INFORMATION HIDING IN ENCRYPTED IMAGES FOR PRIVACY-PRESERVATION AND COVERT COMMUNICATION

A Thesis
Submitted in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY
by

ANKUR
(2K20/PHDCO/507)

Under the supervision of

Dr. Rajeev Kumar (Supervisor)

Delhi Technological University Delhi

and

Prof. Ajay K Sharma (Joint-Supervisor)

National Institute of Technology Delhi

Department of Computer Science and Engineering
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi 110042

JUNE, 2024

CERTIFICATE

Certified that **Ankur (2K20/PHDCO/507)** has carried out his research work presented in this thesis titled **Information Hiding in Encrypted Images for Privacy-Preservation and Covert Communication** under our supervision. The research work was undertaken for the award of the degree of **Doctor of Philosophy** from Delhi Technological University, New Delhi, India.

The thesis embodies the results of original research and studies conducted by the student himself. The contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else from this University or any other University/Institution.

Dr. Rajeev Kumar
Assistant Professor
Computer Science and Engineering
Delhi Technological University
Delhi, India
Place: Delhi
Date: June 27, 2024

Prof. Ajay K Sharma
Director
National Institute of Technology
Delhi
Delhi, India
Place: Delhi
Date: June 27, 2024

CANDIDATE DECLARATION

I, Ankur (2K20/PHDCO/507), hereby declare that the thesis titled **Information Hiding in Encrypted Images for Privacy-Preservation and Covert Communication**, submitted to **Delhi Technological University, Delhi**, in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy in the Department of Computer Science and Engineering, is my original work.

This research work has been completed under the supervision of **Dr. Rajeev Kumar** (Supervisor), Department of Computer Science and Engineering, Delhi Technological University, Delhi, India, and **Prof. Ajay K Sharma** (Joint-Supervisor), Director, National Institute of Technology, Delhi, India.

The explanations presented in this thesis are based on my understanding and comprehension of the original texts. I have not submitted this work to any other institution for the award of any degree, diploma, associate-ship, fellowship, or other title or honor.

Ankur
2K20/PHDCO/507
Computer Science and Engineering
Delhi Technological University
(Formerly Delhi College of Engineering)
Delhi-110042, India
Place: Delhi
Date : June 27, 2024

Brief Description of Ph.D. work

The adoption of digital data have accentuated the importance of secure communication, pushing the advancements in data hiding techniques. Reversible Data Hiding in Encrypted Images (RDHEI) stands out as a important domain, addressing the dual requirements of robust security of cover image and data integrity. This thesis embarks on an exploitative journey through the RDHEI domain, identifying key challenges and proposing innovative solutions that mark significant strides in this domain.

The thesis first proposes a groundbreaking method that introduces a novel link chain driven RDHEI method. The proposed method leverages neighborhood similarity to create variable-sized link chains for high-capacity data embedding. This innovative approach introduces a novel pixel collection strategy, enhances embedding space creation, ensures reversibility within the encrypted domain, and provides flexibility in balancing error correction and image distortion.

In our second work, a high-capacity RDHEI method based on difference image transfiguration is introduced. The proposed method utilizes advanced image processing techniques such as the Median Edge Detector and difference predictor to generate images with lower magnitude prediction values, creating space for data embedding. Additionally, this method introduces an innovative cleaning process, novel difference location techniques, and Huffman encoding for weight matrix size reduction, contributing to enhanced security, direct data embedding, improved efficiency, and the advantage of bit-plane-wise image pre-processing.

The third work introduces a novel multi-level blocking and quad-tree encoding based RDHEI method, which addresses the limitations of variable block size approaches. This technique employs a quad-tree-based bit-plane encoding technique and an inter bit-plane redundancy mitigation strategy, effectively compressing the final bit stream by removing redundancies across bit-planes. The multi-level blocking RDHEI method demonstrates superior embedding capacity while maintaining flawless original image reconstruction, introducing a novel high-capacity RDHEI technique and effectively utilizing inter and intra bit-plane redundancies.

The fourth work proposes a novel RDHEI method that employs a block-level division approach, innovating with 2×2 non-overlapping blocks to enhance spatial correlation exploitation for improved embedding rates and guaranteed reversibility. This method introduces a novel block-shifting mechanism, enabling quicker and more secure RDHEI without the need for a block-label map, thereby enhancing both security and operational efficiency. It also introduces an index-based approach for seamless integration of hidden data within image content, avoiding additional overhead or keys.

The fifth work introduces an Adaptive Two-Stage RDHEI method which utilizes prediction error expansion and pixel value ordering to overcome the limitations of overflow, underflow, and limited embedding capacity. This method combines image encryption and RDHEI methods, enhancing security and adaptability while maintaining visual security and correlation through block rotation. It demonstrates increased embedding capacity compared to existing methods and provides flexible and confidential recovery of data and images based on key availability.

ACKNOWLEDGEMENT

First and above all, I thank God Almighty for giving me the strength, wisdom, and persistence needed throughout this academic journey. The faith in a higher power has been my anchor, offering both comfort and inspiration. This belief has not only sustained me through the challenging moments but also heightened the joy of my successes, reminding me that every step forward is part of a larger journey.

Most importantly, my heartfelt appreciation goes to my supervisor, **Dr. Rajeev Kumar**. Your exceptional mentorship, profound expertise, and steadfast belief in my potential have been the cornerstone of my research journey. Your guidance has been more than just academic advice; it has been a source of constant motivation, encouraging me to explore uncharted territories and embrace challenges with resilience. Similarly, my gratitude extends to my joint-supervisor, **Prof. Ajay K Sharma**. Your detailed feedback and constructive critiques have been invaluable, pushing me to refine my work and strive for excellence. Your perspectives have not only enhanced the quality of my research but have also broadened my own understanding and approach to problem-solving.

My thanks also go to the members of the Student Comprehensive Report Committee and Departmental Research Committee for their valuable advice and thought-provoking suggestions. Your contributions have been crucial in making my research stronger and more focused.

I am thankful for my colleagues and friends at the Department of Computer Science, Delhi Technological University. Your friendship, insightful discussions, and readiness to share your knowledge have been a great help throughout this journey. Your support has been a pillar for me during both good and tough times.

Most importantly, my profound gratitude is reserved for my family - my wife, daughter, and all my loved ones. Your unconditional love, unwavering support, and endless patience have been the foundation of my achievements. The sacrifices you've made and the faith you've shown in me have fueled my ambition and sustained my spirit during the most challenging periods. Your encouragement has been the light guiding me towards this accomplishment.

I am eternally grateful to the divine force that has guided me with strength and wisdom on

this academic path. In moments of doubt and success, my faith in a greater purpose has offered comfort and motivation, helping me to tackle the complex challenges of this research.

This thesis stands as a testament to the collective efforts and support of these remarkable individuals. Their contributions have shaped not only my academic growth but also my personal development, making this journey truly enriching and unforgettable.

Place: Delhi Ankur

Date: June 27, 2024

LIST OF ABBREVIATIONS

AES Advanced Encryption Standard's

AI Auxiliary Information

AM2PHC Adaptive Multi-MSB Prediction and Huffman Coding

AMBTC Absolute Moment Block Truncation Coding

APE Absolute Prediction Error

ATS Adaptive Two-Stage

BBC Binary-Block Coding

BLM Block Label Map

BMPR Block-based MSB Plane Rearrangement

CTR Counter Mode

DH Data Hiding

DIT Difference Image Transfiguration

EC Embedding Capacity

ECB Electronic Code Book

EI Extracted Information

ELM Encoded Label Map

EMSP Encrypted MSBs of Starting Pixels

ER Embedding Rate

ERLE Extended Run-Length Encoding

HCR Huffman Coding Rules

HBVLC Hierarchical Block Variable Length Coding

HC-RDHEI High Capacity RDHEI

HMBQ Huffman-MBQ

HQTC Hierarchical Quad-Tree Coding

HS Histogram Shifting

IBRM Inter Bit-plane Redundancy Mitigation

IAMBTC Interpolative Absolute Moment Block Truncation Coding

IoT Internet of Thing

LAI Length of Auxiliary Information(AI)

LCD Link Chain Driven

LDPC Low-Density Parity Check

LDT Location Difference Technique

LELM Length of ELM

LM Label Map

LSB Least Significant Bit

LSBs Least Significant Bits

MBQ Multi-Level Blocking

MED Median Edge Detector

MSE Mean Squared Error

MSB Most Significant Bit

MSBs Most Significant Bits

NPCR Number of Pixel Changing Rate

PAI Partial Auxiliary Information

PBTL Parametric Binary Tree Labeling

PE Prediction Error

PEE Prediction Error Expansion

PEs Prediction Errors

PI Process Information

PRNG Pseudo Random Number Generator

PSI Primary Supporting Information

PSNR Peak Signal-to-Noise Ratio

PVO Pixel Value Ordering

RDH Reversible Data Hiding

RDHEI Reversible Data Hiding in Encrypted Images

RRBE Reserving Room Before Encryption

SHMBQ Separate Huffman-MBQ

SI Supporting Information

SOTA State-Of-The-Art

SSI Second Supporting Information

SSIM Structural Similarity Index

TLS Total Link Strength

UACI Unified Averaged Changed Intensity

VRAE Vacating Room After Encryption

WoS Web of Science

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In the modern digital age, the exponential growth of internet-connected devices, e-commerce platforms, and social media networks has led to an unprecedented generation of digital data [1]. This growth has been further amplified by the adoption and advancements in the field of Artificial Intelligence [2], and 5G communication technology [3, 4]. With 15.14 billion connected Internet of Thing (IoT) devices [5] and 61.54% of marketers incorporating Artificial Intelligence in content creation [6], the volume of data is increasing exponentially. This is also evident from the content sharing statistics of social media platforms like X (earlier Twitter) and Instagram as these platforms are witnessing the sharing of 66,000 photos per minute [7–9]. However, this rapid expansion of digital data presents various challenges, especially in terms of securing and preserving the confidentiality of private information [10, 11]. This growth requires an urgency in prioritizing robust protective measures for data privacy [12, 13].

Addressing the aforementioned challenges, Data Hiding (DH) has emerged as one of the best solutions for secure and covert communication [14–16]. In the field of data hiding, a fundamental principle is the embedding of secret/private information, which can include various data formats such as text, images, audio, video, or any other bit-stream representations. This secret data is embedded within a cover medium, which could also range from text to multimedia formats, effectively masking the presence of the hidden bit-stream. The primary objectives in this process are to ensure robustness, security, and Embedding Capacity (EC). It is important to note that both the EC and the embedding technique employed are strongly dependent on the structural characteristics of the cover medium, which is why the selection of an appropriate cover medium is considered a critical step in DH techniques.

Notably, perceptual cover media are frequently chosen for embedding purposes because they can tolerate minor modifications without significantly affecting the overall perception of the media. Examples of perceptual cover media include images, audio, and video, which can undergo slight alterations without noticeably compromising their quality or meaning. In contrast, non-perceptual media, such as text and executable code, demand lossless processing. This is

due to their inherent sensitivity to changes; even the slightest modification can alter the intended meaning or functionality, making them unsuitable for embedding purposes where modifications are inevitable. As a result, multimedia elements like images, videos, and audio tracks are frequently utilized as the primary covers for data embedding, owing to their characteristics that are favorable for data hiding. Among the different types of perceptual cover, images have gained more attention of researchers due to their widespread use and availability in digital communication. Moreover, the inherent characteristics of natural images, including their high redundancy, imperceptibility to minor modifications, and ease of transmission, make them ideal cover signals for DH techniques [17]. Therefore, the traditional DH methods can embed large amount of secret data within an image.

There are multiple ways to categorize DH methods, but one straightforward approach is based on whether the original image can be recovered after data has been hidden within it. Methods that typically result in irreversible changes to the original image, thereby making it difficult to restore the image to its original state after data extraction, are known as irreversible data hiding methods. Conversely, Reversible Data Hiding (RDH) methods have revolutionized this field by allowing the full recovery of the original image upon the retrieval of the embedded secret data [18, 19]. RDH methods encompassing both extraction of data and reconstruction of image, like a two-part framework. This procedure initiates with the pre-processing phase of the cover image, wherein a location map is generated. The location map is used to record the boundary pixels and carter the issues such as overflow and underflow during the embedding process. Subsequently, both the location map and important data are seamlessly embedded within the image, resulting in a marked image. Upon receiving the marked image, the recipient can analyze it to retrieve the location map and auxiliary data. Through extraction and reconstruction algorithms, the hidden data can be losslessly recovered, and image can be reconstructed, effectively reversing the embedding process. This remarkable feature sets RDH apart from traditional data hiding methods, offering a comprehensive solution for secure and lossless data transmission.

1.1 Emergence of Reversible Data Hiding in Encrypted Images

In the era of digital data, where data transmission predominantly occurs in image formats, the traditional RDH methods face a significant hurdle in ensuring the privacy of the cover image, which often remains vulnerable in its original form. In response to this limitation, the concept of RDHEI is developed [20–22]. The RDHEI enhances security in terms of confidentiality and privacy by allowing embedding in an encrypted image. Fundamentally, the RDHEI is designed to protect both the original image and the embedded secret data concurrently. Specifically, one of the key advantages of the RDHEI is its ability to maintain the confidentiality of the cover

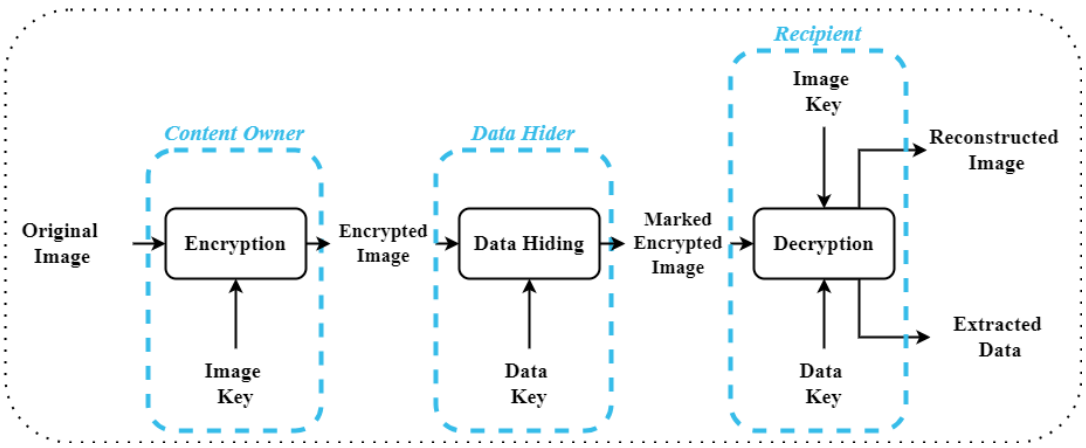


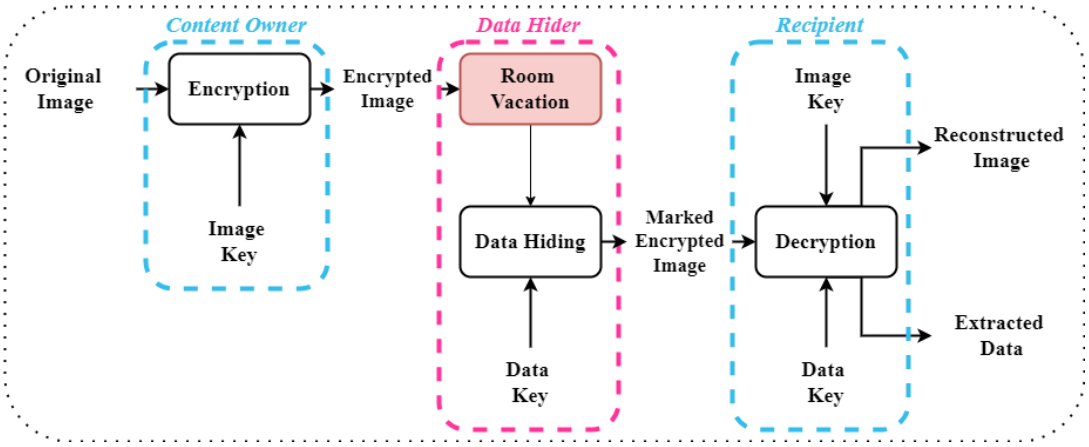
Fig. 1.1: Block diagram of RDHEI

image throughout the embedding and transmission process. By encrypting the original image before embedding the data, RDHEI ensures that the content remains protected, even if the marked encrypted image is accessed by unauthorized parties. This added layer of security is particularly crucial in sensitive domains such as cloud computing scenarios for privacy protection, as well as its predominant use in medical or military, or intelligence operatives [23, 24]. However, RDHEI faces a fundamental challenge in settling the balance between EC and image distortion. As larger amounts of data are embedded, the distortion in the original cover image inevitably increases, highlighting the inherent trade-off between these two objectives.

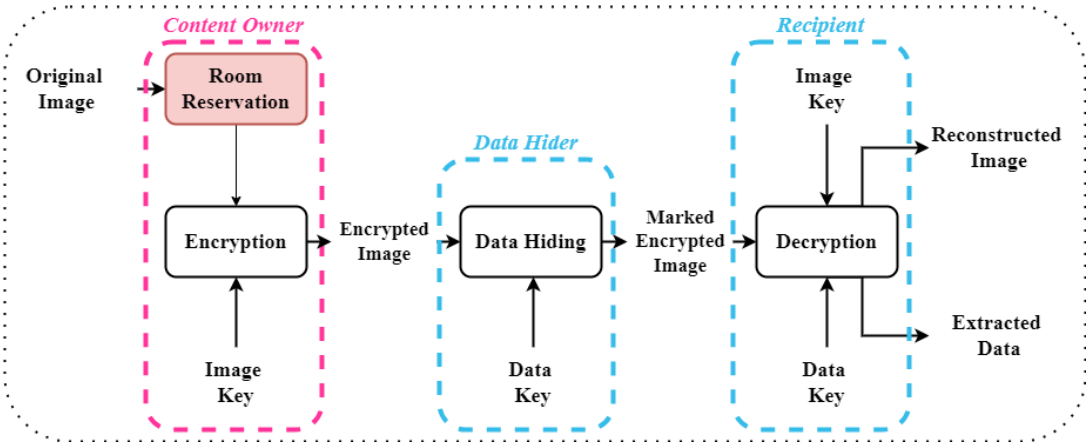
1.2 RDHEI Architecture

The RDHEI architecture typically comprises three primary stages as depicted in Fig. 1.1, each involving distinct entities with specific roles and responsibilities:

- **Content Owner:** The content owner holds the original image and the encryption key. They are responsible for encrypting the original image using the encryption key, ensuring its confidentiality and preventing unauthorized access to the image content.
- **Data Hider:** The data hider possesses the confidential data and the key for data encryption. After encrypting the data using encryption key, the data hider embed the confidential data within the encrypted image, effectively using the image as a covert signal for transmitting confidential data.
- **Recipient:** The recipient may receive either an image decryption key, a data decryption key, or both, depending on the specific requirements and security protocols in place. Based on the availability of keys, the recipient can retrieve the original image, extract the hidden data, or perform both operations, respectively.



(a) Block diagram of VRAE methods



(b) Block diagram of RRBE methods

Fig. 1.2: Block diagram of different RDHEI categories

1.3 Classification of RDHEI Methods

Over the past few years, various RDHEI methods have been developed, employing various techniques such as data compression, spatial correlation, histogram distribution, and advanced encryption algorithms to create room for data embedding within the image. These methods can be classified from the perspectives of both the encoder and the decoder, leading to different architectures and approaches.

1.3.1 In the Purview of Encoder:

Based on the perspective of an encoder, RDHEI methods can be categorized into two categories, namely Vacating Room After Encryption (VRAE) [25–27] and Reserving Room Before Encryption (RRBE) [28, 29]. These classifications are determined based on room allotment by the encoder for data, detailed as follows:

- *Vacating Room After Encryption*: In this category of RDHEI, the content owner encrypts the image using a key before transmitting it to the data hider. The primary task of the

data hider is to reserve room within the encrypted image to accommodate the encrypted data. This room reserving process is challenging due to the high entropy of the encrypted image, which makes manipulating the image to vacate room without altering its content difficult. As a result, the EC of these methods is typically low. Nonetheless, some methods manage to achieve higher EC by compromising on encryption strength, allowing for a certain level of redundancy in the original images to be maintained in specific portions. This approach is visualized in the architecture of the VRAE RDHEI methods, as depicted in Fig. 1.2a. These methods navigate the intricate balance between maintaining encryption strength and optimizing data EC, illustrating the complex interplay between security and functionality in RDHEI applications.

- *Reserving Room Before Encryption*: In this variant of RDHEI, the content owner undertakes an essential preprocessing step before the encryption process. This preprocessing exploits the natural redundancy within the image to reserve a large room for embedding. Allocating this specialized space before the encryption allows for the incorporation of a substantially greater volume of secret data within the image. The structural framework and guiding principles of the RRBE-RDHEI are detailed in Fig. 1.2b. Despite the enhanced EC offered by these methods, they require the image owner to predict the future use of the image as a cover. This level of foresight may not always be feasible for the content owner.

1.3.2 In the Purview of Decoder

From the decoder's standpoint, RDHEI methods can be broadly categorized into two types: joint and separable. The defining characteristic of joint methods is their reliance on a unique interdependence between the image and data keys for recovery. The problem is that the absence of either key within joint methods leads to an incomplete recovery, whether it be for the purpose of reconstructing the image or retrieving data, thereby requiring both keys for comprehensive functionality of recovery. However, the separable methods provide a more flexible and user-friendly approach. These methods allow for the extraction of data with only the data key and the reconstruction of the image with solely the image key, without the need for cross-dependency. This independence between data and image recovery processes not only simplifies the recovery but also significantly enhances the applicability, desirability, and security of the method. The distinct separation of keys and their respective functionalities in separable methods make them popular, especially in scenarios demanding flexible and efficient data and image management.

1.4 Applications of RDHEI

RDHEI stands as a prominent technique addressing the challenges of digital data, including security, privacy, and confidentiality, for both secret data and cover images. RDHEI plays a

crucial role in a wide array of applications such as military and defense, cloud storage, education and e-learning, healthcare, the automotive industry, smart cities and the Internet of Things, digital rights management, as well as the entertainment and gaming industry, among others [30–33].

In the cloud storage sector, RDHEI is crucial for improving data privacy and management. It works by adding metadata to encrypted files, which helps in efficiently retrieving and managing data while keeping secret data secure. RDHEI is also used in digital archives and libraries, where it secure the large collections of digital documents. By embedding metadata in encrypted digital copies, organizations can effectively control document history and access permissions, ensuring the preservation of historical and cultural assets for future generations.

In the military and defense area, RDHEI is used for the secure handling of sensitive information. It enables protected communication and the management of files in a way that can be tailored to different levels of security clearance. This application is crucial in military contexts, where the ability to securely manage and communicate information is very importance. In the world of education and e-learning, RDHEI helps protect educational content and check the authenticity of student assignments. With more people using online learning, it is important to make sure that educational materials and student submissions are genuine and have not been tampered. RDHEI makes this easier by putting unique key in student work, which helps in quickly checking their authenticity while keeping their information private.

In the healthcare sector, RDHEI tackles the critical issue of keeping patient information confidential while making sure medical data is available to those who are allowed to see it at different levels. By using encryption on diagnostic images and embedding patient details, it helps keep patient records safe and easy to manage at different levels. Similarly, in the automotive field (self-driving cars), RDHEI plays a key role in protecting the large amount of data needed for things like navigation, safety, and talking between vehicles. By putting important details into encrypted images that vehicles and systems share with each other, RDHEI helps these autonomous systems work safely.

In the area of digital rights management, RDHEI also plays a crucial role by enabling content distributors to provide safity for intellectual property. This is done through the embedding of watermarks or buyer-specific information within digital content, enhancing traceability and preventing unauthorized distribution. In the entertainment and gaming sector, RDHEI is used to keep digital content and online game chats safe. It helps secure the sharing of games and stops cheating by embedding special marks in in-game items or digital content.

RDHEI's extensive applications not only showcase its wide-ranging utility and critical role in enhancing digital security and privacy but also its evolving significance across various fields. With the progression of digital technologies, the potential for RDHEI applications is poised to expand, further solidifying its impact on the future of secure digital communication.

Conclusively, application of RDHEI has a wide range of sectors to enhancing security and privacy in cloud storage, digital archives, the military, to protecting intellectual property in

digital rights management and enriching experiences in healthcare, education, the automotive industry, and entertainment. The ability to embed and manage data within encrypted images offers a unique solution to the challenges of data privacy, security, and integrity, making it indispensable in the increasingly digital world.

1.5 Metrics and Benchmark Datasets for Evaluating RDHEI Methods

The performance evaluation of RDHEI methods focuses on three key aspects: EC, visual quality of the reconstructed image, and encryption security level. The EC quantifies the amount of data that can be hidden, typically measured in bits per pixel (bpp). Visual quality and security level metrics such as Peak Signal-to-Noise Ratio (PSNR) [34], Structural Similarity Index (SSIM), correlation coefficient [35], Shannon entropy [36], Number of Pixel Changing Rate (NPCR) [37], Unified Averaged Changed Intensity (UACI) [37], and histogram distribution are employed to evaluate the security strength of the marked encrypted images.

PSNR is most commonly used to measure the quality of reconstructed image, determined using Mean Squared Error (MSE). It measures the average squared difference between the original and a reconstructed image. Let consider a original image I with size $M \times N$ and I_r is reconstructed image with same size then PSNR is determined as follows:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}} \right), \tag{1.1}$$

where MAX_I is the maximum possible pixel value of the image. Specifically, for an 8-bit grayscale image $MAX_I = 255$, and $\text{MSE} = \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} \left(I_{(i,j)} - I_{r(i,j)} \right)^2$.

SSIM is improved version which also considers the changes in structural information, texture, and luminance. The SSIM index is a decimal value between -1 and 1, where 1 indicates perfect similarity. For a window of size $X \times Y$, it is determined as follows:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}, \tag{1.2}$$

where μ, σ^2, c are average intensities, variances, and variables to stabilize the division with weak denominator; σ_{xy} is the covariance of x and y; $c_1 = 6.5025$, $c_2 = 58.5225$.

Both PSNR and SSIM possess distinct applications and constraints. PSNR is widely used due to its simplicity and clear physical meaning, but its correlation with perceived visual quality can be inconsistent. Whereas, SSIM is designed to closely aligns with human visual perception. However, it can be more complex to compute and interpret.

Correlation measures how changes in one variable are associated with changes in another. It can be determine about horizontal, vertical, and diagonal directions to describe the relationship

between pixel intensities in these directions. Horizontal, vertical, diagonal correlation are shows the relationship between pixels along the same row, same column, and diagonals, respectively. Let X and Y are the two set of pixels of length of n, then correlation coefficient C_c is determined as follows:

$$c_c = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2 \sum_{i=1}^{n}(Y_i - \bar{Y})^2}}, \tag{1.3}$$

where $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$. The denominator normalizes this covariance, resulting in a dimensionless coefficient C_c that ranges from -1 to 1, where 1 means perfect positive linear correlation, -1 means perfect negative linear correlation, and 0 means no linear correlation.

Shannon entropy is a fundamental concept in information theory that quantifies the uncertainty or randomness in a image. Let $p_0, p_1, \ldots p_{255}$ represent the probability of each pixel intensity from 0 to 255 in gray scale image, then the entropy E can be determine using Eq. (1.4).

$$E = -\sum_{i=0}^{255} p_i \log_2(p_i). \tag{1.4}$$

The value of E will be near 8 bits for an image where the pixel intensities are highly distributed, meaning that the image has a high level of randomness or variability in its pixel values. An entropy value close to 8 suggests that the image makes full use of the 8-bit grayscale range, with a uniform distribution of pixel intensities. This is typical for complex or highly textured images. Lower entropy values indicate less randomness, suggesting more uniform or smooth images.

NPCR measures the percentage of pixels that change in the encrypted image when a single pixel in the original image is changed. If I_e is the encrypted image

$$\text{NPCR} = \frac{\sum\sum D_{(i,j)}}{N \times M} \times 100. \tag{1.5}$$

where

$$D_{(i,j)} = \begin{cases} 0, & \text{if } I_{(i,j)} = I_{e(i,j)} \\ 1. & \text{if } I_{(i,j)} \neq I_{e(i,j)} \end{cases}$$

High NPCR Value indicates that the encryption algorithm is highly sensitive to changes in the input image, which is a desirable property for preventing differential attacks.

UACI measures the average intensity of differences between the original and encrypted images, reflecting the impact of the encryption on pixel values. It quantifies the average change in the intensity of the pixels between the two encrypted images, I and I_e, providing insight into how much the pixel values have been altered on average.

$$\text{UACI} = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{i=1}^{N}\frac{|E_{1(i,j)} - E_{2(i,j)}|}{255} \times 100\%. \tag{1.6}$$

High UACI value suggests that the encryption method significantly alters pixel intensities, making it difficult for attackers to infer any meaningful information from the encrypted image.

These metrics ensure that the message embedding phase does not compromise the security level of the encryption method, and that the embedded encrypted message remains undetectable in the encrypted domain. The three aspects i.e., EC, visual quality, and visual security level, are crucial for comprehensive performance evaluation of RDHEI methods.

The development and progress of RDHEI have recorded the importance of utilizing comprehensive and diverse datasets to systematically evaluate the effectiveness and robustness of proposed methods. In this context, the BOSSBase [38], BOWS-2 [39], UCID [40], and selected images from the USC-SIPI [41] image database have emerged as primary resources. These datasets, each with unique characteristics and challenges, offer a rich testing ground for RDHEI methods across a wide range of scenarios. This ensures that the developed methods are not only theoretically sound but also practically clear and capable of performing reliably in real-world applications.

1.5.1 BOSSBase

The BOSSBase dataset is a widely used resource for evaluating image processing algorithms, particularly in steganalysis. It consists of 10,000 uncompressed, 512×512 pixel grayscale images in the portable Graymap format, sourced from various cameras and captured under diverse conditions. This dataset provides a rich testing ground for evaluating RDHEI algorithms on high-quality, uncompressed images without visual distortions or compression. The BOSSBase images cover a wide range of subjects and scenarios, including natural scenarios, cityscapes, portraits, and still-life scenes, ensuring a diverse representation of real-world use cases. The absence of compression and visual distortions allows researchers to assess the performance of RDHEI algorithms in a controlled environment, focusing on the fundamental challenges of imperceptible data embedding and extraction.

1.5.2 BOWS-2

Designed explicitly for evaluating DH methods in compressed images, the BOWS-2 dataset comprises 10,000 images of 512×512 pixel grayscale images in the JPEG format, with quality factors ranging from 90 to 100. These images are sourced from professional photographers and popular repositories, ensuring a diverse representation of real-world scenarios. BOWS-2 enables researchers to assess the robustness of RDHEI algorithms when dealing with compressed image formats, a common requirement in practical applications. The dataset includes images of various subjects, such as landscapes, portraits, architecture, and still-life scenes, captured under different lighting conditions and with varying levels of complexity. The inclusion of a wide range of quality factors allows researchers to evaluate the performance of RDHEI algorithms

under varying degrees of compression, simulating real-world scenarios where image quality and file size are often balanced.

1.5.3 UCID

The UCID dataset is a comprehensive collection of uncompressed color images containing a diverse range of contents and attributes. It consists of 1,338 images with varying resolutions, from 512×384 to 3264×2448 pixels, including natural scenes, objects, textures, and artificial patterns. The dataset is subdivided into several categories, such as natural scenes, buildings, animals, and textures, allowing researchers to test their RDHEI algorithms on specific types of images or across a broad range of content. Additionally, the UCID dataset provides metadata information, such as EXIF data and ground truth segmentation masks, enabling researchers to explore context-aware and semantically-driven approaches to RDHEI. This metadata can be leveraged to develop algorithms that adapt to image content and characteristics, potentially improving the performance and robustness of RDHEI methods.

1.5.4 SIPI

The USC-SIPI image database is a comprehensive collection from the Signal and Image Processing Institute at the University of Southern California, known for its wide range of standard test images used in the field of digital image processing. This database includes various categories such as textures, aerial, miscellaneous, and sequences that have been repeatedly used for research in image processing, compression, analysis, and other areas. The USC-SIPI database offers a diverse array of test images with varying texture complexities, providing a comprehensive testing ground for evaluating the robustness and versatility of RDHEI methods. Highly textured images like $Baboon$, $Boat$, etc, rich in details and complex patterns, assess the methods capability to embed data imperceptibly within textures regions while ensuring accurate data recovery. Medium-textured images such as $Lena$, Man etc, featuring a balance of smooth regions and detailed areas, challenge the methods to adapt seamlessly to different texture levels within a single image. Conversely, smooth images like $Airplane$, $Splash$, etc, predominantly characterized by uniform textures, test the methods ability to subtly embed data. Some of the standard test images from the USC-SIPI dataset are shown in Fig. 1.3 such as the $Lena$, $Baboon$, $Airplane$, $Jetplane$ (F-16), Man, $Boat$, $Tiffany$, $Barbara$, $Pepper$, and $Crowds$ images, offers a rich tapestry for exploring and refining methods in the fields of RDHEI, RDH, watermarking, steganography, and cryptography. Each of these images presents unique challenges and opportunities for testing the limits and capabilities of various digital image processing and security algorithms.

It is evident that the combination of these datasets and images provides a robust framework for evaluating RDHEI methods, ensuring that they not only meet theoretical expectations but also stand up to the challenges of real-world applications. The collective use of BOSS-

Base, BOWS-2, UCID, and test images from the USC-SIPI dataset is instrumental in advancing the field of RDHEI, proving that these resources are sufficient to evaluate the performance of RDHEI methods comprehensively.

1.6 Research Gaps and Problem Statement

The field of RDHEI has witnessed significant progress over the years, with researchers proposing various methods to address the inherent challenges. However, despite these advancements, several research gaps remain, hindering the development of optimal solutions that can strike a delicate balance among the competing dimensions of embedding capacity, image quality, reversibility, separability, and security. The following research gaps have been identified through a comprehensive literature review:

- From the literature review, it is found that high embedding capacity is achieved prediction based methods. However, due to the issue of encoding limitation in only block based or bit-plane based, both are not able to fully utilize the spatial correlation.
- Most of the literature cover only two aspects i.e., reversibility and embedding capacity whereas the security is also an important concern.
- In RRBE based RDHEI techniques, content owner has to pre-process the image which (in itself) is a highly-complex task.
- To avoid the problem of underflow/overflow and to provide reversibility, large overhead in terms of label/location map is there that limits the embedding capacity.

These notable gaps in existing literature highlight the limitations of current RDHEI techniques in addressing the multi direction requirements of high embedding capacity, preserved image quality, reversibility, separability, and robust security measures.

Problem statement: The major challenge lies in developing an RDHEI technique that can optimally manage the trade-off between embedding capacity and image quality, while simultaneously addressing the additional requirements of reversibility, separability, privacy, and security.

1.7 Objectives

The research gaps identified in the domain of RDHEI highlight the need for comprehensive solutions that address the multifaceted challenges of EC, image quality, reversibility, separability, and security. To address these concerns, a thorough study and experimental evaluation of available data-hiding methods in the encrypted domain are crucial. Such an analysis, conducted on a consistent test dataset, can provide valuable inputs into the strengths and limitations of existing

Fig. 1.3: Test images of (a) $Lena$, (b) $Baboon$, (c) $Airplane$, (d) $Jetplane$, (e) Man, (f) $Boat$, (g) $Tiffany$, (h) $Barbara$, (i) $Pepper$, and (j) $Crowd$

methods, forming the way for the development of novel methodologies. Another critical aspect is the trade-off between EC and the overhead required for reversibility and underflow/overflow prevention. Large overheads in the form of labels or location maps often limit the achievable capacity, necessitating the exploration of new methods that can strike an optimal balance between capacity and overhead while maintaining reversibility and preventing underflow/overflow. In response to the identified gaps, objectives are established to accelerate the domain of RDHEI forward. These objectives aim to develop the solutions that are both robust and efficient, while also ensuring security. Such advancements are critical for meeting the diverse demands of variuos applications and real-world scenarios. The research objectives are delineated as follows:

- **Objective 1**: To study, analyze and experimentally evaluate various available data hiding techniques in the encrypted domain on a consistent test data set
- **Objective 2**: To develop novel data hiding scheme(s) in encrypted domain for optimal utilization of spatial correlation in order to achieve high-embedding capacity
- **Objective 3**: To develop high-capacity reversible data hiding scheme(s) for providing the separability as well the complete reversibility of the cover media
- **Objective 4**: To develop new privacy-preserving data hiding scheme(s) based on prediction error expansion in the encrypted domain to address the concern of high overhead with improved prediction accuracy

Building on our findings, this thesis aims to propose novel data hiding methods optimizing spatial correlation for increased EC, while also prioritizing privacy preservation, separability and reversibility .

1.8 Contributions of Thesis

The research work presented in this thesis makes significant contributions to the domain of RDHEI, addressing several critical gaps and challenges identified. Through detailed experimental evaluation on the qualitative aspects, this research provides a comprehensive analysis of existing RDHEI techniques, highlighting their strengths, limitations, and areas for improvement. This evaluation serves as a critical resource for future research in RDHEI, offering insights into the effectiveness of different approaches. This thesis also incorporates a bibliometric analysis, offering quantitative insights into publication trends, influential contributors, and collaborative networks within the RDHEI domain. This synergistic combination of qualitative and quantitative perspectives helps to identifies the researchers to identifies the challenges and gaps of the domain.

Furthermore, one of the primary contributions of this thesis is the development of novel RDHEI methods, including the Link Chain Driven (LCD) method, the Difference Image Transfiguration approach, the Bit-Plane based method utilizing Multi-Level Blocking with Quad Tree

encoding, continuous space allocation and the Adaptive Two-Stage RDHEI method. These methods offer unprecedented improvements in EC, image quality, and security, marking a significant advancement in RDHEI research.

The thesis successfully addresses several critical research gaps, including the underutilization of spatial correlations, the balancing act between EC and image quality, and the need for improved security measures in RDHEI techniques. By presenting innovative solutions that tackle these issues, the research contributes to the refinement and advancement of RDHEI methodologies.

The proposed RDHEI methods have significant implications for practical applications. By enhancing the security, capacity, and efficiency of data hiding techniques, this research facilitates the development of more robust systems for protecting sensitive information. Lastly, the thesis outlines future directions for RDHEI research, recommending topics and areas for further investigation and advancement.

Collectively, these contributions constitute a advancement in the RDHEI domain, introducing new methodologies that improve the security, EC, and efficiency of data hiding techniques. This research not only bridges current gaps in the field but also establishes a foundation for subsequent methods in secure communication and data protection.

1.9 Thesis Organization

This thesis comprises across eight chapters, progressively building from introduction to conclusion to present a comprehensive research journey. The arrangement of the chapters is outlined as follows:

- **Chapter 1. Introduction:** The starting chapter of the thesis sets a foundation, starting with the prevalence of digital data and the necessity for secure communication today. Further, it discusses DH as essential for ensuring security and privacy for secret data, then moves to RDH, which guarantees the reconstruction of cover image. The chapter narrows its focus to RDHEI, a version of RDH for encrypted images, proving security for cover image. Furthermore, it discusses RDHEI categories, applications, metrics and benchmark datasets for evaluation of the RDHEI methods. Lastly, this introduction lays the groundwork for the thesis, outlining the research objective and significance.

- **Chapter 2. Literature Review:** The literature review chapter offers an in-depth analysis of current research in RDHEI. More specifically, the chapter presents a bibliometric analysis to highlight publication trends and key contributors in RDHEI research. Additionally, the chapter details state-of-the-art methods and solutions to field-specific challenges. Thus, this comprehensive overview combines qualitative and quantitative insights, preparing the ground for further contributions in the domain.

- **Chapter 3. Link Chain Driven Reversible Data Hiding in Encrypted Images for High Payload:** This chapter presents a novel LCD method for RDHEI, addressing gaps in prior research by using neighborhood correlations to create variable-sized link chains, enhancing data EC. This method employs pixel averages or dual quantization for space creation within images for secret data. Further, it outlines significant contributions like the innovative pixel collection, improved embedding space, reversibility in encryption, and flexible error correction/image distortion balance. Furthermore, experimental results which show LCD-RDHEI's superiority in EC while ensuring separability, over the previous methods, are also presented.

- **Chapter 4. High-Capacity Reversible Data Hiding in Encrypted Image based on Difference Image Transfiguration:** This chapter introduces another novel RDHEI method difference Image transfiguration which enhances data embedding capacity through image processing. Specifically, the proposed method employs Median Edge Detector (MED) and difference predictors to prepare images for embedding by cleaning Most Significant Bit (MSB) planes. For this, a unique cleaning process for freeing up space in MSB planes has been introduced. The chapter also introduces a difference location techniques and adopts Huffman encoding for efficient image reconstruction. Therefore, the proposed method emphasizes enhanced security with advanced encryption, direct data embedding, high capacity on benchmark datasets, quicker embedding times, and the benefits of bit-plane-wise pre-processing over direct encoding.

- **Chapter 5. Bit-Plane based Reversible Data Hiding in Encrypted Image Using Multi-Level Blocking With Quad Tree:** Diverging from traditional methods, Chapter 5 introduces a novel Multi-Level Blocking (MBQ) RDHEI as a solution to the limitations of variable block sizes. Utilizing a quad-tree for bit-plane encoding, this method significantly compresses data by exploiting sparsity and implementing a strategy to mitigate inter bit-plane redundancies. Demonstrating exceptional EC and impeccable image reconstruction, the chapter highlights this novel approach's efficiency in handling redundancies and its impressive performance across different variations tested on benchmark datasets.

- **Chapter 6. High Capacity Reversible Data Hiding with Contiguous Space in Encrypted Images:** This chapter is devoted to issues of security, efficiency, and synchronization in data hiding within encrypted images. It presents a breakthrough RDHEI method that utilizes a 2×2 block division for enhanced spatial correlation, leading to better embedding rates and ensured reversibility. A key innovation is the block-shifting mechanism, eliminating the need for complex mapping and significantly improving both security and efficiency. An index-based embedding strategy further simplifies integration, avoiding extra keys. Key advancements include improved embedding through strategic spatial correlation, efficient data placement without label maps, and enhanced security by

synchronizing data and image content. The proposed approach demonstrates exceptional embedding rates and robust security, showcasing its superiority over existing techniques.

- **Chapter 7. Adaptive Two-Stage Reversible Data Hiding in Encrypted Image using Prediction Error Expansion:** Different from the proposed RRBE method, this chapter introduces a VRAE based Adaptive Two-Stage RDHEI method that creatively tackles common challenges such as overflow, underflow, and limited EC. By dividing images into two set of bit-planes and further into variable-sized blocks, it sets the stage for enhanced data embedding. Scrambling and pixel shuffling ensure encryption without losing block correlation. The core technique of prediction error expansion and pixel value ordering enables efficient and reversible embedding, significantly improving EC and visual security. Key highlights include a novel combination of encryption and RDHEI for heightened security, superior EC, and flexible data recovery, demonstrating the method's effectiveness and efficiency in secure data transmission.

- Chapter 8. **Conclusion and Future Research Directions:** This chapter serves as the concluding segment of the thesis, offering a comprehensive summary of the most significant findings and insights derived from the research. This chapter concludes with a discussion of the most prominent future research directions, highlighting potential areas for further exploration and development in the field of hyperspectral image classification.

CHAPTER 2

LITERATURE REVIEW

This chapter presents an extensive investigation into the domain of RDHEI, an area that has experienced remarkable growth since its emergence in 2008. The RDHEI lies in its dual advantage for both the image owner and the data hider, making it an important subject for research within the domain of confidentiality and security. This unique benefit has captured the attention of scholars and researchers worldwide, leading to an extensive collection of literature that encompasses a variety of methodologies and approaches.

