction and QuadTree Decomposition

Muhannad Alqahtani and Atef Masmoudi *

College of Computer Science, King Khalid University, Abha 62529, Saudi Arabia; mb.alqhtani@gmail.com
* Correspondence: amasmoudi@kku.edu.sa

Abstract: Over the past few years, a considerable number of researchers have shown great interest in reversible data hiding for encrypted images (RDHEI). One popular category among various RDHEI methods is the reserving room before encryption (RRBE) approach, which leverages data redundancy in the original image before encryption to create space for data hiding and to achieve high embedding rates (ERs). This paper introduces an RRBE-based RDHEI method that employs pixel prediction, quadtree decomposition, and bit plane reordering to provide high embedding capacity and error-free reversibility. Initially, the content owner predicts the error image using a prediction method, followed by mapping it to a new error image with positive pixel values and a compressed binary label map is generated for overhead pixels. Subsequently, quadtree decomposition is applied to each bit plane of the mapped prediction error image to identify homogeneous blocks, which are then reordered to create room for data embedding. After generating the encrypted image with the encryption key, the data hider employs the data hiding key to embed the data based on the auxiliary information added to each embeddable bit plane's beginning. Finally, the receiver is able to retrieve the secret message without any error, decrypt the image, and restore it without any loss or distortion. The experimental results demonstrate that the proposed RDHEI method achieves significantly higher ERs than previous competitors, with an average ER exceeding 3.6 bpp on the BOSSbase and BOWS-2 datasets.

Keywords: pixel prediction; quadtree decomposition; bit planes reordering; image encryption; data hiding; data extraction; image recovery

check for updates

Citation: Alqahtani, M.; Masmoudi, A. High-Capacity Reversible Data Hiding in Encrypted Images Based on Pixel Prediction and QuadTree Decomposition. *Appl. Sci.* **2023**, *13*, 12706. https://doi.org/10.3390/app132312706

Academic Editors: Luis Javier García Villalba and Ki-Hyun Jung

Received: 24 October 2023
Revised: 21 November 2023
Accepted: 22 November 2023
Published: 27 November 2023

1. Introduction

Recently, the concept of reversible data hiding in encrypted images (RDHEI) has gained importance due to its usefulness in confidential areas such as cloud, military, and medical applications. For example, a content owner may want to store an original image in the cloud, which needs to be encrypted to prevent unauthorized access [1]. RDHEI can be used to add additional data to the encrypted image for easier management without compromising the original content.

As shown in Figure 1, the RDHEI framework includes three users: the content owner, data-hider, and receiver. The content owner encrypts the image content, the data-hider embeds additional data into the encrypted image, and the receiver can extract the hidden data and/or recover the original image. A good RDHEI method should achieve reversibility, security, and high embedding capacity (EC).

RDHEI techniques can be classified into two categories based on the encryption order: vacating room after encryption (VRAE) and reserving room before encryption (RRBE). Furthermore, the data extraction and image recovery processes can be carried out jointly or separately. RRBE methods rely on exploiting the data redundancy present in the original image before encryption to create space that can be used for data hiding. However, in the VRAE techniques, the data hider processes the encrypted image in order to insert its secret information.

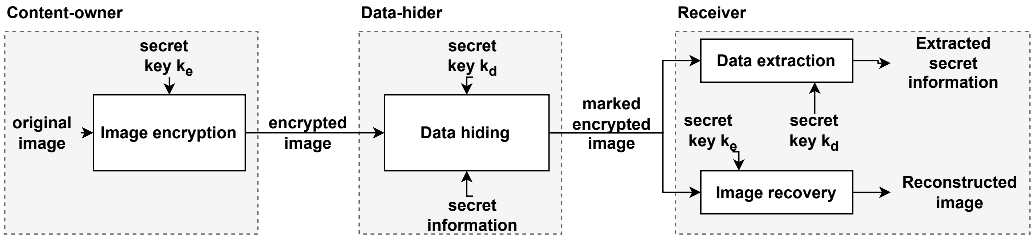


Figure 1. The RDHEI framework.

The initial VRAE method was suggested by Puech et al. [2]. It is a joint method where each 4×4 block of pixels in the original image is encrypted using the Advanced Encryption Standard (AES). After encryption, one bit of the confidential data is embedded into each encrypted block using bit substitution. The outcome is known as the marked encrypted image, and the confidential data can be extracted by acquiring the bits in the substituted positions. During decryption, the local standard deviation analysis is executed to regain the original image. The payload of this approach is 0.0625 bpp, which is very small.

Zhang [3] suggested encrypting the image by applying the bit-XOR operator between the original bits and the pseudo-random bits produced from a standard stream cipher. Following that, a data-hider can modify the encrypted image marginally to embed extra data without knowing the original image content. In the data embedding process, the encrypted image is firstly segmented into different non-overlapping blocks. Next, each block's pixels are pseudo-randomly divided into two sets based on a data-hiding key. Afterward, three least significant bits (LSBs) of each encrypted pixel from the first or the second sets are flipped according to the values of the embedded bits. The average ER is 0.033 bpp.

Zhang [4] introduced the first separable approach, in which the encrypted image's LSBs are compressed using a data-hiding key to create a sparse space to accommodate the additional data. The embedded data can be retrieved at the receiver side using the data-hiding key. As the data embedding only affects the LSB, decrypting the encrypted image with the encryption key can yield an image similar to the original. The use of both encryption and data-hiding keys enable the receiver to have a successful extraction of the embedded data and a perfect recovery of the original image by exploiting the spatial correlation present in the natural images.

While Puech [2] was the pioneer in proposing a VRAE-based RDHEI method, Ma et al. [5] introduced an innovative and practical approach based on the RRBE framework. They proposed the following three steps: image partition, self-reversible embedding, and image encryption. In order to accommodate messages, they proposed to substitute some LSBs in the encrypted image. Compared to the techniques described in [3,4,6], this method is likely to accommodate payloads that are more than 10 times larger while maintaining an acceptable PSNR close to 40 dB.

Yi et al. [7] have suggested to embed message bits according to the binary-block embedding method. This method embeds binary bits in several LSB planes of the original image into its MSB planes, and then encrypts it and hides the secret data inside its LSB planes. After embedding but before encryption, a bit-level scrambling process is employed to ensure resistance against noise and data loss attacks. The maximum ER of 10,000 images in the BOWSBase dataset is 2.2034 bpp, which outperforms many of the existing RDHEI methods.

A new RDHEI method was suggested by Yin et al. [8] with higher capacity and error-free data extraction and image decryption based on multi-MSB prediction and Huffman coding. Specifically, based on its predicted value, a tag value of each pixel is obtained and the total data embedding capacity of an image is determined. The advantage of using Huffman coding is to compress the label map to accommodate more space for embedding information. In this method, the obtained average ERs in two well known image datasets, BOSSbase and BOWS-2, reached 3.361 bpp and 3.246 bpp, respectively, leading to an increment in terms of the net payload of approximately 1 bpp compared to the previous works.

In [9], Wang et al. proposed a RDHEI based on multi-MSB embedding strategy. Through experimentation, they demonstrated that their approach can achieve an average ER of 1.721 bpp for the BOW-2 database.

Yin et al. [10] introduced an RDHEI algorithm that incorporates pixel prediction and multi-MSB plane rearrangement. The prediction errors are calculated through the median edge detector (MED) predictor. They are then separated into two components: the sign of the errors represented by one bit plane, and the absolute values of the errors represented by other bit planes. These bit planes are further divided into small blocks of 4×4 pixels and categorized as uniform or non-uniform blocks. This categorization enables the rearrangement of the blocks to accommodate auxiliary information and additional data. It is important to note that using small blocks of pixels increases the amount of auxiliary information, which, in turn, decreases the ER.

A RDHEI with extended parametric binary tree labeling was introduced in [11]. The experimental results demonstrated that the ERs for commonly used datasets, including Bossbase, BOWS-2, and UCID, are 3.2305 bpp, 3.1619 bpp, and 2.8113 bpp, respectively.

Additional methods [12–20] for RDHEI and many others can be found in [21], which is a survey that presents the birth and evolution of RDHEI methods over 12 years.

The motivation for our research in RDHEI stems from the critical need to find innovative solutions that strike a balance between data security and the processes of securely hiding and retrieving data within encrypted content. This paper presents a significant improvement in the advancement of a secure and efficient RDHEI technique, which holds substantial potential across various domains, including secure communication, healthcare, and cloud computing.

From the exploration of the existing methods for RDHEI, it is evident that various approaches have been developed to strike a balance between data hiding capacity and the preservation of error-free reversibility. Notably, among the methods under consideration, the RRBE approach consistently stands out, showcasing superior performance metrics, particularly in ER and error-free reversibility when compared to VRAE methods as they face significant challenges in achieving a satisfactory EC due to the lack of redundant space in encrypted images.

The main contributions of this article, provided with the proposed solution, are:

- RRBE approach: One of the primary contributions of our method is the utilization of the RRBE approach. This strategy leverages the data redundancy present in the original image before encryption to create space for data hiding. This approach ensures security as it maximizes the embedding capacity without altering the encryption process.
- Integration of pixel prediction: Our method introduces pixel prediction in the RDHEI process. By predicting the error image, the efficiency of data embedding is enhanced.
- Quadtree decomposition: Quadtree decomposition is applied to each bit plane of the mapped prediction error image. This step identifies homogeneous blocks within the image, which can be rearranged to create room for data embedding. Quadtree decomposition allows for a more fine-grained allocation of embedding space, potentially increasing the embedding capacity.
- Bit plane reordering: The method incorporates bit plane reordering as part of the data hiding process. This technique likely enhances the efficiency of embedding data by optimizing the allocation of space within each bit plane.
- Error-free reversibility: One of the most important contributions of our method is its ability to achieve high ERs while maintaining error-free reversibility. This means that the hidden data can be extracted without any loss or distortion, and the original image can be fully restored after decryption. Error-free reversibility is a crucial requirement in RDHEI to ensure that the quality and integrity of the image are preserved.
- Privacy preservation: Encryption is applied to all images being transmitted or stored, ensuring the privacy and confidentiality of sensitive information.

The rest of this paper is organized as follows. While Section 2 details the proposed separable high-capacity RDHEI scheme, Section 3 presents the obtained results for different image datasets, as well as a comparison with the state-of-the-art RDHEI methods. In addition, the reversibility, security, and embedding capacity of the proposed scheme are analyzed and discussed. Finally, general conclusions of this paper are drawn in Section 4.

2. The Proposed RDHEI Method

This section presents a novel RDHEI method that is highly efficient, reversible, and secure. The method is based on several steps that can be summarized into three main parts, which are the following:

- RRBE based on pixel prediction, quadtree decomposition, and bit plane reordering. The reserved room will be used for inserting auxiliary information and embedding additional data after image encryption. The content owner is responsible for both the room reservation and image encryption processes,
- data embedding which is performed by the data-hider,
- data extraction and image recovery performed by the receiver.

Figure 2 illustrates the various stages involved in reserving the embeddable room, encrypting an image, and embedding secret data. A detailed explanation of each of these stages will be provided in the upcoming sections.