The growth of RDHEI research underlines the importance of a comprehensive review that not only displays the domain advancements but also provides a critical assessment of the methodologies employed, using both quantitative and qualitative perspectives. Accordingly, this chapter i.e. literature review is designed to offer a complete overview of the RDHEI domain. It incorporates quantitative analysis that uncovered the trends, collaborations, and impactful contributions that have defined the domain in the last five years, while also engaging in qualitative examination to explore the theoretical foundations, practical implications, methodological soundness, and future prospects of RDHEI research. Accordingly, this chapter integrates publication trend and Productivity Analysis (quantitative analysis) and detailed review of prominent RDHEI methods (qualitative exploration) in sections 2.1 and 2.2, respectively, to provide a comprehensive understanding of the current state of RDHEI research.

2.1 Publication Trend and Productivity Analysis

This section offers a comprehensive exploration of the publication trends and productivity analysis applied to RDHEI methodologies through bibliometric studies conducted over the last five years. The quantitative analysis plays a crucial role in this context, providing the scholarly contributions, trends, and advancements within the RDHEI field. The publication trend analysis aims to illuminate the growth and evolution of RDHEI research over time, revealing the increasing significance of this domain in the digital age. This investigation facilitates a deeper under-

standing of the academic engagement and the spread of knowledge pertaining to the RDHEI domain. Simultaneously, the productivity analysis assesses the impact and scope of RDHEI research through detailed evaluation of publication outputs from researchers, institutions, and countries. This provides an objective measure of the key contributors and their influence in the domain.

2.1.1 Data Collection Source and Strategy

To investigate the academic trend of RDHEI, this study utilized the Web of Science (WoS) database as its primary source of scholarly articles. WoS is celebrated for its extensive collection of peer-reviewed research, encompassing various indices such as the Science Citation Index Expanded, the Social Sciences Citation Index, and the Book Citation Index – Science. This diversity makes WoS an ideal platform for accessing high-quality studies.

The search for relevant RDHEI literature was conducted with a carefully designed query on January 31, 2024, spanning a period from January 1, 2019, to January 31, 2024. The search string formulated for this purpose was: (RDHEI OR ("reversible data hiding in encrypted images") OR ((hiding OR embedding OR RDH) NEAR/3 encrypted)). This approach resulted in the identification of 455 research articles, each contributing to the discourse on RDHEI. These articles were subsequently analyzed bibliometrically, as discussed subsequently in this section.

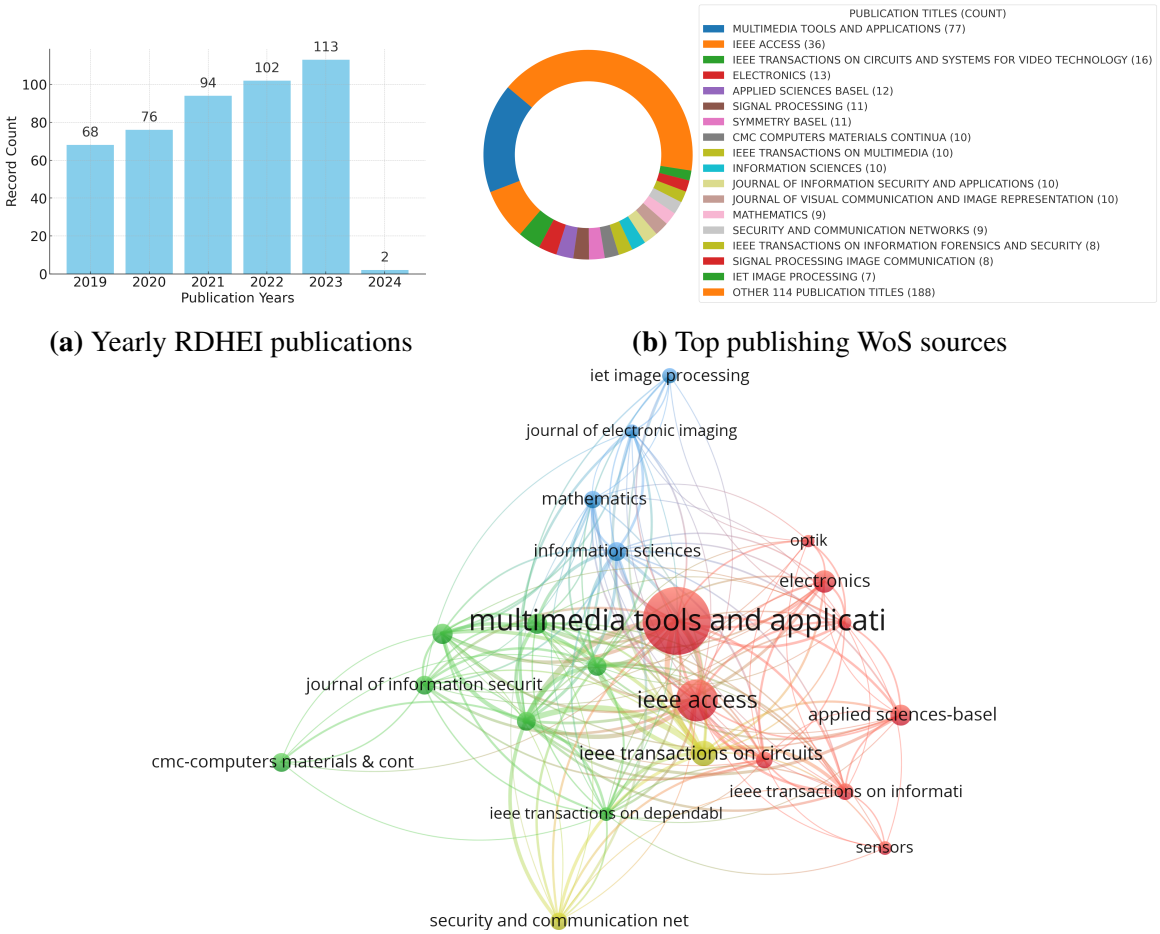
2.1.2 Publication Structure Analysis

The initial step involved a comprehensive analysis of 455 research articles, categorizing them into various verticals like annual publications, publishing sources, productive researchers and organizations and Country trends.

A. Annual Publications and Sources

Fig. 2.1 provides an overview of the annual publication trends and the types of documents that constitute the body of research in the RDHEI domain. Specifically, Fig. 2.1a is depicted which illustrates the upward trajectory of research publications related to RDHEI from 2019 to 2024, as cataloged by the WoS. The initial count of 68 publications in 2019 evidences a steady climb in academic contributions, culminating in a zenith of 113 publications in 2023. This reflects the growing interest in the RDHEI domain. The early count for 2024 is double publication, suggesting continued engagement with RDHEI research, although it is premature to forecast the total contributions for the year. Such a pattern accentuates the evolving and pertinent nature of RDHEI studies within the scholarly community.

Furthermore, Fig. 2.1b presents a visualization for the source in the RDHEI domain. The figure showcases the distribution of publications across various sources, with 'Multimedia Tools and Applications' leading at 77 publications, accounting for 16.923% of the total 455 papers an-



(a) Yearly RDHEI publications

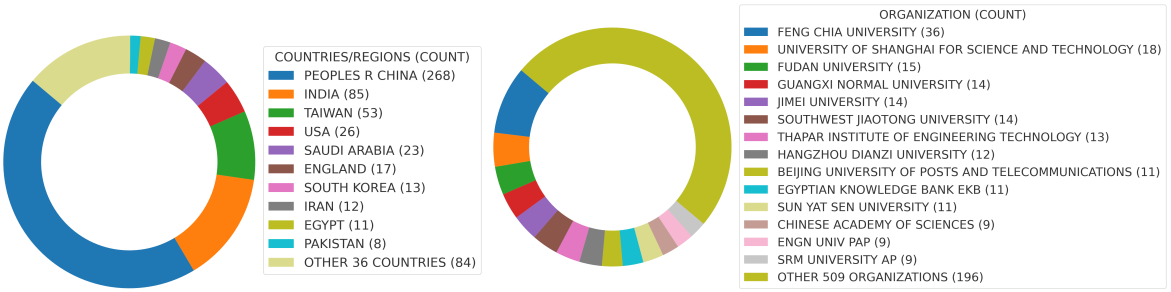
(b) Top publishing WoS sources

(c) Linkage network of sources with $\geq$ 20 documents/source

Fig. 2.1: Annual publications and their source from 2019 to 2024 within the domain of RDHEI

alyzed. This is followed by 'IEEE Access' and 'IEEE Transactions on Circuits and Systems for Video Technology', indicating these journals' significant roles in advancing RDHEI research. Notably, the category 'Other 114 Publication Titles' comprises 188 publications, reflecting a diverse range of contributing sources. This graphical representation highlights the breadth of research activity and the key journals contributing to the field of RDHEI.

For an in-depth analysis of the 455 documents, the VoSviewer tool is utilized to create linkage networks among the sources based on citations. By setting a threshold of 20 documents per source as shown in Fig. 2.1c, only those meeting this criterion are considered, which helps identify the more influential sources. The resulting visualizations expose the central nodes of scholarly influence within the network, with publications like 'IEEE Transactions on Circuits and Systems for Video Technology', 'Signal Processing', and 'Multimedia Tools and Applications' being notably prominent. These sources are highlighted for their significant citation counts and Total Link Strength (TLS) in the accompanying table, emphasizing their impact and importance in the field. 'IEEE Transactions on Information Forensics and Security' and 'Journal of Visual Communication and Image Representation' are also distinguished, not just for their citation volume but for their capacity to bridge different research areas, as indicated by



(a) Countries trends in publications. **(b)** Top contributing organizations

Fig. 2.2: Country and organization trends

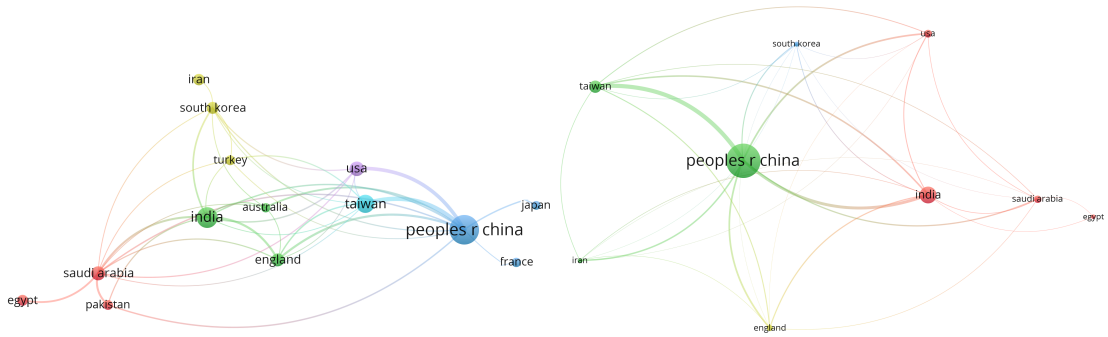
their TLS. This reflects the interdisciplinary appeal of their publications. The analysis reveals a robust network of sources, with 'IEEE Access' being particularly noteworthy for its extensive coverage and accessibility to the research community. Understanding these connections is vital for recognizing foundational studies and identifying emerging trends in the RDHEI field.

B. Country and Organization Trends

In examining global research patterns within the specialized domain of RDHEI, the provided Fig. 2.2a offers a concise geographical narrative. Derived from the WoS repository, figure underscores China's predominant contribution, with a notable 268 publications, indicative of a robust research emphasis in the region. India and Taiwan also emerge as significant contributors, emphasizing the Asian continent's active engagement in the domain. The visualization captures the essence of international collaboration and scholarly interest, with the United States and several European and Middle Eastern countries marking their presence. This bibliometric snapshot, while capturing the 455 papers due to some cross-country collaborations, describes the worldwide momentum towards the RDHEI domain.

Country Co-author and Citation Linkages: The investigation of co-author and citation linkages among countries, based on a dataset of 455 documents, offers a detailed network of global research interconnected and scholarly influence. Fig. 2.3a displays a threshold of five documents per country and encompasses a wider network of 14 countries based on co-author, with the People's Republic of China (PRC), the India and USA, appearing as pivotal nodes, indicating their extensive collaborative reach and high publication output. Fig. 2.3b focused on the collaborative dynamics with a threshold of at least ten documents per country based on citation, showing a network of 9 countries. This network reveals a rich collection of international research efforts, with a total of 9 countries emerging in the large set. This representation spotlights the most progressive nations whose research not only has significant volume but also high citation impact, marking them as influential players in this field.

Figures present a comprehensive view of the RDHEI country co-author and citation, where countries like the PRC, India, Taiwan, and the USA emerge as central nodes of research activity and citation. Additionally, they reveal a core consortium of nations that, although producing



(a) Network co-author min document count 5 **(b)** Network citation min document count 10

Fig. 2.3: Country co-author and citation networks

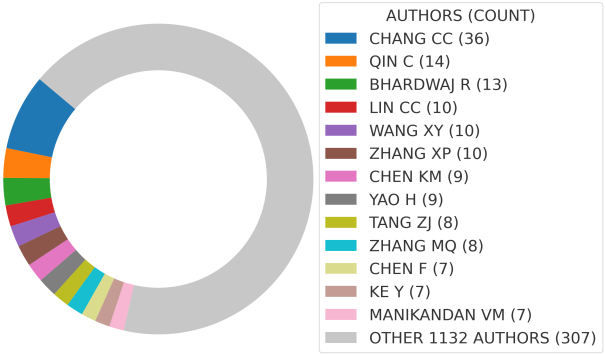


Fig. 2.4: Top contributing authors.

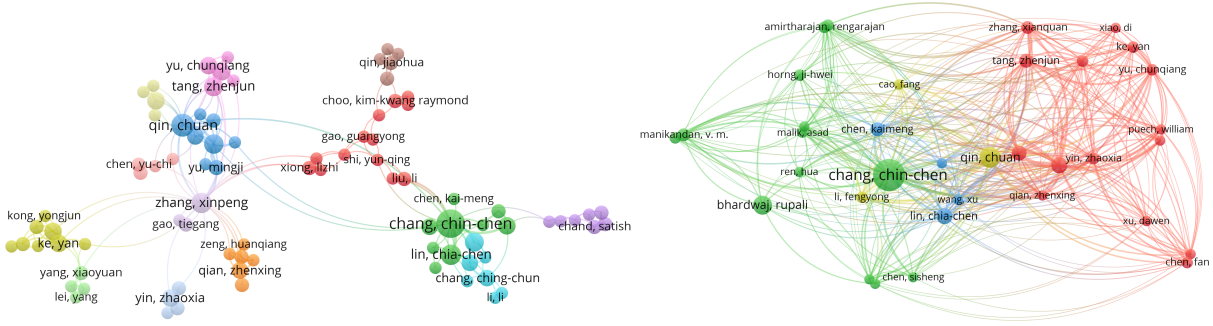
a smaller volume of publications, wield significant influence through their scholarly contributions.

C. Productive Researchers:

In this sub-section, the quantifiable overview of the most prolific contributors to the field is shown with the help of Fig. 2.4 and Table-2.1. Fig. 2.4 identifies 'Chang, Ching-Chun' as the most productive author with 36 papers, complementing the organizational data and underscoring the synergy between individual and institutional productivity. Notable contributions from authors affiliated with mainland Peoples R China and Taiwan signify a regional concentration of expertise in RDHEI, while 'Thapar Institute of Engineering & Technology's' representation points to a growing research interest in India. *Author Co-author and Citation Linkages:* This analysis adopted a detailed counting method to evaluate co-authorship links among researchers, applying a publication threshold minimum of 2 for inclusion criteria while excluding publications authored by more than 25 contributors. Following these guidelines, 245 authors qualified under the threshold of 2. An in-depth analysis of the most comprehensive networks is illustrated in Fig. 2.5a, revealing that 87 authors formed co-authorship linkages. This visual representations are characterized by distinct colors to differentiate between diverse co-authorship networks. The node sizes within these maps indicate the quantity of an author's publications,

Table 2.1: Top ten author based on record count

Author Name	Country	Organization	Record Count
Chang, Ching-Chun	Taiwan	Feng Chia Univ	36
Qin, Chuan	Peoples R China	Univ Shanghai Sci & Technol	14
Bhardwaj, Rupali	India	Thapar Inst Engn & Technol	13
Lin, Chia-Chen	Taiwan	Natl Chin Yi Univ Technol	10
Zhang, Xinpeng	Peoples R China	Fudan Univ	10
Yao, Heng	Peoples R China	Univ Shanghai Sci & Technol	9
Chen, Kaimeng	Peoples R China	Jimei Univ	9
Tang, Zhenjun	Peoples R China	Guangxi Normal Univ	8
Zhang, Minqing	Peoples R China	Engn Univ PAP	8
Zhang, Xianquan	Peoples R China	Guangxi Normal Univ	7



(a) Extensive network minimum of 5 documents **(b)** Extensive network minimum of 5 documents

Fig. 2.5: Author co-author and citation networks

with larger nodes signifying a more substantial body of work. Additionally, the thickness of the lines connecting the nodes reflects the co-authorship bond strength, with denser lines indicating more robust collaborations. Prominent authors, such as 'Chang, Chin-Chen', 'Qin, Chuan', and 'Zhang, Xinpeng', are marked by larger, colored circles to denote their significant contribution to the work, while authors with fewer citations are represented by smaller circles, demonstrating a proportional relationship between a node's size and an author's citation volume.

Furthermore, the author citation linkage is determined using the threshold 5 as depicted in Fig. 2.5b. The network includes 48 authors, with 35 in the large set as shown in Fig. 2.5b. This trimmed network highlights the most prolific authors and their associated documents, pointing the central figures within the community whose work is frequently referenced and who have a substantial impact on the direction of research within the field.

To clearly analyze the citation, Table 2.1 delineates the top ten authors ranked based on citations, offering a distinct perspective on their academic impact and collaboration dynamics. Unlike the previous table, which focused on co-authorship linkages, this table emphasizes citation counts, providing insights into the scholarly influence and reach of these authors' works. 'Zhang, Xinpeng' leads this list with 10 documents with 326 citations and a total linkage strength of 29, which shows the prolific collaboration and significant academic impact. Following him, 'Chang, Chin-Chen' reflecting a substantial contribution to the field both in terms of

Table 2.2: Top ten documents based on citations

Id	Document Title	Authors	Year	Citations	Links
55	Separable and Reversible Data Hiding in Encrypted Images Using Parametric Binary Tree Labeling	Yi, Shuang; Zhou, Yicong	2019	114	54
221	A novel triple-image encryption and hiding algorithm based on chaos, compressive sensing and 3D DCT	Wang, Xingyuan; Liu, Cheng; Jiang, Donghua	2021	110	3
39	Reversible Data Hiding in Encrypted Images Based on Multi-MSB Prediction and Huffman Coding	Yin, Zhaoxia; Xiang, Youzhi; Zhang, Xinpeng	2020	104	42
438	FPGA Realization of a Reversible Data Hiding Scheme for 5G MIMO-OFDM System by Chaotic Key Generation-Based Paillier Cryptography Along with LDPC and Its Side Channel Estimation Using Machine Learning Technique	Shajin, Francis H.; Rajesh, P.	2022	93	4
115	High-capacity reversible data hiding in encrypted images based on extended run-length coding and block-based MSB plane rearrangement	Chen, Kaimeng; Chang, Chin-Chen	2019	76	40
145	An efficient coding scheme for reversible data hiding in encrypted image with redundancy transfer	Qin, Chuan; Qian, Xiaokang; Hong, Wien; Zhang, Xinpeng	2019	70	32
78	New Framework of Reversible Data Hiding in Encrypted JPEG Bitstreams	Qian, Zhenxing; Xu, Haisheng; Luo, Xiangyang; Zhang, Xinpeng	2019	68	23
149	Effective reversible data hiding in encrypted image with adaptive encoding strategy	Fu, Yujie; Kong, Ping; Yao, Heng; Tang, Zhenjun; Qin, Chuan	2019	60	22
286	Securing Data in Internet of Things Using Cryptography and Steganography Techniques	Khari, Manju; Garg, Aditya Kumar; Gandomi, Amir H.; Gupta, Rashmi; Patan, Rizwan; Balusamy, Balamurugan	2020	60	4
14	An Improved Reversible Data Hiding in Encrypted Images Using Parametric Binary Tree Labeling	Wu, Youqing; Xiang, Youzhi; Guo, Yutang; Tang, Jin; Yin, Zhaoxia	2020	57	28

productivity and influence, with a linkage strength of 51. This highlights Chang's central role in collaborative networks and his research's wide acknowledgment. 'Wang, Xingyuan' occupies the third position with 8 documents and 263 citations, indicating a notable influence with a relatively lower linkage strength of 21, suggesting impactful yet selective collaborations. Further down the list, authors such as 'Yin, Zhaoxia', 'Qin, Chuan', and 'Xiang, Youzhi' are ranked based on their citation counts, showing the diversity of contributions and the extent of their scholarly impact. This analysis shows the multifaceted nature of academic contributions, where both collaboration and citation metrics are crucial for understanding an author's influence and role within the scholarly community.

2.2 Detailed Review of Prominent RDHEI Methods

This subsection provides a detailed review of the methodologies of the existing and prominent RDHEI methods, highlighting their principal mechanisms and inherent limitations. As previously discussed, RDHEI methods can be broadly classified into two categories based on the time of room reservation: VRAE and RRBE. Each category presents distinct benefits and challenges. To elaborate on the growth and development of RDHEI methods, the text is divided into two subsections. Subsection 2.2.1 offers an in-depth examination of the important methods within the VRAE category, while subsection 2.2.2 focuses on the RRBE category.

2.2.1 Growth of VRAE-RDHEI Methods

In 2008, Puech *et al.* [25] developed first RDHEI method that encrypts images into 16-pixel blocks using Advanced Encryption Standard's (AES) in Electronic Code Book (ECB) mode and embeds a secret message bit into each block. Utilizing a data hiding key and a Pseudo Random Number Generator (PRNG), specific pixels are targeted for embedding. The decryption process involves extracting the message through marked pixels and employing a standard deviation analysis to differentiate between correctly and incorrectly decrypted blocks, relying on the data hiding key for accurate image reconstruction. Zhang *et al.* [26] developed next RDHEI method, the image is encrypted using stream cipher and dividing them into small blocks to embed one bit of a secret message per block. By organizing pixels into two groups based on an embedding key, the method flips the three Least Significant Bits (LSBs) to embed the message during encryption. Decryption attempts to reconstruct the original image, perfectly recovering the five Most Significant Bits (MSBs) but altering the 3 LSBs. The secret message is extracted by evaluating inconsistencies within the pixel groups. Despite increasing block size to improve reconstruction chances, the technique is irreversible and offers a low payload capacity (less than 0.1 bpp), as it embeds only one bit per block. Hong *et al.* [42] proposed an enhancement of Zhang's method to increase data extraction accuracy, addressing the original method's limitations in utilizing pixel correlations within and across block borders. This improved version introduces a refined approach for assessing block smoothness and integrates a side-match scheme, aimed at lowering the error rate in bit extraction and enhancing error correction capabilities. While this adaptation successfully reduced the error rate by 0.87%, it still does not achieve full image reconstruction.

To enhance the practicality of RDHEI, Zhang *et al.* [43] introduced the first separable method. This methods allows a data hider to compress the LSBs of an encrypted image using a data-hiding key, thereby creating sparse space for embedding additional data. A receiver possessing the data-hiding key can extract this additional data without needing knowledge of the image content. Conversely, a receiver with the image encryption key can reconstruct the image. However, the resembling the image but lacks the ability to extract orignal image. Wu and Sun [44] introduced joint and separable RDHEI methods, embedding data in a chessboard

pattern to segregate pixel sets for secret data carriage. This innovation offered flexibility in data extraction and image decryption, albeit with challenges in ensuring perfect image recovery. Liao [45] addressed this reversibility issue by developing an accurate function to determine image block complexity, reducing extraction error rates. The hidden bits can be estimated using the calculated complexity, greatly improving reversibility potential. Khanam *et al.* [46] further built on [45] by implementing a more precise complexity estimation method, striving for zero extraction error. Similarly, Qian *et al.* [47] also introduced a technique aimed at enhancing image quality through progressive recovery, in which the data hider separates the encrypted image into three channels. Within each channel, varying amounts of additional bits are embedded, resulting in a marked encrypted image. This method significantly reduces distortion in the recovered image, albeit at the cost of a limited EC.

Furthermore, Qian and Zhang [48] introduced a separable VRAE method, wherein the original image is encrypted by the content owner using a stream cipher. Subsequently, the data hider divides the image into four sub-images and compresses selected bits from the encrypted image using Slepian-Wolf encoding with Low-Density Parity Check (LDPC) codes to create space for embedding secret data. This approach ensures the decoding phase is completely separable. One unaltered sub-image is decoded and upsampled through bilinear interpolation to serve as a reference for reconstructing the marked image in the clear domain. Additionally, to recover the original image, LDPC codes are decoded using a sum-product algorithm. While this method increases EC, it also introduces higher distortion to the image. [49] introduced a RDHEI technique that similarly utilizes additive modulo 256 for encryption and employs the property of mean values for embedding data within the encrypted image. This method ensures the preservation of mean values, facilitating efficient watermark extraction and image recovery. It achieves complete reversibility, is computationally straightforward, and operates with significantly low computational time. However, this approach is characterized by a low EC. Liu and Pun [50] proposed a novel approach by focusing on redundant space transfer to facilitate efficient data embedding, diverging from previous methods with limited embedding rates. Conversely, the conversion from the original image's MSB to the Least Significant Bit (LSB) of the encrypted image simplified the decryption task for adversaries, offering only six outcomes. Qin *et al.* [51] improved upon this by incorporating bit plane disordering, block and pixel scrambling, alongside redundancy transfer and sparse block encoding, to enhance rate-distortion performance, despite the high overhead due to block diversity. Prediction-error expansion methods for data embedding often face overflow/underflow issues, necessitating additional information transmission and potentially exposing encrypted images to security risks. Therefore, Ren *et al.* [52] proposed RDHEI method addresses these issues by integrating embedding with re-encryption, utilizing a permutation ordered binary number system.

Unlike traditional Prediction Error Expansion (PEE) approaches that alternate between peak and zero points, our method employs dual peak points without the need for shifting, enhancing both security and efficiency, but limit in EC. To enhance EC, Tang *et al.* [53] introduced

a method using differential compression to exploit special correlations within blocks during encryption, utilizing this space for data embedding. Similarly, Fu *et al.* [54] enhanced data embedding by analyzing the distribution of the MSB layers, identifying blocks suitable for embedding, and generating auxiliary data. They then created space for data by adaptively compressing the MSB layers of these blocks based on their occurrence frequencies. However, both methods exhibit limitations in the strength of their encryption techniques. Bhardwaj *et al.* [55] extended the research by employing a symmetric crypto-system to improve reversibility and EC. Malik *et al.* [56] introduced a high-capacity RDHEI scheme using chaotic encryption based on PWL-memristor and multi-layer embedding, demonstrating robustness against attacks but facing challenges related to auxiliary information overhead. Chen [57], in the same year, aimed at compressing pixel differences within encrypted blocks to vacate space for data embedding, achieving higher capacity and visual quality, yet struggling in complex image scenarios.

Yu *et al.* [58] developed a method using adaptive difference recovery, preserving spatial redundancy within encrypted blocks for efficient data hiding, constrained by a fixed pixel scanning pattern. Gao *et al.* [59] opted for an adaptive block-type labeling strategy, leveraging image compression to create spare room for embedding, with a somewhat narrow focus on the similarity of upper block planes. Yang *et al.* [60] presented a novel error-preserving encryption method (DEP)-RDHEI in 2022. The DPE-RDHEI aims to replace natural images with encrypted images embedded using the popular RDH technique called Pixel Value Ordering (PVO). Yang *et al.* use error-preserving encryption to maintain the correlation between neighboring encrypted pixels. The PVO-based embedding technique exploits this correlation to hide secret data, achieving high EC while preserving reversibility. Anushiadevi *et al.* [61] proposed a new RDHEI method that first encrypts each pixel and embeds data in the pixel error histogram. The Embedding Rate (ER) of 0.0807 bpp is achieved by this method, while maintaining complete reversibility. However, these techniques often have a limited EC. Wang *et al.* [32] introduced a high-capacity RDHEI method leveraging popular RDH techniques like PVO and Histogram Shifting (HS) for embedding secret data. The method utilizes block-level encryption to encode the original image content while preserving the correlation, and then embeds secret data using popular RDH techniques like PVO and HS, achieving high EC and ensuring reversibility. However, the privacy of the original image content still poses a concern. Rai *et al.* [62] introduced a two-pass pixel value ordering strategy, aiming for high EC and reversibility, though facing challenges in encryption strength and EC. In the same line, Han and Guohua [63] proposed a method focusing on redundancy and block similarity to improve embedding, albeit with weaker encryption methods, illustrating the continuous effort to balance encryption strength, EC, and reversibility in RDHEI methods.

The growth and advancement of VRAE-RDHEI methods have been driven by the continuous pursuit of improving EC, encryption strength, and reversibility. Table 2.3 presents an overview of notable VRAE methods, highlighting their principal mechanisms, decoding types, average EC, reversibility, and limitations. This comprehensive table serves as a valuable refer-

Table 2.3: Overview of notable VRAE methods of RDHEI

Method	Year	Principal mechanisms	Decoding Type	Average EC	Reversibility	Limitations
[50]	2018	Redundant space transfer from the original to the encrypted image to facilitate easy and efficient data embedding.	Separable	1.6	No	Limited configurations for encrypted image blocks.
[51]	2019	Encryption includes bit plane disordering, block and pixel scrambling, with redundancy transfer and sparse block encoding for efficient embedding.	Separable	1.7	Yes	High overhead of label map due to block diversity.
[56]	2020	Uses chaotic encryption based on PWL-memristor and multi-layer embedding for high-capacity RDHEI.	Joint	1.2	No	High overhead from auxiliary information, leading to lower EC.
[57]	2020	Compression of pixel differences in blocks using block-level encryption to create room for data embedding.	Separable	1.56	No	Worse performance in complex images due to pixel difference distribution.
[64]	2021	Block-based adaptive MSB encoding technique for enhanced security, embedding rate, and visual quality.	Separable	2.78	Yes	Limitation in considering bit-plane similarity after encryption.
[58]	2022	Adaptive difference recovery in encrypted blocks preserves spatial redundancy for efficient data hiding.	Separable	3.15	Yes	Fixed pixel scanning pattern in blocks may limit embedding flexibility.
[59]	2022	Analyzes image blocks to adaptively decide block-type labeling, compressing the image for spare room creation.	Separable	2.25	Yes	Focuses only on the similarity of upper block planes.
[60]	2022	Combines histogram shifting and difference expansion in small blocks while preserving pixel differences.	Separable	0.12	Yes	Very low EC due to the method's constraints.
[32]	2023	Utilizes pixel-value-ordering and histogram shifting in blocks for high-capacity embedding in IIoT security.	Joint	0.47	Yes	Low EC despite improvements.
[62]	2023	Employs a novel two-pass pixel value ordering strategy for embedding, aiming for high capacity and reversibility.	Separable	0.34	Yes	Lower EC compared to potential and weaker encryption methods.
[63]	2023	Uses redundancy in small blocks and employs two compression methods based on block similarity.	Joint	2.56	Yes	Weaker encryption, focusing on redundancy and similarity.

ence, encapsulating the key aspects of these methods and their respective trade-offs.

2.2.2 Growth of RRBE-RDHEI Methods

The evolution of RRBE-RDHEI has been marked by significant method advancements, beginning with the work by Ma *et al.* [28], which first categorizing image blocks into textured and homogeneous groups, then embedding the LSB plane of textured blocks into homogeneous ones using histogram shifting. This process creates space for the secret message, which is embedded into the textured blocks' LSBs. The image is then encrypted using a stream cipher, with EC noted in the LSBs of the initial textured block pixels, allowing straightforward message insertion. This approach, however, did not guarantee perfect image reconstruction. Progressing from this, [65] integrated the Paillier cryptosystem to strike a more favorable balance between EC and image distortion, although the method encountered challenges with large payloads. [66] furthered this progression by proposed a new RDHEI method, this method divides the original

image into patches, using sparse encoding for representation. Patches with minimal residual errors are selected for data hiding through their sparse coefficients. Residual errors from other patches are hidden using a conventional algorithm, followed by stream encryption for security. The secret message is then segmented and embedded into the prepared spaces. Decoding is separable and reversible, enabling the original image's lossless reconstruction from the unembedded patches' residual errors and the retrieval of the secret message. However, the method fails in lossless data extraction and image recovery like [28, 65], if the amount of hidden data is large.

To address the concern of deteriorated image quality, Yi *et al.* [67] discused an RDHEI method with binary block embedding. The method introduces the usage of the MSB-planes for embedding the secret information seeing the prevalent high spatial correlation among the MSB-planes of neighboring pixels. The shift from LSB to MSB resulted in a substantial enhancement of the EC compared to previous RDHEI methods. However, the method does not guarantee optimal performance in all scenarios as it only explores the local correlation. Borse *et al.* [68] developed a RDHEI method using the Discrete Wavelet Transform, which captures frequency and location information efficiently. The method employs patches over pixels for greater hiding capacity and uses sparse coding to approximate and embed significant residual errors within the image, thus avoiding data loss. Additionally, by embedding a learned dictionary into the encrypted image resulting in an approximate 9.18 percent increase in PSNR as compare to [66]. However, does not guaranty the full reconstruction of image. Xiao *et al.* [69] adapted the PVO technique to embed data within homomorphically encrypted images, generating a histogram of pixel differences that matches the clear domain's zero-centered Laplacian distribution. This method divides the image into 2x2 blocks, rearranges pixels in ascending order, and embeds data in the extreme values. It maintains pixel order, adjusting the histogram for data hiding but is limited to 0.2 bpp due to over/underflow risks. To address this, a location map excludes unusable blocks, affecting the payload.

Subsequently, Puteaux *et al.* [70] extended the MSB-based embedding work by introducing a prediction method to further increase the EC. In fact, Puteaux *et al.* [70] discused two RDHEI methods: corrected predicted error (CPE) and embedded prediction error (EPE). The first method initially calculates the Prediction Errors (PEs) and then pre-processes the original image to conceal all of them. Afterward, the pre-processed image is encrypted and the secret message is embedded by replacing MSBs with the bits of the secret message. Thus, the total ER of the marked encrypted image can reach 1 bpp. However, complete reversibility of the original image is not ensured. In contrast, the second method i.e., EPE first encrypts the original image and then embeds the PEs in the encrypted image followed by embedding of secret information using the MSB replacement strategy. Thus, reversibility of the original image is ensured with reduced total ER size i.e., less than 1 bpp. Puyang *et al.* [71] extended the Puteaux *et al.*'s EPE [70] by leveraging neighborhood correlation and additionally utilizing the second MSB for data embedding. Therefore, the ER of 1.35 bpp is achieved. However,

the method is dependent on the redundancy of only two MSB-planes for embedding which results in limited EC. On similar lines to incorporate secret information in several MSBs and embed additional data, Mohammadi *et al.* [72] introduced a local difference predictor-based RDHEI method. This approach involves dividing the cover image into blocks, with a specific pixel within each block used for predicting the other pixels in the same block. The maximum Absolute Prediction Error (APE) of each block is used to generate a block label, which, in turn, determines the block's EC. As a result, a substantial amount of data, up to 2 bpp, can be embedded into the image. Shiu *et al.* [73] introduced a high-payload RDH method for encrypted images using an enhanced Absolute Moment Block Truncation Coding (AMBTC) technique. This approach divides the image into 4x4 blocks, classified as 'hiding' or 'saving' based on their variance. Lower variance blocks are used for data hiding, while higher variance blocks are marked and reserved. The method involves computing average and quantization values for each hiding block, creating a partial bitmap for data representation. It employs Huffman encoding to compress the differences between original and modified blocks, integrating these with secret data for encryption. This RDH method supports separable image recovery and data extraction, catering to different types of receivers based on the keys they possess. However, these method primarily focuses on smaller blocks and doesn't fully exploit the spatial correlation of the entire image. However, the method fails to efficiently utilize the correlation of bit-planes. For this, Yu *et al.* [74] proposed an RDHEI method based on hierarchical embedding which is basically an extension of Mohammadi *et al.* [72]. The method firstly generates a hierarchical label map based on a prediction technique. Next, the label map is split into three categories based on the magnitude of PEs. The smallest and largest magnitude PEs are then used for embedding the secret information. Thus, embedding performance is somewhat improved over the SOTA high capacity RDHEI methods, but the computational cost increases. Hua *et al.* [75] expanded upon the methodology introduced by Yu *et al.* [74], by investigating the potential for increased embeddable bits within the LSB plane, employing a block-wise approach. This development was aimed at improving EC and data synchronization in encrypted images. Nonetheless, the method overlooks the interdependence among bit-planes, a factor that could potentially restrict further exploration of redundancy.

Differing from the aforementioned methods, Yin *et al.* [76] predicated on the observation that the MSBs of the predicted value and the original value have a high degree of similarity and introduce a method which generates a label-map to identify embeddable MSBs which are then compressed with the Huffman encoding and in turn hidden within the encrypted image. Thus, the method is able to further increase the EC but also fails miserably in a scenario, where the gap between the predicted and the original values is small. Gao *et al.* [77] introduced an expanded iteration of Yin *et al.*'s work [76]. In this version, the determination of image geometries precedes the generation of label maps. Among the four available image geometries, the selection is done based on their capacity for high-level data embedding. Subsequently, a refined variant of Huffman compression is employed to compress the resulting label map, ensuring greater pre-

cision in the process. However, it is noteworthy that the EC of this particular method remains nearly equivalent to that of [76], primarily due to the fact that it does not comprehensively address the inherent limitations of the approach. To enhance EC, Puteaux *et al.* [78] introduced a recursive approach that compresses labels from the MSB to the LSB after determining predicted values in a recursive manner. While this method successfully increases the EC, it does so at the cost of significantly higher time consumption. Similarly, Gao *et al.* [29] employed multilinear regression prediction for pixel estimation and manage the absolute PEs using two levels of label-maps. Although accurate PEs can enhance the EC, the regression method notably prolongs the processing time. Li *et al.* [79] take a unique approach by employing the LSB plane for lossless data embedding. Their method begins by vacating the MSB plane through the use of a MED predictor and subsequently generates a label map for MSB. Subsequently, a bitwise XOR operation is applied to the remaining bit-plane to transfer the vacated space to the LSB plane. Nevertheless, akin to Puteaux *et al.* [70], this method utilized a single plane for embedding, resulting in a comparable EC. Sui *et al.* [80] followed a similar path by segregating the image into MSB and the LSB planes. The MSB plane is directly predicted based on neighbouring pixels, while the remaining LSB planes are treated as a seven-bit plane image and processed akin to Yin *et al.* [76]. Initially, a tag is assigned to each pixel after evaluating the prediction error, and Huffman coding is applied to compress these tags, reducing the additional information overhead. Liu *et al.* [81] discussed a new RDHEI method using quad-tree to compress the bit-plane in different size blocks by assigning the code to each block with the help of four components, namely depth coding, value coding, number coding, and path coding. Thus, the method is able to save more space inside the cover image for embedding. Similarly, Ping *et al.* [82] adopted an analogous encoding mechanism, enhancing EC through the integration of a CNN-based predictor. Nonetheless, the limitations identified in [76] method persist in [82].

Yi *et al.* [83] developed a method utilizing a PBTL, which preserves the spatial correlation between pixels from the clear to the encrypted domain within small blocks. This approach uses selected pixels as reference points to calculate prediction errors, which are then emphasized via the binary tree's labeling. For message embedding, the spatial correlation is leveraged to insert secret message bits through substitution, achieving a payload of approximately 2 bpp. To enhance the ER, Wu *et al.* [85] expanded this technique by computing binary tree labels for the entire image, thus fully utilizing the image's spatial correlations and significantly increasing the ER. However, this method faces limitations in overhead compression efficiency. Chang *et al.* [84] discused a new RDHEI method using Block-based MSB Plane Rearrangement (BMPR) strategy. The BMPR adapts the BBC approach and provides a highly compressible bit-stream by rearranging MSB planes in blocks, which are then compressed using Extended Run-Length Encoding (ERLE), to exploit the high redundancy of MSB planes. Advancing the work of [67, 84] methods and along with additional significant contributions, Fu *et al.* [89], applied a modified BBC strategy specifically designed for lower bit-planes, separating and encoding the bit-planes accordingly. However, this method encountered overflow conditions similar to those

Table 2.4: Overview of notable RRBE methods of RDHEI

Method	Year	Principal Mechanisms	Decoding Type	Average EC	Reversibility	Limitations
[70]	2018	Utilizes MED to compute prediction errors for MSB embedding. Flags are introduced for identifying non-embeddable bits.	Separable	0.96	Yes	Lack of synchronization between image and data content, and faces data recovery risks due to flag usage. Limited exploration of complete image redundancy.
[71]	2018	Employs neighborhood correlation for embedding in two MSBs, enhancing capacity while attempting to maintain image integrity.	Separable	1.52	Yes	Focuses only on two MSBs, neglecting potential redundancy in remaining bit-planes and challenges in synchronization.
[83]	2019	Introduces Parametric Binary Tree Labeling (PBTL) to exploit spatial correlations within small blocks for redundancy exploration.	Separable	2.26	Yes	Limited by edge effects and inability to fully exploit image redundancy, particularly in small regions.
[84]	2019	Implements block-based MSB plane rearrangement and compression via Run Length Encoding, targeting MSB to LSB redundancy.	Separable	2.28	Yes	Generates significant overhead due to uniform treatment of MSB to LSB planes, affecting efficiency.
[76]	2020	Predicts pixels using MED and compresses Label Map through Huffman encoding, from MSB to LSB comparison.	Separable	3.30	Yes	Struggles with embedding in scenarios of minimal pixel value differences, limiting data capacity.
[85]	2021	Advances PBTL to consider entire image redundancy, enhancing spatial correlation usage.	Separable	2.54	Yes	Ineffectively utilizes sparse regions for data embedding, overlooking potential embedding spaces.
[86]	2022	Classifies blocks of PEs into eight labels for Huffman compression, focusing on prediction error block categorization.	Separable	-	Yes	Overlooks bit-plane sparsity, missing opportunities for increased compression and embedding.
[74]	2022	Utilizes a hierarchical method based on bit-plane labeling, compressing with Huffman encoding for improved capacity.	Separable	3.57	Yes	Fails to consider intra-block correlation, missing out on redundancy exploitation within smaller regions.
[87]	2022	Adapts to block sparsity by introducing three variability levels in block treatment for enhanced embedding.	Separable	3.70	Yes	Confronts challenges in block variability and synchronization, limiting optimal data embedding.
[88]	2023	Segregates bit-planes into stages for variable sparsity exploration, aiming at optimized embedding.	Separable	3.74	Yes	Struggles with precise variability identification for sparsity, affecting embedding efficiency.
[89]	2024	Apply a modified Binary-Block Coding (BBC) strategy for LSB planes, aiming at synchronized and efficient embedding.	Separable	3.16	Yes	Encounters issues with overflow and synchronization, limiting EC and image integrity.

highlighted in [70], thus limiting the EC in smooth images. On the similar lines, Yin *et al.* [90] investigated MSB-plane redundancy with BBC and compresses the obtained bit-stream using ERLE. The method considerably expands the available EC. With a further refinement to the BBC strategy, Xu *et al.* [87] introduced hierarchical-block variable-length coding based RDHEI method which selects the variable block size for each bit-plane. For this, the method facilitates subdivision of the block into three hierarchical levels by examining the size with the purpose of obtaining the highly compressed bit-stream. The sub-division in the varied size blocks greatly help in bit-plane compression because of their variable sparsity, thereby significantly improving the EC. Regrettably, the method faces a low probability of accurately identifying the ideal block size for each bit-plane, as it only investigates identical layers of sub-blocking. Contrary to [87],

Yao *et al.* [88] focused on discerning similarities within the LSB-planes of block, eschewing exploration of varied levels. This method leverages the lower LSB-plane for embedding, which aids in EC enlargement. However, the method does not consider the redundancy present among the bit-planes, which would have further increased the EC.

Advancements in RRBE-RDHEI methods have been notably significant. Researcher have utilized various approaches to expand the limits of EC and enhance image reconstruction accuracy to optimize the use of redundancy. Table 2.4 provide a detailed summary of very important RRBE methods, outlining their core mechanisms, types of decoding, average EC, the capability for reversibility, and their inherent limitations. The table serves as an resource, effectively summarizing the fundamental characteristics and trade-offs of each method.

2.3 Summary

In summary, the domain of RDHEI has experienced significant growth and innovation, largely driven by the contributions of VRAE and RRBE methods. These advancements have notably enhanced secure data embedding and image restoration techniques. Despite considerable progress in improving EC, encryption strength, reversibility, and redundancy exploitation, the challenge of achieving an ideal balance among these critical factors persists. Some methods excel in embedding capacity but fall short in quality and security, or vice versa, highlighting the need for continued research and development.

As concerns about data security and privacy escalate across various domains, developing RDHEI methods that seamlessly integrate high embedding capacities, robust encryption, efficient redundancy exploitation, and lossless reversibility remains a crucial objective. Researchers have the opportunity to address the limitations of existing methods and push the boundaries of RDHEI, thereby tackling the evolving challenges of data security and privacy in an increasingly interconnected world.

CHAPTER 3

LINK CHAIN DRIVEN REVERSIBLE DATA HIDING IN ENCRYPTED IMAGES FOR HIGH PAYLOAD

Based on the analysis of the related literature, it has been observed that the high embedding capacity is usually achieved by block wise prediction-based methods as the prediction errors can be encoded to create space which can be later used for embedding. However, due to the fixed size blocks, the complete neighbourhood correlation has not been fully utilized which leads to non-accurate prediction sometimes. To address this problem, a new high-capacity data hiding method which tries to exploit the neighbourhood correlation, is proposed. The proposed method basically generates variable size link chains of highly correlated pixels and represents those pixels by one or two values i.e., their average or two quantization levels, respectively, based on the size of link chains. Thus, a large room inside the image is created which is used to embed the secret information.

To set the base for easy comprehension of the proposed work, the next section provides a detailed review of some of the most closely related data-hiding methods in the encrypted domain.

3.1 Review of Related Works

In this section, review of the working of two most closely related works i.e., Yin *et al.* [76] and Shiu *et al.* [73] in the field of RDHEI, is presented. All of these aforementioned works belong to RRBE category and provide good EC. In the presented review, the complete working process of Adaptive Multi-MSB Prediction and Huffman Coding (AM2PHC) followed by Interpolative Absolute Moment Block Truncation Coding (IAMBTC) based RDHEI methods, is precisely explained as follows.

3.1.1 Adaptive Multi-MSB Prediction and Huffman Coding

Yin *et al.* [76] introduced a new RRBE method based on AM2PHC for RDHEI. AM2PHC-RDHEI is a completely lossless method which makes use of prediction strategy for reserving the room inside the image and provides high EC. For this, Yin *et al.* employs the MED predictor used in the LOCO-I algorithm of the JPEG-LS compression for prediction of the pixel values of the original image I of size $M \times N$ pixels. The pixel located at (i, j) is predicted using MED predictor based on three closest causal pixels (i.e., $X = I_{(i-1,j)}$,$Y = I_{(i,j-1)}$ and $Z = I_{(i-1,j-1)}$) as follow:

$$p_{(i,j)} = \begin{cases} \max(X, Y), & \text{if } Z \leq \min(X, Y) \\ \min(X, Y), & \text{if } Z \geq \max(X, Y) \\ X + Y - Z, & \text{otherwise} \end{cases} \tag{3.1}$$

where $2 \leq i \leq M, 2 \leq j \leq N$ and $p_{(i,j)}$ is the predicted value of $I_{(i,j)}$. Next, both the original and predicted pixel values are converted into 8-bits binary representation and compared in bitwise manner from MSB to LSB until a non-match bit is found. Then, a label that is equal to the number of matching bits (till the first non-matching) is created for the pixel. Therefore, the label value for embeddable pixels can vary from 0 to 8, where 0 represents the first MSB itself is different and 8 represents all the bits are matching, thus, total 9 values. The process of prediction and label generation is repeated for the entire image to get a label map, which contains label (t) for each pixel of the image. The generated label map is then encoded using Huffman method which generally results into the following Huffman code i.e., 00,01,100,101,1100,1101,1110,11110,11111 for the label map values based on their frequency. Usually, the least frequent value is represented by maximum bits and vice-versa. Thus, an Encoded Label Map (ELM) in the form of binary stream is obtained. Afterwards, the original image I is encrypted by performing a bit-wise exclusive-or operation on every bit of the original image using an encryption key. The encryption process is same as the one to be discussed in the proposed work. Next, the ELM is embedded in the encrypted image to generate space inside the image. Before embedding the secret data, Huffman Coding Rules (HCR), length of ELM and other Partial Auxiliary Information (PAI) which includes ELM and original reference pixel's values is embedded. This embedding is done in the certain pixels (also knowns as reference pixels) of the first row and the first columns by replacing their values. The partial embedding of auxiliary is done because it is not possible to replace all reference pixels in their original locations due to the size, separability and reversibility constraints. The value of t is used to guide the embedding process which basically enables embedding of t+1 bits from MSB onwards except for the case of t=8, where all the bits of the pixel can be embedded. After embedding of HCR and PAI, the embedding of remaining Auxiliary Information (AI) and bits of secret data is done by data hider using the same embedding process defined above and final marked encrypted image is obtained.

The marked encrypted image is obtained by the receiver who can the extract the hidden information and restore the image based on the availability of only data key, only encryption key or both. In all three cases, first the HCR and AI are extracted. If the data receiver has only the data key, the encrypted data is extracted from the similar position as embedded during hiding, and then decrypted to obtain lossless data. In case the receiver has only encryption key, after extracting the HCR, Length of ELM (LELM) and reference pixels, the image is decrypted using the same process of encryption. Although, the image reconstruction process is done in raster scan manner and pixel values are predicted using Eq. (3.1), with the exception of the first row and column. And the appropriate predicted value bits are used to get the original pixel values. Following the completion of this replacement, a lossless image is obtained. In case both hiding and encryption keys are available with the same receiver, the above defined process of extraction and recovery can be executed in a sequential manner to get both hidden data and recovered original image.

3.1.2 Shiu *et al.*'s IAMBTC

Shiu *et al.* [73] discussed an IAMBTC based high-payload RDHEI method. IAMBTC technique is an extended version of AMBTC approach given by Lema et. al. [91] in 1984. Shiu *et al.*'s method first divides the image into 4×4 blocks into two categories namely hiding and saving based on their variance. If the variance is less than a user-defined threshold then the block is a hiding block, otherwise, it is a saving block. In Fig. 3.1, block 3.1a and 3.1k represents hiding and saving blocks respectively. Next, the saving blocks are marked at the 4^{th} pixel LSB to discriminate them from the hiding block as shown in block 3.1l of Fig. 3.1 and the hiding blocks are encoded using IAMBTC. For this, an average of each hiding block is calculated using Eq. (3.2).

$$H = \left\lfloor \frac{1}{n} \sum_{i=1}^{n} x_i \right\rfloor, \tag{3.2}$$

where n represents the number of pixels in a block and x_i is the intensity of the i^{th} pixel. Next, two quantization values (H^l and H^h) are computed using Eq. (3.3) and (3.4), respectively.

$$H^l = \left\lfloor \frac{1}{p} \sum_{i=1}^{n} x_i \right\rfloor, \quad \text{if } x_i \leq h. \tag{3.3}$$

$$H^h = \left\lfloor \frac{1}{n-p} \sum_{i=1}^{n} x_i \right\rfloor, \quad \text{if } x_i > h \tag{3.4}$$

where p denotes the count of pixels having value less than H and '$\approx$' represents assignment to the nearest whole number. H^l and H^h represent low and high quantization values, respectively, as shown in block b of Fig. 3.1. Additionally, a bitmap is also created which represents

each pixel of the block by a bit which is '1' if the pixel value is greater than H otherwise it is '0', as shown in block c of Fig. 3.1. To further reduce the size, every alternate (even) bit of the bitmap is dropped as shown in block d of Fig. 3.1 to get a partial bitmap (B^M). Thus, the hiding block is represented by trio i.e., $\{H^l, H^h, B^M\}$. To reconstruct the block, odd numbered pixels can be reconstructed using the following Eq. (3.5), as shown in block e of Fig. 3.1.

$$\bar{p_i} = \begin{cases} H^l, & \text{if } B_i^M = 0 \\ H^h, & \text{if } B_i^M = 1 \end{cases} \tag{3.5}$$

where $\bar{p_i}$ is the decoded intensity of the i^{th} pixel of the block. For even numbered pixels, some pre-defined interpolation equations are applied using the reconstructed odd pixel values. Thus, the complete hiding block is estimated as shown in Fig. 3.1f. In the case of hiding blocks as well, the 4^{th} pixel LSB is also marked to discriminate them from the saving block as shown in Fig.3.1g. Now, the difference between the original and new blocks is calculated as shown in Fig. 3.1h and 3.1m for the hiding and saving block, respectively. Following that, the Huffman encoding is applied to the differences of all blocks to condense their size, resulting in the Huffman codes and dictionary as shown in Fig. 3.1n, which are then coupled with the encrypted secret data as depicted in Fig 3.1o. On the other end, IAMBTC trios are stored into the first three pixels. Afterwards, combined blocks are encrypted using a stream cipher with the assistance of a pseudo-random generator and an encryption key, except the first MSB of the 4^{th} pixel of each block, as shown in block i and block p of Fig. 3.1. Finally, the remaining space of hiding blocks is used for embedding the Huffman code, directory, and secret data, and an encrypted image with hidden secret data is obtained which is also known as a marked encrypted image.

For image recovery and data extraction, the reverse of the embedding process is performed, by the receivers. However, there may be three types of receivers. One who has only a data hiding key, he/she can only extract the Huffman code, Huffman directory and secret data and in case the receiver merely has an encryption key, he/she can distinguish between the hiding and saving blocks. Only the first three pixels are decrypted from the hiding block, which contains the trio $\{H^l, H^h, B^M\}$. The trio then can be used to predict all the pixels with the help of interpolation formulas defined in [76]. Saving blocks, on the other hand, are directly decrypted and an image is obtained. In case a receiver has both keys, then he/she can extract the hidden information and can also restore the image using the same process defined above.

Both the works i.e., [76] and [73] are completely lossless and provide decent EC. Basically, most of the existing works which include AM2PHC [76] and IAMBTC [73] either use linear prediction strategy or use block compression to exploit spatial correlation and create a large room inside the image for embedding the secret information. However, none of the existing works seems to make use of variable-size blocks/groups or chains based on the prevalent spatial correlation to make its optimal benefit. So, in this chapter, we propose a new reversible data-

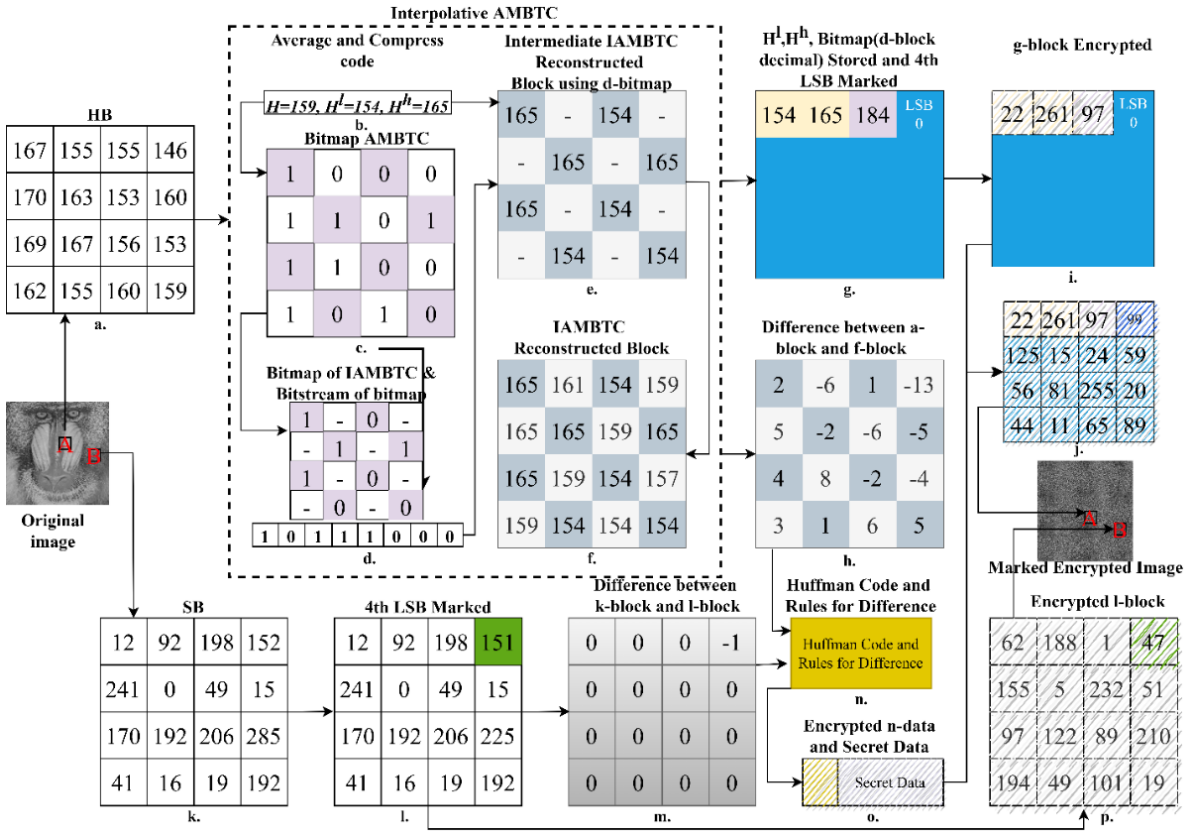


Fig. 3.1: An example of pre-processing, marking, and data hiding of the IAMBTC-RDHEI method

hiding method for encrypted images based on a link chain strategy. The proposed method generates different size and shape link chains based on the prevalent spatial correlation and helps the data hider to embed large amounts of secret data. In the next section, the proposed LCD-RDHEI method is described in detail.

3.2 Proposed Method

This section presents the proposed method which can be seen with the purviews of content owner, data hider and recipient as follows. In the purview of content owner, operations such as image pre-processing, encryption of image, and process of Supporting Information (SI) embedding are done; and in the purview of data hider, operations such as extraction of Process Information (PI), extraction of directory of link chain and embedding of the secret data are implemented; and in the purview of the recipient, operations such as extraction of the data and image decoding either separately or combined subject to the availability of hiding keys are performed. For clear representation of different purviews and operations there-in, Fig. 3.2 is provided that illustrates the framework of the proposed LCD-RDHEI.

As shown in Fig. 3.2, the content owner's major work is to encode the original image I by first pre-processing the image to create room inside for embedding the secret data. In the pre-processing, SI is generated which is embedded inside the image so that subsequent users,

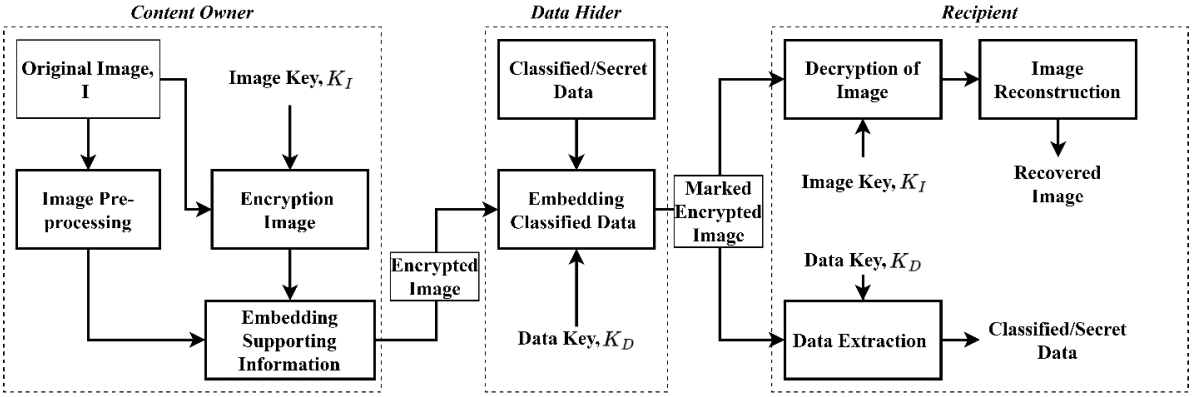


Fig. 3.2: Framework of the proposed LCD-RDHEI method

i.e., data hiders, can extract it to find the created room for embedding. Afterwards, the image is encrypted using an image key K_I to get an encrypted image I_e which is forwarded by the content owner to the data hider after SI embedding ($I_{e'}$), who may be present anywhere in the network. Now, the data hider identifies the pixels from $I_{e'}$ for embedding secret information that can be encrypted using a data key K_D. Thus, a marked encrypted image I_m is obtained which can be forwarded to receivers, who can extract the secret message and recover the image I_r according to the availability of the keys, i.e., K_D and K_I. In the next sub-section, the working process in the content owner's horizon is described in detail.

3.2.1 In Content Owner's Horizon

The content owner has three functions which include image pre-processing, encryption of the image, and process of SI embedding, which are defined as follows:

A. Image Pre-processing

The original image I of size $M \times N$ is scanned in a raster scan manner and pixels are marked by their sequence number (S) of scan instead of their coordinates. The sequence number of a pixel $I_{(i,j)}$ will be $((i-1) \cdot N + j)$, where i and j represent row and column numbers, respectively. For example, the pixel $I_{(3,2)}$ of the 512×512 size image can be represented with its sequence number i.e., $((3-1) \cdot 512 + 2) = 1026$, as I_{1026}. Next, the content owner processes I to form highly correlated groups of pixels. The groups are hereafter called link chains (l_c) as these are also formed externally by making a link between consecutive pixels of the group. The pixels of the link chain are also called nodes. The first node of the link chain, which is also known as the link chain head (h_{lc}), will have the sequence number of the corresponding pixel along with the direction to the subsequent node and the rest of the nodes only contain the direction to their next node. It is to be noted that the link chains can be of any size and shape but will have pixels of very similar intensity or high spatial correlation. Thus, by doing this, a room can be created in the image for embedding the secret information. Next, processing of the untouched pixels

which have not been part of any of the aforementioned link chains is done so that the reserved room can further be extended. Thus, the image pre-processing is done in two phases i.e., a) link chain directory generation and b) untouched pixel identification and their prediction which are discussed as follows:

a) Link Chain Directory Generation: This phase has three steps which include i) link chain generation, ii) categorization of link chains, and iii) node representation in link chains, which are introduced below. For more clarity, the algorithm to create the directory of link chains is provided in sub-section iv.

i) Link Chain Generation: For generating the link chains (l_c), first of all, a list (L_i) is generated which stores all the pixels of I by their intensity value (val) and sequence number (S) in increasing order of their val (except the pixels reserved for (PI), which will be discussed later in Section 3.2.1). Now, the first pixel of the list (L_i) is chosen as a link chain head (h_{lc}) and then any adjacent pixel from I that can be either right, down, left, or up pixel, of the current pixel, is selected for adding the same into l_c, as per Fig. 3.3a. The adjacent pixel can only join l_c, if the intensity difference of the h_{lc} and the adjacent pixel is less than a given user-defined threshold (i.e., variability coefficient (V_c)). The variability coefficient (V_c) shows the maximum allowable intensity difference within a l_c. The addition of the adjacent pixel to the current link chain may lead to a situation where multiple l_c are possible at any stage because more than one adjacent pixel i.e., right, down, left or up, can satisfy the variability coefficient condition. Thus, it may lead to a situation similar to the Hamiltonian path problem [92]. Moreover, this problem is non-polynomial time solvable. So, it is not suitable for the content owner to find the maximum length l_c. To avoid the above situation, the proposed l_c generation method follows the greedy approach [93]. The proposed method uses many recursive calls to determine the maximum length (max_len) for any l_c. In the recursive calls, multiple virtual l_c are formed at the step of adding an adjacent pixel, but only one virtual l_c with the maximum length is added as final, and the others are dropped. As a result, a pixel can be a member of many virtual l_c, requiring length calculations multiple times for a single pixel. To avoid the scenario, this method assumes that the first computation of the length of the link chain from that pixel is perfect and that itself may be applied to any of the remaining virtual link chains from that pixel. Or, to put it another way, if virtual l_c generation has already been completed for any pixel, and any virtual l_c later reaches that pixel, that pixel is attached to the virtual l_c as the next node, and the already formed l_c is presumed to be the virtual l_c's tail. Each virtual l_c must be preserved as part of the aforementioned process, as it can be used immediately during an interdicted call. As a result, this method uses two intermediate lists to store the intermediate information for each virtual l_c, where the first list (Len) contains the max_len of the l_c from the current pixel $I_{(S)}$ and another list (Dir) stores the encoded form of direction of the next possible attached pixel that can be right (`00`), down (`01`), left (`10`), and up (`11`) obtained from the following Eq. (3.6).

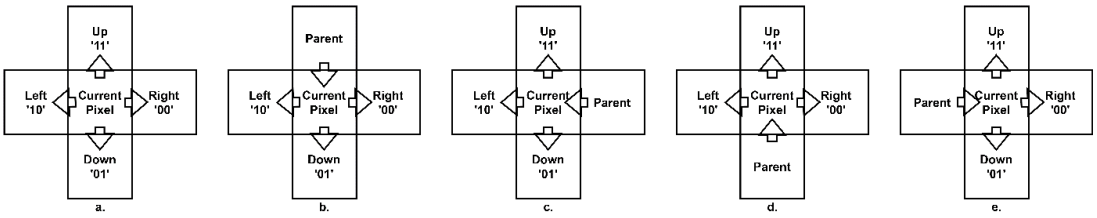


Fig. 3.3: An example of possible link movement for adjacent pixels (a) If the node is link chain head, (b) If the next node is down to the parent move, (c) If the next node is right to the parent move, (d) If the next node is up to the parent move, and (e) If next node is left to the parent move

$$\text{Dir}(S) = \begin{cases} 00, & \text{if } S^c = S + 1 \\ 01, & \text{if } S^c = S + N \\ 10, & \text{if } S^c = S - 1 \\ 11, & \text{if } S^c = S - N \end{cases} \tag{3.6}$$

where S^c is the sequence of the next node of child node to be added in the l_c. Initially, Len and Dir are set null for every new l_c generation to avoid incorrect overlapping with the already generated link chains. Following the determination of a virtual l_c for a pixel, max_len and Dir of the next pixel is updated in Len and Dir, respectively, corresponding to the sequence number S. It can be observed from the function-1: ***Max_Len_LC()*** that $Len(S)$ is checked from the Len before calculating the virtual l_c for a pixel $I_{(S)}$ as all the intermediate virtual l_c share a common intermediate list. Additionally, pixels that are part of (any of) the link chains are regarded to be part of a cover set (C_S), which is initially set to null. The pseudocode to determine the maximum length of a link chain for a pixel $I_{(S)}$ is given as function-1 (***Max_Len_LC***). It can be observed that maximum three recursive calls are possible in the above function ***Max_Len_LC*** for a pixel since one of the directions always points towards parent i.e., already a part of C_S, as illustrated in Fig. 3.3b-e. At last, the l_c is extracted from the intermediate lists as follows: first, a link chain head (h_{lc}) is constructed for the current pixel with sequence S, which also contains two bits of $Dir(S)$. The $Dir(S)$ gives the sequence number (S^c) of the next node (also called as child node) of the link chain as per the following Eq. (3.7).

$$S^c = \begin{cases} S + 1, & \text{if Dir}(S) = \text{‘00’} \\ S + N, & \text{if Dir}(S) = \text{‘01’} \\ S - 1, & \text{if Dir}(S) = \text{‘10’} \\ S - N, & \text{if Dir}(S) = \text{‘11’}. \end{cases} \tag{3.7}$$

Next, the same process of adding the next node in the link list by considering S^c as the sequence number of the current pixel, is continued until the $Dir(S)$ is Null, which represents the last pixel of the link chain. Finally, in the last node of the link chain, the encoded direction

to the penultimate node is updated using Eq. (3.8) which is denoted by $S^p_{\text{direction}}$ whereas the parent node is denoted as S^p.

$$S^p_{\text{direction}} = \begin{cases} \texttt{10}, & \text{if } S^c = S + 1 \\ \texttt{11}, & \text{if } S^c = S + N \\ \texttt{00}, & \text{if } S^c = S - 1 \\ \texttt{01}, & \text{if } S^c = S - N. \end{cases} \tag{3.8}$$

Thus processing, all the link chains from the image can be extracted. In the next sub-section, the categorization of the extracted link chains is discussed.

ii) Categorization of Link Chains: The generated link chains are now categorized in three categories jump, small, and large link chains, for the sake of achieving high EC and low distortion. Two user-defined thresholds i.e., T_1 and T_2 are used to separate them into the above suggested categories. More specifically, if the length max_{len} of any l_c is less than the first threshold T_1 then it can be considered as a jump l_c, which can be represented by l_{cj} and if the length is greater than or equals to T_1 and less than T_2, it is considered small l_c which can be represented by l_{cs}, otherwise, large l_c which can be represented by l_{cl}. Note that, the T_1 and T_2 are fixed, i.e., $T_1 = 3$ and $T_2 = 16$ (for the sake of simplicity) in this method. The proposed method does not mark the entries for jump link chains l_{cj} in the cover set C_s so that those pixels can be utilized for other link chains (if possible). Therefore, there are only small l_{cs} and large l_{cl} link chains, effectively. All the generated link chains are sorted in increasing order based on their head node's sequence number.

iii) Node Representation in Link Chains: Node representation in link chains is crucial for reserving bigger room. The pixel values of link chains are estimated either by their average values or by two quantization levels along with a bitmap, depending on the type of link chain. More specifically, the pixels of small link chains are estimated by their average (H) using Eq. (3.2), and pixels of large link chains are estimated by two quantization levels and a bitmap using IAMBTC, as defined in IAMBTC.

Firstly, H^l and H^h are calculated using Eqs. (3.3) and (3.4) for l_{cl}. To represent every node of l_{cl}, a process defined in Section 3.1.2 is utilized, and every alternate (in other words, even-positioned) pixel representation in the bitmap is dropped, to obtain the partial bitmap (B^M). Moreover, the proposed method stores the difference value (D^{hl}) calculated as $D^{hl} = H^h - H^l$ in place of H^h to further reduce the number of bits required to represent the trio of H^l, H^h, and B^M for representing the nodes of l_{cl}.

Since the D^{hl} can range between 0 to $V_c - 1$, fixed Huffman codes are derived to represent D^{hl}. It should be noted that if $V_c = 1$, then every l_{cl} is treated like l_{cs}, because all of the pixels have the same intensity in l_{cl}, and their average will also be the same, making the production of H^l, H^h, and B^M pointless and a waste of space. The derived Huffman codes are provided in Table 3.1, which can be used by the content owner, data hider, and recipients. Furthermore,

Table 3.1: Fixed bit-sequence for D^{hl} based on V_c, $D^{hl} = H^h - H^l$

V_c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1																		
2	0	1																
3	10	0	11															
4	111	10	0	110														
5	1111	110	0	10	1110													
6	11111	1110	10	0	110	11110												
7	111111	11110	110	0	10	1110	111110											
8	1111111	11110	1101	0	10	1100	1110	111110										
9	11111111	11110	1101	01	00	10	1100	1110	111110									
10	111111111	11110	1101	100	0	101	1100	1110	111110	1111110								
11	1111111111	11100	1100	100	01	00	101	1101	11101	11110	111110							
12	11111111111	111100	11100	1100	100	0	101	1101	11101	111101	111110	1111110						
13	111111111111	1111100	111100	11100	1100	10	0	1101	11101	111101	1111101	1111110	11111110					
14	1111111111111	111100	11100	1100	100	01	00	101	1101	11101	111101	1111100	1111101	1111110				
15	11111111111111	1111100	111100	11100	1100	00	10	01	1101	11101	111101	1111101	11111100	11111101	11111110			
16	111111111111111	111100	11100	1100	100	00	01	101	1101	11101	111101	1111101	1111100	11111100	11111101	11111110		
17	1111111111111111	1111100	111100	11100	1100	100	00	01	101	1101	11101	111101	1111101	11111100	11111101	11111110	111111110	
18	11111111111111111	11111100	1111100	111100	11100	1100	00	10	01	1101	11101	111101	1111101	11111101	111111100	111111101	111111110	1111111110

the binary representation of H for l_{cs} and H^l, D^{hl} & B^M for l_{cl}, are stored as a sequence of bits (H_{bs}) in the same order as the h_{lc}.

iv) Algorithm for Link Chain Directory Generation: The complete step-by-step algorithm for generating the directory of the link chain which consists of all the link chains, is defined as follows.

Algorithm 1 LINK-CHAIN DIRECTORY GENERATION ALGORITHM

Input:

I: Input image of size $M \times N$ pixels,
V_c: Variability Coefficient,
T_1, T_2: Threshold,
$C_s = null$: Cover set

Output:

DL_c: Directory of link chains,
H_bs: Bitstream of link chain representation information.

Step-1: Scan image (I) in raster scan manner and generate a list (Li) of pixels that includes the sequence number (S) and intensity (val) by excluding initial $\lceil (3 \times \lceil \log_2(M \times N) \rceil + 11)/6 \rceil$ pixels to be used for PI embedding.

Step-2: Call the recursive function Max_Len_LC($I, C_s, val(x), V_c, S(x), Len^*, Dir^*$) defined in Section-3.1.1, to get the max_len of the l_c, where Len and Dir are null lists, each of dimensions $M \times N$ for storing the intermediate lengths and the intermediate directions. The Len^* & Dir^* in the function Max_Len_LC implies that reference to Len & Dir, respectively, is being made by their address so that only single copy of the lists are maintained during execution of the recursive function.

Step-3: if $max\_len < T_1$, which means the l_c would be a jump chain, then go to step-7 .

Step-4: else the link chain (l_c) is going to be either a small or large l_c. In both the cases, first add the current pixel's sequence number (S) as the link chain head (h_{lc}), update C_S as $C_S \cup S$ to include the sequence S and direction to the next adjacent node given by $Dir(S)$. Next, do the followings:

- Update the sequence number of the next adjacent node of the link chain based on $Dir(S)$ and updated sequence is denoted as S^c which is calculated using following Eq. (3.6).
- If $Dir(S) \neq Null$, add the next node in the link chain in the form of 2-link bits as $Dir(S^c)$, Else, add the last node of the list in the form of direction towards parent node using following Eq. (3.7)
- Update C_S as $C_S \cup S^c$ to include the sequence S^c of the current pixel in the cover set C_S and also update the S as S^c.
- while $Dir(S) \neq Null$

Step-5: If $T_1 \leq max\_len < T_2$, which indicates the l_{cs}, then apply Eq. (3.2) to get the average (H) intensity of the corresponding pixels of link chain generated in step-4. Afterwards, record H with the h_{lc}.

Algorithm 1 LINK-CHAIN DIRECTORY GENERATION ALGORITHM (CONTINUED)

Step-6. elseif $max\_len \geq T_2$, which indicates the $l_c l$, then calculate H^l and H^h using Eqs. (3.3) and (3.4), respectively and bitmap (B^M) from the corresponding pixels of link chain generated in step-4. Next calculate D^{hl} and determine the bits of D^{hl} from Table 3.1. Afterwards, record $\{H^l, D^{hl}, B^M\}$ with the h_{lc}.

Step-7. Repeat steps 2–6 for all the elements of Li.

Step-8. Arrange the link chains in ascending order of their head's sequence number and store them in a directory of link chains (DL_c). The kth link chain from the DL_c is represented by l_c^k, where k belongs to $\{1, 2, 3, \ldots, n\}$ and n is the number of link chains.

Step-9. Process each link chain of DL_c to get a bitstream of link chain representation information (H_bs) that contains average (H) for small link chains, low quantization (H^l), quantization difference (D^{hl}), and bit map (B^M) for large link chains as follows:

- If $T_1 < max\_len(l_c^k) < T_2$ then $H_{bs} = H_{bs} \parallel l_c^k.H$
- else $H_{bs} = H_{bs} \parallel l_c^k.\{H^l, D^{hl}, B^M\}$, where $l_c^k.H$, $l_c^k.\{H^l, D^{hl}, B^M\}$ and $\parallel$ denote average value, IAMBTC triplet and concatenation operation, respectively.

Step-10: Stop and exit.

b) Untouched Pixel Identification and their Prediction: The pixels, that have not been covered in any of the link chains (or PI, which will be discussed later in subsection 3.2.1), are called untouched pixels ut_p. In the proposed work, untouched pixels are also being used for embedding the secret information in order to increase the EC of an image. For this, AM2PHC [76] (which has already been discussed in section 3.1.1) has been employed. Prediction for remaining pixels and generation of Huffman code and rules, on the other hand, is done in the same way as in AM2PHC [76].

B. Encryption of the Image

In view of the multitude of techniques available for image encryption, the AES-Counter Mode (CTR) [94] has emerged as one of the most reliable and popular open-source alternatives, to ensure the confidentiality of the image. In the proposed method for image encryption using AES-CTR, the process commences with the generation of a pseudo-random matrix (ξ) of size $M \times N$ utilizing the image key (K_I) as the seed. Then, the image is encrypted using Eq. (3.9) and link chains are encrypted using Eq. (3.10), as follows.

$$I_{e(i,j)} = I_{(i,j)} \oplus \xi_{(i,j)}, \tag{3.9}$$

where $\oplus$ denotes the bit-wise exclusive-or operation; $1 \leq i \leq M$, $1 \leq j \leq N$ and $I_{e(i,j)}$ is an encrypted pixel at coordinates i, j. After turning the l_e into a bit stream, 8 bits for each integer, H_{bs} is encrypted as follows:

$$\overline{H_{bs}}(k) = H_{bs}(k) \oplus l_e(i), \tag{3.10}$$

where $\overline{H_{bs}}$ represents the encrypted bit stream and $1 \leq k \leq$ size of H_{bs}. Although, m_e and l_e can also be populated using a chaotic generator or AES algorithm in output feedback mode.

C. Process of Supporting Information Embedding

SI is an additional information required by the receivers to extract the hidden secret data and decrypt the image. This information must be embedded in the image for the sake of hiding and extraction of secret data and decoding of the image. The details regarding SI and its embedding algorithm are presented in the following two subsections:

a) Details of Supporting Information: SI basically, consists of following information components:

- Variability coefficient (V_c): The variability coefficient shows the maximum intensity difference (possible) within a l_c. Since the pixel values in a grayscale image, ranges from 0-255, the maximum possible absolute difference between any two pixels can be 255 which can be represented by 8 binary bits. Therefore, 8 bits are fixed for storing the value of V_c in the present work.

- Location of the first link chain head d_l^1: The location of the first link chain head which is denoted by d_l^1, is represented in $\lceil log_2(M \times N) \rceil$bits.

- Greatest location difference (d_l^{max}) and location difference sequence (dl_{bs}): As mentioned in the preceding sub-section 3.1.1, the link chains are sorted based on the location sequence number of their head nodes in a directory of link chains (DL_c). To get the greatest location difference between any two consecutive link chains, the location difference $(d_l^{(k+1)}$ between every two consecutive heads i.e., $h_l c^k$ and $h_l c^{(k+1)}$, is calculated using following Eq. (3.11).

$$d_l^{(k+1)} = (h_{lc}^{(k+1)}.S - h_{lc}^{k}.S), \tag{3.11}$$

  where $h_{lc}^{k}.S$ and $h_{lc}^{(k+1)}.S$ represent the sequence number of head nodes of k^{th} and $(k+1)^{th}$ link chains, respectively. From the calculated difference sequences, the greatest location difference d_l^{max} is identified, and all the difference sequences are converted into $\lceil \log_2(d_l^{max}) \rceil$ bit binary and stored in dl_{bs}.

- Huffman code rules HCR: In the proposed work, a multi-MSB matching [73] based data hiding scheme has been used for embedding in the untouched pixels (ut_p) as described in 3.1.2. In the scheme, a label map is generated which can have 9 different values. For optimally storing the label map, Huffman encoding has been used as described in [76]. For the Huffman encoding, some rules have been devised as in [73] which can be represented in 32 bits.

- Encrypted link chain representation information : In subsection 3.2.1, a bitstream of link chain representation information (H_{bs}) that contains average (H) for small link chains, low quantization (H^l), quantization difference (D^{hl}), and bit map (B^M) for large link chains are generated. Afterward, the bitstream H_{bs} is encrypted in subsection 3.1.2 to get the encrypted bitstream of link chain representation information.

- Encoded label map (ELM): ELM is obtained form subsection 3.2.1.b for identification of untouched pixels and their prediction.

- Encrypted MSBs of Starting Pixels (EMSP): Six (6) MSBs of starting $\left\lceil \frac{3\times\lceil \log_2(M\times N)\rceil+11}{6} \right\rceil$ pixels of image I which are not part of any of the link chains are stored in EMSP. This information is required for recovering the image by the decoder and therefore needs to be recorded.

All of the above-defined information components (if concatenated together) represent SI.

b. SI Embedding Algorithm: As discussed in the preceding subsection, the SI consists of several information components. However, these information components have been grouped into two different sets, namely PI and AI, based on their requirement and for the sake of simplicity to employ the proposed method. The AI is majorly required by the hider for the embedding of the secret data and by the receivers as well during their process. In contrast, PI is required to initiate any of the phases such as hiding, decoding, and/or extraction. More specifically, AI consists of a concatenation of dl_{bs} (of size number of $n \times \lceil \log_2(d_l^{max}) \rceil$ bits), HCR (of size 32 bits), ELM, EMSP, and $\overline{H_{bs}}$, in the given order. The size of ELM, EMSP, and $\overline{H_{bs}}$ are dependent on multiple parameters such as the number of untouched pixels, PI, and the number of link chains and their type, etc. As far as PI is concerned, it is a concatenation of d_l^{max} (of size $\lceil \log_2(M \times N) \rceil$ bits), d_l^1 (of size $\lceil \log_2(M \times N) \rceil$ bits), V_c (of 8 bits), and Length of Auxiliary Information(AI) (LAI) (of size $\lceil \log_2(M \times N) \rceil$ bits) in the given order, where LAI represents the length of AI. Thus, the total size of PI is $(3 \times \lceil \log_2(M \times N) \rceil + 11)$ bits. The PI is basically embedded in the starting pixels of the encrypted image I_e in a raster scan manner by replacing 6 MSBs of each pixel with the PI bitstream. These pixels can be called PI-enabled pixels and their total count (NP_{PI}) is $\left\lceil \frac{3\times\lceil \log_2(M\times N)\rceil+11}{6} \right\rceil$. Next, AI is embedded in the 6 MSBs of each pixel, which are part of link chains. The algorithm-2 to embed SI, which includes both AI and PI, is provided as follows:

3.2.2 In the Purview of Data Hider

A data hider is a person who basically embeds the secret information inside a cover media. Here, the cover media is an encrypted image with SI i.e., $I_{e'}$. Before starting the embedding of secret information, the data hider first needs to extract PI, link chains, and ELM, respectively as defined in the following subsections. Subsequently, he/she performs the embedding of secret data in the image.

Algorithm 2 SUPPORTING INFORMATION EMBEDDING

Input:
  I_e: Encrypted image,
  DL_c: Directory of link chains,
  SI: Supporting information
Output:
  $I_{e'}$: Encrypted image with SI
Step-1: Replace 6 MSBs of first NP_{PI} pixels of I_e by 6 bits of PI in raster scan manner.
Step-2: Set $k = 1$;
Step-3: Obtain sequence number S using $h_{lc}^k.S$ for l_c^k.
Step-4: Set child node (S^c) of S as $S^c = S$.
Step-5: Do the followings:

- Set parent node (S^p)=S and $S = S^c$
- Replace 2 LSBs of $I_{(S)}$ by 2 bits $l_c^k(x)$ where x is the node number in l_c^k.
- If $S > NP_{PI}$ && AI is not fully embedded then replace 6 MSBs of pixel of $I_{(S)}$ by next 6 AI bits.
- Set S^c using Eq. (3.7)
- While $S^p \neq S^c$

Step-6: If all the bits of AI are not embedded and $l_c^{(k+1)}$ and $S \neq 0$ then set $k = k + 1$ and repeat steps 3-5. Else stop and exit.
Step-7: Stop and exit.

A. Extraction of PI

For the extraction of the PI, the reading of 6 MSBs in raster scan is started from the very first pixel of $I_{e'}$ and, first of all, $\lceil \log_2(M \times N) \rceil$ bits, i.e., 19 bits in the case of a 512×512 pixels image, are extracted from the first four pixels to get the greatest location difference (d_l^{max}) by their decimal value. Then, the next $\lceil \log_2(M \times N) \rceil$ bits are extracted which are used to determine d_l^1 and subsequent 8 bits to get V_c. And lastly, the next $\lceil \log_2(M \times N) \rceil$ bits are used to determine the length of AI, which is represented by LAI. Thus, a total of $3 \times \lceil \log_2(M \times N) \rceil + 11$ bits are read.

B. Extraction of Link Chains and Embedding of Secret Data in Link Chain

The link chain extraction process is somewhat similar to the link chain generation process. Here, the data hider first reaches the head of the first link chain using d_l^1, which is the pixel $\overline{I}(S)$ and reads 6 MSBs of the pixel (except the PI-enabled ones) for EI (which is the same as AI in the encoding/hiding phase, but being denoted differently to avoid any confusion) extraction and two LSBs for the direction to its next (or child) pixel of the link chain. Next, read again the 6 MSBs of the next pixel and 2 LSBs for the direction to its child node using Eq. (3.7). This process is continued until the complete AI is not retrieved. In case the direction of the next (or child) points to its parent, which in turn signifies the end of the k^{th} link chain (l_c^k), the

next link chain ($l_c^{(k+1)}$) needs to be picked. To get the next link chain head node location, the hider reads bits $\beta + 1$ to $\beta + \lceil \log_2(d_l^{max}) \rceil$ and converts them into decimal to get $d_l^{(k+1)}$, where $\beta = (k-1) \times \lceil \log_2(d_l^{max}) \rceil$. Next, $d_l^{(k+1)}$ is added to the current link chain head node location to get the sequence number of the $(k+1)^{th}$ link chain head. As soon as all the bits of EI are extracted, the same process is repeated for embedding the bits of secret data by just substituting the process of reading the 6 MSBs of the pixels with replacing 6 MSBs with the bits of secret data. The complete process is reiterated until $d_l^{(k+1)} = 0$ which signifies that there is no more link chain in the image.

C. Embedding of Secret Data in Remaining Pixels

To embed the secret data in the untouched pixels, firstly dl_{bs}, HCR, ELM, EMSP, and $\overline{H_{bs}}$ are extracted from the Extracted Information (EI). Next, the image is scanned in a raster scan manner by skipping the PI-enabled pixels and the pixels corresponding to the link chains for identifying the untouched pixels. Afterwards, information is embedded in each of the untouched pixels (ut_p) based on the corresponding label map information using AM2PHC, which is also reviewed in Section 3.1.1.

Thus, all the pixels of the image are utilized for embedding the secret information. In the next subsection, Algorithm 3 for the embedding of secret data is presented in a step-wise manner.