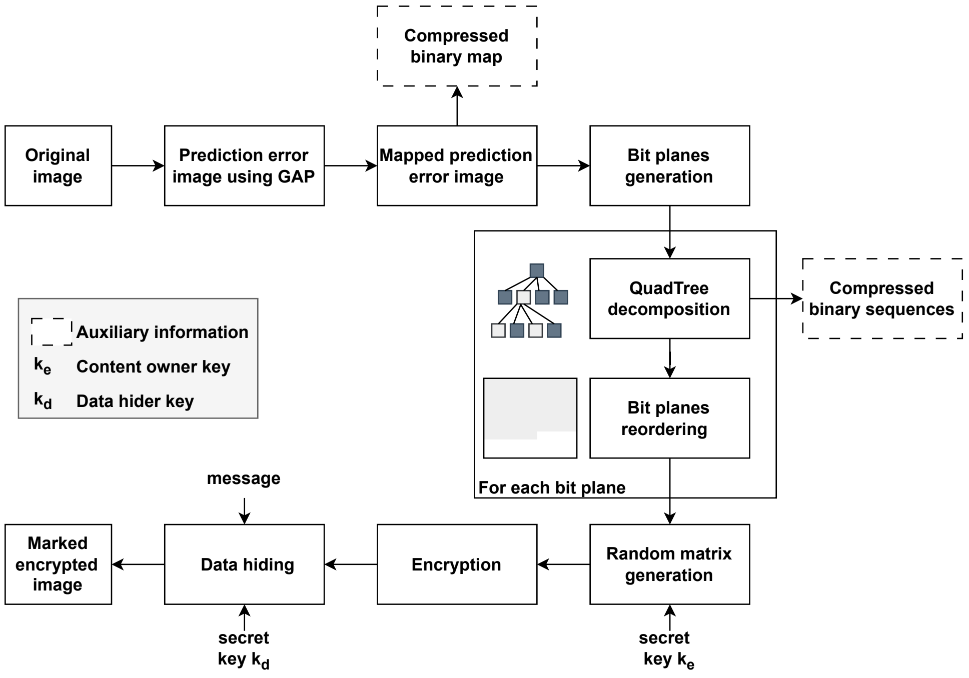


Figure 2. Overview of the Proposed RDHEI Methodology, featuring the RRBE approach, pixel prediction integration, quadtree decomposition, bit plane reordering, image encryption, and data hiding.

This section elaborates on the proposed method in detail. Firstly, Section 2.1 explains how to reserve room for data embedding and outlines the image encryption process. Besides, Section 2.2 details the procedure for embedding additional data. Finally, Section 2.3 provides a thorough explanation of data extraction and image recovery.

2.1. Room Reservation and Image Encryption

2.1.1. Pixel Prediction

Predictors are utilized in various image processing applications to exploit the spatial correlation among neighboring pixels. They make use of the values of adjacent pixels to estimate the value of a pixel. In image compression, predictors are used to reduce redundancy in the data by encoding the difference between the predicted value and the actual value of a pixel [22–24]. This difference is typically smaller than the actual value of the pixel, which means that the amount of data that needs to be stored or transmitted can be significantly reduced while maintaining the original image quality.

There are several types of predictors, and the choice of a specific predictor depends on the requirements of the target application and the characteristics of the images being used.

The prediction causal template used by the studied predictors is depicted in Figure 3, where x denotes the current sample, and A to G are the neighboring samples in the relative positions.

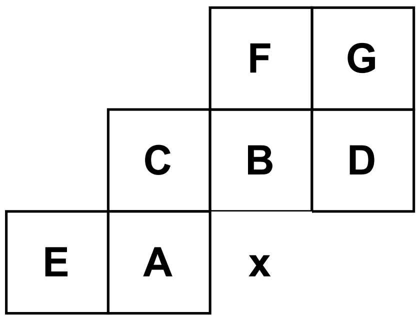


Figure 3. The causal template used by the studied predictors.

(a) ***The MED predictor***

The Median Edge Detector (MED) [22,25], is a type of image processing technique used to detect edges in an image. It operates by considering neighboring pixel values along edges within a local context. The predictor identifies pixels along edges by analyzing the differences between adjacent pixel values. It then computes a prediction for the current pixel by considering the median of these neighboring differences.

(b) ***The GAP predictor***

The Gradient Adjusted Prediction (GAP) [26] algorithm is designed to improve the accuracy of prediction by taking into account the local gradient of the image. In the GAP algorithm, the predicted value of a pixel is determined by using a linear combination of neighboring pixels.

(c) ***First improvement of the MED predictor***

The performance of the MED technique is insufficient when it comes to diagonal edges, resulting in a significant prediction inaccuracy. Jiang [27] proposed a method for improving diagonal edge detection. His method evaluates the neighboring pixels' local gradients and estimates the presence of a diagonal edge. If there is a diagonal edge, the prediction is made using a different method and two thresholds.

(d) ***Second improvement of the MED predictor***

The second improvement detects various sorts of edges based on the set thresholds. In the study by Edirisinghe et al. [28], it is demonstrated that the edge detection techniques employed by the MED predictor are constrained to recognizing solely horizontal and vertical edges. As a result, a low-complexity, cost-effective method was introduced to effectively identify diagonal edges.

(e) ***Time complexity***

The predictors under investigation make estimations of pixel values in a target image by considering neighboring pixels within a local neighborhood, as depicted in Figure 3. The computational efficiency of these predictors primarily relies on two key factors:

- The size of the local neighborhood (e.g., 3 pixels for MED and 7 pixels for GAP).
- The algorithm used to calculate the predicted value from the neighborhood.

Given that the size of the local neighborhood is relatively small (e.g., 3 or 7), the complexity associated with predicting the value of a target pixel can be simplified to

$\mathcal{O}(1)$. Consequently, the overall time complexity for each of the investigated predictors when applied to an image of dimensions $n \times m$ can be succinctly expressed as $\mathcal{O}(nm)$.

Whatever the prediction technique used, the predicted error pixel is always calculated as $e(u,v) = x(u,v) - \hat{x}(u,v)$ and the resulting errors range from -255 to 255 for 8-bit grayscale images, which requires 9 bits to encode each error value. Figure 4 shows the original Lena image, its histogram, the prediction error Lena image, obtained by using the GAP predictor, and its histogram. Throughout this paper, in all histogram figures, the horizontal axis is dedicated to representing pixel intensities, while the vertical axis is employed to illustrate the frequency associated with each intensity level.

From Figure 4d, it can be clearly seen that most of the prediction errors are very close to 0. Moreover, Figure 5 shows the original Jetplane image, its histogram, the prediction error Jetplane image, and its histogram.

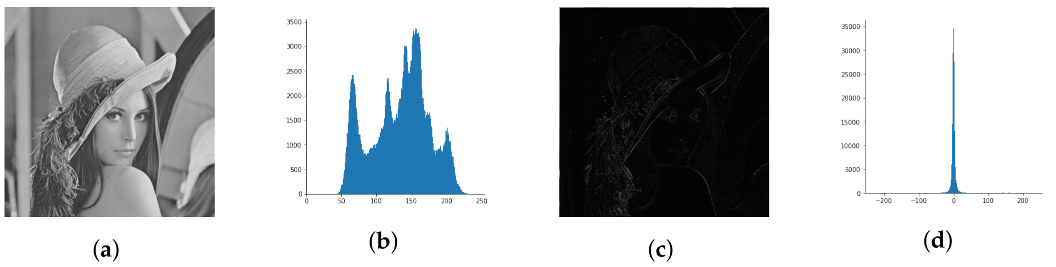


Figure 4. (**a**) The Lena image and (**b**) its histogram. (**c**) The prediction error Lena image and (**d**) its histogram obtained by using the GAP predictor.

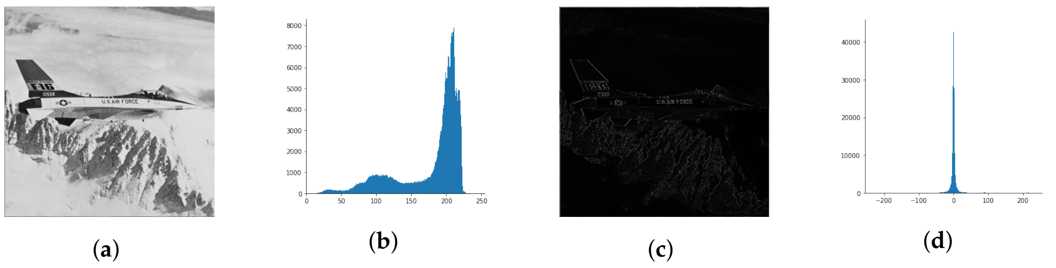


Figure 5. (**a**) The Jetplane image and (**b**) its histogram. (**c**) The prediction error Jetplane image and (**d**) its histogram obtained by using the GAP predictor.

Furthermore, Figure 6 illustrates two bit-plane decomposition schemes that have been applied to the Lena image. The columns represent the bit planes, starting from the sign bit and progressing from the least significant bit to the most significant bit. The rows show the bit-plane decompositions of the original image and the prediction error image achieved through the utilization of the GAP predictor.

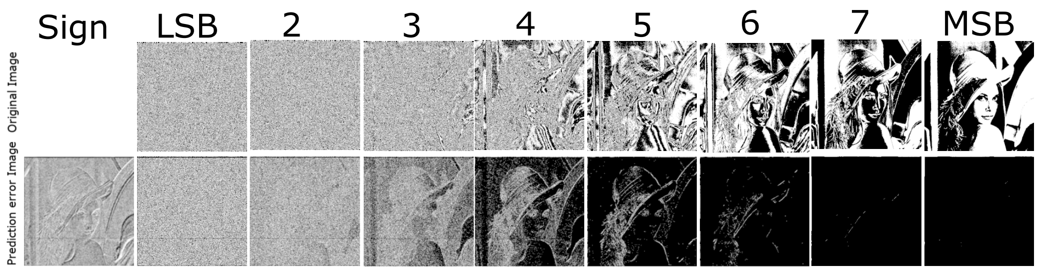


Figure 6. The Lena image undergoes two bit-plane decomposition schemes. The columns represent the bit planes, starting from the sign bit and progressing from the least significant bit to the most significant bit. The rows display the bit-plane decompositions of the original image and the prediction error image achieved through the GAP predictor.

In our method, we propose to map the prediction errors to non-negative values in order to reduce the error range from $[-255, 255]$ to $[0, 255]$ by applying Equation (1).

$$e(u,v) = \begin{cases} x(u,v) & if|e(u,v)| > 127 \\ (|e(u,v)| << 1)|SIGN & otherwise \end{cases}, \quad (1)$$

where $|.|$ denotes the absolute value function, $<<$ is the bitwise left shift operator, and $|$ is the bitwise OR operator. $SIGN$ is a binary variable that contains 1 if the predicted error $e(u,v)$ is negative and 0 otherwise. It should be noted that, if the absolute value of the error is less than or equal to 127, $|e(u,v)| \leq 127$, only 7 bits are required to encode it, and additionally one more bit representing the sign is concatenated to its binary representation for ensuring that the process is fully reversible. Finally, when the error is greater than 127 or less than -127, it is considered an overflow pixel and modified by $e(u,v) = x(u,v)$. Figure 7a,c show the bit-planes of the mapped prediction error images for Lena and Jetplane images, respectively.

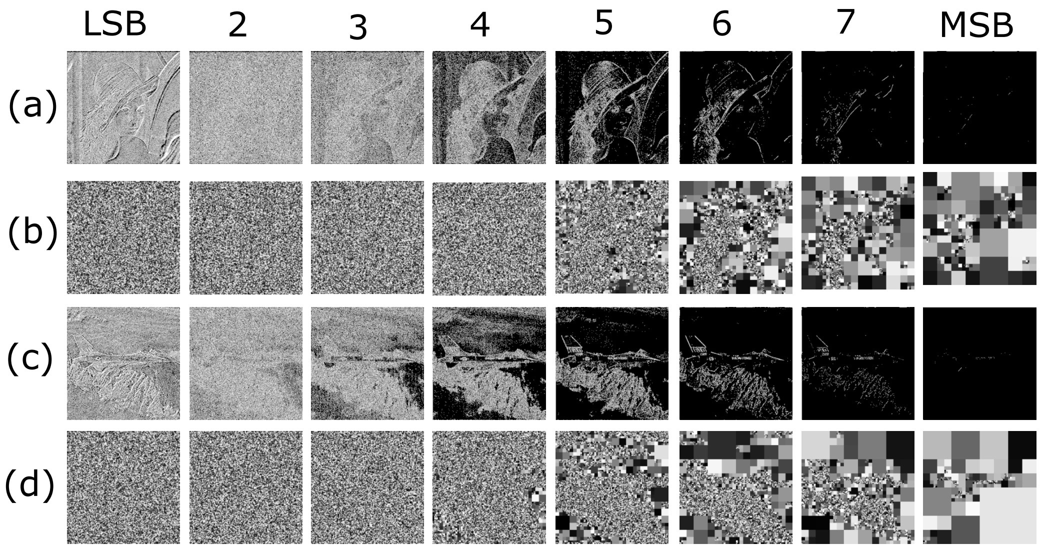


Figure 7. (**a**) Bit-planes of the mapped prediction error Lena image, (**b**) and their corresponding quadtree decompositions. (**c**) Bit-planes of the mapped prediction error Jetplane image, (**d**) and their corresponding quadtree decompositions. The bit planes start with the LSB to the MSB. The mapped prediction error images were obtained by using the GAP predictor.

Algorithm 1 details the process for the prediction error mapping. The outputs of this algorithm are the mapped prediction error image as well as the binary label map. This label map is used to distinguish between the different formulas used in identifying each prediction error. It is a binary map with the same size of the original image, which is filled as follows: $labelMap(u,v)$ is set to 1 for the overflow pixels and 0 otherwise. It is also compressed by using arithmetic coding [24,29–31] and then inserted into the mapped prediction error image as an auxiliary information. It is worth mentioning that for most of the tested images, this label map contains zeros only, and consequently, only one bit is required to represent it. For a few images, even if the label map contains non-zero values, a large number of 0 and a fairly small number of 1 are found, which leads to an effective compression by arithmetic coding and therefore a very small number of bits is required for its representation.

Using this label map with the mapped prediction errors instead of representing the prediction error image with 9 bit planes, including the sign bit plane and bit planes from the LSB to the MSB, offers a significant advantage in terms of efficiency. This advantage primarily stems from the label map's requirement for far fewer bits to represent the same information compared to the sign bit plane and multiple bit planes.

Algorithm 1 mappedPredictionError algorithm.

```
function mappedPredictionError($x$)
Input: An Image $x$ of size $n \times m$.
Output: A mapped prediction error image $e$, and a Label map $labelMap$.

$labelMap$: an array of size $n \times m$ initialized with zeros.
for $u \leftarrow 1$ to $n$ do
    for $v \leftarrow 1$ to $m$ do
        $L \leftarrow listOfNeighbors(u, v)$
        $\hat{x}(u, v) \leftarrow prediction(L)$
        $e(u, v) \leftarrow x(u, v) - \hat{x}(u, v)$
        if $|e(u, v)| > 127$ then
            $e(u, v) \leftarrow x(u, v)$
            $labelMap(u, v) \leftarrow 1$
        else
            $SIGN \leftarrow 0$
            if $e(u, v) < 0$ then
                $SIGN \leftarrow 1$
            end if
            $e(u, v) \leftarrow (|e(u, v)| << 1)|SIGN$
        end if
    end for
end for

return $labelMap, e$
end function
```

It is worth mentioning that the article is supplemented with an Abbreviations part containing a table enumerating the symbols and functions utilized in the algorithms, providing clear explanations of their meanings to enhance reader comprehension.

2.1.2. The Quadtree Decomposition

Quadtree is a hierarchical data structure that recursively subdivides an image into square blocks until a certain condition is met. Quadtree partitioning is widely used in various applications, including image processing, computer graphics, and computational geometry. Figure 7a,c demonstrate the MSBs of the mapped prediction errors for the Lena and Jetplane images, respectively. The significant bit planes exhibit multiple uniform regions. This can serve as an effective way to provide room for data hiding. Rather than dividing each bit plane into non-overlapping blocks of equal size (usually 4×4) [8], we suggest employing quadtree decomposition to generate large, uniform blocks with minimal auxiliary information.

Figure 8 shows the MSBs of the mapped prediction errors for the Lena and Jetplane images, and their corresponding quadtree decompositions. As observed from Figure 8b,d, extremely larger uniform blocks are found. A binary string is then used to represent the quadtree decomposition of each bit plane. Additional bit planes and their corresponding quadtree decompositions are shown in Figure 7.

In the conventional implementation of the quadtree decomposition, an image is recursively divided into smaller sub-blocks until each block becomes homogeneous, indicating that all pixels in the block have the same intensity value, or until the size of the sub-blocks becomes less than a defined limit (2×2 in our case). The quadtree decomposition can be seen as a depth-first traversal (DFS) of a tree structure, where it starts from the root node (i.e., the whole image) and recursively visits its child nodes until it reaches the leaf nodes (i.e., homogeneous sub-blocks, or sub-blocks with size 2×2), meaning that it visits all child nodes of a parent node before moving to its siblings.

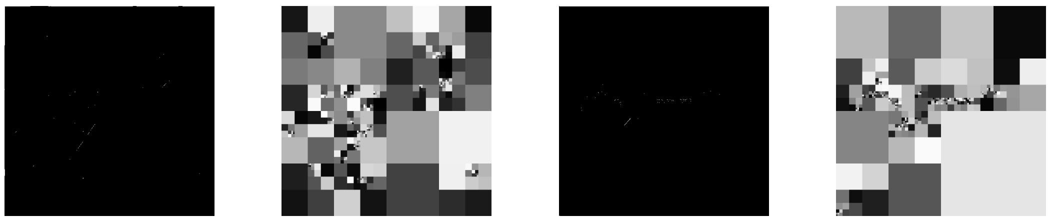


Figure 8. (**a**) The MSB plane of the mapped prediction error Lena image, (**b**) and its quadtree decomposition. (**c**) The MSB plane of the mapped prediction error Jetplane image, (**d**) and its quadtree decomposition.

The conventional implementation of the quadtree decomposition does not meet well with the requirements of our room reservation method. When using the quadtree decomposition based on the DFS approach, the algorithm tends to traverse very small, homogeneous, and non-homogeneous blocks first. However, reordering the homogenous blocks into the beginning of each embeddable bit plane is needed, and we propose a new implementation that prioritizes the traversal of the largest homogeneous blocks first. This allows us to gradually reorder each bit plane as the quadtree decomposition continues processing it. Algorithm 2 details the proposed quadtree decomposition method. For each embeddable bit plane, the algorithm outputs the binary string, corresponding to its quadtree decomposition, and the list of homogeneous blocks that will be used in the data hiding process.

2.1.3. The Bit Plane Reordering

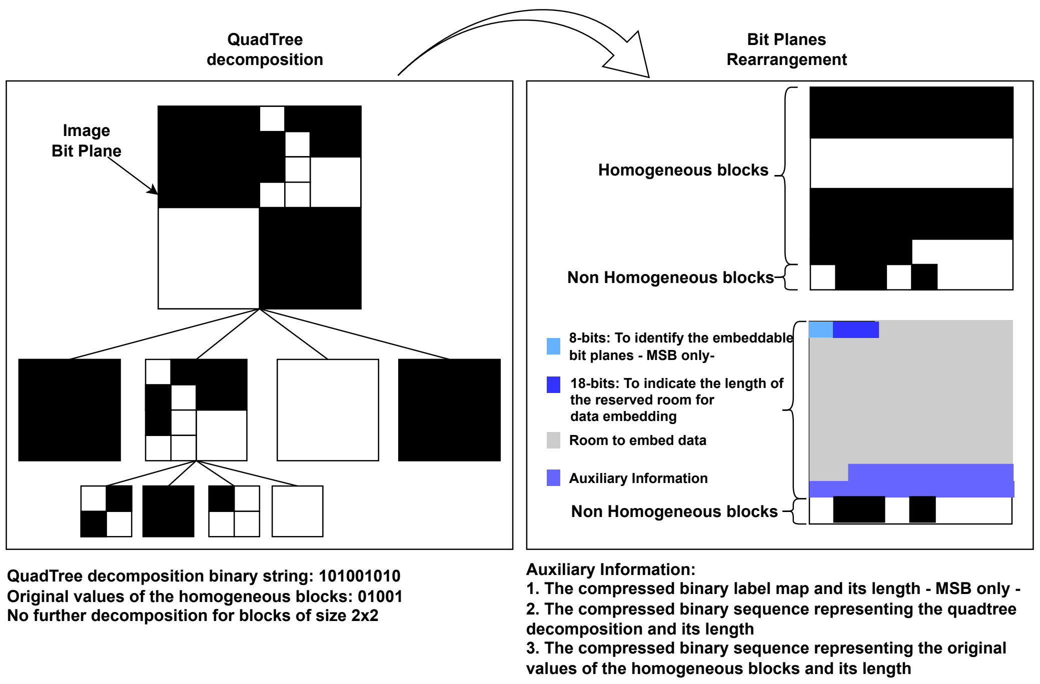
As presented in Algorithm 2, the quadtree decomposition tries to find homogeneous blocks that are as large as possible to reserve room for data embedding with minimum auxiliary information. This algorithm exploits the high redundancy of the most significant bit planes obtained by the mapping prediction error process. The outputs of this algorithm are used in the reordering process and in the reconstruction process in order to recover the original image after decryption. The algorithm iterates through the homogeneous blocks for each bit plane in accordance with the quadtree decomposition steps, with homogeneous blocks arranged at the start and non-homogeneous blocks at the end, as shown in Figure 9. After bit planes reordering, the auxiliary information can now be inserted into the homogenous part of each bit plane leaving room for data embedding. Further details about the auxiliary information can be found in Figure 9. It is noteworthy that all auxiliary information are compressed using arithmetic coding in order to reduce the amount of bits required for their representations.

Figure 9 details the quadtree decomposition and the bit plane reordering steps. For each bit plane, one bit is required to indicate if it is embeddable or not. As we have 8-bit planes, 8 bits are required to identify the embeddable ones. The next 18 bits are used to indicate the length in bits of the embeddable room. These two pieces of auxiliary information will be used by the data hider to insert hidden data into the encrypted bit planes. After the last bit of the embeddable room, we propose to insert the following auxiliary information in order:

- The compressed binary label map and 10 bits indicating its length. This information needs to be inserted in the MSB plane only.
- The compressed binary string representing the quadtree decomposition, with 16 bits indicating its length.
- The compressed binary string corresponding to the original values of the homogeneous blocks, which also requires 16 bits to represent its length.