D. Algorithm for Embedding the Secret Data

The complete step-by-step algorithm for embedding the secret data in the image is provided in Algorithm 3.

3.2.3 Example-1: Link Chain Generation and Categorization for a Sample Image Block

The complete process of link chain generation and their categorization for a sample image block of 5×5 pixels is illustrated in Fig. 3.4 using an example. In the block, the first four pixels are marked for PI as shown in the grey color of Fig. 3.4b, because the length of PI (discussed in Section 3.2.1) is 23 bits (i.e., 5 bits d_l^{max}, 5 bits d_l^1, 8 bits V_c, and 5 bits LAI). After scanning the image in a raster scan order, a list (L_i) is generated which has pixel values with sequence number (S). The pixels in the list are arranged in sorted order according to their intensity value (val) as shown in Fig. 3.4c. The link chain (l_c) generation process is then initiated from the first node in the list (L_i) which has $val = 122$ and its $S = 5$ as depicted in Fig. 3.4c. The very first pixel of the l_c is also marked as link chain head (h_{lc}). Next, the process of adding more adjacent pixels from the image is initiated by considering $V_c = 4$. As depicted in Fig. 3.4e, a link chain which starts from the pixel sequence number 5 and ends at 2 via 4, 3, and 2 is generated. The

Algorithm 3 EMBEDDING OF SECRET DATA

Input:

$I_{e'}$: Encrypted Image with SI, Secret Data

Output:

I_m: Marked Encrypted Image

Step-1: Read 6 MSBs of first NP_{PI} pixels of $I_{e'}$ by scanning the image in raster scan manner to get PI. Next, split the PI as follows:

- $\lceil \log_2(M \times N) \rceil$ bits represent the d_l^{max};
- $\lceil \log_2(M \times N) \rceil$ bits represent the sequence (d_l^1) of first chain head (h_{lc}^1);
- 8 bits represent the variability coefficient;
- Remaining bits ($\lceil \log_2(M \times N) \rceil$) represents the LAI.

Step-2: Set $S = d_l^1$, $k = 1$, Cover set $C_s = Null$ and EI = Null;

Step-3: Set child node (S^c) of S as $S^c = S$ and $S^{old} = S$.

Step-4: Do the followings:

- Set parent node (S^p)=S, $S = S^c$ and update $C_s = C_s \cup S$
- If $S > NP_{PI}$ then
  - If length of EI $\leq$ LAI, add 6 MSBs of pixel $I_{e'(S)}$ in EI
  - Else replace 6 MSBs of pixel of $I_{e'(S)}$ by next 6 secret data bits.
- Update S^c using Eq. (3.7) by treating 2 LSBs of $I_{e'(S)}$ as $Dir(S)$.
- While $S^p \neq S^c$,

Step-5: Convert EI ($\beta + 1 : \beta + \lceil \log_2(d_l^{max}) \rceil$) to decimal to get $d_l^{(k+1)}$, where $\beta = (k-1) * \lceil \log_2(d_l^{max}) \rceil$. Also set $S = S^{old} + d_l^{(k+1)}$.

Step-6: If $d_l^k \neq 0$ then set $k = k + 1$ and repeat steps 3-5. Else go to step-7.

Step-7: Get dl_{bs}, HCR by reading the first $n * \lceil \log_2(d_l^{max}) \rceil$ and next 32 bits of EI, respectively. Subsequently delete the read bits i.e., $* \lceil \log_2(d_l^{max}) \rceil + 32$ bits from EI.

Step-8: For $i = NP_{PI} + 1 : M * N$

- If $i \notin C_s$ then get ELM for the untouched pixel $\bar{I}(i)$ from the EI and delete the bits from EI. Next, decode ELM using HCR to get label map value (t)
- If $t < 8$ then replace $t + 1$ MSBs of $I_{e'(i)}$ with the $t + 1$ bits of secret data else replace all 8 bits of $I_{e'(S)}$ with 8 bits of secret data

Step-9: Stop and exit.

encoded direction of l_c is also shown in Fig. 3.4f. The pixels which are part of the generated link chains so far are marked and treated as part of a cover set (C_s) as depicted in the first part of Fig. 3.4d. Next, for another l_c generation, we need to traverse L_i from the next element onwards and find out the next pixel which has not been part of any of the link chains generated so far or cover set (C_s). So, the next link chain can be begun from the L_i's next node as that is not part of the cover set (C_S) as depicted in Fig. 3.4f. Therefore, the next l_c is generated from pixel sequence number 6, the process of adding more adjacent pixels from the image is initiated as depicted in Fig. 3.4e and 3.4g, a l_c which starts from the pixel sequence number 6 ends at 19 via 11, 12, 13, 14, 15, 20, 25, 24, 23, 22, 21, 16, 17, 18, and 19. Now, C_S is updated as shown in Fig. 3.4d. Further, because earlier pixels are in the cover set, the sequence number of the next suitable pixel for l_c generation is 10. The process of adding more adjacent pixels from the image is the same and we find that this link chain which starts from the pixel sequence number 10 ends at the next node because no other pixel is in the V_c criteria. As categorization threshold (T_1) is considered three, then this generated l_c will be treated as a jump link chain as shown in Fig. 3.4h, which means all the pixels that are part of the l_c will not be marked in the C_S and hence can be considered in the next l_c. The same process of l_c generation imitated for the next pixel of L_i as shown in Fig. 3.4i, but again it results in a jump link chain. So, all the pixels are not marked in the set C_S. The above process is repeated for all the pixels of L_i which results in the l_c depicted in Fig. 3.4h to 3.4k. It can be seen from Fig. 3.4 that except the first and second l_c, all the other link chains are jump l_c which cannot be used for data hiding if V_c is considered as 4. As the first chain is a small l_c (l_{cs}) with a length of 4, so the average is calculated using Eq. (3.1) as depicted in Fig. 3.4l along with the first difference (d_l^1) i.e., the sequence for the first l_c. In contrast, the second l_c i.e., a large l_c (l_{cl}) makes use of a bitmap (B^M) along with quantization levels (H^l & H^h), to represent its pixels of the link chain. To further reduce the size of B^M, every alternate pixel has been dropped as shown in Fig. 3.4n. The H^l, H^h, and bits of D^{hl} are computed using Eqs. (3.3), (3.4), and Table 3.1, respectively as shown in Fig. 3.4n. Fig. 3.4o represents the total bits of PI and AI. Finally, Fig. 3.4p shows the embedding of SI, and the location of data embedding.

3.2.4 In the Purview of the Recipient

Since the proposed method is a separable RDHEI method, there can be two different receivers, namely the decoder and the extractor. The decoder is the person who holds the image key (K_I), and can get back the contents of the original image whereas the extractor holds the extracting key (K_D) to get the hidden contents. It should be emphasized that the decoder and extractor can be the same person if they possess both of the aforementioned keys.

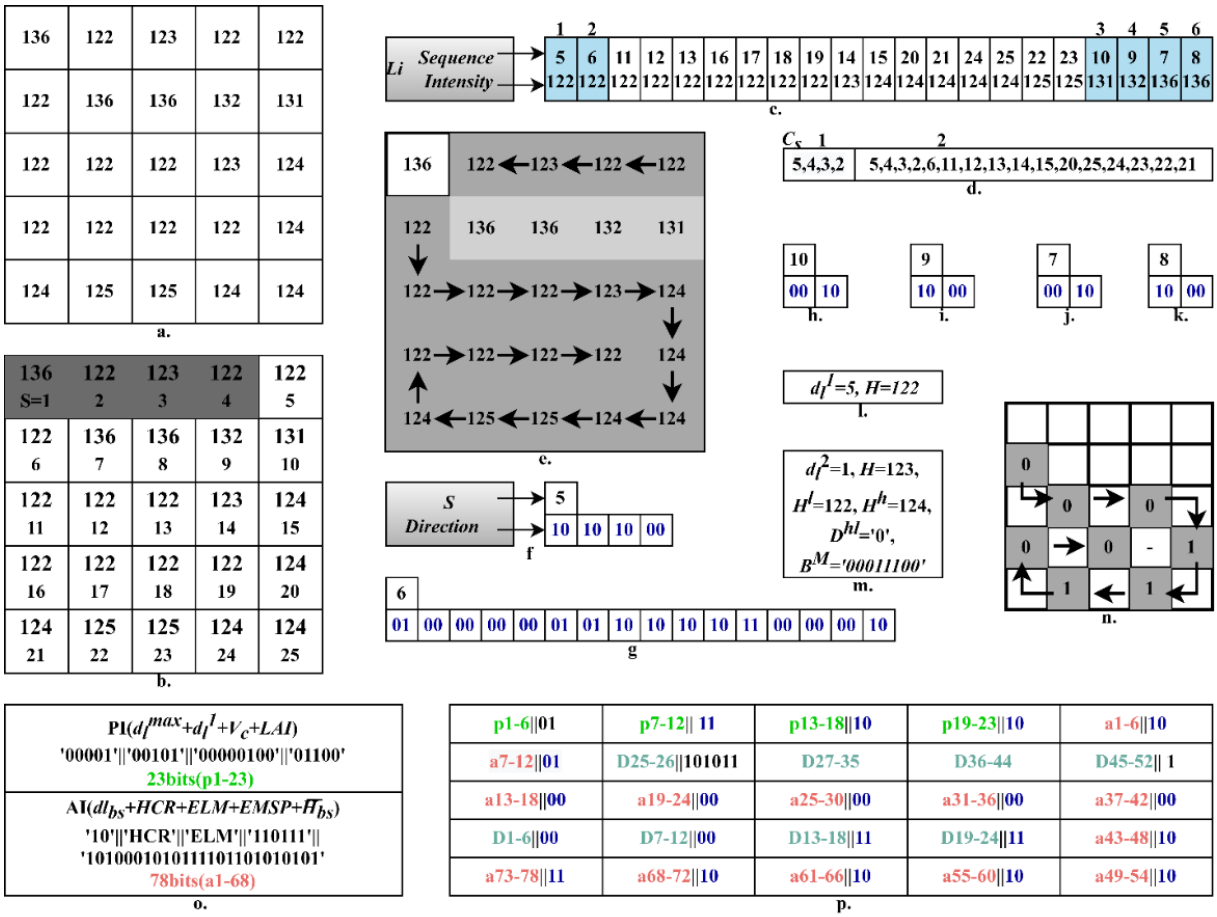


Fig. 3.4: An example of link chain generation (a) Original image block, (b) Pixel reserved for PI, (c) PI marking list (Li) of the block, (d) Cover set information (e) Link chain directions for the original image block for all possible link chains with V_c=4, (f-k) Possible link chains with V_c=4, (l) SI of link chain of Fig. 3.4f, (m) SI of link chain of Fig. 3.4g, (n) Bitmap of IAMBTC of link chain of Fig. 3.4g, (o) Calculated SI, and (p) Place and flow of data embedding

A. Data Extraction

The data extraction process is basically similar to the hiding process (described in Section 3.2.4). Here, the extractor first needs to extract the PI, link chain, and AI like the hider and then reads the six bits from each pixel instead of replacing them as in the case of data hiding to get the encoded secret information from the link chain pixels. Next, the untouched pixels (ut_p) are processed using the label map to get back the hidden data. Finally, the extractor decodes the extracted information using the key (K_D) to get back the hidden information.

B. Image Decoding

As discussed earlier, the image decoding process is performed by the decoder who holds the decoding key (K_I). In the image decoding process, first of all, PI, directory of link chains (DL_c), Label Map (LM), and AI is extracted on similar lines to the hider (described in Section 3.2). Afterwards, dl_{bs}, HCR, ELM, EMSP, and $\overline{H_{bs}}$ are extracted as per the steps 7, 8, and 10

defined in the decoding algorithm. Next, the secret data is then extracted from the untouched pixels with the help of AI and PI. Afterwards, a pseudo-random number list $\tilde{l}_e$ is created using the key K_I with the help of a pseudo-random number generator and $\overline{H_{bs}}$ is decrypted using Eq. (3.12), as follows:

$$H_{bs}(i) = \overline{H_{bs}}(i) \oplus \tilde{l}_e(i), \tag{3.12}$$

where $1 \leq i \leq$ size of $\overline{H_{bs}}$. Additionally, image decryption is done according to Eq. (3.13) by first creating a pseudo-random number matrix ξ of size $M \times N$ with the help of key (K_I) using a pseudo-random number generator.

$$I_{r(i,j)} = I_{m(i,j)} \oplus \xi_{e(i,j)}, \tag{3.13}$$

where $1 \leq i \leq M$ and $1 \leq j \leq N$. Now to restore the image, all l_c of DL_c are processed in their order as follows: If the l_c is small then 8 bits are removed from H_{bs} and converted into decimal to get average (H). Afterwards, a list for recovered intensity (R) of size $max\_len(l_c^k)$ is generated where each element of R is assigned average (H value. In case the l_c is large then 8 bits are removed from H_{bs} and converted into decimal which represents low quantization (H^l). Next, D^{hl} bits are removed from H_{bs} and decoded using 3.1. Afterwards, high quantization (H^h) is computed as $H^h = H^l + D^{hl}$ and partial bitmap (B^M) is obtained by extracting $max\_len(l_c^k)/2$ bits from H_{bs}. The complete bitmap (B_c^M) is estimated using the following Eq. (3.14):

$$B_c^M(x) = \begin{cases} B^M(\lceil x/2 \rceil), & \text{if } x\%2 = 1 \\ -1, & \text{if } x\%2 = 0 \end{cases} \tag{3.14}$$

where x is a positive integer $(1, 2, 3, \ldots, max\_len(l_c))$. Thus, odd locations are filled by the values of the partial bitmap and $' - 1'$ is set to the even locations of the bitmap. Next, a list (R) to recover intensity values of the pixels is constructed using Eq. (3.15) as follows:

$$R(x) = \begin{cases} H^l, & \text{if } B_c^M(x) = 0 \\ H^h, & \text{if } B_c^M(x) = 1 \\ R_{avg}, & \text{if } B_c^M(x) = -1 \text{ and } x < max\_len(l_c) \\ R(x-1), & \text{if } B_c^M(x) = -1 \text{ and } x = max\_len(l_c) \end{cases} \tag{3.15}$$

where $R_{avg} = \lfloor \frac{1}{2}(R(x-1) + R(x+1)) \rfloor$. All the pixels of $\widetilde{I}$ corresponding to l_c are then replaced by a corresponding value of recovered intensity (R). Thus, the pixels corresponding to the link chains are recovered. Next, the untouched (ut_p) pixels are reconstructed using AM2PHC as described in Section 3.1.1. Thus, the complete image is recovered and secret data is extracted.

Algorithm 4 DATA EXTRACTION AND IMAGE RECONSTRUCTION ALGORITHM

Input:
I_m: Marked Encrypted Image, K_D: Data key, K_I: Image key
Output:
I_r: Reconstructed Image, Secret data
Step-1: Read 6 MSBs of first NP_{PI} pixels of I_m by scanning the image in raster scan manner to get PI. Next, split the PI as follows:

- $\lceil \log_2(M \times N) \rceil$ bits represent the d_l^{max};
- $\lceil \log_2(M \times N) \rceil$ bits represent the sequence (d_l^1) of first chain head (h_{lc}^1);
- 8 bits represent the variability coefficient;
- Remaining bits ($\lceil \log_2(M \times N) \rceil$) represents the LAI.

Step-2: Set $S = d_l^1$, $k = 1$; $C_s = Null$; and $EI = Null$;
Step-3: Set child node (S^c) of S as $S^c = S$ and $S^{old} = S$.
Step-4: Do the followings:

- Set parent node (S^p)=S and $S = S^c$ and update $C_s = C_s \cup S$
- If $S > NP_{PI}$ then
  - If length of EI $\leq$ LAI, add 6 MSBs of pixel $I_{m(S)}$ in EI
  - Else read 6 MSBs of pixel of $I_{m(S)}$ and add them in the secret data.
- Update S^c using Eq. (3.7) by treating 2 LSBs of $I_{m(S)}$ as $Dir(S)$.
- While $S^p \neq S^c$,

Step-5: Convert EI ($\beta + 1 : \beta + \lceil \log_2(d_l^{max}) \rceil$) to decimal to get $d_l^{(k+1)}$, where $\beta = (k-1) * \lceil \log_2(d_l^{max}) \rceil$. Also set $S = S^{old} + d_l^k$.
Step-6: If $d_l^k \neq 0$ then set $k = k + 1$ and repeat steps 3-5. Else go to step-7.
Step-7: Get dl_{bs}, HCR by reading and removing the first $n * \lceil \log_2(d_l^{max}) \rceil$ and next 32 bits of EI.
Step-8: For $i = NP_{PI} + 1 : M * N$

- If $i \notin C_s$ then get ELM for the untouched pixel $I_{r(i)}$ from the EI and delete the bits from EI. Next, decode ELM using HCR to get label map value (t)
- If $t < 8$ then extract $t + 1$ MSBs of $I_{r(i)}$ to get $t + 1$ bits of secret data else extract all 8 bits of $I_{r(i)}$ for 8 bits of hidden secret data

Step-9: Decrypt the obtained secret data with data key (K_D) to retrieve its original form
Step-10: Decrypt the image I_r and remaining bits of EI that represents H_{bs} using image key (K_I) by using Eqs. (12) and (13), respectively.
Step-11: Set $S = d_l^1$ and $k = 1$;
Step-12: If $max\_len(l_c^k) \leq T_2$ then remove 8 bits from H_bs (decrypted H_{bs}) and converted into decimal to get average (H). Generate a list for recovered intensity(R) of size $max\_len(l_c^k)$ where each element of R is assigned average (H) value. Afterwards, all pixels of I_r corresponding to the l_c are replaced by H.

Algorithm 4 DATA EXTRACTION AND IMAGE RECONSTRUCTION ALGORITHM(CONTINUED)

17: **Step-13:** Else

- Remove 8 bits from H_bs i.e., H^l,
- Get D^{hl} by reading next few bits with the help of 3.1 and calculate the $H^h = H^l + D^{hl}$
- Construct partial bitmap (B^M) by extracting $max\_len(l_c^k)/2$ bits from H_bs.
- Estimate the complete bitmap (B_c^M) using following Eq. (3.14)
- Generate a list for recovered intensity(R) of size $max\_len(l_c^k)$ using following Eq. (3.15)

18: **Step-14:** Set $x = 1$, do the followings:

- Set parent node (S^p)=S , $S = S^c$
- $I_{r(S)} = R(x)$
- Update S^c using Eq. (3.7) by treating bits of $l_c^k(x)$ as $Dir(S)$. and set $x = x + 1$
- While $S^p \neq S^c$

19: **Step-15:** If $d_l^k \neq 0$ then set $k = k + 1$ and repeat steps 11-14. Else, go to step-16.

20: **Step-16:** For $S = NP_{PI} + 1 : M * N$,

- If $S \notin C_s$ then predict a value using MED predictor for pixel $I_{r(S)}$, and based on the label map, replace label (t) number of MSB bits of pixel $I_{r(S)}$ with predicted value MSB bits,
- If $t < 8$ then replace the next $(t + 1)$ MSB bit of pixel $I_{r(S)}$ with transpose of corresponding MSB of predicted value.

21: **Step-17:** Stop and exit.

The complete flowchart to elaborate the process flow of the proposed LCD-RDHEI is presented in Fig. 3.5. The flowchart demonstrates the working in all three purviews defined in Section 3.2.1. In the preview of the content owner, various link chains (l_c) are generated, categorized, and compressed either by IAMBTC or by average based on their length after dropping the jump link chain. The encrypted chain averages (H) or low quantifier (H^l) are collected in support information along with PI, link chain direction, and D^{hl}. On another end, AM2PHC is applied to untouched pixels (which are not part of any of the above link chains, i.e., small or large) and generates ELM, LELM, and HCR, which are also combined with SI. The aforementioned SI is then embedded in an encrypted image (I_e), which is recovered using the image key from the original image's encryption. After receiving the aforementioned SI-enabled encrypted image (I'_e), the data hider reads the SI and regenerates the link chain, LM, and other information, then embeds the encrypted data in the image. At the receiver's side, the recipient first

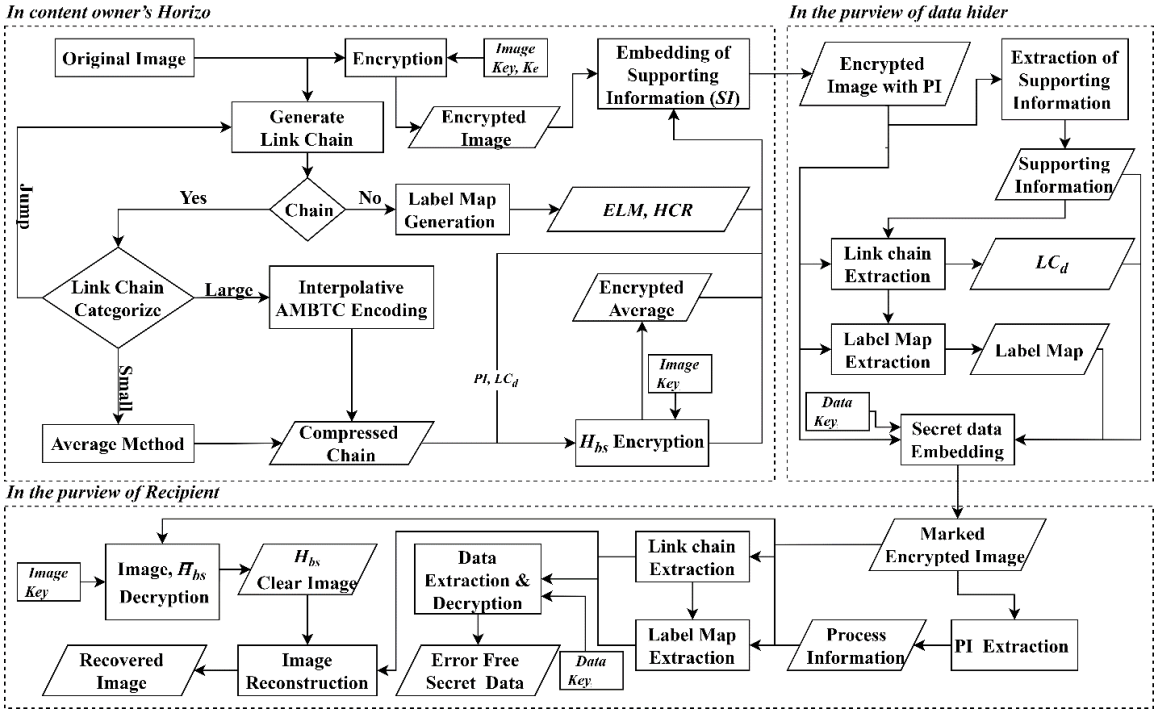


Fig. 3.5: Complete flow chart of the proposed LCD-RDHEI method

extracts SI, transforms it into the link chain, LM, and other necessary information, and then extracts data or reconstructs the image based on the kind of available keys. Next, encrypted data is extracted from the marked encrypted image (I_m) with the help of the link chains and LM described previously. After decrypting encrypted data using a data key, the hidden data is losslessly recovered. If the image key is available with the receiver, then it is used to decrypt the above image with the help of average, quantization list, and LM and l_c.

3.3 Experimental Results and Discussion

This section elaborates upon experimental results and their analysis in order to compare and evaluate the performance of the proposed LCD-RDHEI method with respect to the existing and related RDHEI methods. The discussion and analysis of the experimental results has been done in four sub-sections as follows. The first sub-section presents a detailed explanation of the LCD-RDHEI considering image $Lena$ as a use case, the second sub-section discusses performance analysis on different test images and datasets such as BOSSbase [38], BOWS-2 [39] and UCID [40], the third sub-section provides discussion around security of the method under consideration, and final sub-section presents a comparative analysis of the LCD-RDHEI with respect to the state of art modern methods. In these sub-section, the main focus is on EC/ER, illustrating how much data can be incorporated either in an entire image or per pixel. This section also exploresPSNR, which is a logarithmic metric devised to measure the variations between an original image and its reconstructed version. Infinite PSNR implies absolute reversibility,

Table 3.2: ER (in bpp) and PSNR (in dB) of the recovered images under different variability coefficients for eight test images

	Lena		*Baboon*		*Airplane*		*Barbara*		*Pepper*		*Boat*		*Man*		*Crowd*	
V_c	EC	PSNR	EC	PSNR	EC	PSNR	EC	PSNR	EC	PSNR	EC	PSNR	EC	PSNR	EC	PSNR
1	2.6522	$+\infty$	1.1011	$+\infty$	3.7309	$+\infty$	1.9622	$+\infty$	2.2484	$+\infty$	2.1953	$+\infty$	2.2591	$+\infty$	3.0015	$+\infty$
2	2.6634	55.7917	1.1090	59.2415	3.7114	55.0909	1.9748	56.6469	2.2452	56.5927	2.2025	57.1161	2.2636	57.0842	3.1090	57.0980
3	2.8621	52.3290	1.1430	55.8434	3.7286	51.9711	2.1446	53.2048	2.4230	52.9935	2.2832	53.6151	2.3533	53.5386	3.2708	53.1144
4	2.9993	48.8056	1.2116	51.4982	3.7842	49.0838	2.3325	49.6571	2.6435	49.0722	2.4708	49.6266	2.4627	49.7176	3.3278	50.2469
5	3.2148	46.7481	1.3056	49.0467	3.8544	47.3645	2.5233	47.6214	2.9823	46.7486	2.6519	47.2370	2.6250	47.4295	3.4633	48.1430
6	3.5079	45.1297	1.4144	46.6721	3.9135	45.9519	2.6943	45.9546	3.2329	44.7996	2.8517	45.1664	2.8770	45.3513	3.5956	46.4290
7	3.6165	43.9491	1.5288	44.9644	3.9712	44.9060	2.9187	44.7239	3.4533	43.4199	3.0381	43.6630	3.0492	43.8820	3.7045	45.1233
8	3.7175	42.9513	1.6507	43.3884	4.0606	43.9731	3.0517	43.6203	3.5701	42.2220	3.2168	42.3416	3.2017	42.6042	3.8238	43.8163
9	3.8580	42.1339	1.7699	42.1509	4.1575	43.1754	3.1669	42.7224	3.7422	41.2986	3.3841	41.2788	3.3492	41.5804	3.8488	42.9207
10	3.9791	41.4396	1.8873	41.0052	4.2457	42.4313	3.2683	41.8952	3.8941	40.4986	3.5314	40.3664	3.4770	40.6645	4.0061	41.7956
11	4.1484	40.8425	1.9990	40.0576	4.3202	41.8408	3.3639	41.1428	4.0296	39.8714	3.6721	39.6069	3.5946	39.9559	4.0341	41.1150
12	4.2498	40.2966	2.1125	39.1408	4.3852	41.2446	3.4526	40.4455	4.1538	39.3214	3.7963	38.9490	3.7008	39.2721	4.1711	40.3522
13	4.3208	39.8431	2.2176	38.3837	4.4407	40.7525	3.5295	39.8445	4.2560	38.8342	3.9062	38.3626	3.7973	38.7076	4.1836	39.7799
14	4.3533	39.3855	2.3910	37.6694	4.5040	40.2873	3.5590	39.2607	4.2924	38.3845	4.0041	37.8565	3.8895	38.1678	4.2436	39.1837
15	4.4703	38.9945	2.5004	37.0363	4.5563	39.8581	3.6268	38.7486	4.4239	38.0574	4.0854	37.4399	3.9762	37.6836	4.3098	38.6887
16	4.5265	38.6329	2.6014	36.4276	4.5964	39.4697	3.6817	38.2352	4.4877	37.7202	4.1648	37.0263	3.9867	37.2479	4.3564	38.1931
17	4.5735	38.2881	2.6959	35.8910	4.6284	39.1124	3.7891	37.7709	4.5158	37.4239	4.2426	36.6624	4.1195	36.8363	4.4133	37.6758
18	4.6177	37.9611	2.8662	35.3917	4.6634	38.7960	3.8023	37.3264	4.5744	37.1495	4.2562	36.3268	4.1293	36.4639	4.4514	37.3114

whereas superior to 30dB is regarded as satisfactory reversibility.

3.3.1 A Detailed Explanation of the LCD-RDHEI Considering Image Lena as a Use Case

The trade-off of ER and PSNR i.e., if V_c increases the ER increases but PSNR decreases has been depicted in Table 3.2, where ER and PSNR are dependent on the variability coefficient V_c. The proposed LCD-RDHEI method is applicable for V_c = 1 to 256, but for sake of simplicity and for comparative ease, V_c = 1 to 18 or PSNR $>$ 40dB has been taken into consideration in the presented experiments. However, the content owner can decide the value of V_c according to his/her application needs.

A test image $Lena$ is chosen to experimentally study different aspects such as pixel distribution of original and reconstructed image, representation of original, encrypted and reconstructed images etc., as provided in Fig. 3.6 for V_c=12. In Fig. 3.6a, the original image of $Lena$ of 512 × 512 size is shown and its pixel distribution using a histogram is presented in Fig. 3.6b. Since the proposed method starts its process with link chain (l_c) generation as discussed above, the coverage of pixels by link chains in the $Lena$ image is also presented in Fig. 3.6c. The black colour in Fig. 3.6c represents the pixel covered by the link chains. However, the LCD-RDHEI method generates a total of 13585 l_c out of which 2666 are large l_c and remaining 10919 are small l_c, where the maximum size of l_c thus formed is 1553 pixels whose initial location is at coordinate (233, 475) or sequence number (119259). In addition to this, a total of 223119 pixels have been covered in the link chain which includes 149619 pixels of large l_c. Two bits are reserved for direction, therefore the preceding l_c can reserve six bits per pixel, or a total of 1338714 bits. Following that, other room in the image is reserved using AM2PHC on the remaining 39025 pixels, and a label map of 158814 bits is generated for a total room of 246001 bits. The total SI includes 85 bits of PI (that includes 18 bits for d_f^l size identification, 9 bits

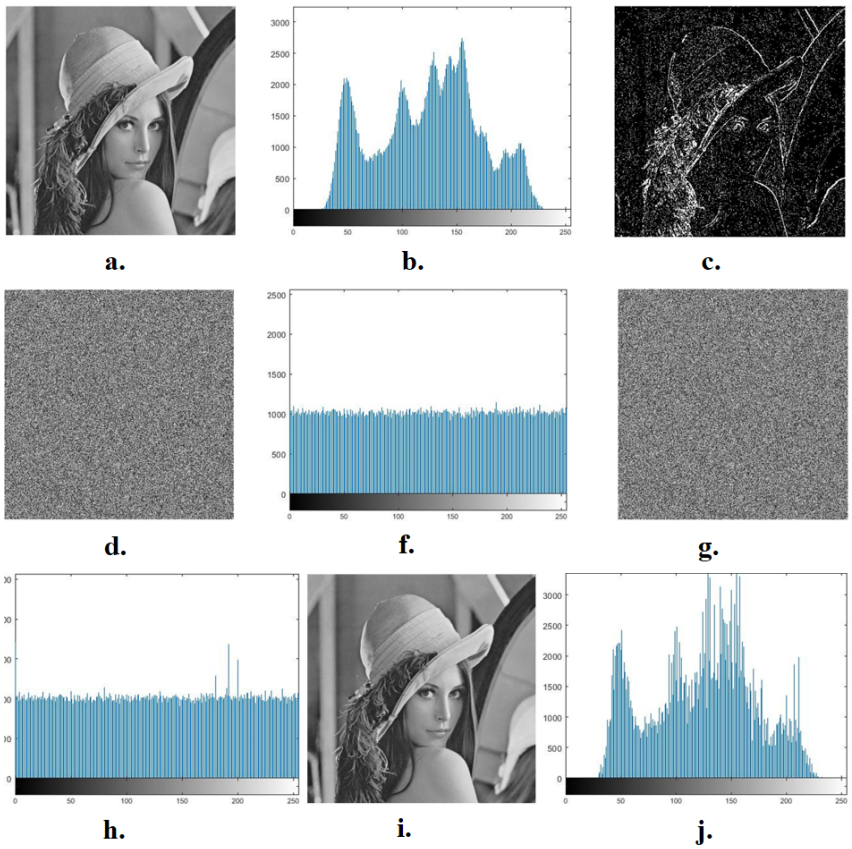


Fig. 3.6: Experimentation on image $Lena$ considering $V_c = 12$ (a) Test image, (b) Histogram of the test image, (c) Coverage of Link Chains, (d) Encrypted image, (e) Histogram of encrypted image, (f) Marked image, (g) Histogram of Marked image, (h) Reconstructed image, and (i) Histogram of the reconstructed image

for d_f^l, 8 bits for V_c, 32 bits for Huffman code, 18 bits for E^f), 236925 bits for all d_f^l, H, H^l and D^{hl} for all l_c, 74807 bits for bitmap and 158814 bits of label map, which will embed after encryption. In Fig. 3.6d, the encrypted image which has been encrypted by considering K_D = 143, is shown. The histogram of the encrypted image is presented in Fig. 3.6e, which clearly shows that pixels are equally distributed after encryption and there is no similarity left between the original image i.e., Fig. 3.6a and encrypted image i.e., Fig. 3.6d. Then, 470631 bits SI are embedded in total reserved room, which is the sum of the 1338714 bits generated by l_c's and the 246001 bits generated by AM2PHC. Next, the data hider can embed secret data at a rate of 4.2498 bpp in the aforesaid SI-enabled encrypted image. Fig. 3.6f displays the marked encrypted image containing 1114084 bits of secret data. The pixel distribution of the marked encrypted image has been provided in Fig. 3.6g. This distribution is closer to the encrypted version provided in Fig. 3.6e which implies that the marked encrypted image also does not have any clue about the original image. Finally, the recovered image is shown in Fig. 3.6h after error-free extraction of the hidden data. The PSNR between the original image, shown in Fig. 3.6a, and the reconstructed image, shown in Fig. 3.6h, is 40.2966 dB. The histogram of the recovered image, which is substantially identical to the original image, is shown in Fig. 3.6i.

Table 3.3: ER (bpp) and PSNR (dB) of the recovered images with $V_c = 1$ and $V_c = 10$ for three datasets

	$V_c = 1$			$V_c = 10$		
	BOSS Base	**BOWS-2**	**UCID**	**BOSS Base**	**BOWS-2**	**UCID**
ER	3.2348	3.0415	2.704	4.2325	4.0012	3.8497
PSNR	$+\infty$	$+\infty$	$+\infty$	42.8769	40.9913	41.0458

3.3.2 Performance Analysis on Different Test Images and Datasets

In order to analyse the efficiency of the proposed LCD-RDHEI technique, the 8 standard test images namely $Lena$, Man, $Baboon$, $Pepper$, $Airplane$, $Boat$, $Barbara$ and $Crowd$, each of size 512×512 pixels, as shown in Fig. 1.3, have been taken into consideration. The experimental results in terms of ER (in bpp) and PSNR (in dB) by setting $V_c = 1$ to 16, have been presented in Table 3.2. At $V_c = 1$, PSNR is infinite as the receiver can losslessly get the original image after extraction of the secret data. However, as the value of V_c increases, the PSNR decreases but EC increases and vice-versa is also true, which represents the traditional trade-off between the ER and PSNR. It can be observed from Table 3.2 that the smother images such as $Airplane$, and $Lena$ provide more ER as compared to the rough images such as $Baboon$. In other words, the maximum ER i.e., 3.0789 is achieved in the case of the smoothest image i.e., $Airplane$ and in contrast, minimum ER i.e., 1.0923 is provided by the most complex image i.e., $Baboon$ with infinite PSNR. This suggests that the LCD-RDHEI method efficiently utilizes the spatial correlation. As a result, around half of the bits (i.e., 4 bits) in a smother image are easily utilized for embedding while keeping the PSNR over 40dB. It can be observed that in few cases, ER at $V_c = 2$ is lower than that of at $V_c = 1$ because LCD-RDHEI, at $V_c = 2$ and onwards, employs IAMBTC to represent the pixels of link chain by trio of H^l, D^{hl} and bitmap, which results in more overhead instead of one average value (in case of $V_c = 1$). The performance of the LCD-RDHEI is also analysed on three data sets at $V_c = 1$ and 10. For this, the results of embedded capacity of each data set and corresponding PSNR are provided in Table 3.3. At $V_c = 10$, the ER for 500 randomly selected images from the BOSSBase, BOWS-2, and UCID are 4.2325, 4.0012, and 3.8497 respectively, while the PSNR values are 42.8769, 40.9913, and 41.0458. Conversely, at $V_c = 1$, the proposed method attains infinity PSNR for the similarly selected 500 images, delivering ER values of 3.2348, 3.0415, and 2.704bpp for the BOSSBase, BOWS-2, and UCID databases respectively.

3.3.3 Security Analysis

The LCD-RDHEI method aims to secure hidden data as well as the cover image using two keys: a data key for encrypting the data and an image key for anonymizing the cover image. It is to be noted that the data hider can use any secure method to encrypt the data, reducing data leakage risks. Meanwhile, AES-CTR with the image key is used for encrypting the processed

Table 3.4: Evaluation of security for seven encrypted test images

Test Images	Horizontal Correlation		Vertical Correlation		Shannon Entropy		NPCR
	Original	Encrypted	Original	Encrypted	Original	Encrypted	
Lena	0.9726	0.052603	0.9859	-0.027929	7.445061	7.999200	99.811542
Baboon	0.8719	-0.018643	0.7576	-0.019225	7.358336	7.999361	99.798248
Airplane	0.9731	0.013771	0.9613	0.026329	6.702462	7.999136	99.814731
Barbara	0.8554	-0.044477	0.9597	-0.008613	7.466426	7.999335	99.807739
Pepper	0.9706	-0.029432	0.9819	-0.004811	7.593654	7.999215	99.807204
Boat	0.9258	0.040320	0.9718	-0.042878	7.191370	7.999290	99.794006
Man	0.9757	-0.032377	0.9707	0.010273	7.484219	7.999836	99.802970
Crowd	0.9675	-0.067165	0.9659	0.024945	6.789260	7.999200	99.811581

cover image. To visually assess security, the original, encrypted, marked encrypted, and reconstructed images of $Lena$ are shown in Fig. 3.6. It's clear from these images that the encrypted and marked encrypted ones are garbled, making the original content unrecognizable. Further, histograms of all these images are shown in Fig. 3.6. Unlike the original and reconstructed images, the encrypted ones have uniform distributions, showing the method's effectiveness in providing perceptual security To further evaluate the security, different statistical matrices such as horizontal & vertical correlation coefficients, Shannon entropy, NPCR and PSNR have also been taken into consideration. The experimental results in relation to horizontal, vertical correlation [35], Shannon entropy [36] and NPCR [37] are presented in Table-3.4. From the Table, it is clear that that horizontal and vertical correlation analysis is very high in the original image $Lena$, which is almost nothing in the encrypted image of the same. In addition to this, the grey level dispersion of Shannon entropy is uniform and reaches a high near maximum for both original and encrypted images. The extreme values of the NPCR test elucidate for robustness against differential attacks. The probability of predicting the image without the key is equal to $M \times N \times 8$ bits with two possibilities each for values 0 and 1, which is really a giant number to brute force upon. Thus, the original and encrypted images are considerably different, as one would expect. Additionally, employing AES-CTR for image encryption, along with the method's flexibility in encrypting hidden data using a strong algorithm, greatly reduces the chance of brute-force attacks. The method's data hiding relies entirely on the SI and its separability, making it resilient to collage attacks. Thus, it can be conclude that this method exhibits notable robustness against various attack methods.

3.3.4 Comparison with State-of-Art Methods

To comprehensively evaluate the performance of the proposed LCD-RDHEI method against some of the notable and recently developed existing methods such as the works of Liu *et al.* [95] (2023), Qin *et al.* [96] (2023), Li *et al.* [79] (2023), Gao *et al.* [97] (2022), Wang *et al.* [98] (2021), Xiao *et al.* [99] (2020), Yin *et al.* [76] (2019), Shiu *et al.* [73] (2019), Puteaux *et al.* [70] (2018), Cao *et al.* [66] (2015), Zang *et al.* [65] (2015) and Ma *et al.* [28] (2013), have been

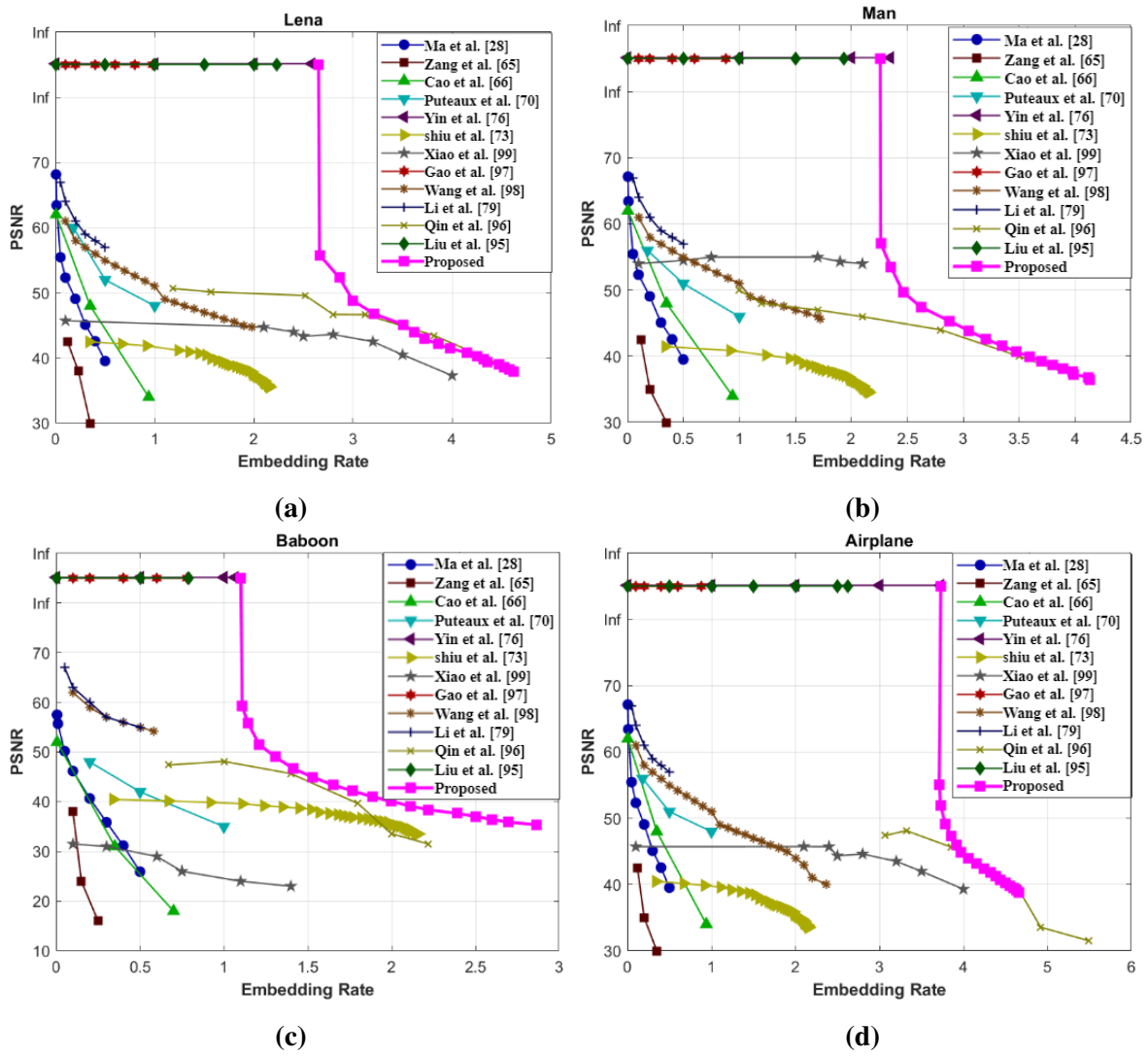


Fig. 3.7: Comparative analysis of proposed method w.r.t. SOTA methods for four test images of $Lena$, Man, $Baboon$ and $Airplane$

comparatively analyzed. The aforementioned state-of-the art methods belong to both VRAE and RRBE. Fig. 3.7 graphically represents the comparative analysis between the State-Of-The-Art (SOTA) and the proposed method, focusing particularly on PSNR and ER as the pivotal performance metrics. The comparative analysis is given using four different characteristics test images $Lena$, Man, $Baboon$, and $Airplane$. As inferred from Fig. 3.7, the ability to embed a substantial payload is feasible in every case while maintaining an acceptable level of PSNR.

From Fig. 3.7a which repents the ER of image $Lena$, the case of infinite PSNR, the ER stands at 2.6522, surpassing even the finest SOTA method, exemplified by [76] at 2.583, offering infinite PSNR. Conversely, under variable PSNR conditions, the method surpass the [99], which presented the maximum ER of 4 bpp at a PSNR value of 38.90. For the same method, in the instances of the $Baboon$ and $Airplane$ images at infinite PSNR, the proposed methodology surpasses [76], noteworthy considering the structurally complex nature of the $Baboon$ image and the smoothness of the $Airplane$ image. Nonetheless, the proposed method does not outstrip

[76] in the scenario of the Man image at infinite PSNR. Importantly, with minimal distortion, marked at 53.5386 dB, the proposed method achieves an ER of 2.3533, marginally eclipsing the 2.349 rate of [76]. The illustrative insights from Fig. 3.7b reveal that the LCD-RDHEI method achieves a high ER, even with negligible distortion. Additionally, [76] achieves ER of 1.8999, 2.1868, 2.1323, and 2.9432bpp for the images $Barbara$, $Pepper$, $Boat$, and $Crowd$, respectively. In comparison, the proposed method realizes gains of 0.0623, 0.0616, 0.63, and 0.592bpp respectively for each image. Moreover, it's pivotal to highlight that, unlike [76] which is constrained to a fixed ER, the proposed method offers enhanced flexibility. This adaptability allows for increased capacity through manageable increments in distortion, providing a versatile solution that can be tuned for optimal performance.

3.4 Summary

In this chapter, a high EC link chain driven RDHEI method based on pixel correlations, was proposed. The proposed LCD-RDHEI method collects the highly correlated pixels in terms of link chains which are of variable sizes and shapes. This collection of correlated pixels based on a variability co-efficient, helps in creating a big room before encryption which in turn helps in embedding the large amount of secret data. It is to be noted that the proposed method works as completely reversible data hiding method in encrypted domain if the variability co-efficient is zero, in other words, when all the pixels have the same intensity value within a link chain, which results into minimal EC. At larger embedding capacities where the values in a link chain are subject to a user-defined range based on the variability co-efficient, some of the contents of the cover media might be lost at the time data extraction and image restoration. To our knowledge, this is the first method that suggests the usage of variable size and shape of groups so that highly correlated groups can be formed. Experimentally, the proposed LCD-RDHEI method provides significantly greater EC than the existing and related data hiding methods.

CHAPTER 4

HIGH-CAPACITY REVERSIBLE DATA HIDING IN ENCRYPTED IMAGES BASED ON DIFFERENCE IMAGE TRANSFIGURATION

It has been observed that the traditional RDHEI methods achieve low EC or result in a lossy reconstruction of the original image, in other words, have a trade-off between EC and reconstructed image quality. In order to mitigate the aforementioned trade-off, this chapter presents an RDHEI method based on Difference Image Transfiguration (DIT). This method starts by initially exploring the hybrid predictor combination of difference predictors and MED predictors, following which the MSB planes of the resultant matrix (difference matrix) undergo a cleaning process. Consequently, a transfigured image is obtained. Moreover, a space is created within the transfigured image to embed the MSB planes originating from the difference matrix. Furthermore, the content generated from the MSB planes of the difference matrix is compressed employing a novel location difference technique. The proposed method aims to achieve a very high EC while ensuring errorless reconstruction of the cover image. The subsequent section is devoted to providing a detailed explanation of the proposed methodology.

4.1 Proposed Method

This section introduces a novel RDHEI method based on difference image transfiguration. The proposed method involves four stages, as illustrated in Fig. 4.1. The first stage is difference image transfiguration, where the content owner generates an image with multiple cleaned MSBs. In the second stage which is image encryption, the content owner encrypts the image with an image key and embeds SI while reserving space for data embedding. The third stage involves data embedding, where the data hider embeds secret information in the reserved space. In the final stage, extraction and recovery, the recipient extracts the hidden data and reconstructs the original image. Thus, an effective method for secure and efficient data hiding is introduced.

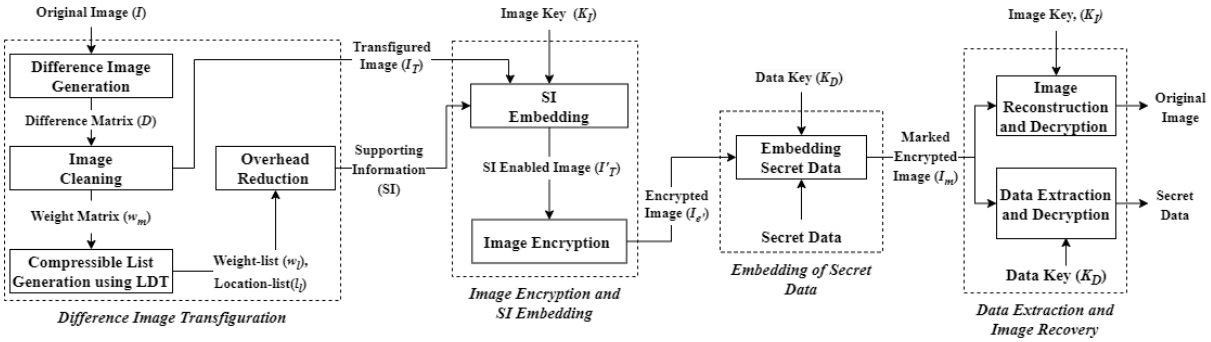


Fig. 4.1: Block diagram of the proposed DIT-RDHEI method

4.1.1 Difference Image Transfiguration

This sub-section comprises four parts: difference image generation, multi-MSB cleaning, compressible list generation using the Location Difference Technique (LDT), and overhead reduction. Following the generation of the difference matrix in the first part, the cleaning process yields a transfigured image and weight matrix. Subsequently, the weight matrix is compressed using the LDT, and finally, overhead is minimized and results in SI.

A. Difference Image Generation

The original gray-scale image I of size $M \times N$ is transformed into a difference matrix (D) of the same size i.e., $M \times N$ using the difference and MED predictors, both. This process of the transformation explores the redundancy of the input image I and can be executed by Eq. (4.1), defined below:

$$D_{(i,j)} = I_{(i,j)} - \begin{cases} I_{(i,j)}, & \text{if } i = 1 \ \& \ j = 1 \\ I_{(i,j-1)}, & \text{else if } i = 1 \\ I_{(i-1,j)}, & \text{else if } j = 1 \\ I_{med(i,j)}, & \text{otherwise} \end{cases} \tag{4.1}$$

where $D_{(i,j)}$ represents the difference matrix element corresponding to the original image pixel $I_{(i,j)}$ and I_{med} is determined using Eq. (4.2), $1 \leq i \leq M$ and $1 \leq j \leq N$.

$$I_{med(i,j)} = \begin{cases} max(B_p, U_p), & \text{if } O_p \leq min(B_p, U_p) \\ min(B_p, U_p), & \text{elseif } O_p \geq max(B_p, U_p) \\ B_p + U_p - O_p, & \text{otherwise} \end{cases} \tag{4.2}$$

where $B_p = I_{(i,j-1)}$, $U_p = I_{(i-1,j)}$ and $O_p = I_{(i-1,j-1)}$ denote back-pixel, upper-pixel, oblique-pixel, respectively, with respect to $I_{(i,j)}$. It is to be noted that the original image (I) can be reconstructed at the recipient end by having access to the difference image (D) and the value of the first pixel (i.e., $I_{(1,1)}$), which is chosen as the reference point. By designating the $I_{(1,1)}$ as

Table 4.1: Count of '1' in each bit-plane

Image	Bit-plane MSB to LSB								
	$Sign$	7^{th}	6^{th}	5^{th}	4^{th}	3^{rd}	2^{nd}	1^{st}	0^{th}
Lena	115550	0	123	1725	9633	31781	80121	115585	129660
Airplane	78373	5	429	1513	3686	10116	30812	133767	114573

the reference pixel (p_{ref}), the corresponding difference pixel $D_{(1,1)}$ can be initialized to zero.

where $D_{(i,j)}$ denotes the difference between the original $I_{(i,j)}$ and predicted pixels $P(i,j)$ located at location i,j, $1 \leq i \leq M$ and $1 \leq j \leq N$. Thus, the difference (matrix) D will have both positive and negative values in the range of $[-255...255]$, as $I(i,j)$ and $P(i,j)$ both have the pixel values in the range (0 to 255). It means total of nine bits are required to represent each value of D as one bit is exclusively required to represent sign and other 8 bit can represent the magnitude of difference. Thus, total 9 bit-planes of size $M \times N$ are needed to represent D. The sign (or 8^{th}) bit-plane of D is usually highly random or in other words has very less sparsity as both positive and negative values are equally possible, in comparison to other MSB-planes which usually have large sparsity but in decreasing order. The reason of this is the prevalent correlation among adjacent pixels of the original meaningful images. The same is evident in Table 4.1 where two standard test image $Lena$ and $Airplane$ have been considered for experimentation. It can be observed from that table that the sparsity i.e., large number of zeros and fewer number of ones, increases with the bit-plane number except for the sign bit-plane.

Since the 8^{th} bit-plane is highly random as it represents the sign bit, the bit-planes are initially rearranged by doing a circular one-bit left shift as follows so that all the least sparse bit-planes can be gathered on LSB side:

$$D'_{(i,j)} = 2 \times |D_{(i,j)}| + \begin{cases} 1, & \text{if } D_{(i,j)} < 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.3}$$

where $|.|$ represents the absolute value of '.'. Thus processing, the modified difference image (D') is obtained.

To clearly understand the difference image generation process, an illustrative example is presented in Fig. 4.2. In the example, a 6×6 image (I) is considered and the pixels are predicted using MED predictor. For the first row and first column, the pixels are predicted directly by their last pixel such that $P_{(1,j)} = I_{(1,j-1)}$ and $P_{(i,1)} = I_{(i-1,1)}$, as shown in the upper and left part of the image, respectively. Thus, only the very first pixel i.e., 169, is required to be retained as the reference pixel (PR). Next, the differences are calculated for each pixel using Eq. (4.1) and the bit-planes of the matrix D as shown in Fig. 4.2c are rearranged using Eq. (4.3) to get the modified difference image D' as depicted in Fig. 4.2d.

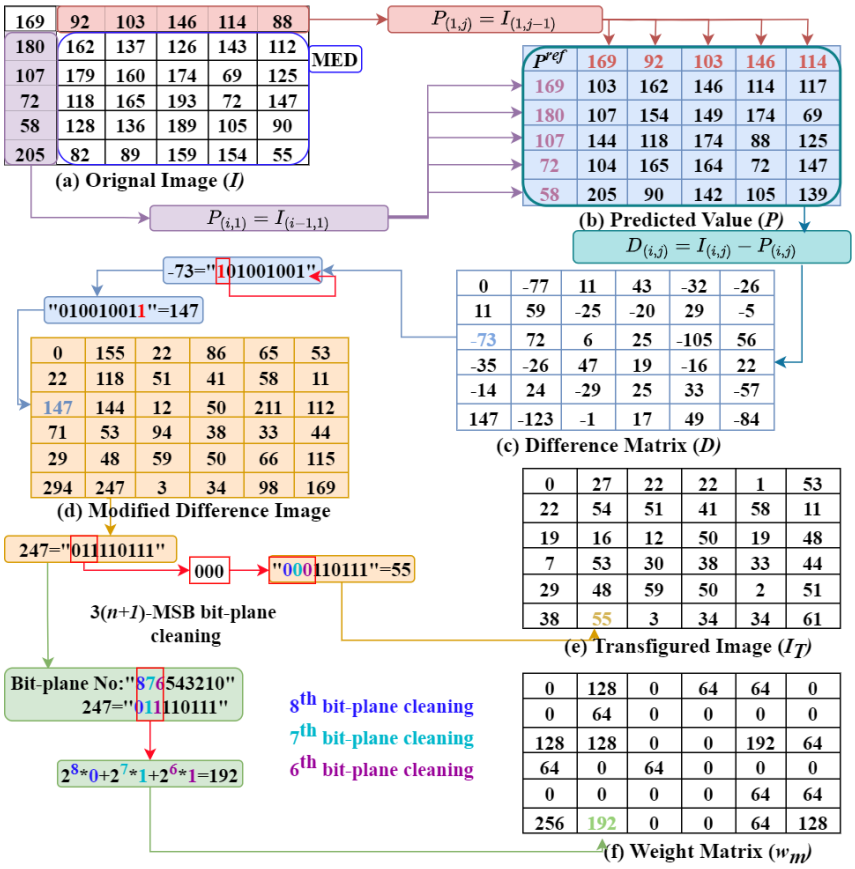


Fig. 4.2: An illustrative example for difference image generation and cleaning process

B. Multi-MSB Cleaning

The bit-plane sparsity is directly related to the embedding space or room. More specifically, the sparser the bit-plane is the bigger the embedding space will be. As seen, the sparseness is the highest at MSB-plane i.e., the modified difference image $D'_{e^{[8]}}$ which keeps on decreasing towards the LSB-planes i.e., $D'_{e^{[0]}}$. The MSB-planes of D' generally contain 0's, with few exceptions i.e., 1's because of the prevalent spatial correlation among the adjacent pixels of the original image. These exceptions can be avoided by cleaning the bit-planes so that a large room can be created for data embedding. More specifically, the 1's in the selected bit-planes are substituted by 0's to make them cleaned so that secret information can be embedded. However, a bit-plane $D'^{[k]}$ is assumed to be $Clean$ if all of its preceding bit-planes are also cleaned along with satisfying the property of cleanliness i.e., $D'^{[k]}(i,j)$ ='0' where $1 \leq i \leq M$,$1 \leq j \leq N$, which means all the bits of the bit-plane must be '0'. The cleaning process starts from the sparsest bit-plane i.e., MSB-plane (8^{th}) and continues to move towards LSB-plane of D' until Eq. (4.4) is true. The Eq. (4.4) suggests that the pure EC provided by n bit-planes is less than the same of $(n-1)$ bit-planes, or in other words, overhead in the form of SI added by n^{th} bit-plane is more than the added bits of the secret data. As a result, the n^{th} bit-plane cannot be

used to boost the EC.

$$EC^{PET}(n) \leq EC^{PET}(n-1), \tag{4.4}$$

where $8 \geq n \geq 1$ and EC^{PET} denotes pure EC that can be computed using Eq. (4.5) defined as follows:

$$EC^{PET}(n) = (M \times N \times n) - L^{SI}, \tag{4.5}$$

where L^{SI} represents the length of SI. The details about SI are described in the next subsection 4.1.1. Thus, the value of n which represents the number of embeddable bit-planes, is determined. The value of n is dependent on the characteristics of the original image. More specifically, the higher the smoothness of the original image, the greater the value of n. Next, the aforementioned cleaning process is performed with the help of Eq. (4.6) for $n+1$ MSB-planes of D'.

$$I_{T(i,j)} = \begin{cases} D'_{(i,j)} - 2^k, & \text{if } f(D'_{(i,j)}, k) = 1 \\ D'_{(i,j)}, & \text{otherwise} \end{cases} \tag{4.6}$$

where $f(x,y) = \left\lfloor \frac{x \bmod 2^{y+1}}{2^y} \right\rfloor$ is a function for extracting the y^{th} LSB bit of x, 'mod' is modulo operator and k represents the bit-plane number such that $8 \geq k \geq (8-n)$. For ensuring the reversibility, these replacements (from 1's to 0's during the cleaning) are iteratively recorded by making modifications in a zero matrix namely weight (w_m) of size $M \times N$, as per the Eq. (4.7) defined as follows:

$$w_{m(i,j)} = \sum_{k=8}^{8-n} D'_{(i,j)} \times 2^k, \quad \text{if } f(D'_{(i,j)}, k) = 1. \tag{4.7}$$

Thus, for each replacement in the k^{th} bit-plane, weight is increased by 2^k. Hence, the values of weights in w_m can lie in range (0 to 511). To demonstrate this, the example shown in Fig. 4.2 is further considered, where 3 MSB-planes of D' are cleaned by replacing all the 1's by 0's, to get the cleaned (or transfigured) image I_T as shown in Fig. 4.2e. At the same time, corresponding replacements for the cleaned bits are recorded in the weight matrix as shown in Fig. 4.2f, with the help of Eq. (4.7).

C. Compressible List Generation using Location Difference Technique

As evident from the weight matrix w_m of the example-1 and also in general, the count of '0' elements, remains high as replacements are done only for $n+1$ MSB-planes of D'. This sparsity presents an opportunity to reduce the size of w_m, which is essentially an overhead, by compressing it. However, the number of unique elements in the weight matrix w_m is limited, and they are scattered in small groups due to the correlations. If their location in the same sequence is available, the weight can be collected in a list after skipping the 0' elements. To achieve an optimal compression rate, a new technique called the LDT has been developed. This

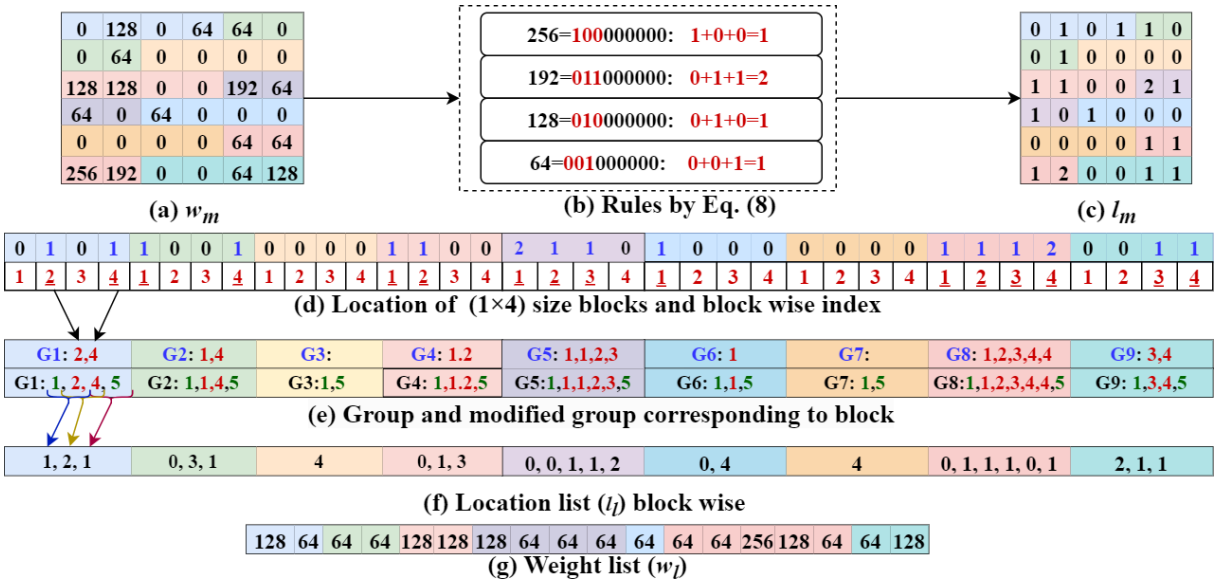


Fig. 4.3: An illustrative example of location difference technique

technique collects the given weights in a weight-list by utilizing their location, enabling optimal compression performance through Huffman encoding.

More specifically, the LDT first generates a location matrix (l_m) to mark the locations of replacements as well as the number of replacements, according to Eq. (4.8), defined as follows:

$$l_{m(i,j)} = \sum_{k=8}^{8-n} f(w_{m(i,j)}, k). \tag{4.8}$$

Next, both the matrices w_m and l_m are uniformly partitioned into $\alpha \times \beta$ sized non-overlapping blocks, where α and β are user-defined thresholds such that $\alpha \ll N$ and $\beta \ll M$, where $\ll$ represents "significantly lesser than" sign. Now, each block of l_m is scanned in raster scan manner and groups are formed, where the location of each element is marked as many times as its magnitude. For example: if the first element is '2', or '1' or '0' then two times '1' or one time '1' or no entry/representation in the corresponding group, is made, respectively. Subsequently, '1' and $(\alpha \times \beta) + 1$ are added as the first and the last element of each group. Now, the difference between two consecutive elements of each group is computed and all the differences are concatenated to form a list namely location list (l_l). Similarly, w_m is first divided into blocks of the same size as l_m was partitioned and then each block of w_m is also scanned in raster scan manner. Next, every element is represented in 9-bit binary representation. For every non-zero bit, its position value is added into a list namely weight list (w_l). Thus, obtained w_l represents the magnitude of change due to replacement and the l_l represents the corresponding location at which changes have been made in the same order. It can be observed that the image partitioning into blocks of $(\alpha \times \beta)$ helps in significantly lowering the range and enhance the frequencies of elements in the lists. As the new range of elements in the location list is (0 to $\alpha \times \beta$) which is significantly lesser than the range if no division is considered i.e., (0 to $M \times N$).

Thus, the frequencies of elements (representing the difference) are significantly increased which will immensely help while employing the compression method.

To clearly demonstrate the process, example-1 has been extended in example-2 as depicted in Fig. 4.3. The same weight matrix w_m depicted in Fig. 4.2e has been considered in Fig. 4.3a of the example-2. The location matrix (l_m) as depicted in Fig. 4.3c, is generated with the help of Eq. (4.8), for indicating number of 1's replacements done during cleaning for each pixel as shown in Fig. 4.3b. Next, the matrix is divided into blocks of size 1×4 as shown in Fig. 4.3d using different colors, by considering $\alpha = 1$ and $\beta = 4$. Thus, a total of 9 blocks are obtained as $B_1\{0,1,0,1\}$, $B_2\{1,0,0,1\}$,..., $B_9\{0,0,1,1\}$ if the image as well as elements of the blocks are scanned in raster scan manner as shown in first row of Fig. 4.3e. Afterwards, groups are formed as $G_1\{2,4\}$, $G_2\{1,4\}$, $G_3\{\}$, $G_4\{1,2\}$, $G_5\{1,1,2,3\}$, $G_6\{1\}$, $G_7\{\}$, $G_8\{1,2,3,4,4\}$ and $G_9\{3,4\}$, where the location of each element in (l_m) is marked as many times as its magnitude as shown in Fig. 4.3e. Next, all the groups are altered to have 1 and $(\alpha \times \beta) + 1$ i.e., $(1 \times 4) + 1 = 5$ as their first and the last element, respectively as depicted in the second row of Fig. 4.3e. Now, the difference between two consecutive elements of each group is computed as shown in Fig. 4.3f and all the differences are concatenated to form a list namely l_l. Thus, l_l =1,2,1,0,3,1,4,0,1,3,0,0,1,1,2,0,4,4,0,1,1,1,0,1,2,1,1 is obtained.

As discussed above to get the w_l, w_m is also first divided into blocks of same size 1×4 as represented in Fig. 4.3a by different colors. Next, the elements of each block are represented in 9-bit binary form and for every non-zero bit, its position value is added to a list namely the weight-list (w_l). Thus, w_l =128, 64, 64, 64, 128, 128, 128, 64 , 64, 64, 64, 64, 64, 256, 128, 64, 64, 128 is obtained as shown in Fig. 4.3g, where 191 is composite in 128 and 64.

D. Overhead Reduction

The generated lists l_l and w_l from LDT are now compressed using Huffman encoding. Since w_l represents the bit position at which replacements (from 1's to 0's,) were done during the cleaning process, the number of distinct elements is only $n + 1$. Moreover, their frequency is also in decreasing order. More specifically, the lower magnitude elements are the most repetitive and the higher magnitude ones are the least. So, a pre-fix Huffman tree of the maximum height n with $2n + 1$ nodes can be used to provide optimal compression. As a result, the length of the Huffman codes will be $1, 2, \ldots, n$, and n bits where the most frequent and the least frequent are represented by '1' Afterward bit Huffman code. Therefore, total $n(n + 3)/2$ bits are required to represent the prefixed Huffman code. Next, w_l is turned into bit-stream (b^{wgt}) by replacing each element with its determined Huffman code, for further processing. Similarly, l_l can also be compressed. Though l_l contains only $(\alpha \times \beta) + 1$ unique elements but their frequencies are highly random. Because of this, the Huffman tree cannot be fixed but devised based on their frequencies. Thus, Huffman code rules of [$\lceil \log_2((\alpha \times \beta) + 1) \rceil \times ((\alpha \times \beta) + 1)$+ length of code] are obtained, where each $\lceil \log_2((\alpha \times \beta) + 1) \rceil$ bit represents any of the code. The codes are concatenated to form a code rule bit-stream (h^l), Finally, l_l is converted into a bit-stream

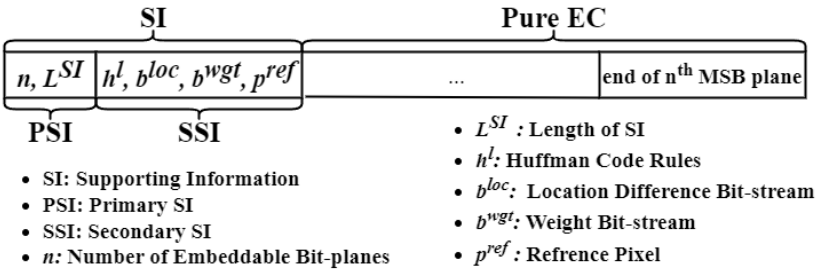


Fig. 4.4: Supporting information management

Table 4.2: Performance of LDT, Huffman Encoding (HE) and Arithmetic Encoding (AE)

Supporting information ($bits$) **/ Computational time** ($seconds$)	**Compression technique**		
	LDT+HE	**HE**	**AE**
Matrix w_m **of Lena**	219655/0.037247	321070/0.027705	231032/1.708585
Matrix w_m **of Airplane**	255943/0.03728	339695/0.029919	271329/ 1.965141

(b^{loc}) by replacing each element with its Huffman code. It is to be noted that the Huffman (or even Arithmetic) encoding can also be directly employed without first encoding with LDT, but directly encoding with these methods won't help much as the total number of toinct values in w_m are 2^{n+1}. To prove the point and show the contribution of the proposed LDT, experimental results to show the compressed size and running time for all the three encoding methods, are provided in Table 4.2. For the experiments, two standard test image $Lena$ and $Airplane$ have been considered. It is clearly visible from Table 4.2 that the LDT method significantly helps in achieving the optimal compression performance with computation time.

Thus processing, different pieces of information such as number of embeddable bit-planes (n), Huffman code rules (h^l), location difference bitstream (b^{loc}), weight bitstream (b^{wgt}), reference pixel (PR), etc., have been obtained which must be recorded in a specific order so that image recovery (at the receiving end,) can be ensured. The following information is referred to as SI, and its length is denoted by L^{SI}. SI is separated into two sections based on the need: Primary Supporting Information (PSI) and Second Supporting Information (SSI) as shown in Fig. 4.4. PSI includes only n and L^{SI} of size i.e., $[6 + \lceil \log_2(M \times N) \rceil]$ bits, which is necessary for data hiding, data extraction, and image reconstruction. SSI includes h^l, b^{loc}, b^{wgt} and PR which are concatenated in the specified sequence and is required for image reconstruction. In the next stage, the content owner encrypts the obtained transfigured image with an image key while also embedding the SI and reserving space for data embedding.

4.1.2 Image Encryption and SI Embedding

For encrypting the image contents, the content owner uses a stream cipher to encrypt the original image I_T similar to Subsection 3.2.1 (Chapter-3). The generation of a pseudo-random matrix (ξ) of size $M \times N$ utilizing the image key (K_I) as the seed.

Now before applying the mandatory encryption, the SI is embedded in I_T so that this information can also be encrypted. Therefore, enhanced security against the unauthorized decoding of the image can be achieved. The SI is embedded in a raster scan manner from the MSB-plane to the LSB-plane of I_T by the bit-replacement technique of Eq. (4.9), defined as follows.

$$I'_T(i,j) = I_T(i,j) + \begin{cases} ((b^{SI} \times 2^{k-1}), & \text{if } f(I_T(i,j),k) = 0 \\ ((b^{SI} - 1) \times 2^{k-1}), & \text{otherwise} \end{cases} \tag{4.9}$$

where $k = 7, 6, \ldots, 8-n$ and b^{SI} represents to be embedded bit of SI. The first case of Eq. (4.9) always holds for SI embedding, as the Multi-MSB of I_T is always zero. The second case, on the other hand, addresses the issue of overflow that may occur during data embedding. Following the aforementioned embedding of SI, the encryption is performed on image I'_T using Eq. 3.9 and image $I_{e'}$ is obtained. It is to be noted that initial $[6 + \lceil \log_2(M \times N) \rceil]$ MSBs in raster scan are not encrypted by Eq. (3.9) because it contains PSI, which is required during data hiding.

4.1.3 Embedding of Secret Data

In this subsection, the data hider first read the PSI to know the number of embeddable bit-planes (n) that can hold data and the length of SI (L^{SI}) from the encrypted image ($I_{e'}$) with SI, prior to data hiding. For this, the data hider extracts $[6 + \lceil \log_2(M \times N) \rceil]$ bits using the function $f(I_{e(i,j)}, k)$, by reading the image in raster scan manner where the initial 3 bits give n and the remaining $[3 + \lceil \log_2(M \times N) \rceil]$ bits represent L^{SI}. Next, the secret data is encrypted using the data key (K_D) and embedded in the image $I_{e'}$ using a bit replacement technique or by Eq. (4.9) after bypassing the pixels having SI bits. The end result is a marked encrypted image (I_m).

4.1.4 Data Extraction and Image Recovery

As the proposed DIT-RDHEI method supports separability property of RDHEI, all the three cases 1) the receiver possesses only the data key; 2) the receiver possesses only the image key; and 3) the receiver possesses both the keys can be considered. In any of the case, PSI is first read/extracted in the raster scan from the MSB-plane to the LSB-plane from the image I_m as done by the data hider.

Case-1: the recipient has access only to the data key K_D which means he is exclusively responsible for the data extraction. In this situation, the recipient reads the bits of all the n bit-planes like PSI extraction, by skipping the first L^{SI} bits and then decrypts the above remaining bits with K_D to obtain the lossless data.

Case-2: the recipient has access only to the image key (K_I), which means he can only reconstruct the identical image. For this, a pseudo-random matrix (ξ) of size $M \times N$ using key K_I is generated and image decryption is performed using Eq. (3.13) to get the decrypted image (I_d). Next, the SSI is extracted from I_d in a manner identical to the PSI extraction,

where the length of SI is L^{SI}, which is already obtained from PSI. After skipping the PSI bits, $\lceil \log_2((\alpha \times \beta) + 1) \rceil \times ((\alpha \times \beta) + 1)$ bits are obtained from the SSI, where the decimal value of each $\lceil \log_2((\alpha \times \beta) + 1) \rceil$ bits sequentially represents each code length of the Huffman code rule. Following that, Huffman coding rules are extracted from the leftover SSI based on the lengths previously stated for each code.

Using the remaining SSI and the previously obtained Huffman code, $M \times N$ dimensional location matrices with initial zero values are then created. To convert the remaining bits of SI to l_m, Huffman code values are successively read from SI and after decoding substituted in a $(\alpha \times \beta)$-sized block of l_m during raster scan, with each block also being contracted during raster scan.

After updating the entire l_m blocks, prefixed Huffman codes for weight matrix are determined similarly to section 4.1.1. Following that, w_m are recovered from the leftover SSI. Because the Huffman code for weight is present, the Huffman value is retrieved from the SSI and decoded. Then weights are updated in w_m for every non-zero l_m in raster scan manner After detaching all the weights, only 8 bits are left in SSI, which reflect the reference pixel (PR). Next, the cleaned image I_T is determined from LSB planes of I_d. Then, to obtain the modified difference image (D'), n MSB-planes of the I_T are replaced by the weight matrix (w_m) and image D' is obtained. Afterwards, bit-planes rearrangement is reversed to obtained the difference matrices D as follows:

$$D(i,j) = \begin{cases} +\frac{D'(i,j)}{2}, & \text{if} D(i,j) \bmod 2 = 0 \\ -\lfloor \frac{D'(i,j)}{2} \rfloor, & \text{otherwise} \end{cases} \tag{4.10}$$

For the image reconstruction, reference pixel (PR) is substituted at $\bar{I}(1,1)$, first row and column is obtained by following substitution, $\bar{I}(i,1) = D(i,1) + \bar{I}(i-1,1)$ where $2 \leq i \leq M$, $\bar{I}(1,j) = D(1,j) - \bar{I}(1,j-1)$ where $2 \leq j \leq N$; and remaining pixels are predicted by MED predictor, and $\bar{I}(i,j) = D(i,j) + \bar{P}(i,j)$, where $\bar{P}(i,j)$ is the predicted value. The obtained final image is identical to the original image which indicates lossless image reconstruction and in turn proves reversibility of the DIT-RDHEI.

Case-3: if the data-hiding key K_D and the image key K_I are available with recipient then he can extract the hidden data and can also reconstruct the original image without loss by following the first case and the second case, respectively.

4.2 Experimental Results and Analysis

This section discusses the experimental results of the proposed (DIT-RDHEI) method and their analysis on comparison with the SOTA methods. The discussion and analysis of the experimental results has been divided into five parts detailed as follows. In the first section, a comprehensive explanation of the DIT-RDHEI using the test image $Lena$ as an example is provided.

The second part discusses performance analysis on different test images and datasets such as BOSSbase [38] and BOWS-2 [39]. In the third part, the security analysis of the method under consideration is done followed by comparison of the DIT-RDHEI with the SOTA, in the penultimate part. In the last part, time complexity analysis is done.

4.2.1 A Complete Example for the DIT-RDHEI Method

In this sub-section, a comprehensive illustration of the proposed DIT-RDHEI method is provided. For this, grayscale test image $Lena$ of 512×512 pixel as depicted in Fig. 4.5a is considered. The histogram of the image $Lena$ to show the frequency distribution of 256 levels of grayscale, is shown in Fig. 4.5b. As discussed in proposed work, the DIT-RDHEI first generates the difference image which is shown in Fig. 4.5c, with the help of MED predictor. It can be observed that the difference image adds to the visual security aspects as it hardly shows the original contents of the cover image. Next, the sign bit-plane (as it is highly random which is also evident from Table 4.1), is relegated to the LSB position and the bit-plane cleaning process is initiated from the new MSB-plane to fully utilize the correlation/sparsity. Initially, $n = 1$ is considered which means top two MSB-planes i.e., $n + 1$ are to be cleaned. During the cleaning of the eighth and seventh planes, 123 locations are cleaned. To clearly demonstrate the sparseness of bit-planes, all the cleaned locations are labelled as '1' in a binary image as depicted in Fig. 4.5d. The image shows that almost all of the pixels are black except very few whites, which in turn indicates the greater sparsity. However, this cleaning process results into SI mentioned in Table 4.3. As discussed above, the cleaning process is continued until Eq. (4.4) is true. For the image $Lena$, it has been found that for $n = 5$ the pure EC decreases as the SI is determined to be of the length 502566 bits in comparison to 219655 bits for $n = 4$. The binary images for each of the bit-planes cleaning i.e., n=2,3,4 and 5 are shown in Fig. 4.5e-4.5h to demonstrate the sparsity of each of the bit-planes. It can be clearly observed that white regions keep on increasing as the value of n increases which means sparsity keeps on decreasing. The binary image displayed in Fig. 4.5h showcases a notable excess of white pixels, signifying a substantial generation of overhead in terms of SI, which violates Eq. (4.4) due to the high overhead. Next, after cleaning the top MSB n+1 bit-planes of the image, a transfigured image I_T is obtained as shown in Fig. 4.5i. The histogram of the transfigured image is shown in Fig. 4.5k, where it can be clearly observed that pixel values are lying only in the range of 0 to 15 as 4-MSB-planes of the difference image have already been cleaned to get this transfigured image. The transfigured image is divided into $\alpha \times \beta$ blocks. To clearly understand the effect of different block sizes with $n = 4$, experimental data which depicts the bifurcation of SI, EC, and ER for $Lena$, is presented in Table 4.4. It can be observed that the image's pure EC increases until the block size reached 12×12, afterwards it begins to drop.

Next, SI of 219655 bits is embedded and the resultant image as shown in Fig. 4.5l, is obtained. Next, the image is encrypted using stream cipher as per Eq. (4.10). The encrypted

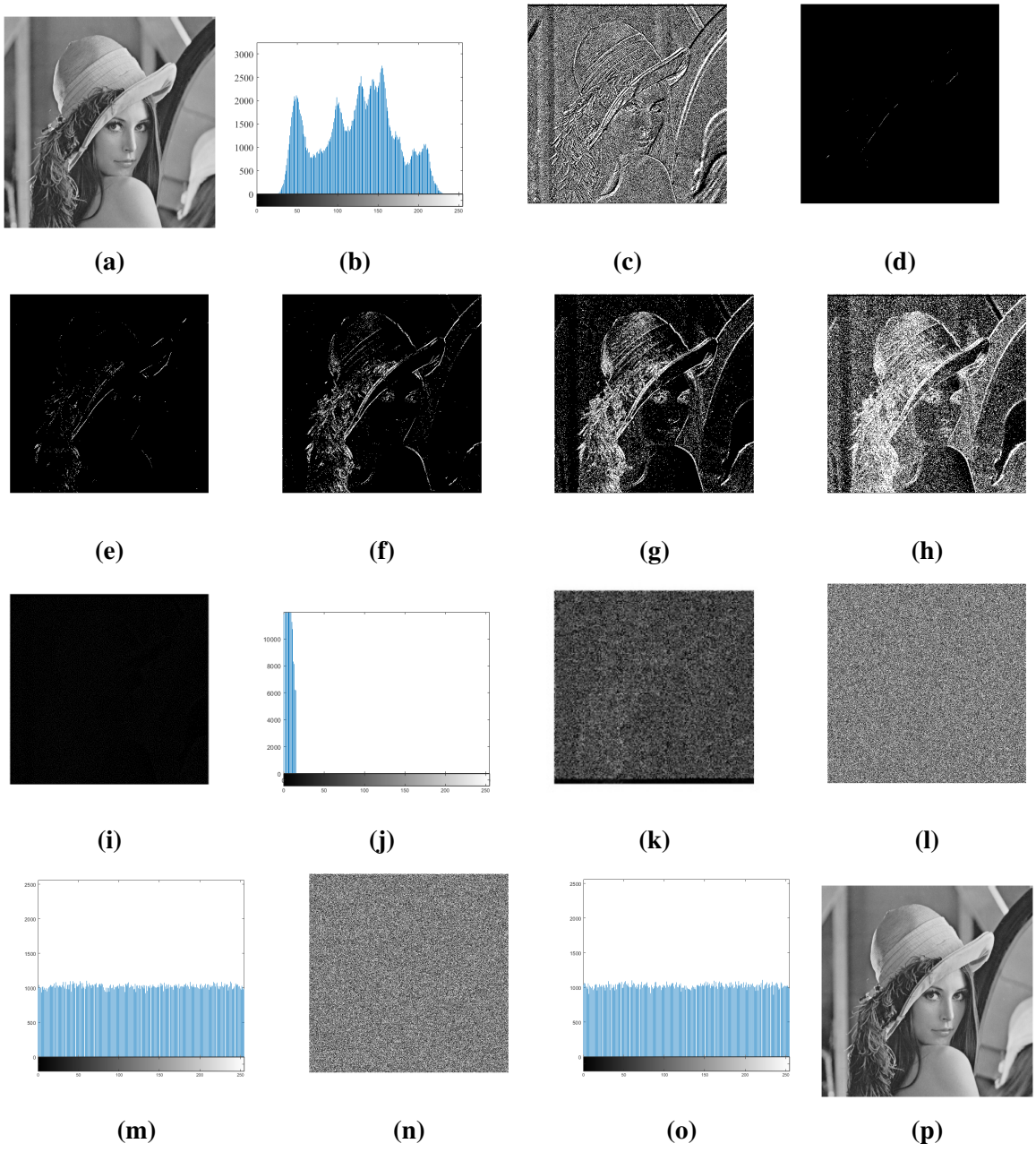


Fig. 4.5: Illustration of the proposed DIT-RDHEI method on image $Lena$ of size 512×512 (a) Original image I, (b) Histogram of original image I, (c) Difference image D, (d-h) Binary image to demonstrate the level of sparsity for n =1, 2, 3, 4, and 5, (i) Transfigured image I_T, (j) Histogram of image I_T (k) SI enabled image I'_T, (l) Encrypted image $I_{e'}$, (m) Histogram of $I_{e'}$, (n) Marked encrypted image I_m after embedding the 828921 bits of secret data, (q) Histogram of image I_m and (r) Recovered lossless image

Table 4.3: SI bifurcation, pure EC and ER for different values of n

n	**Size of h^l ($bits$)**	**Size of b^{loc} ($bits$)**	**Size of b^{wgt} ($bits$)**	**Size of SI ($bits$)**	**Pure embedding capacity ($bits$)**	**Embedding rate (bpp)**
1	6498	2894	123	9547	252597	0.9636
2	2705	15248	1971	19956	504332	1.9239
3	2672	56258	13452	72414	714018	2.7238
4	2842	160067	56714	219655	828921	3.1621
5	5318	317119	180097	502566	808154	3.0829

Table 4.4: SI bifurcation, pure EC and pure ER for increasing value of $\alpha \times \beta$ with $n = 4$

$\alpha \times \beta$	**Size of h^l ($bits$)**	**Size of b^{loc} ($bits$)**	**Size of b^{wgt} ($bits$)**	**Size of SI ($bits$)**	**Pure embedding capacity ($bits$)**	**Pure embedding rate (bpp)**
2×2	29	227150	56714	283928	764648	2.9169
4×4	170	190191	56714	247107	801469	3.0574
8×8	1040	167637	56714	225423	823153	3.1401
10×10	1747	163411	56714	221904	826672	3.1535
11×11	2204	161637	56714	220587	827989	3.1585
12×12	2842	160067	56714	219655	828921	3.1621
13×13	3464	159492	56714	219702	828874	3.1619
14×14	4202	158835	56714	219783	828793	3.1616
16×16	7163	157570	56714	221479	827097	3.1551
20×20	25148	155959	56714	237853	810723	3.0927

image and its histogram are presented in Fig. 4.5m and 4.5n, respectively, which clearly shows complete distribution as compare to the original image. Afterwards, the data hider embeds the secret information i.e., 828921 bits in the encrypted image. The marked encrypted image and its histogram are depicted in Fig. 4.5o and 4.5j, respectively. Again, it is evident from the histogram that there is no correlation between the original image and the encrypted-marked image. Following this, the receiver extracts hidden data and recovers the original image based on key availability. As shown in Fig. 4.5p, the result is identical to Fig. 4.5a, proving the method's reversibility.