Algorithm 2 Quadtree decomposition of a mapped prediction error bit plane.

```
struct block
$x$ {Starting row index}
$y$ {Starting column index}
$size$ {Size of the block}
$value$ {One bit to represent the value of an homogeneous block}
end struct

Input: An Image bit plane.
Output: $binaryStr$: a string that represents the quadtree decomposition of the input bit plane. $listHomogeneousBlocks$, the list of homogeneous blocks.

function quadtreeDecomposition($bitPlane$)
$rows, cols \leftarrow shape(bitPlane)$
$binaryStr \leftarrow ''''$
$listHomogeneousBlocks \leftarrow [\,]$
struct block root $\leftarrow \{0, 0, rows, ''''\}$
$q \leftarrow Queue()$
$q.put(root)$

while $not\ q.empty()$ do
  $node \leftarrow q.get()$
  $currentBlock \leftarrow getBlock(bitPlane, node)$
  if $isHomogeneous(currentBlock)$ then
    $binaryStr \leftarrow binaryStr + ''0''$
    $node.value \leftarrow currentBlock[0, 0]$
    $listHomogeneousBlocks.append(node)$
    continue
  end if

  $binaryStr \leftarrow binaryStr + ''1''$
  //The block is not subdivided into 4 sub-blocks if its size is equal to $2 \times 2$ pixels
  if $node.size == 2$ then
    continue
  end if

  // If the block is not homogeneous, split it into four sub-blocks and add them to the queue
  $halfSize \leftarrow node.size//2$

  // Add the new 4 sub-blocks to the queue
  $q.put(block(node.x, node.y, halfSize, ''''))$
  $q.put(block(node.x, node.y + halfSize, halfSize, ''''))$
  $q.put(block(node.x + halfSize, node.y, halfSize, ''''))$
  $q.put(block(node.x + halfSize, node.y + halfSize, halfSize, '''')$

end while

return $binaryStr, listHomogeneousBlocks$
end function
```



QuadTree decomposition binary string: 101001010
Original values of the homogeneous blocks: 01001
No further decomposition for blocks of size 2x2

Auxiliary Information:
1. The compressed binary label map and its length - MSB only -
2. The compressed binary sequence representing the quadtree decomposition and its length
3. The compressed binary sequence representing the original values of the homogeneous blocks and its length

Figure 9. Overview of the quadtree decomposition and the bit plane reordering steps.

In the proposed method, the total embedding capacity (in bits) is then calculated using Equation (2).

$$Total\ EC = \sum_{i=1}^{length(BPs)} \sum_{k=1}^{length(HBs)} HomogeneousBlocks[i][k].size^2 - sizeOf(ReferencePixels) \times 8, \tag{2}$$

where BPs corresponds to the list of the embeddable bit planes and HBs is the list of the homogeneous blocks for each bit plane. The function $sizeOf(ReferencePixels)$ returns the total number of bytes representing the reference pixels used by the predictor. It is worthy to note that the two first rows (i.e., from the top), the two first columns (i.e., from the left), and the last column of an image are considered reference pixels when using the GAP predictor. The reference pixels maintain their original values throughout all the processes, except in the encryption process.

The total auxiliary information (in bits) is calculated according to Equation (3).

$$\begin{aligned} Auxiliary\ Information &= 8 + 18 \times length(BPs) + 10 + sizeOf(compressed(labelMap)) \\ &+ \sum_{i=1}^{length(BPS)} (16 + sizeOf(compressed(binaryStr[i]))) \\ &+ \sum_{i=1}^{length(BPS)} (16 + sizeOf(compressed(\ concat_{k=1}^{length(HBs)}(HomogeneousBlocks[i][k].value)))). \end{aligned} \tag{3}$$

The net payload (in bits) available for data embedding by the data hider is calculated using Equation (4).

$$Net\ Payload = Total\ EC - Auxiliary\ Information \tag{4}$$

Consequently, the ER is computed using Equation (5).

$$ER = \frac{Net\ Payload}{n \times m} \tag{5}$$

where $n \times m$ is the image size. The ER is a fundamental metric, measured in bits per pixel (bpp), which quantifies the average number of bits that can be concealed within each pixel of an encrypted image. A higher ER signifies the ability to hide more data within the image.

Figure 10 shows the bit-planes of the reordered mapped prediction errors for two images, Lena and Jetplane, before and after adding their corresponding auxiliary information. The figure shows the conversion of binary values $\{0,1\}$ obtained from the mapped prediction error bit planes into $\{0,255\}$ to represent the results in a more appropriate manner. In addition, the binary values representing the auxiliary information are transformed from $\{0,1\}$ to $\{0,128\}$ in order to differentiate between the different components of each bit plane.

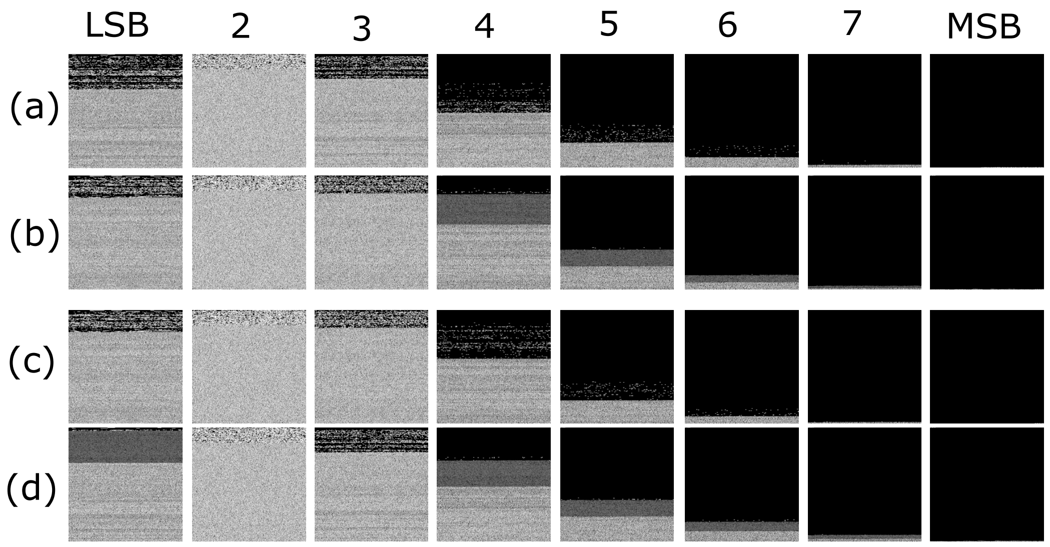


Figure 10. Bit-planes of the reordered mapped prediction error Lena image before (**a**) and after (**b**) adding the corresponding auxiliary information. Bit-planes of the reordered mapped prediction error Jetplane image before (**c**) and after (**d**) adding the corresponding auxiliary information.

2.1.4. Image Encryption

In this section, we encrypt each reordered mapped prediction error bit plane using an encryption key K_e. We first generate a pseudo-random matrix r, with the same size of the original image, using the key K_e. We then encrypt the reordered mapped prediction error image according to Equation (6) and generate the encrypted image x_e.

$$x_e(u,v) = e(u,v) \oplus r(u,v), \tag{6}$$

where $1 \leq u \leq n, 1 \leq v \leq m$, and $\oplus$ is the exclusive-or (XOR) operation.

Figure 11 presents the encryption process. With no loss of generality, in this work the Piecewise Linear Chaotic Map (PWLCM) [32–35] is used to generate the encryption matrix, taking into account that any other efficient chaotic system can be used [36–38]. It is to be noted that the bits required by the data hider, which are located at the top-left corner of each bit plane, remain unencrypted.

The PWLCM is a map that can be described as in Equation (7):

$$x_i = F(x_{i-1},\eta) = \begin{cases} \dfrac{x_{i-1}}{\eta} & 0 < x_{i-1} < \eta \\ \dfrac{x_{i-1}-\eta}{0.5-\eta} & \eta \leq x_{i-1} < 0.5 \\ F(1-x_{i-1},\eta) & 0.5 \leq x_{i-1} < 1 \end{cases}, \tag{7}$$

where $\eta \in (0,0.5)$ is the control parameter, and $x_0 \in (0,1)$ is the initial value.

In our encryption scheme, we suggest using the PWLCM to generate the random matrix r to be scored with the reordered mapped prediction error image e, as depicted in Equation (6). We employ two distinct PWLCMs to generate two sequences, denoted as $A = a_1, a_2, \ldots, a_{nm}$ and $B = b_1, b_2, \ldots, b_{nm}$. It is important to maintain that both the image to be encrypted and the random matrix must share the same dimensions, specifically n rows and m columns, corresponding to the original image's size. It is noteworthy that different control parameters (η_1, η_2) and initial values (x_0, y_0) are employed to generate the two sequences A and B. These initial values and control parameters serve as the secret key, designated as $k_e = (x_0, \eta_1, y_0, \eta_2)$.

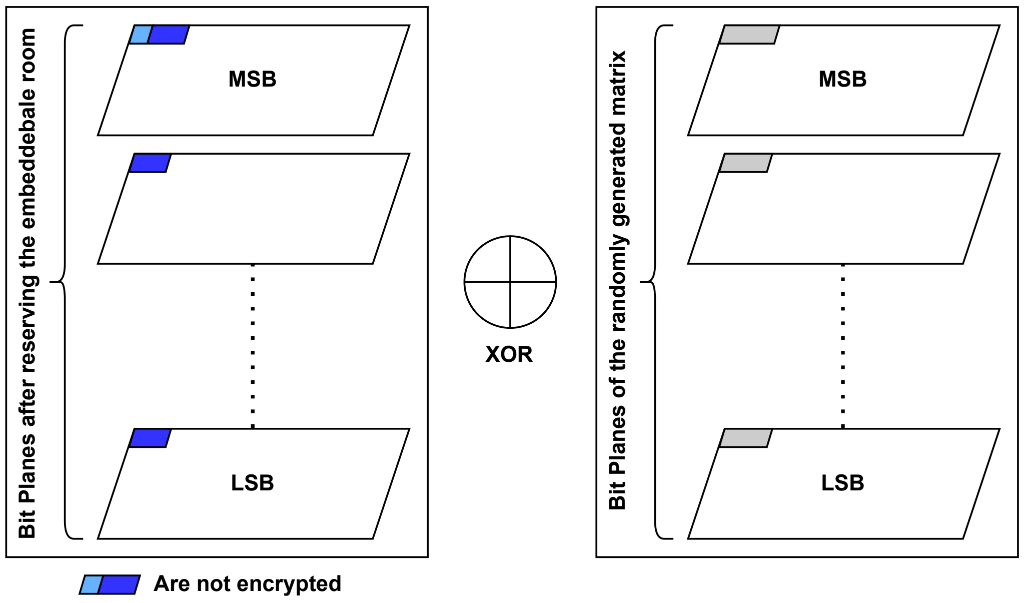

Bit Planes after reserving the embeddebale room
MSB
LSB
XOR
Bit Planes of the randomly generated matrix
MSB
LSB
Are not encrypted

Figure 11. The encryption process.

The first PWLCM employs Equation (8) to convert the sequence $X = x_1, x_2, \cdots, x_{nm}$ to the integer sequence A by truncating it, where $\lfloor . \rfloor$ is the floor function. Consequently, each integer a_i within the sequence A comprises 14 digits. We propose further decomposing a_i into two integers, a_i^{1-7} and a_i^{8-14}, each composed of 7 digits. As an example, if $a_i = 11223344556677$ the $a_i^{1-7} = 1122334$ and $a_i^{8-14} = 4556677$.

$$A = \lfloor X * 10^{14} \rfloor. \tag{8}$$

The second PWLCM uses the following equation to convert $Y = y_1, y_2, \cdots, y_{nm}$ into a binary sequence B:

$$B = mod(\lfloor Y * 10^{14} \rfloor, 2), \tag{9}$$

where $mod(x, y)$ is the function that returns the remainder when x is divided by y. Finally, each integer within the random matrix r is generated according to the following equation:

$$r_i = \begin{cases} mod(a_i^{1-7}, 256) & y_i = 0 \\ mod(a_i^{8-14}, 256) & otherwise \end{cases}. \tag{10}$$

This process effectively produces the random matrix r, which is crucial for the encryption process.

Algorithm 3 implements the $processImage$ procedure by calling the following functions:

- $mappedPredictionError$ for the mapped prediction error and label map calculation.
- $quadtreeDecomposition$ for the quadtree decomposition and the generation of the binary string and the list of the homogeneous blocks for each bit plane.
- $calculateAuxInfo$ for the calculation of the auxiliary information as detailed in Figure 9.
- $bitPlaneReordering$ for arranging the homogeneous blocks at the start and non-homogeneous blocks at the end of each bit plane. Furthermore, this function incorporates the compressed auxiliary information at the end of the homogeneous segment, allocating the remaining bits for data embedding through bit replacement.
- $encrypt$ to encrypt the resulting image with the data owner key k_e.