4.2.2 Performance Analysis on Different Test Image and Dataset

In this sub-section, the efficacy of the proposed DIT-RDHEI method is evaluated. For performance evaluation, commonly used metrics: PSNR and EC in terms of ER, are adopted to test the reversibility and measure the payload capacity of the method, respectively. Firstly, six standard test images: $Lena$, $Baboon$, $Airplane$, $Jetplane$ (F-16), $Tiffany$ and Man (Male), each of size 512×512 pixels as shown in Fig. 1.3, from SIPI dataset [41], are considered. The experimental results are presented in Fig. 4.6 where EC (in bpp) is displayed for different block sizes considering different values for $\alpha \times \beta$. In this analysis, $\alpha = \beta$ has been considered

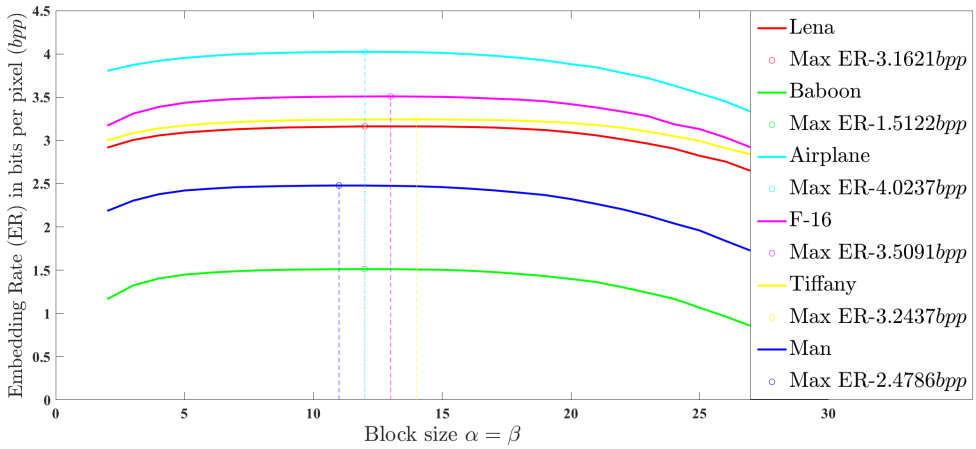


Fig. 4.6: Performance analysis of the proposed DIT-RDHEI method on test images on different thresholds

for simplicity. It is to be noted that the achieved PSNR by the DIT-RDHEI between the original and the recovered image for all the test images is positively infinite ($+\infty$), which proves that the DIT-RDHEI is a pure reversible data hiding method. Since the achieved PSNR (in all the cases) is $+\infty$ dB, the same is not shown in figure to avoid meaning less entry. As shown in Fig. 4.6, a smooth image such as $Lena$ provides greater ER than a complex image such as $Baboon$. It is because the number of the embeddable bit-planes in $Lena$ is greater than the same in $Baboon$. In other words, the maximum ER i.e., 4.0237 is achieved in case of the smoothest image i.e., $Airplane$ and in contrast, the minimum ER i.e., 1.5122 is provided by the most complex image i.e., $Baboon$. This indicates that the DIT-RDHEI method makes effective use of spatial correlation. As a result, approximately half of the bits (i.e., 4 bits) in a smother image are easily utilized for embedding while maintaining the identical image, and it is also analyzed that the EC is highest when the threshold ($\alpha \times \beta$) is in the range of 11×11 to 14×14, so it is fixed as 12×12 for further analysis. Further, the DIT-RDHEI is evaluated in two datasets: BOSSBase and BOWS-2, both having 10,000 grayscale images with size 512×512. In Table V, the experimental results for ER at $\alpha = \beta = 12$ are presented. The obtained PSNR is $+\infty$ dB, which again proves the reversibility aspect. For BOSSBase and BOWS-2, net ER is 3.7010 and 3.5784 *bpp*, with PSNR $+\infty$ dB, respectively, in the average case. The greatest ER i.e., 6.8181, 6.6604 and 5.9598 *bpp* is achieved by three images at index number 5162, 5215 and 5542 *bpp*, respectively in the BOSSBase dataset. Similarly, in BOWS-2 dataset, image at index 1478, 8414 and 1559, provide the highest ER i.e., 6.3224, 5.9301 and 5.7915 *bpp* respectively. Although the ER of worst image in BOSSBase is 0.9410 *bpp* while in BOWS-2, it is 0.8439 *bpp*. This analysis proves that the DIT-RDHEI has acceptable performance and can be applied in privacy protection.

Table 4.5: Statistical analysis on image $Lena$

Image	Original image (I)	Encrypted image ($I_{e'}$)	Marked encrypted image (I_m)
Horizontal correlation	0.9719	0.0034	-0.0001
Vertical correlation	0.9850	0.0010	0.0048
Diagonal correlation	0.9593	-0.0027	0.0014
Information entropy in (8-bpp)	7.4451	7.9993	7.9993
NPCR (I and $I_{e'}$/I_m) in (%)	-	99.61	99.60
UACI (I and $I_{e'}$/I_m) in (%)	-	73.00	72.95
PSNR (I and $I_{e'}$/I_m) in (dB)	-	9.2311	9.2363
SSIM (I and $I_{e'}$/I_m)	-	0.0374	0.0364

Table 4.6: Experimental results on two datasets at $\alpha = \beta = 12$ and $+\infty dB$ PSNR

dataset	Average ER(bpp)	Best ER (bpp)	Worst ER(bpp)
BOSSBase	3.7010	6.8181	0.9410
BOWS-2	3.3353	7.1584	0.6666

4.2.3 Security Analysis

The DIT-RDHEI method is designed to ensure security for both the hidden data and the image. For this, the method uses two keys: an image key for cover image encryption and a data key for data encryption. A suite of statistical metrics, including horizontal and vertical correlations [35], Shannon entropy [36], NPCR [37], UACI [37], PSNR, and SSIM [34], serves as the benchmark to assess its performance concerning the mage. Performance metrics based on these benchmarks for the test image, $Lena$, are delineated in Table 4.5.

Figures 4.5a, 4.5m, and 4.5o visually represent the original, encrypted, and marked encrypted versions of $Lena$, respectively. Both encrypted versions, $I_{e'}$ and I_m, appear substantially obfuscated, showing the imperceptibility of the original content. The histogram distributions for encrypted images, displayed in figures 4.5o and 4.5j, are notably uniform, different from the histogram of I in Fig. 4.5b. This signifies that the DIT-RDHEI achieves commendable perceptual security.

The I shows a high correlation across horizontal, vertical, and diagonal axes orientations, as shown in Table 4.5. From Table, the correlation for $I_{e'}$ and I_m veers close to nil. This suggests the efficacy of the method in disrupting I's inherent correlation. Information entropy measures the uncertainty in the distribution of pixel intensities within encrypted images. In grayscale images, intensities span from 0 to 255, capping entropy at 8. A larger entropy denotes a more uniform pixel distribution. As shown in Table 4.5, the entropy of the encrypted version is near 8, denoting their unpredictable nature.

NCPR and UCID also determine the percentage of image pixel change. The NPCR of I_m is close to 100%, i.e., 99.60, indicating a high secure image, as the greater number of NPCR denotes a more mismatch. Despite the fact that the percentage value of UCID is higher, visual

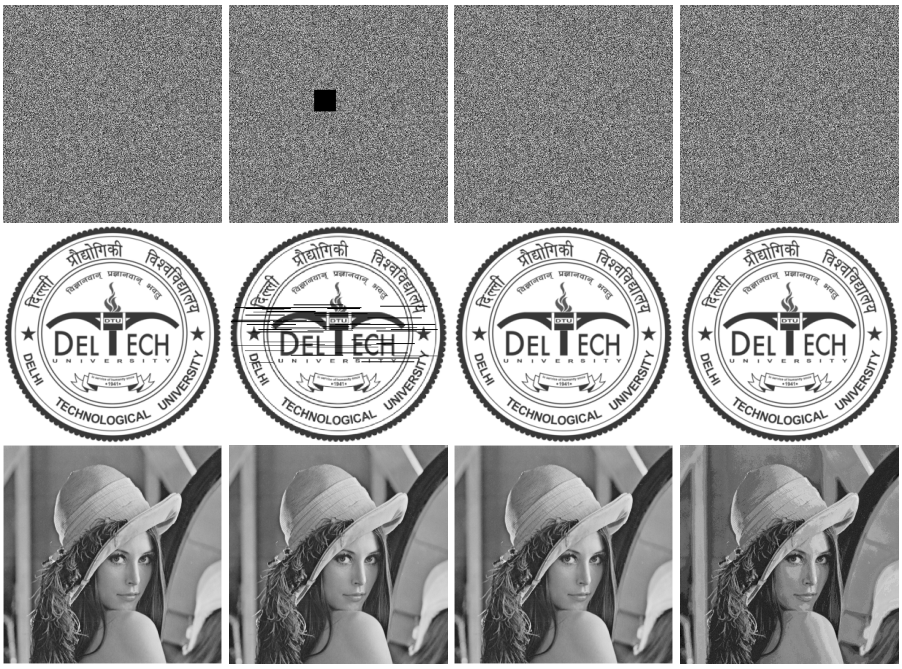
security is considered to be of high quality. The UCID for $I_{e'}$ and I_m is 73.00 and 72.95, respectively, as indicated in Table 4.5. Lastly, the PSNR and SSIM for $I_{e'}$ and I_m were analyzed, primarily to discern the extent of dissimilarity. Both metrics are diminished for the said images relative to I, confirming their significant divergence and structural distinctness from the original, I.

Moreover, the DIT-RDHEI has been subjected to different malicious attacks, including tampering, brute-force, and collage attacks. To evaluate its robustness against tampering attacks, a logo was embedded in the encrypted $Lena$ image, and various removal attacks were conducted, including patch removal, 4^{th} bit-plane removal, and 5^{th} bit-plane removal. The results of these experiments can be observed from Fig. 4.7. The top row depicts the I_m, the middle row shows the extracted secret logo messages, and the bottom row displays the restored cover images. Removal attacks were conducted on the second, third, and fourth columns, respectively. In each scenario, the recovered cover image shown negligible distortion, and the extracted logo has remained recognizable. Nonetheless, removing SI bits could result in complete degradation of the cover image, and removing the bit-planes housing the majority of the data complicates the recovery of the logo. The utilization of AES in image encryption, in combination with the possibility of encrypting data using any robust algorithm, renders brute-force attacks almost impossible. Lastly, during the collage attack, a new image is formed by merging portions of various encrypted images. Since the DIT-RDHEI is separable and solely dependent on the SSI for data embedding, the receiver can extract the data. Thus, it can be concluded based on the aforementioned results the DIT-RDHEI has good resilience against various forms of attacks.

4.2.4 Comparison with State-of-Art-Methods

To comprehensively and comparatively assessment the performance of the DIT-RDHEI method, some of the notable and recently developed high-capacity SOTA methods such as the works of Yu *et al.* [74], Yi *et al.* [83], Wu *et al.* [85], Yin *et al.* [76], Gao *et al.* [77], Chen *et al.* [100] and Yin *et al.* [90], have been considered. The aforementioned SOTA methods majorly belong to the RRBE-RDHEI category as RRBE-RDHEI methods usually provide higher EC than VRAE-based ones. Since the DIT-RDHEI always losslessly reconstructs the image which leads to an $+\infty$ dB PSNR with one SSIM, ER has been considered as a major performance metric for comparison with SOTA methods. To persuasively analyze the performance, the parameters for each of the aforementioned methods are tuned to get the highest ER. As evident from Fig. 4.8, the DIT-RDHEI achieves the highest ER for the $\alpha \times \beta$ ranging between 11×11 to 14×14. Moreover, the ER fluctuation within the aforesaid range is also negligible.

A comprehensive comparison of the DIT-RDHEI with leading SOTA methods was conducted across six images: $Lena$, $Baboon$, $Airplane$, $F-16$, $Tiffany$, and Man. From the observations in Fig. 4.8, the DIT-RDHEI method displayed superior performance, yielding an ER of 3.162 bpp for the $Lena$ image. This surpasses the previous best ER of 3.075 bpp reg-



(a) **(b)** **(c)** **(d)**

Fig. 4.7: The experimental results obtained under various removal attack scenarios (a). Original, (b) Patch removal, (c) Fourth bit-plane removal and (d) Fifth bit-plane removal

istered by Yin *et al.* [90]. Regarding the most complex image $Baboon$, DIT-RDHEI achieved an ER of 1.512 bpp surpassing Yu *et al.*'s [74], which registered 1.385 bpp. This indicates an enhancement of 0.1270 bpp in the ER by the DIT-RDHEI method for the $Baboon$ image. In the case of the $Airplane$ image which is a smooth texture, the DIT-RDHEI surpassed existing benchmarks by achieving an ER of 4.024 bpp, compared to 3.987 bpp for the method by Yu *et al.* [74]. This yields an incremental gain of 0.037 bpp for the DIT-RDHEI method. For images with moderate complexity like $F-16$, $Tiffany$, and Man, the SOTA method by Yin *et al.* [90] previously held the best ERs of 3.402, 3.149, and 3.305 bpp, respectively. However, the DIT-RDHEI method gave significant improvements, outperforming Yin *et al.* by increments of 0.105, 0.094, and 0.173 bpp for images, respectively.

As evident from the above comparison and analysis, two methods: Yu *et al.* [74] and Yin *et al.* [90] among SOTA methods are notable in terms of ER. However, still these methods have a limit in ER. Yu *et al.* [74] makes use of the MED predictor to fully exploit spatial correlation throughout the entire original image. The embedding level of each pixel is ascertained according to the magnitude of its prediction error. A level map is generated, to guarantee reversible and error-free data extraction. Consequently, owing to the magnitude value, fewer MSB bits are encoded. Nonetheless, the DIT-RDHEI utilizes prediction error with sign bits, which greatly aids in expanding the EC. Additionally, the proposed LDT significantly helps in reducing the size of overhead and then in turn creating large room for embedding.

Yin *et al.* [90] used the MED predictor and the sign bits of the prediction value are also

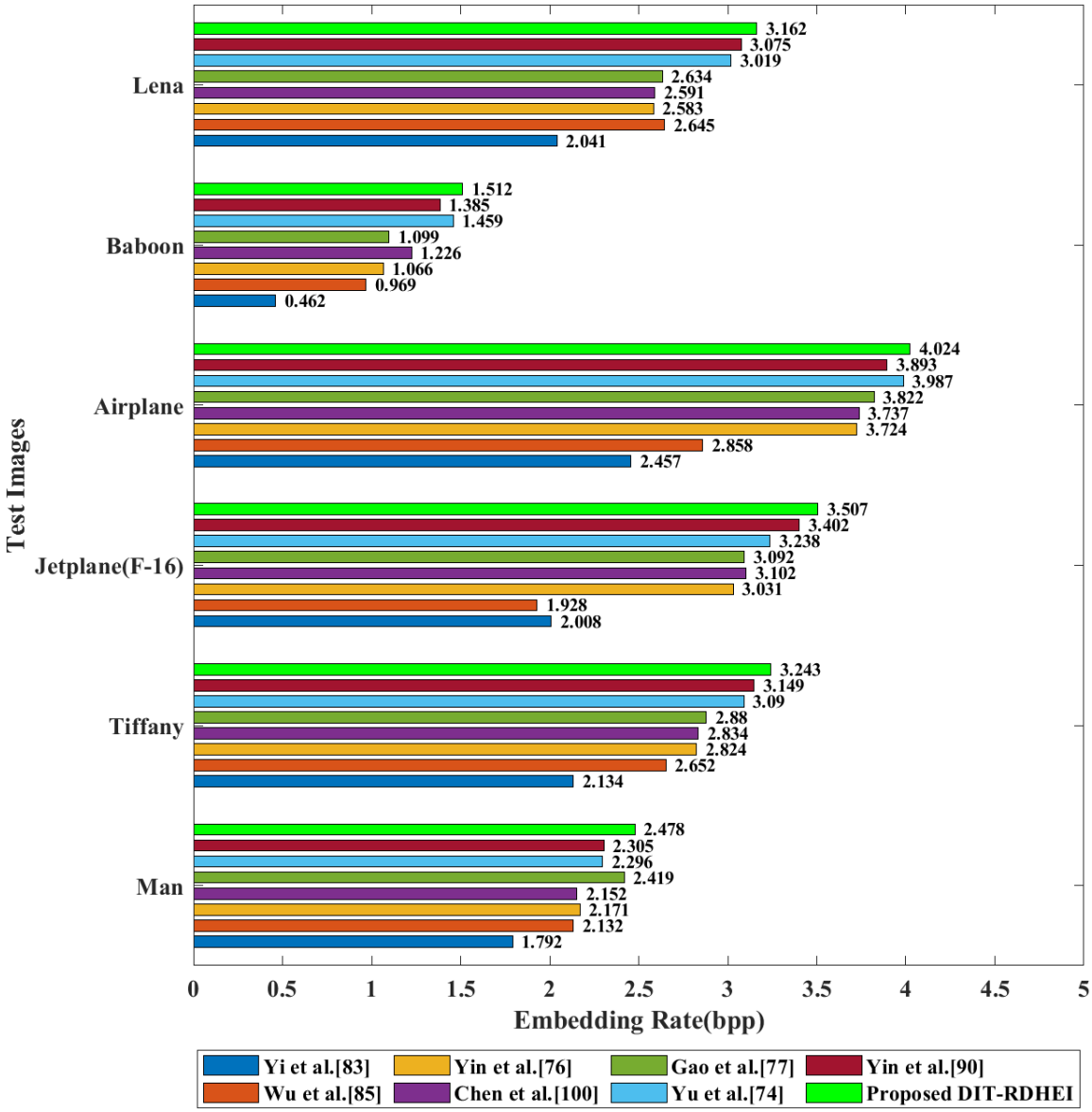


Fig. 4.8: Comparison of ER (*bpp*) on test images between proposed DIT-RDHEI method and SOTA methods

carried with the image to remove the dependencies of the original image. After shifting each '1' bit of the 8^{th} bit-plane and its location to auxiliary information, the sign bit-plane of prediction is stored in the image's 8^{th} bit-plane. In contrast, in the DIT-RDHEI, a cleaned bit-plane is configured to sign bit, and either of storing the '1' bit and their bit-plane location, the extra bit-plane is compressed. Yin *et al.* [90], on the other hand, adopts bit-plane rearrangement compression, which does not treat the dominating feature of '0' values. However, the suggested LDT accounts for the preponderance of '0' values. As a result, it is obvious that the DIT-RDHEI can make better use of image redundancies. Further, two datasets: BOSSbase and BOWS-2 have also been considered for performance comparison of the DIT-RDHEI with the SOTA methods as shown in Fig. 4.9. It is indicated that the proposed technique has higher ER than any of the existing SOTA methods [74, 76, 77, 83, 85, 90, 100] on BOSSBase and BOWS-2

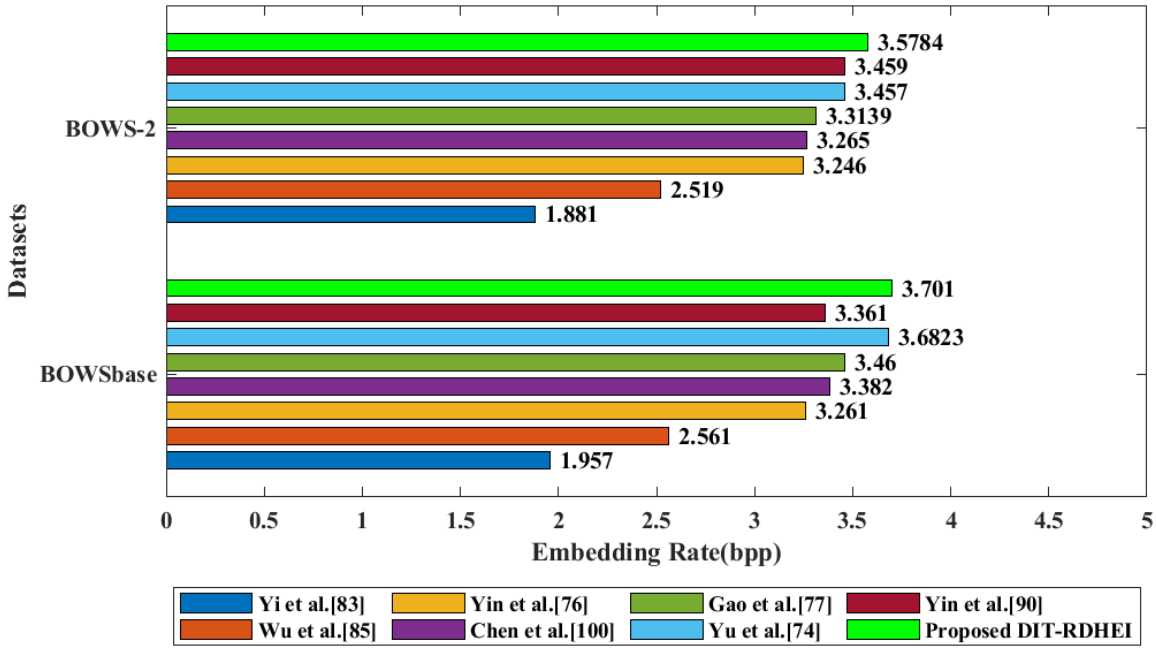


Fig. 4.9: Comparison of average ER (*bpp*) on two datasets between proposed DIT-RDHEI method and SOTA methods

datasets. It can also be observed from figure that any of the existing aforementioned methods except Yu *et al.* [74] for the BOSSbase and BOWS-2, does not surpass 3.5 *bpp* ER for any of the datasets. Thus, the DIT-RDHEI gains 0.128 and 0.019 *bpp* ER, for the BOSSbase and BOWS-2, respectively. The aforementioned extensive evaluation shows that the DIT-RDHEI completely outperforms the existing SOTA methods [74, 76, 83, 85, 90, 100].

4.2.5 Time Complexity Analysis

Given the technical difficulties with executing the code, I will manually calculate the F1 scores for each technique based on the precision and recall values you've provided:

These are approximate values calculated based on the formula for the F1 score. Please note that slight variations may occur due to rounding during manual calculation.

As discussed above, the proposed DIT-RDHEI method comprises four stages, and at each stage, at most $M \times N$ pixels and constant number of bit-planes are processed. Consequently, the worst case time complexity can be $O(M \times N)$, at each stage, because α, β and n are constant. Additionally, two Huffman trees are formed during the overhead processing, however, their number of nodes remains constant, resulting in a constant time complexity. Thus, the overall time complexity of the method is $O(M \times N)$.

To further analyze the time complexity persuasively, the evaluation of computational time for the proposed DIT-RDHEI and each SOTA method, is also performed using MATLAB software on a computer equipped with a 6-core AMD Fx-6300 processor, 8 GB of RAM, and the Windows 10 operating system. The computational time (in seconds) with maximum EC (in

Table 4.7: Comparison of computation time and EC with the Proposed DIT-RDHEI and SOTA methods

Method	Total computational time of 6-test images(seconds)	Total EC of 6-test images (bit)	Average time per embedded bit (seconds/bit)
Yi *et al.* [83]	115.9477	2878944	0.40274×10^{-4}
Wu *et al.* [85]	286.2726	3742053	0.76501×10^{-4}
Yin *et al.* [76]	200.8890	4083817	0.49196×10^{-4}
Chen *et al.* [100]	276.5286	3615464	0.76485×10^{-4}
Yu *et al.* [74]	308.0526	4573023	0.67363×10^{-4}
Yin *et al.* [90]	502.4172	4590510	1.09450×10^{-4}
Proposed DIT-RDHEI	289.9782	4700085	0.61696×10^{-4}

bits), and average time per embedded bit (in seconds/bit) taken by all the methods for six test images (depicted in Fig. 1.3,) is presented in Table 4.7. The DIT-RDHEI was found to be faster than Yu *et al.* [74] and Yin *et al.* [90], which have the highest EC among the SOTA methods. It can also be noted that the DIT-RDHEI takes lesser average time per embedded bit than the most SOTA methods except [83], [76] and, in fact, the proposed method is the fastest at high EC more specifically at EC $\geq$ 4500000 bits. Notably, the slowest method requires only 1.09450×10^{-5} sec to embed a bit among SOTA methods, and the DIT-RDHEI only takes slightly more than the half of this.

4.3 Summary

In this chapter, we proposed a new high-capacity RDHEI method based on difference image transfiguration. The proposed (DIT-RDHEI) method employed MED to generate a highly sparse difference image and transfigured the same to create a big room for embedding. Experimental results demonstrated that the proposed DIT-RDHEI method provides higher ER than the SOTA methods, while ensuring the complete reversibility of the original image. More specifically, the DIT-RDHEI provides the highest average payload of 3.5784 and 3.7010 *bpp* for BOSSbase and BOWS-2 datasets ever, respectively, while ensuring complete reversibility of the original images. Additionally, DIT-RDHEI method provides greater visual security as embedding is done in the transfigured image which is completely different than the original image.

CHAPTER 5

BIT-PLANE BASED REVERSIBLE DATA HIDING IN ENCRYPTED IMAGES USING MULTI-LEVEL BLOCKING WITH QUAD-TREE

This chapter introduces a novel method called MBQ-RDHEI to address the main challenges found in traditional RDHEI methods i.e., the limited space available for hiding data and the restricted ability to utilize redundant information. MBQ-RDHEI begins with a prediction step similar to the one described in Chapter 4, where the difference and MED predictors are utilized to generate difference matrix. Next, the selected MSB planes of difference matrix are converted into a bit-stream using a quad-tree approach. The proposed quad-tree technique differs from traditional quad-trees compression technique in that it focuses on zero values to achieve high compression of sparse bit-planes. Additionally, a novel inter-bit-plane redundancy mitigation technique is proposed to remove redundancies between the bit-planes. Furthermore, the generated bit-streams representing the quad-tree can be further condensed using Huffman encoding, if necessary. Consequently, the proposed MBQ-RDHEI method can be transformed into three variants: Direct-MBQ, Huffman-MBQ, and Separate Huffman-MBQ. This approach creates a large room or space within the difference image, which can be utilized to embed secret information. The following section provides an in-depth examination of preliminary works followed by the discussion on the proposed method.

5.1 Preliminary

In this section, we will discuss some of the existing quad-tree inspired image compression techniques which include the original quad-tree compression [101], hierarchical quad-tree coding [81], and hierarchical block variable length coding [87], in detail. These techniques utilize the spatial correlation of the original images and divide the image/bit-planes into blocks for compression. The detailed compression process of each of the aforementioned techniques is

explained with suitable examples in the following subsections.

5.1.1 Quad-tree Compression

In 1992, Markas and Reif [101] discussed an image compression method by decomposing the image into a hierarchical tree configuration, where each node of the tree signifies a specific square region/block of the image and possesses four distinct child nodes, hence the name "quad" tree. The process of image decomposition begins by determining whether the image block (which is initially the entire image itself), satisfies a predefined criterion (e.g., minimum threshold of the difference between pixel values, in the context of image compression) or not. If it does, its node is marked '$Leaf$', and no further subdivision is done as the process is assumed to be completed for that block, otherwise, its node is marked '$Internal$' and the image block is recursively divided into four smaller blocks until each sub-block satisfies the criterion or reaches a predetermined minimum size. Thus, the root node of the tree represents the entire image, and each of its child nodes represents a quadrant of the image. Next, each node of the tree is assigned a four-bit code signifying the number of child nodes it has. Additionally, each '$Leaf$' node is linked with a fixed size code to denote the average intensity of its corresponding block. To encode the image, the assigned codes are amassed following any traversal sequence, which will give a compressed bit-stream representing the complete image.

It is to be noted that the quad-tree encoding permits the use of larger blocks for representation in homogeneous areas. This adaptability is a crucial aspect of the compression process, as it allows these regions to be represented more compactly, thereby optimizing space usage. In other words, the quad-tree encoding essentially 'summarizes' these uniform areas into larger blocks, achieving significant data reduction and contributing to the overall efficiency of the compression.

5.1.2 Hierarchical Quad-tree Coding

In 2021, Liu *et al.* [81] introduced an RDHEI method based on Hierarchical Quad-Tree Coding (HQTC). The method encodes the cover image in a bit-plane wise manner starting from the MSB to the LSB, to create room for embedding. For this, all the bits of a given bit-plane are checked, if these bits are the same, or in other words, all the bits are either 0 or 1, then the complete bit-plane is encoded by the corresponding bit only as either $Leaf-0$ or $Leaf-1$, and no subdivision is executed like in 5.1.1. Otherwise, the bit-plane is recursively subdivided into four sub-blocks till the aforementioned condition is not met for each sub-block. Next, all the leaf nodes are coded with the help of four unique codes which include $depthcode$ representing the number of levels, $valuecode$ denoting the category of the leaf node ($Leaf-0$ or $Leaf-1$), $numbercode$ indicating the total number of leaf nodes, and $pathcode$ suggesting the route to the particular node. Finally, the codes of each leaf i.e., either $Leaf-0$ or $Leaf-1$, are collected

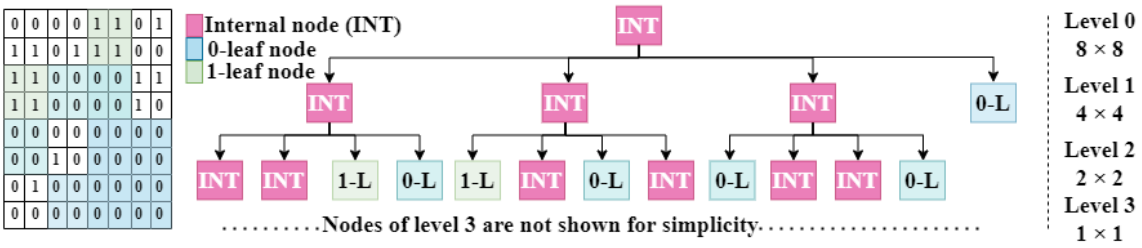


Fig. 5.1: Hierarchical quad-tree coding

for each meaningful level of the tree, to encode the complete image. Thus, a significantly big space for embedding is reserved.

In order to better elucidate the process of the HQTC, an illustrative example is presented in Fig. 5.1. The example considers an 8×8 bit-plane and generates its corresponding quad-tree structure. The tree is coded in a level-wise manner, starting with level 1, which contains only a $Leaf - 0$. This leaf is represented by '01 0 01 11', where the blue-colored bits represent the level number or $depth$ code, the yellow-colored bit represents the type of $Leaf$ or $value$ code, the green-colored bits represent the one $Leaf - 0$ node at the given level number or $number$ code, and the red-colored bits represent the path from the root node or $path$ code. Similarly, for the $Leaf - 0$ and $Leaf - 1$ of level 2, two bit-streams are generated as follows: '10 0 0100 0011 0110 100011' and '10 1 0010 0010 0100', respectively.

5.1.3 Hierarchical Block Variable Length Coding

In 2022, Xu *et al.* [87] discussed a Hierarchical Block Variable Length Coding (HBVLC) for RDHEI. The working of HBVLC seems to be inspired by the quad-tree structure, as the image (bit-plane) is partitioned into a 3-level hierarchical structure having adaptive block sizes. HBVLC initially considers the entire bit-plane of size $\tau \times \tau$ as a level-1 block and checks for the '1' bit distribution. If the distribution is greater than a defined threshold (t_1) which is calculated using Eq. 5.1, then the level-1 block is partitioned into four level-2 sub-blocks of size $\tau/2 \times \tau/2$.

$$t_i = \lfloor (\tau_i^2 - 2)/\log_2 \tau_i^2 \rfloor, \tag{5.1}$$

where $1 \leq i \leq 3$, and τ_i is the i^{th}-level block. Next, the distribution of '1' is checked in all the four level-2 blocks and found to be greater than a defined threshold (t_2) as stated in Eq. 5.1, for any of the sub-blocks then each of the four sub-blocks is divided into level-3 sub-blocks of size $\tau/4 \times \tau/4$. Next, the distribution of '1' is checked in level-3 blocks and if found to be greater than a defined threshold (t_3) as given in Eq. 5.1, then no further partitioning is done but the bit-plane itself is considered as its code. However, the level-1 block is encoded using its level-label concatenated by the code of its sub-blocks, where the code of the sub-blocks includes a type-label followed by its structure information. Thus, HBVLC tries to keep the total code length

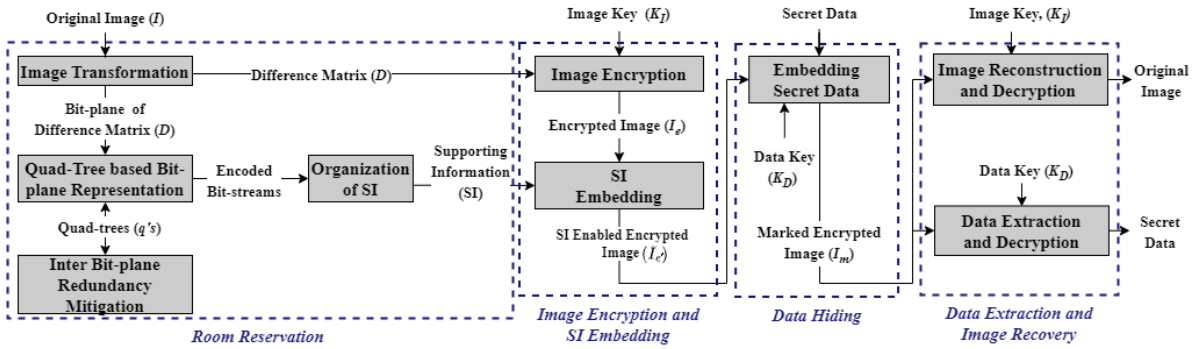


Fig. 5.2: Workflow of the proposed RDHEI using multi-level blocking with quad-tree method

of a block less than the direct recording of the bit-plane and achieves significant performance benefits as far as image compression is concerned.

5.2 Proposed RDHEI using Multi-Level Blocking with Quad-Tree (MBQ-RDHEI) Method

This section provides a detailed description of our proposed MBQ-RDHEI method, which aims to mainly improve the EC while ensuring reversibility and confidentiality of the cover image data. The proposed MBQ-RDHEI works in four stages that include 1) Room reservation, 2) Image encryption and supporting information embedding, 3) Data hiding, and 4) Data extraction with image recovery, as illustrated in Fig. 5.2.

5.2.1 Room Reservation

To reserve a room inside the original cover image (I) for embedding, I is first transformed into a difference matrix with the help of difference predictor andMED predictor [102], which is later converted into bit-planes. Next, the obtained bit-planes are represented into compressible bit-streams guided by quad-tree structure to condense their size. Afterwards, a novel Inter Bit-plane Redundancy Mitigation (IBRM) strategy is employed to further optimize the bit-plane representation. Lastly, the supporting information, generated during the aforementioned process which is essentially an overhead, is prudently organized to lessen its impact on the created room. The detailed process to execute the aforementioned steps is provided as follows.

A. Image Transformation

The original gray-scale image I of size $M \times N$ is transformed into a difference matrix (D) of the same size, akine to the chapter 4. The he difference and MED predictors are explored using Eqs. 4.1 and 4.2 and D image is obtained. Similarly, the original image (I) can be reconstructed at the recipient end by having access to the image D and the value of the first pixel (i.e., $I_{(1,1)}$),

Table 5.1: Count of '0' values in $b^{(k)}$ bit-plane

Test image	k=8	k=7	k=6	k=5	k=4	k=3	k=2	k=1	k=0
Lena	262144	262021	260296	250944	222282	154565	78493	29207	146894
Airplane	262139	261711	260217	257182	248793	222200	107790	103924	183771
Baboon	262140	258013	230106	178159	114437	60158	26465	8950	134330

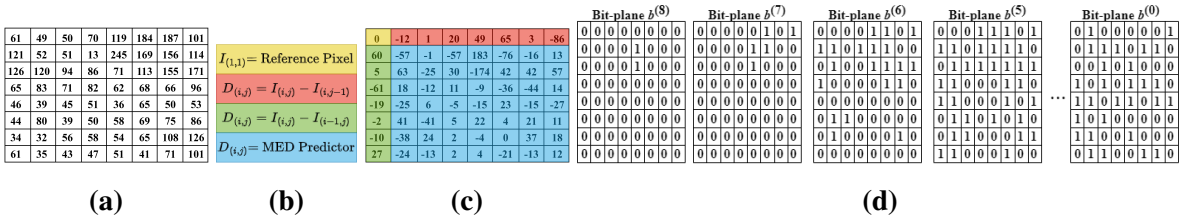


Fig. 5.3: Illustration of the difference matrix's bit-planes creation process, (a) Original image, (b) Rules (c) Difference matrix, and (d) 9 bit-planes ($b^{(8)}$, $b^{(7)}$, ..., $b^{(0)}$) of difference matrix

which is chosen as the reference point. By designating the $I_{(1,1)}$ as the reference pixel (RP), the corresponding difference pixel $D_{(1,1)}$ can be initialized to zero. Since I is a gray-scale image, then $-255 \leq D_{(i,j)} \leq +255$ holds which in turn indicates that $D_{(i,j)}$ must be represented by 9 bits, calculated as follows:

$$b^{(k)}_{(i,j)} = \begin{cases} \begin{cases} 0, & \text{if } D_{(i,j)} < 0 \\ 1, & \text{otherwise} \end{cases}, & \text{if } k = 0 \\ \left\lfloor \frac{|(D_{(i,j)}| mod (2^k)}{2^{k-1}} \right\rfloor, & \text{Otherwise} \end{cases} \tag{5.2}$$

where $b^{(k)}$ denotes the k^{th} bit-plane for D and $8 \geq k \geq 0$. In other words, 8 bit-plane ($b^{(8)}$, $b^{(7)}$, ..., $b^{(1)}$) are required to store the magnitude of D and 1 bit-plane ($b^{(0)}$) for the sign-bit of each element of D. Due to prevalent spatial correlation within the natural input images, it has been commonly observed that the sparsity of a given bit-plane $b^{(k)}$ decreases with the value of k for magnitude bit-planes. This observation is also validated by the experimental finding presented in Table 5.1 for three different characteristics and frequently used test images (i.e., $Lena$, $Airplane$, and $Baboon$), where the number of '0' bits representing the magnitude of sparsity, keeps on decreasing with the bit-plane number. However, it can also be observed that the highest level of randomness is present in the sign bit-plane $b^{(0)}$ which makes the embedding a challenging task.

To demystify the process of the bit-plane generation, an illustrative example is presented in Fig. 5.3. The example considers an 8×8 image, as shown in Fig. 5.3a, and follows the rules for generating a difference matrix as depicted in Fig. 5.3b. The resulting difference matrix is shown in Fig. 5.3c, which is then separated into nine bit-planes, with the first eight bit-planes representing the magnitude and the last bit-plane representing the sign-bit. This separation of bit-planes is depicted in Fig. 5.3d.

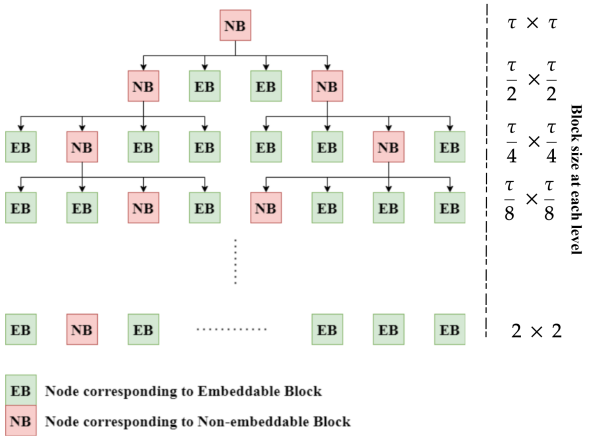


Fig. 5.4: Representation of full quad-tree

B. Quad-Tree based Bit-plane Representation

To condense the size of the obtained bit-planes, each bit-planes ($b^{(k)}$) is converted into a compressed bit-stream ($s^{(k)}$) one by one, starting from the MSB to the LSB-plane. For this, a quad-tree $q^{(k)}$ specifically tailored for each individual bit-plane $b^{(k)}$ is generated adhering to the principles of a traditional quad-tree, as described in 5.1.1. However, the partitioning of a block (or node) into four sub-blocks (or child nodes) is done only if any bit associated with the corresponding block is non-zero, such blocks are also termed as non-embeddable block or node (NB) and remaining ones as an embeddable block (EB).

Algorithm 5 Quad-tree Generation

function GENERATEQUADTREE(τ, bit-plane $b^{(k)}$)
    Create an empty $queue$ and a $root$ node representing the entire bit-plane $b^{(k)}$ of dimension $\tau \times \tau$ as a block
    $enqueue$ the $root$ into the $queue$
    while $queue$ is not empty **do**
        $dequeue$ the $front$ item from $queue$ and assign to $node$
        if the block represented by $node$ is not an EB and the size of the block is greater than 2×2 **then**
            Subdivide the block into four equal size quadrants/sub-blocks
            Create a child node for each quadrant/sub-block and $enqueue$ each child onto the $queue$
        end if
    end while
    Return the $root$ node of the quad-tree.
end function

To provide further insight into the quad-tree generation process, a bit-plane $b^{(k)}$ with dimensions $\tau \times \tau$ has been considered, where $\tau = 2^t$ and $t \in \mathbb{N}$. Fig. 5.4 provides a visual representation of a complete quad-tree $q^{(k)}$ for $b^{(k)}$. It shows that a $\tau \times \tau$ bit-plane can have up to $\log_2 \tau$ possible levels, with each level containing a maximum of 4^{r-1} nodes, where r

ranges from 1 to $\log_2 \tau$. The step-by-step algorithm to generate a complete quad-tree ($q^{(k)}$) for bit-plane ($b^{(k)}$) is provided in Algorithm 5.

The presented Algorithm 5 generates a quad-tree starting with a $root$ node representing the entire bit-plane of dimensions $\tau \times \tau$ as a block using $queue$ as a data structure. The algorithm subdivides a block into four sub-block (i.e., top-left (top_{left}), top-right(top_{right}), bottom-left($bottom_{left}$), and bottom-right($bottom_{right}$)), if it is not an EB. This subdivision is recursively called until each sub-block becomes EB or its size reduces to 2×2. The resulting quad-tree thereby models the entire $\tau \times \tau$ bit-plane, where each EB signifies a sparse region and each NB denotes a complex one within the bit-planes. Thereafter, $q^{(k)}$ is traversed in level-order (also known as breadth-first search (BFS) order) to generate the compressed bit-stream $s^{(k)}$ for the bit-plane $b^{(k)}$, where '0' and '1' are collected for each EB and NB, respectively. The complete step-by-step algorithm to generate the compressed bit-stream ($s^{(k)}$) for $q^{(k)}$, is discussed in algorithm 6.

Algorithm 6 QUAD-TREE ENCODING INTO BIT-STREAM USING LEVEL ORDER TRAVERSAL

function ENCODEQUADTREE($root$, $b^{(k)}$)
    Create an empty $queue$ and empty bitstream $s^{(k)}$
    $enqueue$ the $root$ node into the $queue$
    while $queue$ is not empty **do**
        $dequeue$ the $front$ item from $queue$ and assign to $node$
        Determine the corresponding block of $b^{(k)}$
        if the block is an EB **then**
            Append '0' bit to $s^{(k)}$
        else
            Append '1' bit to $s^{(k)}$
            if the node corresponds to NB and is not a leaf node **then**
                $enqueue(node.top_{left})$ into the $queue$
                $enqueue(node.top_{right})$ into the $queue$
                $enqueue(node.bottom_{left})$ into the $queue$
                $enqueue(node.bottom_{right})$ into the $queue$
            end if
        end if
    end while
    return bitstream $s^{(k)}$
end function

C. Inter Bit-plane Redundancy Mitigation

In order to further optimize the representation of bit-planes, a novel IBRM strategy is employed so that redundancy between the subsequent bit-planes' representations can be minimized. This is being done based on findings of a comprehensive experimental analysis conducted using three commonly used standard test images, namely, $Lena$, $Airplane$, and $Baboon$ with two

Table 5.2: Probability of NB for $b^{(k)}$ with respect to similarly positioned NB for $b^{(k-1)}$

Test image	$k=8$	$k=7$	$k=6$	$k=5$	$k=4$	$k=3$	$k=2$	$k=1$	$k=0$
Lena	1.0000	0.6436	0.7673	0.8199	0.8435	0.8975	0.9334	0.9070	-
Airplane	0.2500	0.6031	0.7018	0.7388	0.7177	0.9353	0.9157	0.8429	-
Baboon	0.0000	0.7104	0.8277	0.8742	0.9069	0.9273	0.9363	0.9435	-

consecutive bit-planes ($b^{(k)}$ and $b^{(k-1)}$) and their respective quad-trees ($q^{(k)}$ and $q^{(k-1)}$) along with compressed bit-streams ($s^{(k)}$ and $s^{(k-1)}$). As discussed in subsection 5.2.1, each bit in the bit-stream represents a block of size ranging from $\tau \times \tau$ to 2×2, where the bit '0' denotes EB and '1' represents NB. From the findings calculated in terms of probability using Eq. (5.3) as provided in Table 5.2, it has been observed that NB for $b^{(k)}$ can significantly impact the corresponding blocks in the subsequent bit-plane $b^{(k-1)}$.

$$p^{(k)} = \frac{\delta^{(k-1)}}{\theta^{(k)}}, \tag{5.3}$$

where $\theta^{(k)}$ denotes the total count of NBs present in $b^{(k)}$, and $\delta^{(k)}$ represents the total count of NBs in $b^{(k)}$ whose corresponding blocks in the subsequent bit-plane $b^{(k-1)}$ are also NBs.

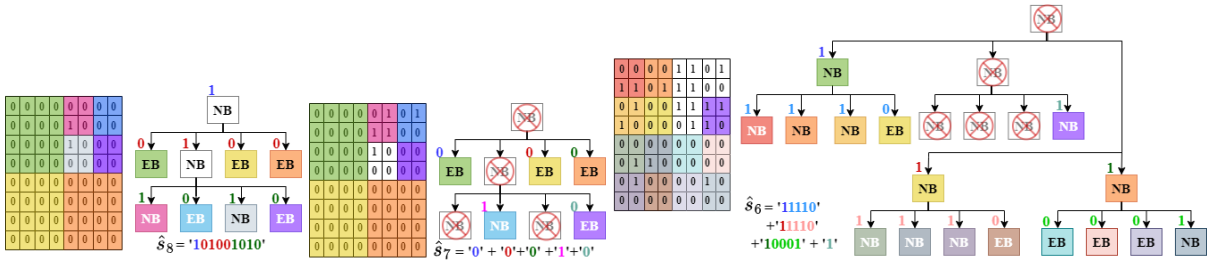
Therefore, the proposed method marks all the corresponding blocks of the subordinate/subsequent planes (i.e., $b^{(k-1)}$ to $b^{(0)}$) as NBs of the NB of $b^{(k)}$, to optimize their representation. Consequently, the entries for each NB of $b^{(k)}$ are excluded from $s^{(k-1)}$. The complete aforementioned IBRM strategy is detailed in step by step manner in Algorithm 7, which can be directly applied on the bit-planes starting from $b^{(7)}$, ..., $b^{(0)}$ resulting into compressed bit-streams i.e., $\hat{s}^{(7)}$, ..., $\hat{s}^{(0)}$.

To better explain the process, the example depicted in Fig. 5.3 is extended and illustrated in Fig. 5.5 to provide a visual representation of how the bit-planes are compressed into bit-streams. More specifically, Fig. 5.5a shows the generation of quad-tree for $b^{(8)}$ and its encoding in compressed bit-stream using level-order tree traversal as per the process defined in subsection 5.2.1. Further, Fig. 5.5b and 5.5c depicts the quad-tree generation and their encoding into compressed bit-stream for the bit-planes $\hat{s}^{(7)}$ and $\hat{s}^{(6)}$, where only EB nodes of previous bit-plane are extended as per the Algorithm 7.

However, it's essential to highlight that the IBRM strategy is maintained until the size of $\hat{s}^{(k)}$ is less than the total number of embeddable bits in the current bit-plane. Natural images might have lower-order bit-planes $b^{(k)}$ with fewer embeddable bits than their corresponding bit-streams $\hat{s}^{(k)}$, leading to possible information loss. Therefore, the number of bit-planes in an image that, when included for embedding, improve the overall EC, are referred to as 'embeddable bit-planes'. The total count of these embeddable bit-planes, denoted by n, can be encoded by a maximum of 3 bits. The process for ascertaining the complete sum of embeddable bits in the k^{th} bitplane, denoted as $T_{EC}^{(k)}$, requires a thorough examination of each node within the

Algorithm 7 INTER BIT-PLANE REDUNDANCY MITIGATION

function IBRM($s^{(k)}$, $b^{(k-1)}$)
    Initialize an empty bitstream $\hat{s}^{(k-1)}$
    for each bit x in $s^{(k)}$ from left to right **do**
        if $x = 0$ **then**
            Determine the sub-tree for the corresponding block of $b^{(k-1)}$ using Function-GENERATEQUADTREE
            Generate sub bit-stream $s_x'^{(k-1)}$ from the sub-tree using Function-ENCODEQUADTREE
            Append $s_x'^{(k-1)}$ to $\hat{s}^{(k-1)}$
        end if
    end for
    return bitstream $\hat{s}^{(k-1)}$
end function



(a) Quad-tree of $b^{(8)}$ **(b)** Quad tree of $b^{(7)}$ **(c)** Quad tree of $b^{(6)}$

Fig. 5.5: Illustrative process of compressed bit-stream ($s^{(8)}$, $s^{(7)}$, $s^{(6)}$) generation

corresponding trees. $T_{EC}^{(k)}$ is determined using Eq. (5.4).

$$T_{EC}^{(k)} = \sum_{c=1}^{\alpha} (h_{(c)} + 2)^2, \quad \text{if } node_{(c)} \in EB \tag{5.4}$$

where $h_{(c)}$ represents the height of the node ($node_{(c)}$) within the tree (which is zero for leaf nodes), and α symbolizes the total count of nodes within the k^{th} bit-plane.

D. Organization of Supporting Information

As discussed above, the proposed method makes use of level-order traversal while generating bit-streams for representing the bit-planes, despite knowing the fact that level-order traversal is more complex than pre-order traversal of the quad-tree. This deliberate choice is made to capture even the most minute redundancies present in the codes so that further optimization using Huffman encoding can be done. In the level-order traversal of the quad-tree, all 15 possible combinations of a node's four child nodes are represented by codes ranging from 0001, 0010, 0011, ..., 1111. Next, the generated bit-streams to represent the quad-tree can be further condensed using Huffman encoding (if needed). Therefore, the proposed MBQ-RDHEI method can transform itself into three versions namely Direct-MBQ (MBQ), Huffman-MBQ (HMBQ),

and Separate Huffman-MBQ (SHMBQ), where the MBQ method does not employ Huffman encoding to compress the received bit-streams (i.e., $\hat{s}^{(8)}$, $\hat{s}^{(7)}$, $\hat{s}^{(6)}$, ..., $\hat{s}^{(8-n+1)}$), instead directly utilizes them. In contrast, HMBQ first traverses the bit-streams from left to right and determines the frequencies of each code to generate the corresponding Huffman codes. Next, all the codes in the bit-streams are replaced by their corresponding Huffman codes to get the Huffman-encoded bit-streams. Likewise, SHMBQ also employs Huffman encoding but separately on each bit-stream. However, it is to be noted that each version of the method offers a different EC with varying execution time.

Further, SI that includes a total number of embeddable bit-planes (n), and compressed bit-streams ($\hat{s}^{(8)}$ to $\hat{s}^{(8-n+1)}$) arranged in descending order of their significance, RP, non-embeddable bits of 8^{th} bit-plane denoted as $NE_{b^{(8)}}$, and/or Huffman code/rules, which is indispensable for identifying the embeddable space in the image, as well as reconstructing the image itself, is also generated, whose length is denoted by SI_{len}. The complete SI is crucial for data hiding/extraction and image reconstruction, that's why it is embedded in the image using a process detailed in the next sub-section.

5.2.2 Image Encryption and SI Embedding

The proposed method also makes use of AES-CTR for the image encryption similar to the methods in chapters 3 and 4. However, the encryption is performed in bit-wise manner. First, a pseudo-random matrix (ξ) of size $M \times N$ is generated. This matrix ξ is initially converted into 8 bit-planes $\xi^{(7)}$, $\xi^{(6)}$, ... , $\xi^{(0)}$ as follows,

$$\xi^{(k)}_{(i,j)} = \left\lfloor \frac{(\xi_{(i,j)})\%(2^{k+1})}{2^k} \right\rfloor, \tag{5.5}$$

where $\%$ and $\lfloor * \rfloor$ represent the modulus operator and the floor function, respectively, and $7 \leq k \leq 0$. Next, pixels of the image are encrypted using bit-wise XOR ($\oplus$) operation defined as follows, to get the encrypted image (I_e).

$$I^{(k)}_{e(i,j)} = I^{(k)}_{(i,j)} \oplus \xi^{(k)}_{(i,j)}. \tag{5.6}$$

Thus, the obtained image also has the 8 bit-planes i.e., $b_e^{(7)}$, $b_e^{(6)}$,..., $b_e^{(0)}$, like the I.

Thereafter, SI is embedded in the encrypted bit-planes as it is required for the efficient operation of data embedding, data extraction, and image reconstruction. For this, the encrypted bit-planes are considered in order of $b_e^{(7)}$, $b_e^{(6)}$, ...,$b_e^{(8-n+1)}$ and embedding in each bit-plane is done in raster scan order by replacing the bits of bit-planes by the bits of SI while collecting the bits belonging to NBs as NE_{bits}. After the SI embedding, NE_{bits} are embedded by replacing the bits belonging to onward EBs only, with NE_{bits}. Thus, the SI-enabled encrypted image $I_{e'}$ is obtained, by collecting the resultant bit-planes i.e., $b_e'^{(7)}$, $b_e'^{(6)}$, ..., $b_e'^{(0)}$ using following Eq. 5.7. To elucidate the SI embedding process, Fig 5.6 offers an illustration where the first block only

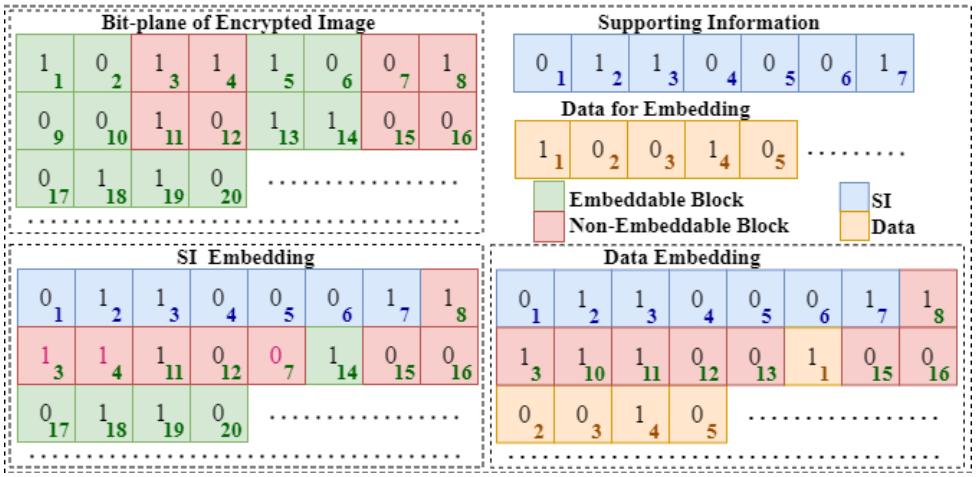


Fig. 5.6: SI and data embedding

shows EB and NB positions in a bit-plane and the second block shows the SI of length 7 bits and data to be embedded of 5 bits. As shown in the third block, the SI replaces all the bits in sequence (that includes bits from both EB and NB), however for the sake of reversibility and lossless extraction the NE_{bits} pinpointed at positions 3, 4, and 7 are recorded and embedded at subsequent embeddable positions (shown in green color). Thus, NE_{bits} undergo a positional shift to locations 9, 10, and 13, to avoid the information loss.

$$I'_{e(i,j)} = \sum_{k=7}^{0} b'^{(k)}_{e(i,j)} \times 2^k. \tag{5.7}$$

5.2.3 Data Hiding

Initially, the data hider must extract the SI from $I_{e'}$ in order to identify the reserved room for data embedding. This is accomplished by reading the image starting from the MSB-plane in raster scan order by first decoding the initial three bits to get n. After skipping the RP bits from SI, the next bits of SI are processed to generate the quad-trees for $b^{(8)}$ using Algorithm 8. Likewise, to generate the quad-tree for bit-plane $b^{(7)}$, the quad-tree for the previous bit-plane i.e., $b^{(8)}$ (in case of $b^{(7)}$) is subjected to level-order traversal and for each EB node, the tree is extended using the steps outlined in Algorithm 8. This process is repeated for $8 - n + 1$ subsequent bit-planes. Next, the data hider starts traversing from the MSB-plane in raster scan order to count the NE_{bits} in the first SI_{len} positions. Next, he/she skips SI_{len} positions followed by NE_{bits} number of positions/bits belonging to EBs to start embedding the secret data bits. However, the secret data is first encrypted using a secure encryption algorithm and the data-hiding key (K_D) to enhance the level of security and then the embedding is done at each EB position by replacing its bits with the secret data bits. The process of embedding is continued till $8 - n$ bit-plane and a marked encrypted image I_m is obtained. To elucidate the data hiding

process, the last block of Fig 5.6 shows the embedding of data at positions - 14, 17, 18, 19, and so on.

Algorithm 8 RECONSTRUCTION OF QUAD-TREE FROM SI

procedure RECONSTRUCTQUADTREE(SI, $block\_size$, m)
    Create a new quad-tree with $root$ node having value $= SI[m]$ of $block\_size \times block\_size$
    Enqueue $node\_block$ pair i.e., $root$ node with its $block\_size$ in $queue$ and set $m \leftarrow m+1$
    while $queue$ is not empty **do**
        Dequeue a $node\_block$ pair from the $queue$
        if If the node's value is 1 and the $block\_size > 2$ **then**
            Partition the block into 4 new sub-blocks, each with size $block\_size/2 \times block\_size/2$
            for i in range 1 to 4 **do**
                Create a new node with the value $= SI[m]$
                Enqueue the new $node\_block$ pair in $queue$
                set $m \leftarrow m + 1$
            end for
        end if
    end while
    return quad-tree and m
end procedure

5.2.4 Data Extraction and Image Recovery

To recover the hidden data and reconstruct the original image, the receiver first extracts the n, SI and $NE_{b^{(8)}}$. Subsequently, the receiver fabricates the quad-tree, mirroring the approach of the data hider as elucidated in subsection 5.2.3. Next, the receiver requires access to K_D and/or K_I to extract the hidden data and/or recover the original image. Thus, three cases are possible, which are defined and discussed as follows:

In a case where the recipient holds only the K_D, the recipient must retrieve the bits from each EB denoted as E_{bits}, sequentially, beginning from the MSB-plane and the second pixel, while traversing in raster scan order. Following the skipping of the initial SI_{len} bits from the collected bits, the recipient decrypts the remaining bits using K_D to recover the secret data, without any loss.

If the recipient only possesses the K_I, he/she must read the image similarly as in the previous case. Next, the NE_{bits} (where SI was embedded), are replaced by the same number of bits belonging to EBs. Afterwards, the recipient generates a pseudo-random matrix (ξ) as in section 5.2.2 and converts the updated marked encrypted image (I_m) into eight bit-planes using equation (5.5). Next, the bit-planes of I_m are decrypted using Eq. 5.6 to get the decrypted bit-planes i.e., $b_d^{(7)}$, $b_d^{(6)}$, ..., $b_d^{(0)}$.

To determine the bit-plane of the difference matrix, the quad-tree of each bit-plane is used. Initially, the MSB-plane of the difference matrix is generated using the first quad-tree and

Table 5.3: Net gain in EC (bits) for image $Lena$ by IBRM

Bit-plane		$b^{(7)}$	$b^{(6)}$	$b^{(5)}$	$b^{(4)}$	$b^{(3)}$	$b^{(2)}$
Without IBRM	$s^{(k)}$/ EC	757/ 261740	6745/ 256952	21845/ 236184	56445/ 177804	84917 /72876	87321/ 27324
With IBRM	$\hat{s}^{(k)}$/ EC	757/ 261740	6487/ 266808	19005/ 234953	44331/ 172932	48571/ 59124	15352/ 6712
Total Difference in Pure EC (With IBRM - Without IBRM)		0	114	1609	7242	10553	-

$NE_{b^{(8)}}$. A bit-plane of the image size is created with all zero values, and all non-embeddable positions are replaced with their respective bits from $NE_{b^{(8)}}$. Subsequently, in order to generate the next $8-n$ MSB-planes, all the positions belong to EBs of $8-n$ MSB-planes ($b_d^{(7)}$ to $b_d^{(8-n)}$) are replaced with '0'. The remaining LSB-planes are given as $b_d^{(k)}$ for $8-n-1 \geq k \geq 0$. These 9 bit-planes are collected using Eq. 5.7 for $8 \geq k \geq 0$, and the difference matrix D is obtained. Finally, the lossless and identical reconstructed image I_r is obtained using the difference matrix and RP from Eq. (4.1).

When both the data-hiding and encryption keys are available with the receiver, he/she can extract the embedded data as in the first case and can also reconstruct the original image as in the second case, without any loss/error.

5.3 Experiment Results and Discussion

This section presents a detailed discussion of the performance of the proposed method and its comparison with the SOTA methods. The discussion is organized into four subsections. In the first subsection, the embedding performance of the proposed method is evaluated and analyzed on well-known test images of the SIPI dataset [41], including $Lena$, $Baboon$, $Airplane$, $Jetplane$ (F-16), $Tiffany$ and Man (Male) as depicted in Fig. 1.3, as well as on the big datasets, namely, BOSSbase [38] and BOWS-2 [39]. The second subsection presents a comparison of the proposed method with the modern SOTA methods, to highlight the effectiveness of the proposed method. The third subsection evaluates the security effectiveness of the proposed method considering the perceptual level, several statistical measures and malicious attacks. In the last subsection, a detailed discussion of the complexity analysis of the MBQ-RDHEI method is presented.

5.3.1 Embedding Performance Analysis

To experimentally demonstrate the embedding performance of the proposed method, the cover image $Lena$ is used as a test image. As discussed, the image is first transformed into a difference matrix having 9-bit-planes ($b^{(8)}$ to $b^{(0)}$), where eight bit-planes, starting from $b^{(8)}$ to $b^{(1)}$, represent the magnitude, while the $b^{(0)}$ contains the sign bit. For each bit-plane, a quad-tree is designed using Algorithm 5, where each node corresponds to a block of the bit-plane. The quad-tree is subsequently transformed into a bit-stream with the help of Algorithm-6, while in-

Table 5.4: Number of different size and type of blocks along with total EC per bit-plane for image $Lena$

Block Type	Block Size	Bit-plane								
		$b^{(8)}$	$b^{(7)}$	$b^{(6)}$	$b^{(5)}$	$b^{(4)}$	$b^{(3)}$	$b^{(2)}$	$b^{(1)}$	$b^{(0)}$
EB	512×512	1	0	0	0	0	0	0	0	0
EB	256×256	0	1	0	0	0	0	0	0	0
EB	128×128	0	3	0	0	0	0	0	0	0
EB	64×64	0	22	19	4	0	0	0	0	0
EB	32×32	0	34	76	54	1	0	0	0	0
EB	16×16	0	58	190	212	43	0	0	0	0
EB	8×8	0	77	441	722	606	14	0	0	0
EB	4×4	0	105	1017	2321	3668	517	4	0	0
EB	2×2	0	167	2006	6414	15857	12489	1662	300	0
NB	2×2	0	101	1233	5464	15505	28452	13103	1378	300
No. of total nodes		1	757	6487	19005	44331	48571	15352	1682	300
Total EC		-	261740	256808	234952	172932	59124	6712	1200	0

ter bit-plane redundancy is efficiently reduced through the implementation of the novel IBRM using Algorithm-7. To demonstrate the efficacy of the proposed IBRM, a detailed analysis using the test image $Lena$ is provided in Table 5.3. Here, the length of the bit-streams with and without IBRM is presented, where it can be clearly observed that the IBRM strategy significantly minimizes the length of the bit-streams which in turn increases the pure EC. Though it may result in a few misclassifications, a reduction in SI size-ably increases the overall net gain in Pure EC. More specifically, the cumulative benefit after employing IBRM for $Lena$ is a reduction of 19518 bits SI.

Next, in Table 5.4, the details about the number and types of nodes and their corresponding block's size along with the total EC, for each bit-plane are provided. This table also helps us elaborate on the size of the quad-tree, which is essentially an overhead. As evident from Table 5.4, the bit-plane $b^{(8)}$ does not contain any '1' bit which means a single node can represent the entire bit-plane, corresponding to a block size of 512×512. Consequently, the subsequent bit-plane i.e., $b^{(7)}$ will only have one quad-tree $q^{(7)}$ that has a total of 757 nodes which can be represented by a total of 757 bits. Similarly, for $b^{(6)}$, a total of 467 quad-trees are constructed having a total of 6487 nodes which can be represented by a total of 6487 bits. Similarly, for other remaining bit-planes, the results detailing about the number of quad-trees, total nodes and their corresponding block size, and total EC are provided in Table 5.4. Further, it can also be observed that the maximum EC i.e., 261780 is provided by $b^{(7)}$ followed by $b^{(6)}$ and so on. This is because of the level of sparsity present in the bit-planes, which also significantly affects the size of the overhead.

Further, Table 5.5, which is organized into three sections where the first, second and third section details the results on bit-plane wise compressed bit-stream size, Huffman code length, SI length (SI_{Len}) and pure EC of the proposed MBQ, HMBQ and SHMBQ, respectively, is provided.

As evident from Table 5.4 and the first section of Table 5.5, the total EC of the bit-planes

Table 5.5: Bit-plane wise performance analysis of all three versions of the proposed method on image $Lena$

Method		Bit-planes								
		$b^{(8)}$	$b^{(7)}$	$b^{(6)}$	$b^{(5)}$	$b^{(4)}$	$b^{(3)}$	$b^{(2)}$	$b^{(1)}$	$b^{(0)}$
MBQ	$\hat{s}^{(8)}$ **to** $\hat{s}^{(0)}$	1	757	6487	19005	44331	48571	15352	1682	300
	SI Length (SI_{Len}) **:** 3 (n) + 8 (RP) + 119152 ($\hat{s}^{(8)}$ to $\hat{s}^{(3)}$) = 119163									
	Pure EC : 985556 (Total EC of $b^{(8)}$ to $b^{(3)}$) - 119163 = 866393									
HMBQ	$\hat{s}^{(8)}$ **to** $\hat{s}^{(0)}$	1	724	6224	18339	42820	47331	15250	1681	300
	Huffman code Length : 127									
	SI Length (SI_{Len}) **:** 3 (n) + 8 (RP) + 115439 ($\hat{s}^{(8)}$ to $\hat{s}^{(3)}$) + 125 = 115577									
	Pure EC : 985556 (Total EC of $b^{(8)}$ to $b^{(3)}$) - 115577 = 869979									
SHMBQ	$\hat{s}^{(8)}$ **to** $\hat{s}^{(0)}$	1	600	5904	17992	42610	46107	14668	1773	300
	Huffman code Length :	0+1	0+1	126+1	126+1	127+1	126+1	142+1	0+1	0
	SI Length (SI_{Len}) **:** 3 (n) + 8 (RP) + 113214 ($\hat{s}^{(8)}$ to $\hat{s}^{(3)}$) + 511 = 113736									
	Pure EC : 985556 (Total EC of $b^{(8)}$ to $b^{(3)}$) - 113736 = 871820									

($b^{(2)}$ to $b^{(0)}$) is less than the size of their compressed bit-streams ($\hat{s}^{(2)}$ to $\hat{s}^{(0)}$), therefore the bit-planes $b^{(2)}$ to $b^{(0)}$) are not used for embedding. It can be further observed that the overall length of SI accounts to 119163 bits, which includes 3 bits for the number of embeddable bit-planes, 8 bits for the RP, 119152 bits for the quad-tree representation of $b^{(8)}$ to $b^{(3)}$ bit-planes, and 0 bits for $NB_{b^{(8)}}$. Hence, the pure EC is 866393 bits for the MBQ method. As far as the HMBQ method is concerned, the required Huffman code's length is 127 bits. It can be further observed that the overall length of SI accounts to 115577 bits. Hence, the pure EC of HMBQ is 869979 bits which is 3586 bits higher than the same of the MBQ. For the SHMBQ approach, the cumulative length of Huffman codes is 511 bits, which incorporates a single bit for each bit-plane indicating the utilization of Huffman codes for that bit-plane. The pure EC of this method is 871820 bits, which is having a net gain of 5427 bits and 1841 bits compared to the MBQ and HMBQ methods, respectively. Therefore, it can be concluded that the SHMBQ method offers the highest EC among all the three versions of the proposed method.

Furthermore, Table 5.6 provides the results of EC, ER and SI for all the test images shown in Fig. 1.3. The presented findings indicate a strong correlation between an image's smoothness and its EC, ER, and SI. Notably, the $Baboon$ image, characterized by a higher presence of textures, exhibits significantly lower ER and EC values compared to the $Airplane$ image, which predominantly consists of smoother regions. Additionally, the length of SI for the $Baboon$ image is comparatively higher, while it is minimal for the $Airplane$ image. More specifically, the proposed methods i.e., MBQ, HMBQ, and SHMBQ, achieve ER of 1.6638, 1.6772, and 1.6796 bpp for the $Baboon$ and 4.0773, 4.0998, and 4.0987 bpp for the $Airplane$, respectively.

The proposed method has been evaluated on two datasets, BOSSBase and BOWS-2, each comprising 10,000 images of size 512×512 pixels. The results of the evaluation are presented in Table 5.7, where the PSNR and SSIM of each reconstructed image are $+\infty$ and 1, respectively. The average net ER for the proposed MBQ, HMBQ and SHMBQ methods in BOSSBase

Table 5.6: Pure EC ($bits$), SI_{Len} ($bits$) and ER (bpp) on the standard test images

Methods		Lena	Airplane	Baboon	Jetplane	Tiffany	Man
MBQ	**Pure EC**	866393	1068952	436146	973997	898815	790014
	SI_{Len}	119163	93736	126298	129395	115069	126446
	ER	3.3050	4.0777	1.6638	3.7155	3.4287	3.0137
HMBQ	**Pure EC**	869979	1074747	439659	977700	902006	793149
	SI_{Len}	115577	87941	122785	125692	111878	123311
	ER	3.3187	4.0998	1.6772	3.7296	3.4409	3.0256
SHMBQ	**Pure EC**	871820	1074449	440288	979118	903635	794267
	SI_{Len}	113736	88239	122156	124274	110249	122193
	ER	3.3252	4.0987	1.6796	3.7350	3.4471	3.0299

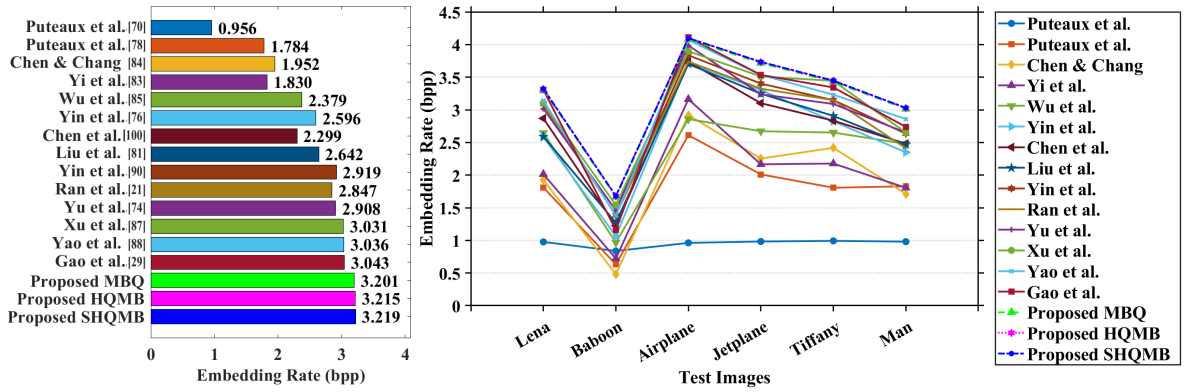
Table 5.7: Pure ER (bpp) on BOSSBase and BOWS2

	BOSSBase			BOWS2		
	Average	Best	Worst	Average	Best	Worst
MBQ	3.9970	7.8648	0.9654	3.9161	7.2973	0.8839
HMBQ	4.0110	7.8650	0.9780	3.9299	7.3016	0.8972
SHMBQ	4.0197	7.8648	0.9825	3.9343	7.3021	0.9002

are 3.9970, 4.0110 and 4.0197 bpp, respectively, while in BOWS-2, they are 3.9161, 3.9299 and 3.9343 bpp, respectively. The best ER achieved for an image in BOSSBase is 7.8648, 7.8650 and 7.8648 bpp for proposed methods, respectively, whereas in BOWS-2, it is 7.2973, 7.3016 and 7.3021 bpp. The worst ER for an image in BOSSBase is 0.9654, 0.9780 and 0.9825 for proposed methods, respectively, while the worst ER for an image in BOWS-2 are 0.8839, 0.8972, and 0.9002 bpp, respectively. Based on these analyses, it can be concluded that the proposed MBQ, HMBQ and SHMBQ methods are efficient and can be used for the privacy and security of data.

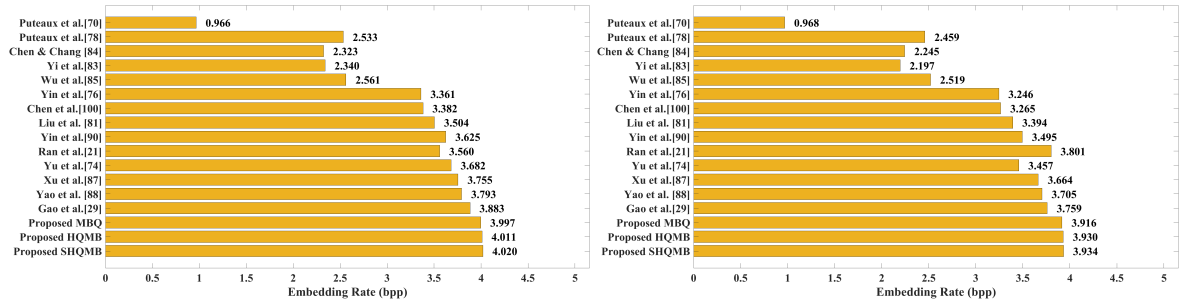
5.3.2 Comparison with State-of-the-Art Methods

To effectively demonstrate the performance of proposed methods, an extensive comparison with fourteen SOTA methods, that include [21, 29, 70, 74, 76, 78, 81, 83–85, 87, 88, 90, 100], have been



(a) AER for test images **(b)** Embedding rate for test images

Fig. 5.7: Comparative results on six standard test images



(a) $BOSSBase$ **(b)** $BOWS-2$

Fig. 5.8: Comparative results on two datasets

made. Since all three versions of the proposed method and the aforementioned SOTA methods are completely reversible, performance comparison only in terms of ER is done. It can be seen from Fig. 5.7 that the proposed method is able to achieve the maximum ER of 3.325 *bpp* for the $Lena$, whereas the highest ER attained by Gao *et al.*'s method [74] among the SOTA methods, is 3.304 *bpp*, for the same image. Thus, a gain of 0.001, 0.015, and 0.021 *bpp* is attained by all three versions of the proposed methods with respect to [29]. Further, it can be observed that the proposed method achieves the lowest ER in the case of $Baboon$, i.e., 1.680 *bpp* whereas among the SOTA methods, Xu *et al.*'s method [87] provides the maximum ER of 1.541 *bpp* for the same $Baboon$ image. Thus, a gain of 0.123, 0.126, and 0.139 *bpp* by the proposed method with respect to Xu *et al.*'s method [87], for the $Baboon$ image, is obtained. Thus, it can be clearly stated based on the observation of Fig. 5.7 that the proposed method easily outperforms all the fourteen aforementioned SOTA methods in terms of ER, for all the six test images (except $Airplane$ where our performance is comparable with the best performing SOTA method i.e., Gao *et al.* [74]). It is worth to be noted that the proposed method has significant performance gain even for the challenging $Baboon$ image. Further, as evidenced by Fig. 5.7a, the proposed method unequivocally demonstrates superior performance, with the average ER for all the six test images consistently exceeding the rates achieved by the other fourteen SOTA methods. This observation also confirms that the proposed method has been able to effectively exploit redundant information present in the natural images, which ultimately leads to superior performance compared to the SOTA methods.

Further, an extensive analysis on two large and benchmark datasets, namely BOSSBase and BOWS-2, to compare the performance of our proposed method with that of the SOTA methods, has been done. As demonstrated in Fig. 5.8, all three versions (i.e., MBQ, HMBQ and SHMBQ) of the proposed method, achieve an impressive ER of 3.997, 4.011 and 4.0197 *bpp* for BOSSBase and 3.9161, 3.9299 and 3.9343 *bpp* for BOWS-2, respectively. As far as the SOTA methods are concerned, Yao *et al.* [88] and Gao *et al.* [29] methods achieve the highest ER of 3.793 and 3.883 *bpp* for the BOSSBase, and 3.705 and 3.644 *bpp* for BOWS-2, respectively. Thus, it can be easily stated that all the versions of the proposed method easily outperform the SOTA methods in terms of ER, for both the datasets.

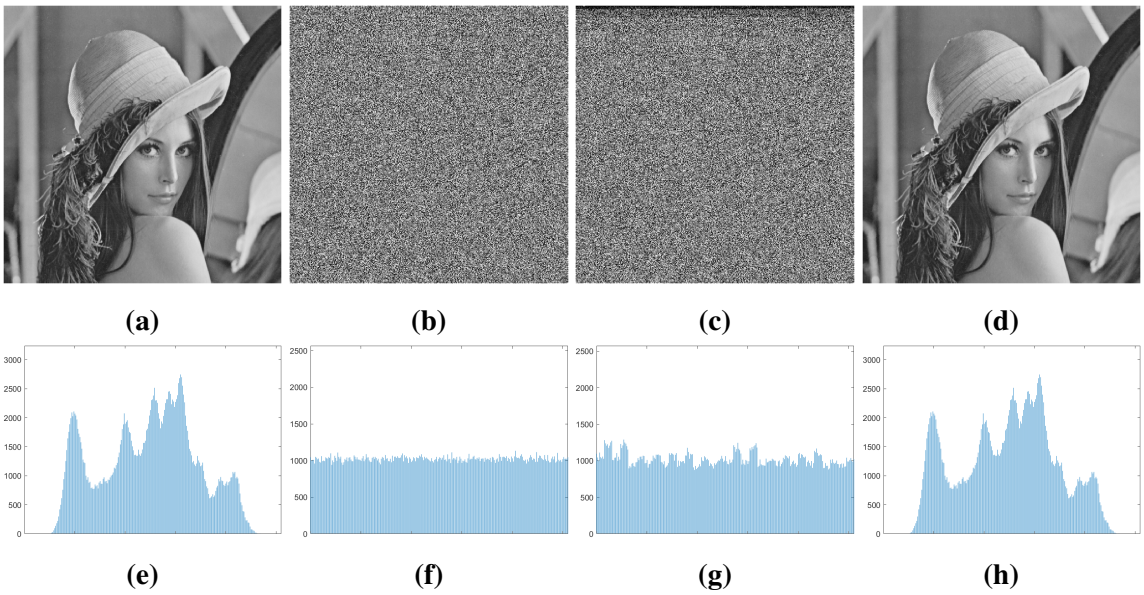


Fig. 5.9: Representation of resultant images and their histogram distribution of each phase of the proposed MBQ-RDHEI method: (a) Original image $Lena$, (b) Encrypted image, (c) Marked encrypted image, (d) Reconstructed Image (e-h) Histogram distribution of (a-d), respectively

5.3.3 Security Analysis

In this subsection, the robustness of the proposed method is analyzed against various security attacks such as Perceptual, Statistical and other Malicious attacks. For this, the subsection is divided into three parts as follows:

A. Robustness Against Perceptual Attacks

As discussed above, the proposed MBQ-RDHEI method uses separate keys to protect the hidden data and cover images. More specifically, K_D encrypts the secret data using any secure encryption technique, making data leakage highly improbable, while K_I encrypts the cover image to anonymize its contents to ensure confidentiality. To evaluate the security resilience of the proposed method, the original $Lena$ image I, as shown in Fig. 5.9a, has been considered. The I is first pre-processed to reserve a room for embedding and subjected to the AES in CTR method, resulting in the encrypted image I_e as shown in Fig. 5.9b. The SI is subsequently embedded into I_e. Next, the data hider proceeds to embed the encrypted secret data in I_e, producing a marked encrypted image I_m, as depicted in Fig. 5.9c. At the receiving end, the recipient extracts the data and reconstructs the image using the available keys, yielding the reconstructed image I_r, as shown in Fig. 5.9d. From Fig. 5.9, it is apparent that both I_e and I_m exhibit significant distortion and pervasive noise, rendering the original content unidentifiable. To further analyze the characteristics of the I, I_e, I_m, and I_r, frequency distribution of the 256 gray-scale levels using histogram representation are presented in Figs. 5.9e, 5.9f, 5.9g, and 5.9h, respectively. The distributions presented in Fig. 5.9f and 5.9g are noticeably uniform, in contrast to the his-

Table 5.8: Statistical analysis on image $Lena$

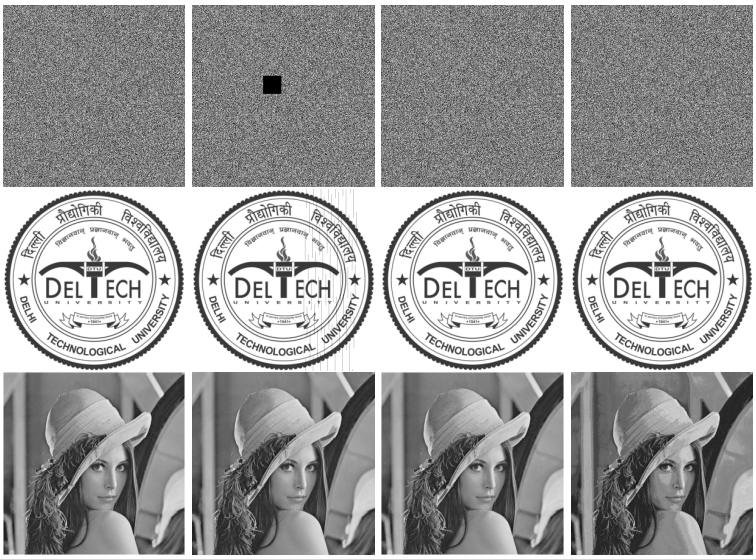
Image	Original image (I)	Encrypted image (I_e)	Marked encrypted image (I_m)
Horizontal Correlation	0.9719	-0.0001	0.0251
Vertical Correlation	0.9850	-0.0005	0.0199
Diagonal Correlation	0.9593	0.0018	0.0241
Information Entropy	7.4451	7.9993	7.9970
NPCR (I and I_e/I_m)	-	99.80	99.70
UACI (I and I_e/I_m)	-	72.98	73.42
PSNR (I and I_e/I_m)	-	9.2390	9.1902
SSIM (I and I_e/I_m)	-	0.0106	0.0108

tograms of I and I_r. This uniformity signifies that the proposed method provides a high level of perceptual security as far as encrypted and marked encrypted images are concerned.

B. Robustness against Statistical Attacks

To evaluate the performance of the proposed MBQ-RDHEI method against various statistical attacks, different statistical measures such as horizontal and vertical correlation [35], Shannon entropy [36], NPCR [37], UACI [37], PSNR, and SSIM [34] have also been considered. The experimental findings for the test image $Lena$ are presented in Table 5.8. As evident from the findings, the correlation in all directions i.e., horizontal, vertical, or diagonal for I, is high (or close to one), whereas the same is almost negligible (or more specifically, close to zero), for both I_e and I_m. This signifies that the proposed method completely destroys the relations among the neighbour pixels to make the resultant image fully anonymized. Further, measured entropy for both I_e and I_m is close to 8, which is near to maximum for an 8-bit gray-scale image, as given in Table 5.8. The high entropy suggests that the pixels are spread more equally within the permitted range i.e., 0 to $2^8 - 1$, which in turn signifies a high degree of unpredictability. Furthermore, it can also be observed from Table 5.8 that NPCR and UACI reach close to 100% for both I_e and I_m, which means almost all the pixels of the images are changed. This in turn indicates high perceptual security.

To conduct a more in-depth assessment of the similarity of I with I_e and I_m, the PSNR and SSIM are evaluated, and corresponding results are presented in Table 5.8. The experimental results show that the obtained PSNR and SSIM, are low for both the images, i.e., I_e and I_m, which in turn indicates that the I_e and I_m are significantly different and structurally dissimilar with respect to I. Thus, it can be stated that the proposed MBQ-RDHEI method demonstrates significant robustness against differential attacks.



(a) **(b)** **(c)** **(d)**

Fig. 5.10: The experimental results obtained under various removal attack scenarios (a). Original, (b) Patch removal, (c) Fourth bit-plane removal, and (d) Fifth bit-plane removal

C. Robustness against Other Malicious Attacks

To evaluate the robustness against other malicious attacks, the MBQ-RDHEI has been subjected to tampering, brute-force, and collage attacks. For this, a logo has been embedded in the encrypted $Lena$ image and various removal attacks such as Patch removal, Fourth bit-plane removal and Fifth bit-plane removal have been conducted as shown in Fig. 5.10. The images presented in the last two rows show that there is only a minimal distortion in the restored cover image, and the extracted logo is also visibly similar. However, there exists the potential for data loss should the bit-planes containing secret data be compromised. Similarly, the removal of bit-plane containing the SI bit-planes within the encrypted image could lead to a loss of the cover image itself. Conversely, the application of brute-force attacks becomes virtually unfeasible given the robust design of the proposed RDHEI method. This method incorporates AES for image encryption, a highly reliable and secure encryption protocol. Furthermore, the flexibility of the proposed method allows for the encryption of data through any robust algorithm selected by the data hider, which can provide a protection for the data against brute-force attacks.

Moreover, since the MBQ-RDHEI is separable and solely dependent on the bit-stream of SI for data embedding, resilience against collage attacks is enabled. Thus, it can be concluded that the MBQ-RDHEI has good resilience against various forms of malicious attacks. It is also important to note that since all three versions of the proposed method i.e., MBQ, HMBQ and SHMBQ share identical characteristics, the aforementioned findings regarding the security resilience are equally applicable to all three versions of the proposed method.

Table 5.9: Comparison of computation time and EC of the proposed method with the SOTA methods

Method	Total computational time of 6-test images (seconds)	Total EC of 6-test images (bit)	Average time per embedded bit (seconds/bit)
Puteaux *et al.* [70]	98.1472	1503142	0.65294×10^{-4}
Puteaux *et al.* [78]	247.9135	2805341	0.88371×10^{-4}
Chen & Chang [84]	187.2173	3070239	0.60978×10^{-4}
Yi *et al.* [83]	115.9477	2878944	0.40274×10^{-4}
Wu *et al.* [85]	286.2726	3742053	0.76501×10^{-4}
Yin *et al.* [76]	200.8890	4083817	0.49196×10^{-4}
Chen *et al.* [100]	276.5286	3615464	0.76485×10^{-4}
Liu *et al.* [81]	302.7732	4154739	0.72874×10^{-4}
Yin *et al.* [90]	502.4172	4590510	1.09450×10^{-4}
Ren *et al.* [21]	484.6203	4478570	1.08208×10^{-4}
Yu *et al.* [74]	308.0526	4573023	0.67363×10^{-4}
Xu *et al.* [87]	481.5319	4767897	1.00994×10^{-4}
Yao *et al.* [88]	302.0317	4774954	0.63253×10^{-4}
Gao *et al.* [29]	422.1275	4786829	0.818521×10^{-4}
Proposed MBQ	310.3405	5034317	0.61645×10^{-4}
Proposed HMBQ	315.1940	5056940	0.62329×10^{-4}
Proposed SHMBQ	324.1043	5063577	0.64007×10^{-4}

5.3.4 Complexity Analysis

The proposed MBQ-RDHEI method has also been analyzed for its time complexity, and found to have $O((M \times N)log(M \times N))$ time complexity. It is mainly because the number of nodes in the quad-tree is proportional to the size of the image, which is $M \times N$, and each node is processed only once. Further, since the method performs a logarithmic number of subdivisions on each node, the total time complexity becomes $O((M \times N)log(M \times N))$.