Algorithm 3 processImage algorithm: by the image owner.

```
Input: An image $x$ of size $n \times m$, encryption key $k_e$
Output: The encrypted image $x_e$.
$e, labelMap \leftarrow mappedPredictionError(x)$
// binaryStr is used to store the quadtree decomposition of each bit plane
$binaryStr$: string
// listHomogeneousBlocks is used to store the list of all homogeneous blocks from each
bit plane
$listHomogeneousBlocks$: list
// Extract 8 bit planes and calculate their quadtree decompositions
for $i \leftarrow 0$ to 7 do
  $bitPlane \leftarrow (e >> i)$ & 1
  $binaryStr, listHomogeneousBlocks \leftarrow quadtreeDecomposition(bitPlane)$
  $auxInf \leftarrow calculateAuxInfo(labelMap, binaryStr, listHomogeneousBlocks)$
  $bitPlaneReordering(e, i, compressedAuxInf, binaryStr, listHomogeneousBlocks)$
end for
$x_e \leftarrow encrypt(e, k_e)$
```

2.2. Data Embedding

To embed additional data into the encrypted image, it is necessary to extract the partial auxiliary information from the top left of each bit plane. Based on this information, one bit indicating if the current bit plane is embeddable or not with 18 bits representing the length of the reserved room, the data hider can embed additional information accordingly. It should be noted that to improve the security of the embedded data, it is encrypted using the data hiding key K_d prior to the embedding process. With no loss of generality, the identical encryption scheme used to encrypt the original image is equally employed for encrypting the embedded data. The final output is a marked encrypted image, as shown in Figure 12. Furthermore, Algorithm 4 provides a comprehensive breakdown of the steps involved in the data embedding process.

Algorithm 4 dataEmbedding: by the data hider.

```
Input: An encrypted image $x_e$ of size $n \times m$, a message, and an encryption key $k_d$.
Output: The marked encrypted image $x_e$.
// Encrypt the data hider message.
$encryptedMsg \leftarrow encrypt(message, k_d)$
// Extract the partial auxiliary information from the encrypted image.
$embeddable, length \leftarrow readPartialInf(x_e)$
// The variable totalEmbeddedBits keeps a record of the number of bits extracted
// from the encrypted message and embedded into the encrypted image by bit replace-
ment.
$totalEmbeddedBits \leftarrow 0$
for $i \leftarrow 0$ to 7 do
  if $embeddable[i] == 1$ then
    $embedBits(x_e, encryptMsg, totalEmbeddedBits, length[i])$
    $totalEmbeddedBits \leftarrow totalEmbeddedBits + length[i]$
  end if
end for
```

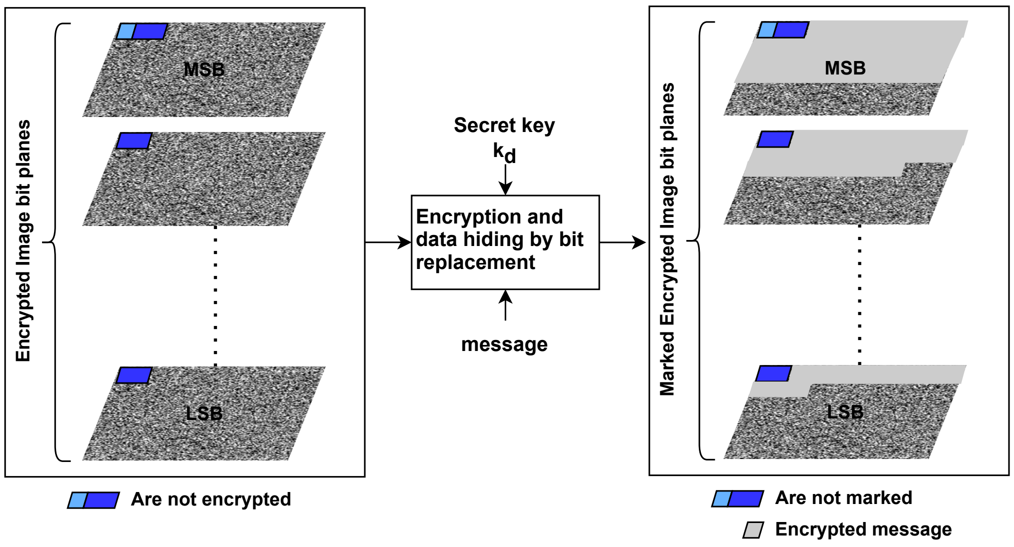

Encrypted Image bit planes
MSB
LSB
Are not encrypted
Secret key
k_d
Encryption and data hiding by bit replacement
message
Marked Encrypted Image bit planes
MSB
LSB
Are not marked
Encrypted message

Figure 12. The data hiding process.

2.3. Data Extraction and Image Recovery

Depending on the key held by the receiver, either the embedded data or the original image can be obtained. There are three possible scenarios based on the receiver's possession of the different keys, as shown in Figure 13:

1. If the receiver possesses the data hiding key K_d, he can successfully retrieve the embedded data without any errors. The receiver begins by dividing the marked encrypted image into 8 bit planes. To identify the bit planes involved in the embedding process and extract the embedded data, the receiver follows the procedure explained earlier, starting from the top left and reading the first bits in each bit plane. These bits indicate whether the bit plane is utilized for data hiding and provide information about the available embedding space. The hidden data are then read from the encrypted image and decrypted using the key K_d. However, since the receiver lacks access to the encryption key K_e, they are unable to decrypt the image and restore it to its original form.
2. If the receiver has the encryption key K_e, they can reconstruct the original image without any loss. The receiver starts by decrypting the image using its encryption key. Following that, they extract the auxiliary information from each bit plane that will be used in the image recovery process. Initially, the label map is extracted from the MSB plane of the decrypted image. Subsequently, the compressed binary strings and their corresponding lengths are extracted and decompressed to recover the positions and the values of all image reordered blocks. Finally, using the extracted decompressed label map, we inverse the mapping prediction error process to recover the original image.
3. When the receiver possesses both the decryption key K_d and the encryption key K_e, they can successfully retrieve the data without any errors. By employing the same detailed steps as in the previous cases, they can decrypt the image and restore it to its original form without any loss or distortion.

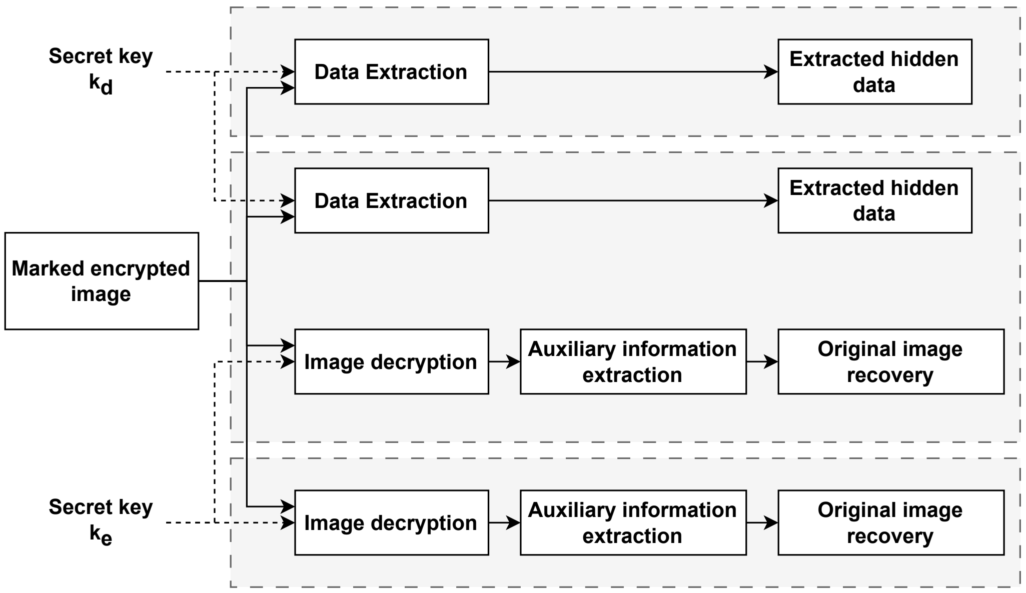


Figure 13. The three possible scenarios depending on the receiver's possession of the different keys for data extraction and image recovery.

3. Experiment Results and Analysis

This section provides an in-depth evaluation of the proposed method, including experimental results and analysis. The experimental results are obtained using five well known and frequently used test images, namely Lena, Man, Jetplane, Baboon, and Tiffany, as shown in Figure 14. In addition, tests are also conducted on the following two image datasets:

- BOSSbase [39]: is a public image dataset from the Break Our Steganographic System! competition [39]. It is formed by 9074 cover images with a size of 512 × 512 pixels.
- BOWS-2 (https://data.mendeley.com/datasets/kb3ngxfmjw/1) (accessed on 21 January 2021): is a public image dataset formed by 10, 000 cover images with a size of 512 × 512 pixels.

Additionally, the section encompasses an evaluation of security, performance, and a comparative analysis with other contemporary techniques in terms of embedding capacity.

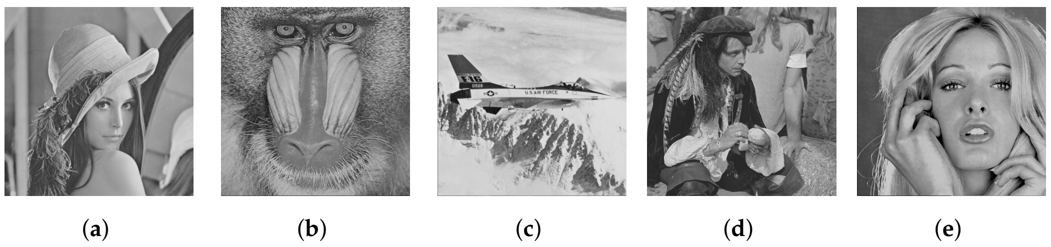


Figure 14. The test images: (**a**) Lena, (**b**) Baboon, (**c**) Jetplane, (**d**) Man, and (**e**) Tiffany.

3.1. Security Analysis

Figures 15 and 16 illustrate the proposed RDHEI method for two test images, Lena and Jetplane, respectively.

Figure 15a showcases the original Lena image. After reordering the bit planes in the mapped prediction error image, the allocation of space for data embedding, and the insertion of auxiliary information in the corresponding bit planes, the resulting image displayed in Figure 15b is obtained. The encrypted image, obtained by the content-owner using their encryption key with a net payload of 3.6 bpp, is depicted in Figure 15c. Subsequently, the data-hider incorporates additional data into the encrypted image, resulting in the creation of a marked encrypted image shown in Figure 15d. The disparity between the encrypted image and the marked-encrypted image is presented in Figure 15e. Finally, the recovered Lena image is showcased in Figure 15f, with a Mean Square Error (MSE) value of 0, affirming its matching with the original one. The same outcomes are observed for the Jetplane image illustrated in Figure 16.

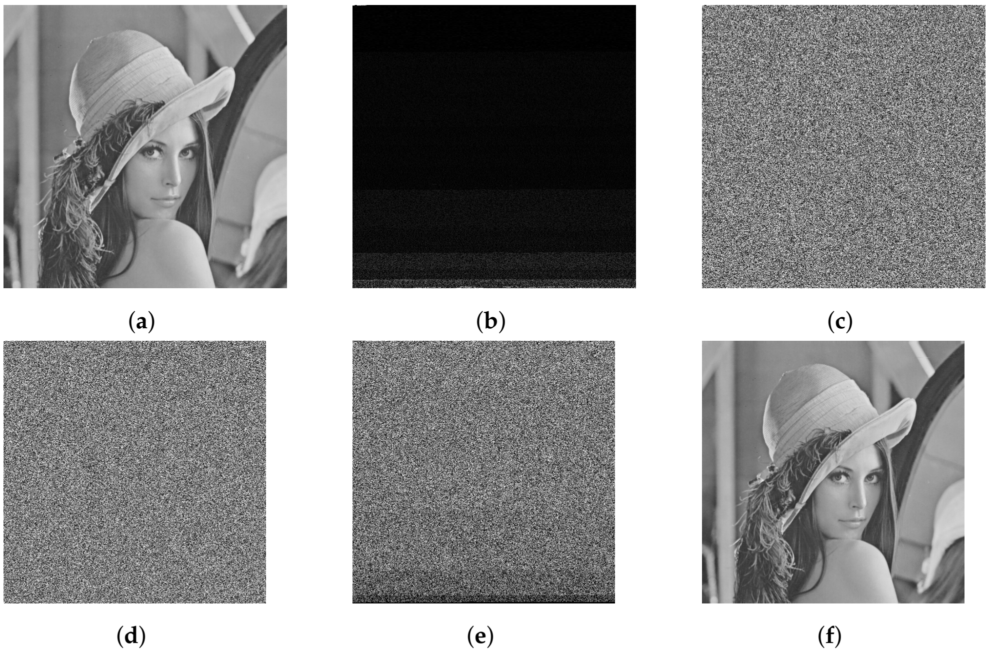


Figure 15. The proposed RDHEI method for the Lena image is depicted as follows: (**a**) the original Lena image, (**b**) the reordered mapped prediction error image, (**c**) the encrypted image, (**d**) the marked encrypted image with a net payload of ER = 3.6 bpp, (**e**) the bitwise XOR operation between the marked and encrypted image, (**f**) and the reconstructed image with an MSE value of 0.

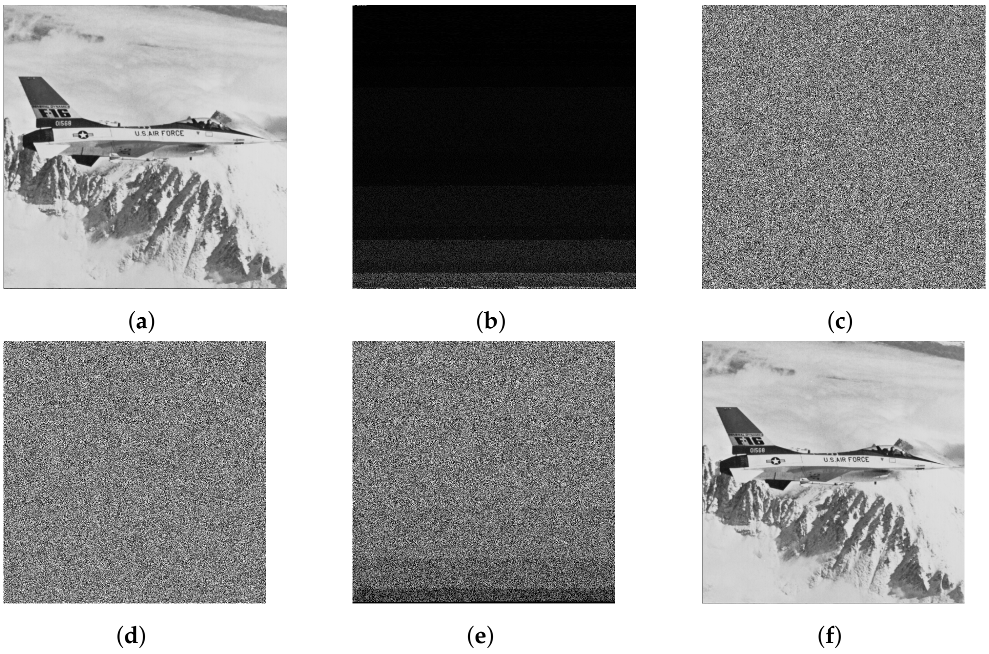


Figure 16. The proposed RDHEI method for the Jetplane image is depicted as follows: (**a**) the original Jetplane image, (**b**) the reordered mapped prediction error image, (**c**) the encrypted image, (**d**) the marked encrypted image with a net payload of ER = 3.64 bpp, (**e**) the bitwise XOR operation between the marked and encrypted image, (**f**) and the reconstructed image with an MSE value of 0.

Additionally, this section aims to demonstrate the robustness of the proposed encryption scheme against different attacks. To achieve this goal, several metrics are employed, namely the histogram, the χ^2 statistic, the correlation coefficient, and the information entropy. These metrics serve as evidence that an ideal encryption algorithm should be resilient to different types of attacks.

(a) ***The key space analysis***

The key space denotes the total count of distinct keys available for utilization in the encryption process. Within our proposed algorithm, the secret keys encompass the control parameters and the initial values of the two used PWLCM maps $k_e = (x_0, \eta_1, y_0, \eta_2)$. Assuming a computational precision equivalent to double-precision numbers, which is approximately 10^{16}, then the key space H_{x_0} and H_{y_0} is approximately 10^{16}, while H_{η_1} and H_{η_2} are around 0.5×10^{16}. Consequently, the total key space H is calculated as $H = H_{x_0} \times H_{y_0} \times H_{\eta_1} \times H_{\eta_2}$, yielding a value of approximately 0.25×10^{64}. The key space equates to approximately 1.519×2^{210}. For an encryption system to be considered effective and resilient against brute-force attacks, the size of the key space should not fall below 2^{100}, making such attacks infeasible [40]. Consequently, the key space of our encryption scheme is large enough to ensure robust protection against all kinds of brute-force attacks.

(b) ***The histogram analysis***

We have analyzed the histograms of several test images and their corresponding encrypted and marked-encrypted images. Figure 17 presents the histograms of the encrypted and marked-encrypted images for Lena and Jetplane.
The histograms shown in Figure 17 indicate that the encrypted and the marked-encrypted images follow a uniform distribution. Therefore, both the encrypted and marked-encrypted images exhibit distinct statistical characteristics from the original image. Similar results for the histogram analysis were found for all the test images.

(c) ***The χ^2 test***

The χ^2 values are computed for the original, the encrypted, and the marked-encrypted images. The χ^2 value is described as follows:

$$\chi^2 = \sum_i \left(f_i - \frac{n \times m}{256} \right)^2 / \frac{n \times m}{256}, \tag{11}$$

where i is the number of gray levels $(0, \ldots, 255)$, f_i is the observed frequency count of each gray level, and $n \times m$ is the image size. It should be noted that the χ^2 test is a measure of how much the observed frequencies deviate from the expected frequencies, which is, in our case, the uniform distribution. Assuming a significant level of 0.05, then the critical value for the given significance level and degrees of freedom is $\chi^2(255, 0.05) = 293$. The χ^2 values of the different images for Lena and Jetplane are presented in Table 1. These values, the χ^2 statistics, are compared to the critical value to determine whether to reject the null hypothesis that the image is uniformly distributed or not. If the calculated χ^2 statistic for an image is greater than the critical value, the null hypothesis is rejected, indicating that the image is not uniformly distributed. If the calculated χ^2 statistic is less than or equal to the critical value, the null hypothesis is not rejected, and the image is uniformly distributed. From Table 1, we observe that for the original Lena and Jetplane images the χ^2 values are very high—242,173 and 715,669, respectively.However, the χ^2 values obtained by the encrypted and the marked-encrypted versions of Lena and Jetplane images are less than the critical value 293, which implies that the distributions of these images are uniform, thus indicating that the encryption scheme is of good quality.

(d) ***The correlation test***

Real images typically exhibit a high correlation between adjacent pixels in the horizontal, vertical, or diagonal directions. However, an effective encryption scheme should produce a ciphertext with a low correlation between adjacent pixel values [41].

Table 1 presents the correlation coefficients of the original images, Lena and Jetplane, as well as their respective encrypted and marked-encrypted versions obtained using our proposed RDHEI method. The correlation coefficients of the original images are close to 1, indicating a high correlation among adjacent pixels. Specifically, the correlation coefficients for Lena and Jetplane are 0.9710 and 0.9357, respectively. However, the encrypted and marked-encrypted images exhibit significantly low correlation coefficients near 0, indicating effective suppression of the correlation between adjacent pixel values. Similar results were obtained for all the test images.

Furthermore, the correlation distributions [42] in the horizontal direction of the original images, Lena and Jetplane, along with their respective encrypted and marked-encrypted images, are illustrated in Figure 18. In this figure, the x-axis corresponds to the pixel intensity at coordinates (x, y) within the image, while the y-axis represents the pixel intensity of the horizontally adjacent pixel at coordinates $(x + 1, y)$. Each point on the plot reflects the relationship between the intensity of a pixel and the intensity of its immediate neighbor in the horizontal direction. A higher density of points around a particular region on the plot means a stronger correlation between the pixel at (x, y) and its adjacent counterpart at $(x + 1, y)$. As depicted in Figure 18, the original images exhibit a high correlation between horizontal pixels, whereas the encrypted and marked-encrypted images show no correlation between adjacent pixels. It is worth noting that the correlation distribution is used to assess the degree of similarity between pixel values at nearby locations in the image.

Both the correlation coefficients and the figures show that the proposed RDHEI method effectively decorrelates the neighboring pixels of the plain images.

(e) ***The information entropy test***

The information entropy is initially defined by Shannon [43]. When an image is encrypted, its entropy should ideally be equal to 8 bpp. If the entropy of the encrypted image is less than 8 bits/pixel, there exists a certain degree of predictability and the encryption algorithm will be threatened in its security. The entropies of the encrypted and marked-encrypted images for Lena and Jetplane are shown in Table 1. While the encrypted and marked-encrypted Lena images have entropies of 7.9994 bpp and 7.9995 bpp, respectively, the encrypted and marked-encrypted Jetplane images have entropies of 7.9993 bpp and 7.9994 bpp, respectively. These values are very close to the theoretical value of approximately 8 bpp, indicating that the proposed encryption scheme is secure against entropy attacks.

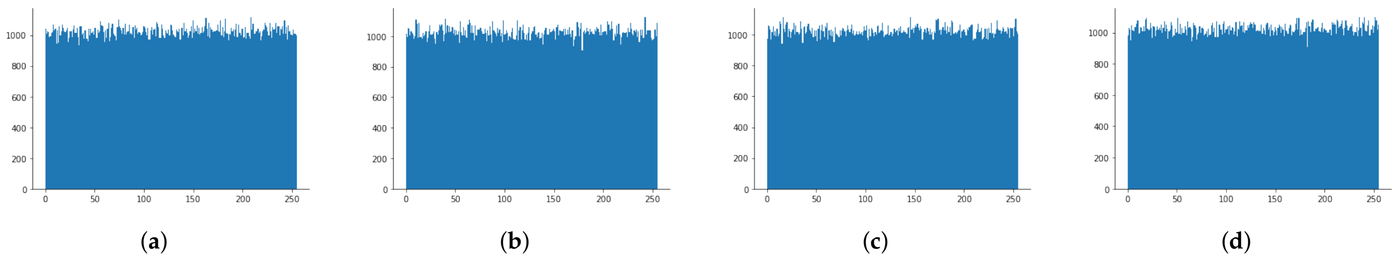


Figure 17. The histogram analysis of the encrypted and the marked-encrypted images for Lena and Jetplane. (**a**) Histogram of the Lena encrypted image. (**b**) Histogram of the marked and encrypted Lena image. (**c**) Histogram of the encrypted Jetplane image. (**d**) Histogram of the marked and encrypted Jetplane image.

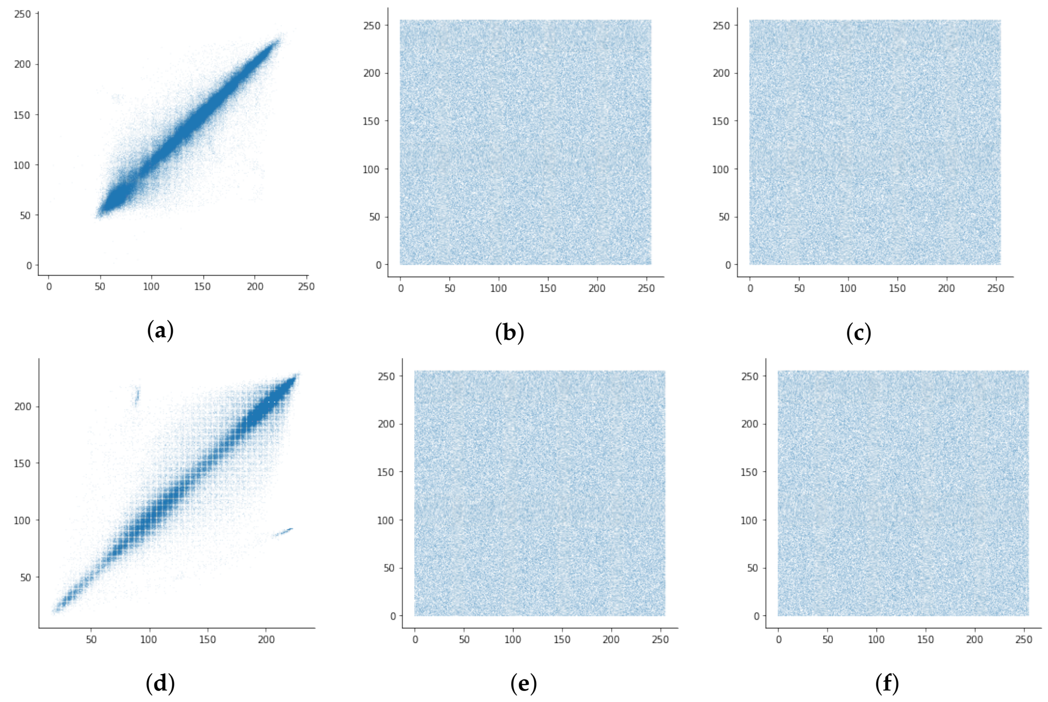


Figure 18. The correlation distributions in the horizontal direction of: (**a**) the original Lena image, (**b**) the encrypted Lena image, (**c**) the marked-encrypted Lena image, (**d**) the original Jetplane image, (**e**) the encrypted Jetplane image, (**f**) and the marked-encrypted Jetplane image. The x-axis represents the pixel intensity value at coordinates (x, y). On the other hand, the y-axis represents the pixel intensity of the horizontally adjacent pixel located at coordinates (x + 1, y).

Table 1. Security evaluation of the proposed method.

Image	**Horizontal Correlation**	**Entropy (in bpp)**	**χ^2 Test**
Original Lena image	0.9710	7.2184	242,173
Encrypted Lena image	−0.0021	7.9994	222
Marked and encrypted Lena image	−0.0003	7.9995	203
Original Jetplane image	0.9357	6.7059	715,669
Encrypted Jetplane image	−0.0001	7.9993	253
Marked and encrypted Jetplane image	−0.0005	7.9994	211

3.2. Performance Analysis

Table 2 shows the ER in bpp using different predictors on the five test images. From Table 2, we can see that for the five test images, the best ERs are obtained when using the GAP predictor. The average ER for these five test images is more than 3 bits per pixel.

Table 2. Embedding rates obtained when using different predictors on the five test images.

Image	**MED**	**MED-IMP1**	**MED-IMP2**	**GAP**
Lena	3.493	3.419	3.476	3.606
Baboon	1.832	1.667	1.824	1.903
Jetplane	3.620	3.387	3.599	3.642
Man	3.002	2.817	2.989	3.058
Tiffany	2.825	2.730	2.813	2.920
Average	2.954	2.804	2.940	3.026

The total embedding capacity, the auxiliary information, and the net payload for the five test images are calculated, and the results using the GAP predictor are shown in Table 3.

Table 3. The embedding capacity and the auxiliary information of five test images by using the GAP predictor.

Image	Total EC (bits)	Auxiliary Information (bits)	Net Payload (bpp)
Lena	1,075,378	129,984	3.606
Baboon	668,056	169,152	1.903
Jetplane	1,084,546	129,808	3.642
Man	976,450	174,776	3.058
Tiffany	876,568	110,872	2.920

Figure 19 shows the obtained ERs for the five test images when using different methods. It can be seen that our method outperforms the other five methods for the first four images with the highest ER.

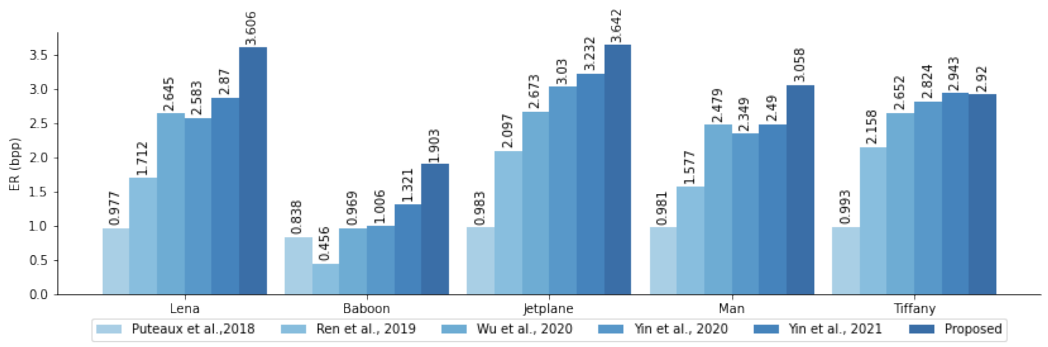


Figure 19. The comparison of ER (in bpp) for the five test images includes Puteaux et al. [12], Ren et al. [16], Wu et al. [19], Yin el al. [8], Yin et al. [10], and Proposed method.

Puteaux et al. [12] proposed to embed additional data into one-MSB only, resulting in a very low ER. For instance, the ERs of their method for the five test images were between 0.838 bpp and 0.993 bpp. Wu et al. [19] proposed an improvement of the Yi et al.'s [18] work by dividing the image pixels into two categories according to a parameter binary tree labeling scheme. By dividing the pixels into embeddable and non-embeddable sets, the overall embedding capacity may be affected. The ERs of the five test images are significantly higher than those obtained by the previous methods and range from 0.969 bpp to 2.6726 bpp. In their work, Yin et al. [8] compressed the auxiliary information by using Huffman coding and embedding multi-bits adaptively by multi-MSB substitution. The scheme requires a large amount of auxiliary information leading to a potential impact on the overall embedding capacity. As an example, the auxiliary information required for the Lena image in Yin et al.'s [8] approach is 793,304 bits, which is over six times greater than that needed in our method, as presented in Table 3. In [10], an RDHEI method is proposed based on pixel prediction scheme and multi-MSB planes reordering. The bit planes were divided into several non-overlapping blocks and the best ERs were found in the blocks of size 4 $\times$ 4. The blocks are classified as uniform or non-uniform, with a binary label map generated for each bit plane to indicate their positions. The data-embedding method used depends on whether the block is uniform or non-uniform. In uniform blocks, one bit is left unchanged to represent the block's value and aid in image recovery. However, this approach fails to exploit the large uniform blocks in the MSB bit planes, resulting in an augmented amount of the auxiliary information.

The obtained ERs by our method for the five images are better than those achieved by previous methods. For the Lena image the ER is 3.606 bpp, which is approximately 26% better than the best result obtained with the [10] method. By mapping the prediction errors, using the quadtree decomposition in order to find as many large homogenous blocks as possible and by reordering the embeddable bit planes, we achieved a significantly higher ERs than those obtained with state-of-the-art methods.

Finally, Table 4 show the ERs of our proposed method and compares them with state-of-the-art methods on the two image datasets, respectively. The average ERs for the image datasets BOSSbase and BOWS-2 are 3.813 bpp and 3.696 bpp, respectively. In addition, it can be seen from Table 4 that our proposed method has a higher ER than the other methods. In Table 4, "Gain" represents the percentage improvement in ER achieved by our approach when compared to other methods. In comparison to the most effective method of Yin et al.'s [10], our results exhibit an improvement of up to 9% and 8.9% for BOSSbase and BOWS-2 datasets, respectively. Comparatively larger improvements are observed when our results are compared to the other methods.

Table 4. Comparison of the average ERs of two image datasets between our method and five state-of-the-art methods.

DataSet	Method										
	Our	[12]		[19]		[8]		[10]		[20]	
	ER	ER	Gain	ER	Gain	ER	Gain	ER	Gain	ER	Gain
BOSSbase	3.813	0.966	294.7	2.561	48.8	3.361	13.4	3.498	9	2.556	49.1
BOWS-2	3.696	0.968	281.8	2.519	46.7	3.246	13.8	3.393	8.9	2.530	46

3.3. Time Complexity Analysis

To assess the time complexity of our RDHEI method, we conducted a detailed analysis of its key operations, which are presented in Algorithms 3 and 4. We considered factors such as the size of the input image $n \times m$ and the number of bit planes processed. Algorithm 3 involves initiating the $mappedPredictionError()$ function, followed by sequential calls to $quadtreeDecomposition()$, $calculateAuxInfo()$, and $bitPlaneReordering()$ for the embedded bit planes (up to 8 in the worst case), and finally calling the $encrypt()$ function. The complexity for most functions is $\mathcal{O}(nm)$, except for $quadtreeDecomposition()$, which has a time complexity of $\mathcal{O}(nm\log(nm))$. Consequently, the overall time complexity of the $processImage$ algorithm is deduced to be $\mathcal{O}(nm\log(nm))$ as the quadtree decomposition process imposes the most significant computational load compared to the other processes. In the data embedding Algorithm 4, three functions—encrypt(), readPartialInf(), and embedBits()—are employed, with each having a time complexity upper-bounded by $\mathcal{O}(nm)$. Considering both algorithms, it can be inferred that the overall time complexity of our RDHEI method is $\mathcal{O}(nm\log(nm))$.

Upon comparing the complexity analysis of the various studied algorithms, it is noteworthy that a significant portion of these algorithms exhibit a time complexity that approximates $\mathcal{O}(nm)$. However, our proposed approach, with a time complexity of $\mathcal{O}(nm\log(nm))$, while slightly higher, remains well within the bounds of efficiency. It is important to emphasize that this $\mathcal{O}(nm\log(nm))$ complexity offers a favorable trade-off between computational cost and performance gains when compared to the $\mathcal{O}(nm)$ alternatives. The computational efficiency is more than acceptable, making our approach a compelling choice.

4. Conclusions

In this paper, we present a novel approach for RDHEI using separable RRBE method with high ERs and error-free reversibility. While most existing RDHEI methods utilize the most or least significant bit planes to hide data, we propose exploiting the redundancy between adjacent pixels to calculate a prediction error image that is then processed to accommodate data in its embeddable bit planes. This image is subsequently converted into a new error image with positive pixel values, which undergoes quadtree decomposition for each bit plane to locate as many homogeneous blocks as possible with minimal auxiliary information. These blocks of varying sizes are then reordered to create space for embedding data. The proposed method enables error-free extraction of the secret message and lossless reconstruction of the original image. Experimental results demonstrate that our method is highly efficient, secure, and outperforms previous state-of-the-art methods.

While our research has made significant advancements in the field of RDHEI, there are several avenues for future work and potential improvements. One avenue of future research entails further exploration of techniques aimed at maximizing embedding capacity while maintaining error-free reversibility. Additionally, we suggest an investigation of RDHEI's potential applications in emerging technologies such as blockchain, which could open doors to novel use cases and challenges.

Author Contributions: Conceptualization, A.M.; Software, M.A.; Validation, M.A.; Writing–original draft, M.A.; Writing—review and editing, A.M.; Supervision, A.M. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deanship of Scientific Research at King Khalid University for funding this work through large group Research Project under grant number RGP2/171/44.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in this paper are publicly available and their links are http://agents.fel.cvut.cz/boss/ [39] and https://data.mendeley.com/datasets/kb3ngxfmjw/1 (accessed on 21 January 2021).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript, Note: This table provides a brief overview, and readers are encouraged to refer to the corresponding sections in the paper for more detailed explanations.

Symbol/Variable/Function	**Meaning**
$x(u,v)$	The original image
$\hat{x}(u,v)$	The predicted image
$e(u,v)$	The prediction error image
$x_e(u,v)$	The encrypted image
$r(u,v)$	A pseudo-random matrix generated with the PWLCM
K_e	The image encryption key
K_d	The data hiding key
labelMap(u,v)	A binary label map. It is set to 1 for the overflow pixels and 0 otherwise
listOfNeighbors(u,v)	A function that returns the neighboring pixel values, as depicted in Figure 3
prediction(L)	A function that takes a list of neighboring pixels L as input, and returns the predicted value based on the used predictor (MED, GAP,$\cdots$)
quadtreeDecomposition(bitPlane)	A function that accepts an image bit plane as input and returns a binary string, denoted as *binaryStr*, which represents the quadtree decomposition of the input bit plane. Additionally, it provides a list of homogeneous blocks extracted from the same bit plane.
readPartialInf(x_e)	A function that extracts the partial auxiliary information from the encrypted image
$embedBits()$	A function that embeds the encrypted hidden data into the encrypted image

References

1. Horng, J.H.; Chang, C.C.; Li, G.L.; Lee, W.K.; Hwang, S.O. Blockchain-Based Reversible Data Hiding for Securing Medical Images. *J. Healthc. Eng.* **2021**, *2021*, 9943402 . [CrossRef] [PubMed]
2. Puech, W.; Chaumont, M.; Strauss, O. A Reversible Data Hiding Method for Encrypted Images. In Proceedings of the SPIE—The International Society for Optical Engineering, San Jose, CA, USA, 1–3 July 2008; Volume 6819.
3. Zhang, X. Reversible Data Hiding in Encrypted Image. *IEEE Signal Process. Lett.* **2011**, *18*, 255–258. [CrossRef]
4. Zhang, X. Separable Reversible Data Hiding in Encrypted Image. *IEEE Trans. Inf. Forensics Secur.* **2012**, *7*, 826–832. [CrossRef]
5. Ma, K.; Zhang, W.; Zhao, X.; Yu, N.; Li, F. Reversible Data Hiding in Encrypted Images by Reserving Room Before Encryption. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 553–562. [CrossRef]
6. Hong, W.; Chen, T.S.; Chang, Y.P.; Shiu, C.W. A high capacity reversible data hiding scheme using orthogonal projection and prediction error modification. *Signal Process.* **2010**, *90*, 2911–2922. [CrossRef]
7. Yi, S.; Zhou, Y. Binary-block embedding for reversible data hiding in encrypted images. *Signal Process.* **2017**, *133*, 40–51. [CrossRef]
8. Yin, Z.; Xiang, Y.; Zhang, X. Reversible Data Hiding in Encrypted Images Based on Multi-MSB Prediction and Huffman Coding. *IEEE Trans. Multimed.* **2020**, *22*, 874–884. [CrossRef]
9. Wang, D.; Zhang, X.; Yu, C.; Tang, Z. Reversible Data Hiding in Encrypted Image Based on Multi-MSB Embedding Strategy. *Appl. Sci.* **2020**, *10*, 2058. [CrossRef]
10. Yin, Z.; She, X.; Tang, J.; Luo, B. Reversible data hiding in encrypted images based on pixel prediction and multi-MSB planes rearrangement. *Signal Process.* **2021**, *187*, 108146. [CrossRef]
11. Feng, Q.; Leng, L.; Chang, C.C.; Horng, J.H.; Wu, M. Reversible Data Hiding in Encrypted Images with Extended Parametric Binary Tree Labeling. *Appl. Sci.* **2023**, *13*, 2458. [CrossRef]
12. Puteaux, P.; Puech, W. An Efficient MSB Prediction-Based Method for High-Capacity Reversible Data Hiding in Encrypted Images. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1670–1681. [CrossRef]
13. Qian, Z.; Zhou, H.; Zhang, X.; Zhang, W. Separable Reversible Data Hiding in Encrypted JPEG Bitstreams. *IEEE Trans. Dependable Secur. Comput.* **2018**, *15*, 1055–1067. [CrossRef]
14. Zhou, J.; Sun, W.; Dong, L.; Liu, X.; Au, O.C.; Tang, Y.Y. Secure Reversible Image Data Hiding Over Encrypted Domain via Key Modulation. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 441–452. [CrossRef]
15. Wu, X.; Sun, W. High-capacity reversible data hiding in encrypted images by prediction error. *Signal Process.* **2014**, *104*, 387–400. [CrossRef]
16. Ren, H.; Lu, W.; Chen, B. Reversible data hiding in encrypted binary images by pixel prediction. *Signal Process.* **2019**, *165*, 268–277. [CrossRef]
17. Chen, K.; Chang, C.C. 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.* **2019**, *58*, 334–344. [CrossRef]
18. Yi, S.; Zhou, Y. Separable and Reversible Data Hiding in Encrypted Images Using Parametric Binary Tree Labeling. *IEEE Trans. Multim.* **2019**, *21*, 51–64. [CrossRef]
19. Wu, Y.; Xiang, Y.; Guo, Y.; Tang, J.; Yin, Z. An Improved Reversible Data Hiding in Encrypted Images Using Parametric Binary Tree Labeling. *IEEE Trans. Multimed.* **2020**, *22*, 1929–1938. [CrossRef]
20. Li, L.; Chang, C.C.; Lin, C.C. Reversible Data Hiding in Encrypted Image Based on (7, 4) Hamming Code and UnitSmooth Detection. *Entropy* **2021**, *23*, 790. [CrossRef]
21. Puteaux, P.; Ong, S.; Wong, K.; Puech, W. A survey of reversible data hiding in encrypted images–The first 12 years. *J. Vis. Commun. Image Represent.* **2021**, *77*, 103085. [CrossRef]
22. Weinberger, M.; Seroussi, G.; Sapiro, G. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. *IEEE Trans. Image Process.* **2000**, *9*, 1309–1324. [CrossRef] [PubMed]
23. Wu, X.; Memon, N. CALIC—A context based adaptive lossless image codec. In Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, Atlanta, GA, USA, 9 May 1996; Volume 4, pp. 1890–1893.
24. Masmoudi, A.; Puech, W.; Masmoudi, A. An Improved Lossless Image Compression Based Arithmetic Coding Using Mixture of Non-Parametric Distributions. *Multimedia Tools Appl.* **2015**, *74*, 10605–10619. [CrossRef]
25. Martucci, S. Reversible compression of HDTV images using median adaptive prediction and arithmetic coding. In Proceedings of the IEEE International Symposium on Circuits and Systems, New Orleans, LA, USA, 1–3 May 1990; Volume 2, pp. 1310–1313.
26. Wu, X.; Memon, N. Context-based, adaptive, lossless image coding. *IEEE Trans. Commun.* **1997**, *45*, 437–444. [CrossRef]
27. Jiang, J. Variations of JPEG-LS and Its Applications. Ph.D. Thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2000.
28. Edirisinghe, E.A.; Bedi, S.; Grecos, C. Improvements to JPEG-LS via diagonal edge-based prediction. In Proceedings of the Visual Communications and Image Processing, 2002, San Jose, CA, USA, 19 January 2002; Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series; Volume 4671, pp. 604–613.
29. Masmoudi, A.; Masmoudi, A.; Puech, W. A New Semiparametric Finite Mixture Model-Based Adaptive Arithmetic Coding for Lossless Image Compression. *Circuits Syst. Signal Process.* **2016**, *35*, 1163–1186. [CrossRef]
30. Masmoudi, A.; Chaoui, S.; Masmoudi, A. A finite mixture model of geometric distributions for lossless image compression. *Signal Image Video Process.* **2016**, *10*, 671–678. [CrossRef]
31. Masmoudi, A.; Masmoudi, A. A new arithmetic coding model for a block-based lossless image compression based on exploiting inter-block correlation. *Signal Image Video Process.* **2015**, *9*, 1021–1027. [CrossRef]

32. Li, S.; Chen, G.; Mou, X. On the dynamical degradation of digital piecewise linear chaotic maps. *Int. J. Bifurc. Chaos* **2005**, *15*, 3119–3151. [CrossRef]
33. Lu, X.; Zhi, L.; Jian, L.; Wei, H. A novel bit-level image encryption algorithm based on chaotic maps. *Opt. Lasers Eng.* **2016**, *78*, 17–25.
34. Wang, X.-Y.; Xu, D.-H. A novel image encryption scheme based on Brownian motion and PWLCM chaotic system. *Nonlinear Dyn.* **2014**, *75*, 345–353. [CrossRef]
35. Wang, X.; Jin, C. Image encryption using Game of Life permutation and PWLCM chaotic system. *Opt. Commun.* **2012**, *285*, 412–417. [CrossRef]
36. Khlif, N.; Masmoudi, A.; Kammoun, F.; Masmoudi, N. Secure chaotic dual encryption scheme for H.264/AVC video conferencing protection. *IET Image Process.* **2018**, *12*, 42–52. [CrossRef]
37. Masmoudi, A.; Puech, W. Lossless chaos-based crypto-compression scheme for image protection. *IET Image Process.* **2014**, *8*, 671–686. [CrossRef]
38. Li, Z.; Peng, C.; Tan, W.; Li, L. A Novel Chaos-Based Image Encryption Scheme by Using Randomly DNA Encode and Plaintext Related Permutation. *Appl. Sci.* **2020**, *10*, 7469. [CrossRef]
39. Bas, P.; Filler, T.; Pevný, T. "Break Our Steganographic System": The Ins and Outs of Organizing BOSS. In *Information Hiding*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 59–70.
40. Alvarez, G.; Li, S. Some Basic Cryptographic Requirements for Chaos-based Cryptosystems. *Int. J. Bifurc. Chaos* **2006**, *16*, 2129–2151. [CrossRef]
41. Wu, X.G.; Hu, H.P.; Zhang, B.L. Analyzing and Improving a Chaotic Encryption Method. *Chaos Solitons Fractals* **2004**, *22*, 367–373. [CrossRef]
42. Askar, S.S.; Karawia, A.A.; Al-Khedhairi, A.; Al-Ammar, F.S. An Algorithm of Image Encryption Using Logistic and Two-Dimensional Chaotic Economic Maps. *Entropy* **2019**, *21*, 44. [CrossRef]
43. Shannon, C. Communication Theory of Secrecy Systems. *Bell Syst. Tech. J.* **1949**, *28*, 656–715. [CrossRef]