Further, the computational efficiency of the MBQ method has also been evaluated and compared with the SOTA methods. For this, all the methods are implemented on MATLAB running on a computer with a 6-core AMD Fx-6300 processor, 8 GB of RAM, and the Windows 10 operating system. The results for computational time (in seconds) along with the maximum EC (in bits) and the average time per embedded bit (in seconds/bit) for six test images shown in Fig. 1.3 are provided in Table 5.9. The presented results demonstrate that the MBQ method outperforms Ren *et al.* [21], Yu *et al.* [74], Gao *et al.* [29], Yin *et al.* [90], and Xu *et al.* [87] in terms of computational time, despite having the highest EC among the SOTA methods. As far as average time per embedded bit is concerned, the proposed MBQ method is only behind Chen & Chang [84], Yi *et al.* [83] and Yin *et al.* [76], while surpassing their EC performance by a big margin.

In addition, the proposed MBQ method has also been evaluated for its space complexity. The space complexity of generating the quad-tree is proportional to the size of the image, with the algorithm creating one node for each NB in the quad-tree. The queue used by the algorithm also contributes to the space complexity, but its size is bounded by the number of nodes in the quad-tree. Thus, the overall space complexity is $O(M \times N)$.

5.4 Summary

In this chapter, a novel high-capacity RDHEI method using multi-level blocking with quad-tree was introduced. The MBQ-RDHEI method transforms the original image into a difference matrix having high sparsity in the MSB-planes. The sparse bit-planes are then encoded using a quad-tree based compression strategy to significantly help in reserving a bigger room for embedding. The encoding exploits both inter as well and intra bit-plane redundancy to significantly reduce their size. Experimental findings indicate that the MBQ-RDHEI methods outperform the SOTA methods, by achieving higher ER. More specifically, three versions i.e., MBQ, HMBQ, and SHMBQ of the proposed method have achieved impressive ER of 3.997, 4.011, and 4.0197 *bpp* for BOSSBase and 3.9161, 3.9299, and 3.9343 *bpp* for BOWS-2, respectively, while ensuring complete reversibility of the original images.

CHAPTER 6

HIGH CAPACITY REVERSIBLE DATA HIDING WITH CONTIGUOUS SPACE IN ENCRYPTED IMAGES

The High Capacity RDHEI (HC-RDHEI) methods are facing several significant challenges such as 1) Security: Since the data hider or recipient is required to ascertain the label map for embedding or extraction, the label map must be left in a clear domain that can attract the attacker to break through the method [72, 74, 76, 79, 83]; 2) Time-consuming: It may require extra time when multiple embedding is performed on the same encrypted image [76]- [74]; 3) Non-synchronization: To achieve the high capacity, many RRBE based methods often use the MSB planes for embedding due to their high spatial correlation. However, at times, the LSB planes are also utilized for increased EC, leading to a lack of synchronization between the data and the cover image content [74, 76, 77, 80]. To address the aforesaid challenges, this chapter proposes a new HC-RDHEI method with contiguous space. The proposed methods explore the hybrid predictor to generate a difference matrix. After dropping the sign bit-plane from the difference matrix, a difference image is constructed. Every 2×2 block of the difference image is labeled, and these labels are compressed using Huffman encoding to identify the embeddable space. Moreover, based on the Quicksort algorithm, all reserved space is collected contiguously in $\mathcal{O}(1)$ time, which helps improve the time consumption for the data hider and provides additional security by encrypting the supporting information, unlike traditional methods of RDHEI. Furthermore, an index-based synchronization is proposed to scatter the data content within the image content. Therefore, the methods provide higher EC with lossless reconstruction and recovery while considering time efficiency, security, and synchronization. The following section provides a detailed elaboration of the proposed method.

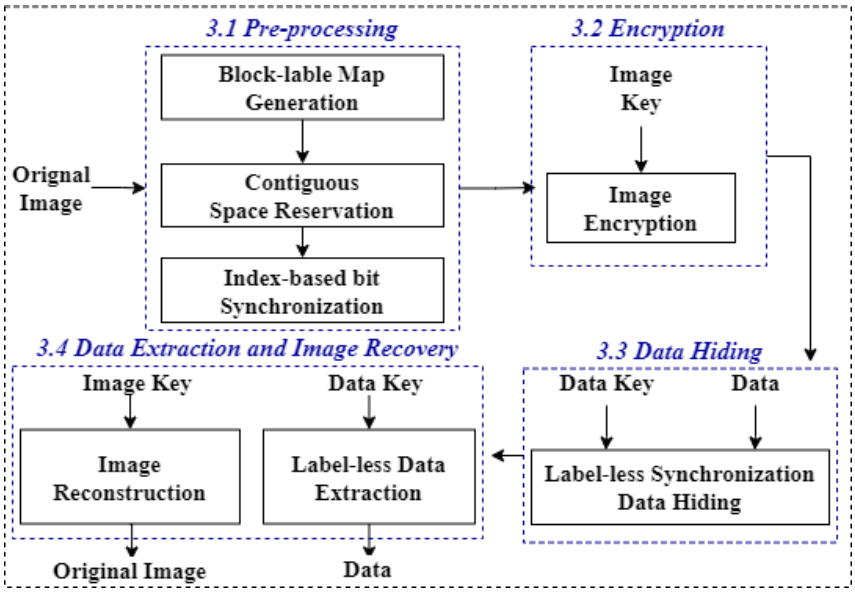


Fig. 6.1: Block diagram of the proposed method

6.1 Proposed Method

In this chapter, we propose an RRBE based HC-RDHEI method that consists of four stages: 1) pre-processing, 2) Encryption, 3) Data hiding, and 4) Data extraction and image recovery, as shown in Fig. 6.1, which are discussed as follows.

6.1.1 Pre-processing

Since the proposed method is an RRBE based HC-RDHEI method, it involves a preliminary pre-processing step on the original image to reserve space for embedding before its encryption takes place. For ease of understanding, the pre-processing phase is divided into three distinct sub-phases: block-label map generation, contiguous space reservation, and index-based bit synchronization. The following sub-sections provide a detailed explanation of the aforementioned sub-phases:

A. Block-label Map Generation

The MED and difference predictors are explored to determine the difference matrix D for original image I, similar to methods in 4 and 5.

The absolute value of the prediction error is determined as follows: $e = |D|$. To determine the Block Label Map (BLM), the matrix is non-overlapping divided into 2×2 size blocks, where the maximum absolute prediction error (e_{max}) of each block determines its block label (l) as follows:

$$l = \begin{cases} 7-k, & \text{if } 2^{(k-1)} \leq e_{max} < 2^k \\ 8, & \text{else } e_{max} = 0 \end{cases} \tag{6.1}$$

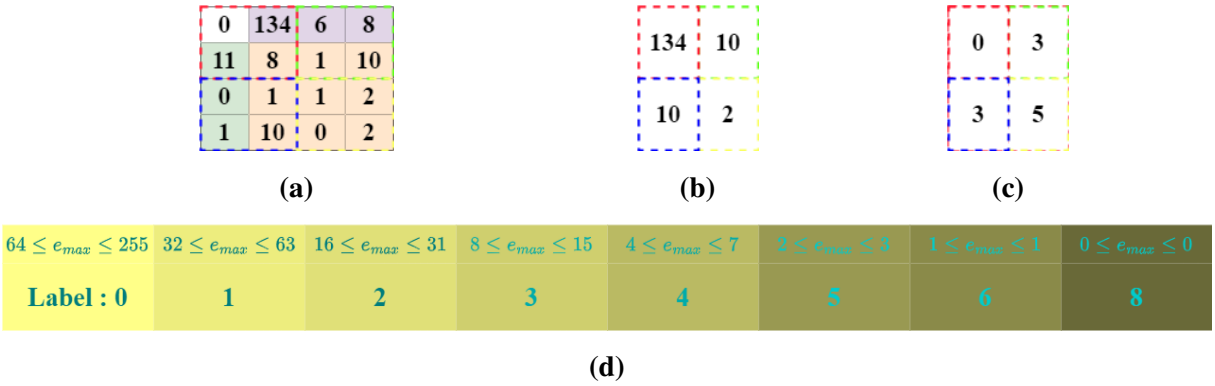


Fig. 6.2: An illustrative example for block label generation

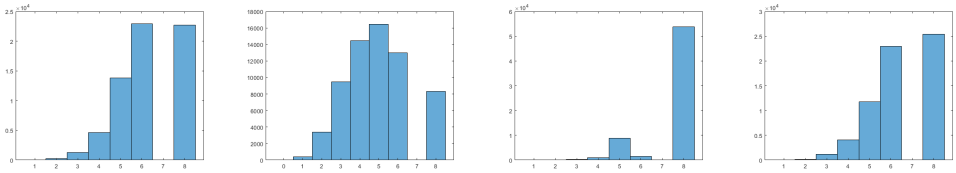


Fig. 6.3: BLM histogram for test image (a) $Lena$, (b) $Baboon$, (c) $Jetplane$, and (d) $Tiffany$

where $7 \geq k \geq 1$. Thus, the generated BLM contains only eight block labels i.e., 0,1,2,3,4,5,6, and 8.

To demystify the process of the BLM generation, an illustrative example is presented in Fig. 6.2. In the example, an absolute prediction error matrix as shown in Fig. 6.2a, is obtained, which is divided into 2×2 blocks, and the maximum e (e_{max}) of each block is used to represent the block, depicted in Fig. 6.2b. Finally, the label of each block is determined based on the max absolute prediction error as depicted in Fig. 6.2c, and BLM is generated by scanning all the l of I in a raster-scan order.

In order to create a big room for embedding, it is important to compress the BLM before embedding it in the image. Since the labels of the BLM are highly uneven in terms of frequency, which is further underscored by the histogram distribution presented in Fig. 6.3, Huffman encoding with fixed code is employed as it has a proven record of providing optimal compression ratio on unevenly distributed data with a significant amount of processing time. Next, each label of the BLM is replaced by a Huffman code (h_f) derived according to its frequency and a compressed sequence (s_c) is received. For lossless image reconstruction at the receiving end, the aforesaid information which includes s_c, h_f for each unique label, and reference pixel are arranged in a specific order to be saved in the image and is known as AI. Thus, the maximum length of AI, which is denoted by AI_{len}^{max}, is $\lceil (log_2(3/4(M \times N) + 35 + 8) \rceil$ bits, as there exist $(M \times N)/4$ total number of labels in BLM and at most 3 bits are needed to represent each of the labels l.

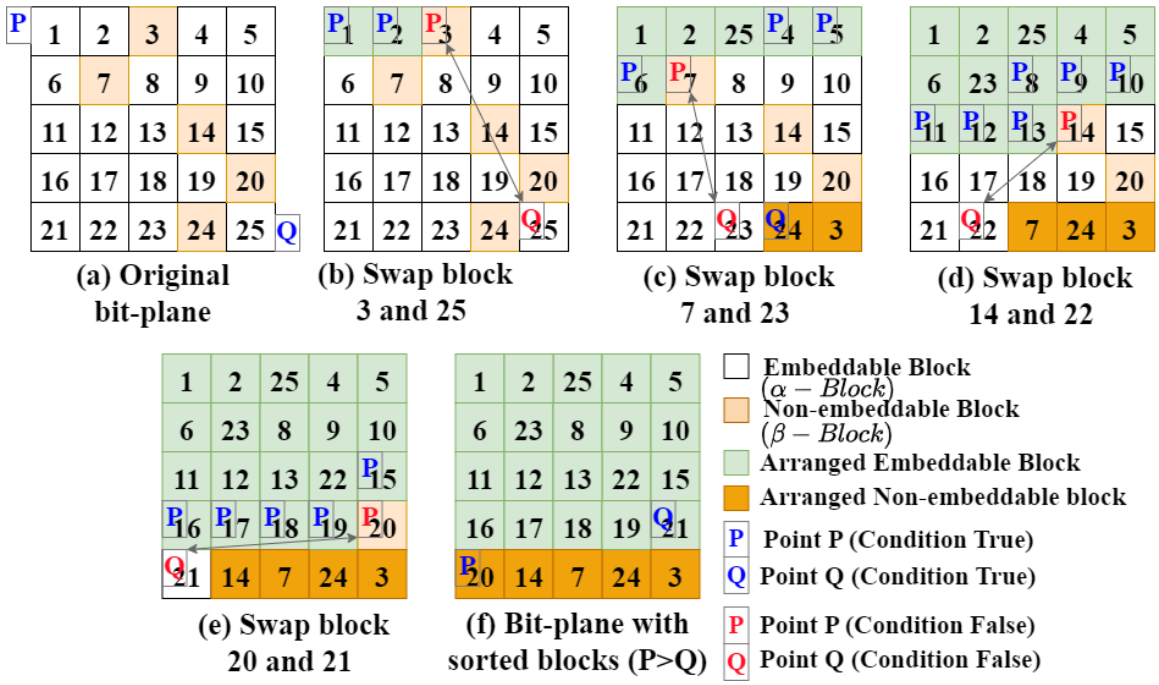


Fig. 6.4: Block arrangement of bit-plane in order to identify the contiguous space

B. Contiguous Space Reservation

For creating contiguous space at the beginning of the image for embedding, the bit-planes are processed starting from MSB to LSB plane (i.e. $k = 7 \sim 0$), in a block-wise manner. For this, the blocks of the bit-plane are first categorized into two distinct groups, namely embeddable ($\alpha - block$) and non-embeddable block ($\beta - block$), based on the block label (l), for each bit-plane. More specifically, if $l > 7 - k$ for k^{th} bit-plane, then block is considered as $\alpha - block$ otherwise $\beta - block$. Further, if a block of k^{th} bit-plane has been $\beta - block$ then the corresponding block of all of its subordinate/following bit-planes, which includes $(k-1, ..., 0)$, are not considered for processing, or in other words, marked as $\beta - blocks$.

Next, all $\alpha - block$'s are shifted to the beginning followed by all the $\beta - block$'s, in each bit-plane. For this, a novel shifting mechanism is employed to complete the shifting process in linear time. This shifting is inspired by the partitioning algorithm of the quick-sort [103]. For this, bit-planes of all the blocks are evaluated in the raster scan order. Further, two pointers P and Q are set to point to the first and the last blocks of bit-plane. Then, P starts moving to the next block (right side) while the block is $\alpha - block$ and Q keeps moving to the next block (left side) while the block is $\beta - block$. When P and Q both stops, P points to $\beta - block$ and Q points to $\alpha - block$. Now, the P and Q bit-plane blocks are swapped along with corresponding blocks of all subordinate bit-planes. This process is continued until $Q > P$, and contiguous space, at the beginning of the image, is obtained.

An illustration presented in Fig. 6.4 provides a comprehensive and intricate depiction of the block rearrangement in the context of contiguous space. Initially, the pointer P is at the first block and the pointer Q is at the last block of the original bit-plane of 25 blocks, as shown in Fig.

6.4a. Now P starts moving to the next block and keeps on moving in the raster scan sequence while the blocks are $\alpha - blocks$ and Q also starts moving in the reverse raster scan direction while the blocks are $\beta - blocks$. Since the third block (i.e., 3,) for P is $\beta - blocks$ and the first block (i.e., 25), for Q is $\alpha - blocks$, swapping of blocks 3 and 25 takes place as demonstrated in Fig. 6.4b. The process continues in a similar manner, with subsequent swaps involving blocks 7 and 23, 14 and 22, and 20 and 21 as shown in Fig. 6.4c, 6.4d, 6.4e. respectively; and the terminating condition $Q > P$ is met. Thus, all the $\alpha - blocks$ and $\beta - blocks$ are effectively separated as depicted in Fig. 6.4f, to provide the contiguous space for embedding.

To enable lossless reconstruction of I, a number of $\alpha - blocks$ are recorded from MSB plane. Let the number of $\alpha - blocks$ (MSB onward) are $b_{\alpha^{(7)}}$, $b_{\alpha^{(6)}}$, $b_{\alpha^{(5)}}$, ..., $b_{\alpha^{(1)}}$; where $b_{\alpha^{(7)}} \geq b_{\alpha^{(6)}} \geq ... \geq b_{\alpha^{(1)}}$. Therefore, $b_{\alpha^{(k)}}$ can be represented by at most $\lceil log_2(b_{\alpha^{(k+1)}}) \rceil$ bits. Since, I has been divided into 2×2 blocks, $b_{\alpha^{(7)}}$ can be represented at most by $\lceil log_2(\frac{M \times N}{4}) \rceil$ bits. In certain cases for some of the LSB planes, however, $log_2(b_{\alpha^{(k+1)}}) \leq 4b_{\alpha^{(k)}}$ may hold, which means the number of bits to represent $b_{\alpha^{(k)}}$ are greater than the amount of secret data that can be hidden in the bit-plane. That also means embedding should only be done until the aforementioned condition is false, and hence the information about the number of embeddable bit-planes (n), which can be represented by 3 bits, is also recorded and made available to the data hider for enabling error free extraction at the receiving end. Additionally, the data hider also needs to skip the pixels storing the AI from the contiguous space to ensure the recovery of the compressed map. Therefore, 3 and AI_{len}^{max} bits to identify n and AI respectively, are prefixed with $\lceil log_2(b_{\alpha^{(k+1)}}) \rceil$ bits of $b_{\alpha^{(7)}}$ and embedded in the $7th$ bit-plane in block-wise raster scan order. Where the bit-plane blocks are sequentially traversed following a raster scan order and within each block, the individual bits are also traversed in a raster scan order. Similarly, numbers of $\alpha - blocks$ for $n - 1$ subsequent MSB planes are also embedded in their corresponding bit-plane. These all embedded bits are recognized as SI which is required at the hiding, extraction and recovery stages. Thereafter, AI bits are embedded from the MSB plane onward, after skipping SI bits and till $4b_{\alpha^{(k)}}$ bits at the k^{th} bit-plane. The positions of AI and SI are depicted in Fig. 6.5 to clarify the bifurcation of the overhead. Further, to illustrate the partition and respective roles of the associated overhead components, Fig. 6.5 presents a visual representation of the AI and SI.

C. Index-based bit Synchronization

To synchronize the reserved room with the cover image pixels, the bits of each pixel within a block are shuffled with circular shift using Eq. (6.2) defined as follows:

$$\hat{p}' = \left(2^{S_K} \hat{p} + \left\lfloor \frac{\hat{p}}{2^{(8-S_K)}} \right\rfloor \right) \bmod 2^8, \tag{6.2}$$

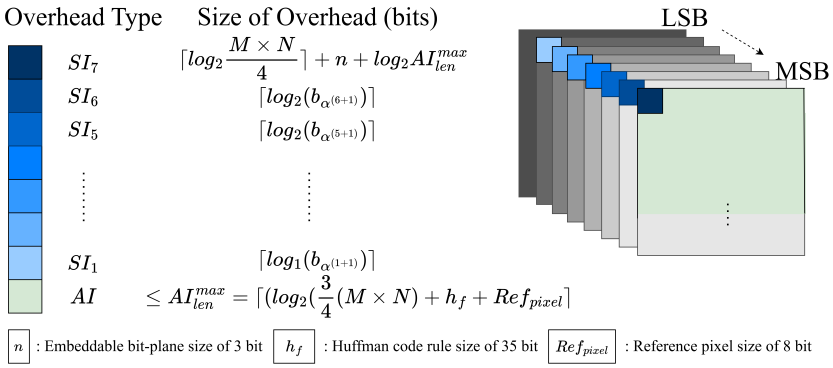


Fig. 6.5: Illustration of method overhead

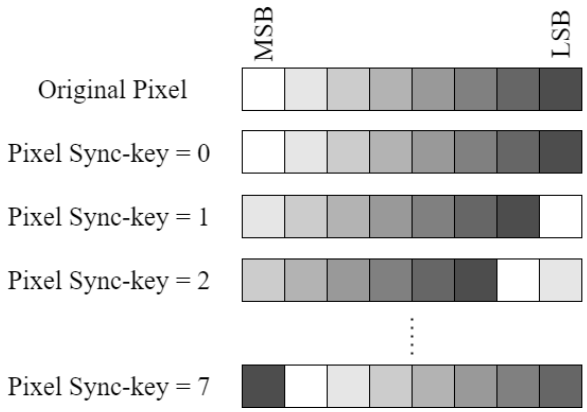


Fig. 6.6: An illustration of bit-shuffling using a sync-key

where $\hat{p}$ denotes a pixel located at coordinates (i, j) within I_s and S_K denotes a sync-key which is derived from the pixel and block indexes in the images using Eq. (6.3).

$$S_K = \left(\left\lceil \frac{i}{2} \right\rceil \left(\frac{N}{2} \right) + \left\lceil \frac{j}{2} \right\rceil \right) \bmod 8. \tag{6.3}$$

Thus, the above derivation of the sync-key (S_K) helps the data hider and the recipient in facilitating the synchronization without any explicit sharing of side/auxiliary/supporting information. To further elucidate the concept, an example to illustrate the above bit-shuffling approach is portrayed in Fig. 6.6, where the MSBs of the pixels are shifted circularly according to the S_K, which is derived from the pixel's indexes.

6.1.2 Image encryption

The AES in CTR mode is adopted to encrypting the image I_s similar to method in 4. Using the Eq. (3.9), the encrypted image with SI $I_{e'}$ is obtained. However, it is important to note that SI bits are skipped during the aforementioned encryption process. Thus, this encryption ensures the security and privacy of the image from any potential threats.

6.1.3 Data Hiding

In order to locate the contiguous space suitable for embedding the data, the data hider commences by identifying the reserved room utilizing the Eq. (6.2). Subsequently, he/she calculates the sync-key S_K using Eq. (6.3) by taking into account the specific block index for each case.

Upon examining the synchronized positions of the bit in each pixel, the data hider proceeds to read the bits of SI. The reading of SI is started from the 7_{th} bit-plane, and SI_7 is decoded to establish the n, count of the $\alpha - blocks$ and AI_{len}^{max}. Subsequently, the SI bits for each n MSB bit-plane are perused and decoded into the respective $\alpha - blocks$ for each bit-plane. It is important to note that the total number of $\alpha - blocks$ in the 0^{th} bit-plane is equivalent to those in the 1^{st} bit-plane, thus eliminating the need to process the 0^{th} bit-plane for its $\alpha - blocks$. Following this, the data hider places the ciphered data within the contiguous space, as identified through Eq. (6.2). This is accomplished by bypassing the initial bits of SI and AI, ultimately yielding a marked encrypted image (I_m).

6.1.4 Data Extraction and Image Recovery

The method under consideration permits the independent execution of data extraction and image recovery, contingent on which keys are available to the receiver. Consequently, three distinct cases become feasible, as elaborated below.

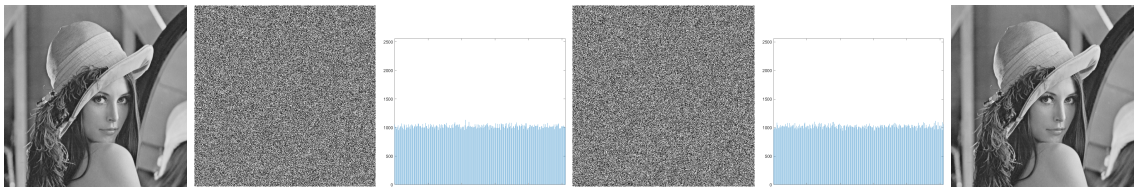
Case1: When the receiver has access only to K_D, he first identifies the contiguous space in I_m after pinpointing the synchronized positions, akin to the data hider as in subsection-6.1.3. Subsequently, the recipient circumvents the AI to extract the hidden bits which undergoes decryption utilizing K_D, ultimately revealing the original hidden data.

Case2: In the availability of only K_I with the receiver, I_m is decrypted using Eq. (3.9) by first generating the matrix χ similar to image encryption, as in subsection-6.1.2. Next, the bits of AI are read from the contiguous space to get the L_c, h_f, and reference pixel after pinpointing the synchronized positions similar to subsection-6.1.1. L_c is then decoded with the help of h_f to get the original block label map (L). Thereafter, to reverse the rotation of the bits of the pixel to their original position, Eq. (6.4) is used after determining the sync-key using Eq. (6.3).

$$\hat{p}' = \left(2^{(8-S_K)}\hat{p} + \left\lfloor \frac{\hat{p}}{2^{(S_K)}} \right\rfloor\right) \bmod 2^8. \tag{6.4}$$

Next, using L, the bit-plane blocks of each plane in I_m are revert-allocated by the shifting mechanism defined in subsection-6.1.1. The label l is assigned to each pixel of the image using L, and $I_r(i,j)$ is reconstructed by $I_m(i,j)$. During the reconstruction, the prediction value of the first row and first pixels are determined by the last and upper pixels, while the remaining pixels are predicted using the MED. This is done after setting the value of $I_r(1,1)$ by reference pixel. Thus, proceeding, I is losslessly reconstructed.

Case 3: When the receiver is in possession of both keys (K_D and K_I), not only can the



(a) $Lena$ **(b)** **(c)** **(d)** **(e)** **(f)** $Lena$

Fig. 6.7: Processed image $Lena$ (a) Original image I, (b) Encrypted image $I_{e'}$, (c) Histogram of $I_{e'}$, (d) Marked image I_m, and (e) Histogram of image I_m (f) Reconstructed image

secret data be extracted as described in case 1, but the original image can also be successfully reconstructed, akin to the procedure outlined in case 2.

6.2 Experimental Results

This section serves to provide compelling results, substantiating the effectiveness of the proposed method. In the evaluation process, four test images (namely, $Lena$, $Baboon$, $Jetplane$, $Tiffany$) are sourced from the SIPI datasets as depicted in Fig. 1.3. Additionally, two renowned datasets, specifically BOSSBase [38], and BOWS-2 [39], are utilized for analysis. Each dataset comprises 10,000 grayscale images with dimensions 512×512.

6.2.1 Security Analysis

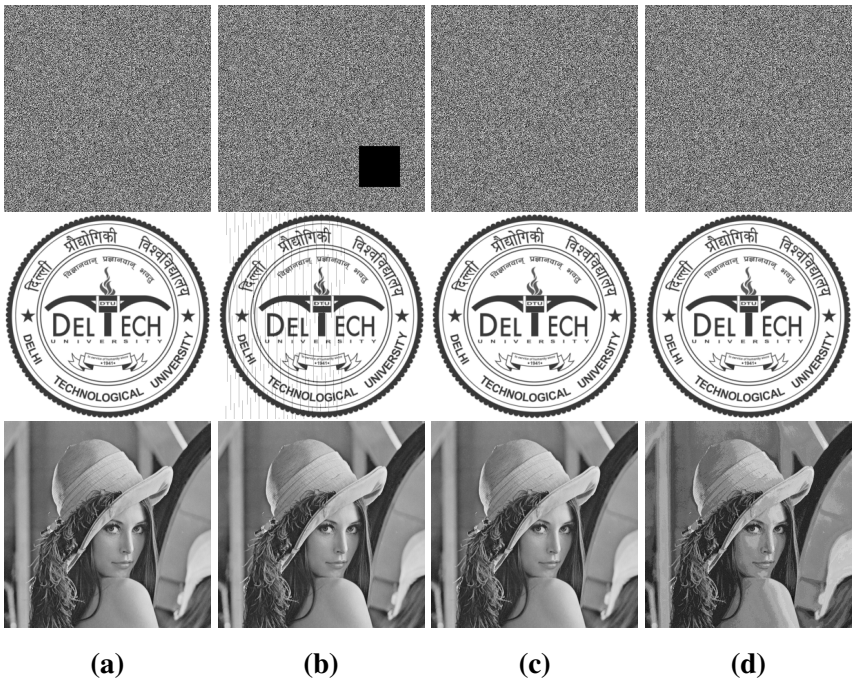
In order to gauge the robustness of the proposed method, we incorporated a selection of esteemed metrics. The metrics encompass Shannon entropy, along with UACI. It also delves into the domain of horizontal and vertical correlation. Furthermore, the evaluation is extended to consider histogram distribution, as well as the metrics of PSNR and SSIM. The analysis is rounded off with the inclusion of the NPCR.

Furthermore, the image of $Lena$, which was illustrated in Fig. 1.3a, had been selected as the representative use case and is further displayed in Fig. 6.7a. Displayed in Fig. 6.7b and Fig. 6.7d are the encrypted ($I_{e'}$) and marked (I_m) versions of the original $Lena$ image, respectively. As evident, both images present a noise-like, scrambled appearance. The uniformity of histogram distributions for both versions is further elucidated in Fig. 6.7c and Fig. 6.7e. This uniformity across intensity levels and noise-like scrambled appearance attests to the high degree of perceptual security by the method.

The statistical metrics of our experiment are detailed in Table 6.1. Near-zero correlation values are observed in both horizontal and vertical dimensions for the encrypted ($I_{e'}$) and marked encrypted (I_m) images, denoting a lack of dependency between adjacent pixels. That is in contrast to the original image (I), which exhibits a correlation nearly equal to 1. Furthermore, the entropy levels for $I_{e'}$ and I_m are marginally higher than for I, pointing to a richer variety of pixel values in the manipulated images. In terms of resistance to differential attacks, the NPCR

Table 6.1: Evaluation of security on test image $Lena$

Test Image	Correlation Horizontal	Correlation Vertical	Entropy	NPCR	UACI	PSNR	SSIM
I	0.9719	0.9850	7.4451	-	-	-	-
$I_{e'}$	0.0012	0.0014	7.9994	99.67	73.58	9.2421	0.0363
I_m	-0.0003	0.0024	7.9993	99.60	72.66	9.2223	0.0341



(a) **(b)** **(c)** **(d)**

Fig. 6.8: The experimental results obtained under various removal attack scenarios (a). Original, (b) Patch removal, (c) Fourth bit-plane removal, and (d) Fifth bit-plane removal

and UACI metrics are impressively high, roughly 100% and 73%, respectively. Conversely, the notably low SSIM and PSNR values emphasize the level of unlikeness between the manipulated and original images. To encapsulate, the data robustly validates that the method offers a formidable shield against a diverse range of security threats, whether they are visual, differential, or statistical. In summary, Table 6.1 is evident that the method is exceptionally resilient to a variety of security threats, encompassing visual, differential, and statistical vectors of attack.

To further investigate the security in terms of robustness, various popular tampering attacks such as patch removal, bit-plane removal, and brute force attacks have been considered. For this, a logo is embedded within the image $Lena$, as shown in Fig. 6.8a. To check the robustness against the patch removal, a patch from the resultant image has been removed. Later on, the proposed image recovery and extraction method is employed, results of which are presented in Fig.6.8b. The presented results show that there is only minimal distortion in the restored cover image, and the extracted logo also maintains its visibility. Next, bit-plane removal attacks by removing 4^{th} and 5^{th} bit-planes are also employed, results for them are also presented in Fig.6.8c and 6.8d, respectively. The presented results clearly show that the recovered image

Table 6.2: Bit-plane wise embedding performance on test image $Lena$

Bit-plane (index)	Embeddable 2×2 blocks count	Auxiliary information (bits)	Supporting information (bits)	Pure Embedding capacity (bits)
7^{th}	65435	137752	16+3+18	123951
6^{th}	64202	-	16	256792
5^{th}	58738	-	16	234936
4^{th}	43233	-	16	172916
3^{rd}	14781	-	16	59108
2^{nd}	1678	-	14	6698
1^{st}	300	-	11	1189
0^{th}	300	-	-	1200
Pure embedding capacity(bits) / rate (bits/pixel)=856790/ 3.2684				

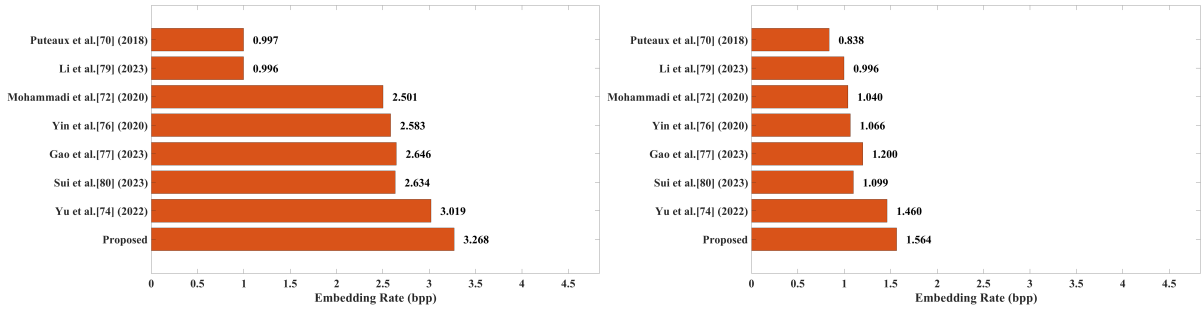
Table 6.3: EC bifurcation of four test images

Test image	Total embedding capacity (bits)	Auxiliary information (bits)	Supporting information (bits)	Pure Embedding capacity (bits)
$Lena$	994668	137752	126	856790
$Baboon$	566284	156276	101	449907
$Jetplane$	1106824	160042	124	966658
$Tiffany$	1028532	143026	122	885384

and the extracted logo have been able to maintain their original visibility. This underlines the importance of carefully managing the extraction and preservation of information within the image. Moreover, the proposed method's incorporation of AES for securing image content, together with the versatility to utilize any robust encryption algorithm for safeguarding data, effectively diminishes the feasibility of successful brute-force attacks. Therefore, one can assert that the method demonstrates exceptional resilience against various forms of tampering attacks.

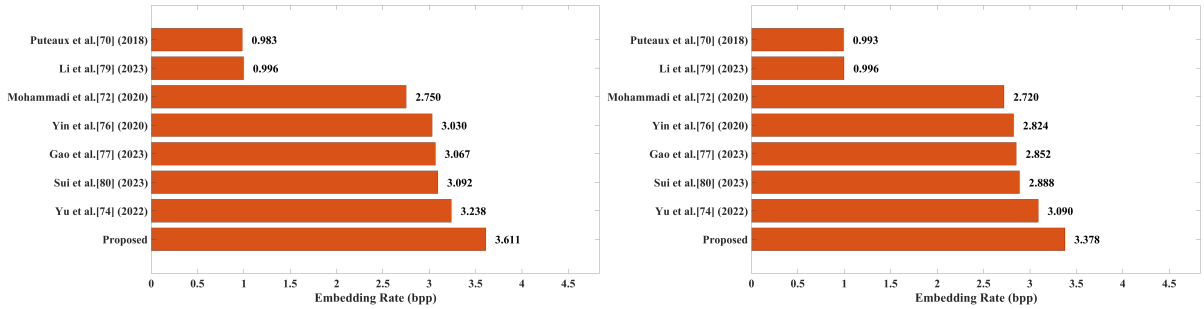
6.2.2 Embedding Performance Evaluation

Here, the embedding performance of the proposed method is analyzed considering the $Lena$ image. The experimental results to showcase bit-plane-wise embedding performance are detailed in Table 6.2. As evident, the number of $\alpha - blocks$ is the highest for the 7^{th} and the lowest for the 0^{th} bit-plane. Similarly, AI and SI which essentially are overhead but required for complete reversibility and lossless extraction, need the highest number of bits for the 7^{th} bit-plane. The $Total\ EC$ of the method is four times of $\alpha - blocks$ number. However, the $Pure$EC which is $Total\ EC$ - $overhead$, is actually 856790 bits. Consequently, the ER amounts to 3.2684 bpp. Additionally, the embedding capacity for other test images has also been examined, as illustrated in Table 6.3. The $Pure$EC values for $Baboon$, $Jetplane$, and $Tiffany$ stand at 449907, 966658, and 885384 bits respectively. These values clearly show that the $Pure$EC metric is directly correlated with the smoothness of the image.



(a) *Lena*

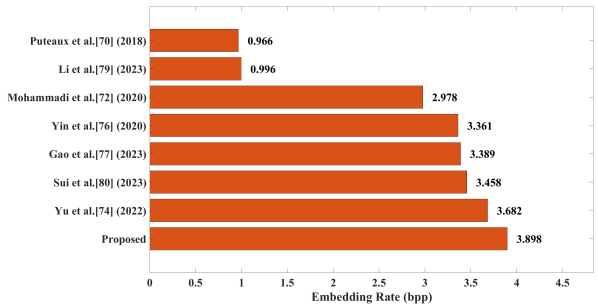
(b) *Baboon*



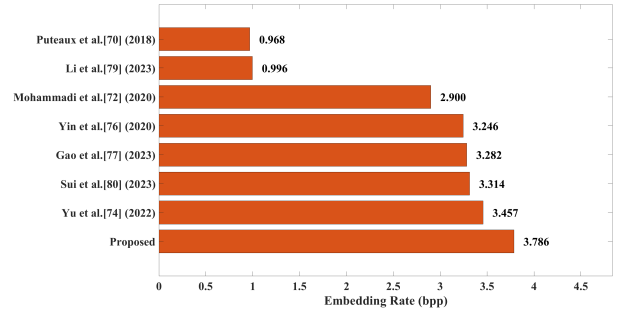
(c) *Jetplane*

(d) *Tiffany*

Fig. 6.9: Comparison of ER (in *bpp*) with SOTA methods on standard test images



(a) BOSSbase



(b) BOWS-2

Fig. 6.10: Comparison of ER (in *bpp*) with SOTA methods on two open datasets

6.2.3 Comparison with State-of-the-art Methods

To convincingly and comprehensively compare the proposed method with SOTA methods, the parameters for each of the SOTA methods are fine-tuned. A comparative visualization of the performance metrics, specifically in terms of ER, can be found in Fig. 6.9. This figure contrasts our approach with notable SOTA techniques, referencing [74], [80], [77], [76], [72], [79] and [70], utilizing four benchmark images from the SIPI dataset. As indicated in Fig. 6.9, the ER values for *Lena, Baboon, Jetplane*, and *Tiffany* stood at 3.2684, 1.5637, 3.6112, and 3.3775 *bpp*, respectively. Remarkably, these figures considerably outpace those recorded by the contemporary SOTA methods. It is to be noted that our method ensures lossless image reconstruction, achieving infinite PSNR and one SSIM.

To augment the validation of the proposed method's effectiveness, performance metrics are

Table 6.4: Comparison of computation time and EC of the proposed method with the SOTA methods

Method	Total computational time of 4-test images(seconds)	Total EC of 4-test images (bit)	Average time per embedded bit (seconds/bit)
Puteaux *et al.* [70] (2018)	79.8163	999037	0.798932×10^{-4}
Li *et al.* [79] (2023)	101.0429	1048568	0.963627×10^{-4}
Mohammadi *et al.* [72] (2020)	177.9376	2365025	0.752372×10^{-4}
Yin *et al.* [76] (2020)	138.2741	2491553	0.554972×10^{-4}
Gao *et al.* [77] (2023)	161.0121	2546313	0.632334×10^{-4}
Sui *et al.* [80] (2023)	157.9483	2561235	0.616688×10^{-4}
Yu *et al.* [74] (2022)	206.4126	2832816	0.728648×10^{-4}
Proposed	214.5174	3098742	0.692273×10^{-4}

evaluated on big datasets, i.e., BOSSbase and BOWS-2. For this, the best, average, and worst-case ERs for each dataset have been considered. Specifically, for BOSSbase and BOWS-2, these ERs are recorded at 0.9796, 3.8977, and 7.6180 bpp, and 0.9035, 3.7863, and 7.0751 bpp respectively. Next, the performance comparison on the aforementioned data sets is done, and experimental results are presented in Fig. 6.10. These visuals confirm that our proposed method outperforms existing SOTA techniques. Specifically, gains of 0.245 bpp and 0.329 bpp are noted for BOSSbase and BOWS-2 datasets, when compared to the top-performing SOTA methods. Conclusively, our proposed method consistently outperforms the SOTA methods, but does so with significant margins in terms of ER across multiple large-scale datasets.

6.2.4 Time Complexity Analysis

The proposed method has been scrutinized for its time complexity and determined to possess $O(M \times N)$ as overall complexity. This is attributed to the single-time processing of each pixel within the block, which amounts to $M \times N$. To furnish a thorough examination of time complexity, we conducted an analytical assessment to measure the computational time consumed by our proposed method as well as by each of the prevailing SOTA methods. In order to provide a more comprehensive analysis of the time complexity, we have assessed the computational time required by the proposed method and each of the SOTA methods. This evaluation was carried out using MATLAB software on a computer configured with a 6-core AMD Fx-6300 processor, 8 gigabytes of Random Access Memory, and operating under the Windows 10 operating system. The computational time (in seconds), maximum embedding capacity (in bits), and average time per embedded bit (in seconds/bit) for all the methods applied to four test images are detailed in Table 6.4. It is evident that our method demonstrates superior performance in terms of Average Time Per Embedded Bit, outpacing all referenced methods except Yin *et al.* [76]. This makes the proposed method more appealing as it provides the highest EC among all the methods.

6.3 Summary

In this section, we highlight the effectiveness of the proposed HC-RDHEI method. This method's strength lies in its adept exploitation of spatial correlations, both at the block and bit-plane levels, resulting in a significant enhancement of its embedding capabilities. By recognizing redundancy in original images and generating prediction errors, the proposed method can effectively distinguish between embeddable and non-embeddable blocks. It's worth noting that blocks are strategically arranged to ensure embeddable blocks precede non-embeddable ones for each bit-plane, and the images produced after this process are thoughtfully bit-shuffled. Furthermore, the integration of the counter mode of AES in the encryption process adds an extra layer of robustness, promoting the overall security of the image. Throughout our extensive experiments, the proposed method consistently outperformed existing works, demonstrating an ER gain of 0.216 and 0.329 bpp with respect to the best-performing existing method for BOSSbase and BOWS-2 datasets, respectively. These results clearly surpass the performance benchmarks set by the SOTA methods.

CHAPTER 7

ADAPTIVE TWO-STAGE REVERSIBLE DATA HIDING IN ENCRYPTED IMAGES USING PREDICTION ERROR EXPANSION

As per the analysis of existing literature, VRAE methods often have a limited EC. Moreover, the VRAE methods generate an overhead in terms of location map, which is used to address the issue of overflow/underflow. To address this concern, this chapter presents a novel adaptive two-stage RDHEI method based on prediction error expansion to enhance security and adaptability across a range of applications. The image encryption procedure involves transforming the image into two images using bit-plane division, followed by block scrambling, pixel shuffling, and encryption while preserving the correlation of the blocks. In the data hiding phase, a pixel value ordering method is used to embedding the secret data as well as managing probable overflow/underflow issues using the label map. In the next section, a thorough explanation of the implemented method is presented.

7.1 Proposed Method

This section presents a new Adaptive Two-Stage (ATS)-RDHEI method that utilizes two-stage embedding to improve the performance of the embedding. The ATS-RDHEI method falls under the VRAE-RDHEI category, as it preserves the correlation in small blocks in two distinct parts within the encrypted image while employing this correlation for data embedding. Moreover, the method separates the images' bit-planes, to increase the EC as well as effectively reduce the size of the LM. This section is organized into three subsections: image encryption, data embedding, and data extraction alongside image recovery. To provide a comprehensive understanding of the ATS-RDHEI method, Fig. 7.1 presents a block diagram illustrating the entire process.

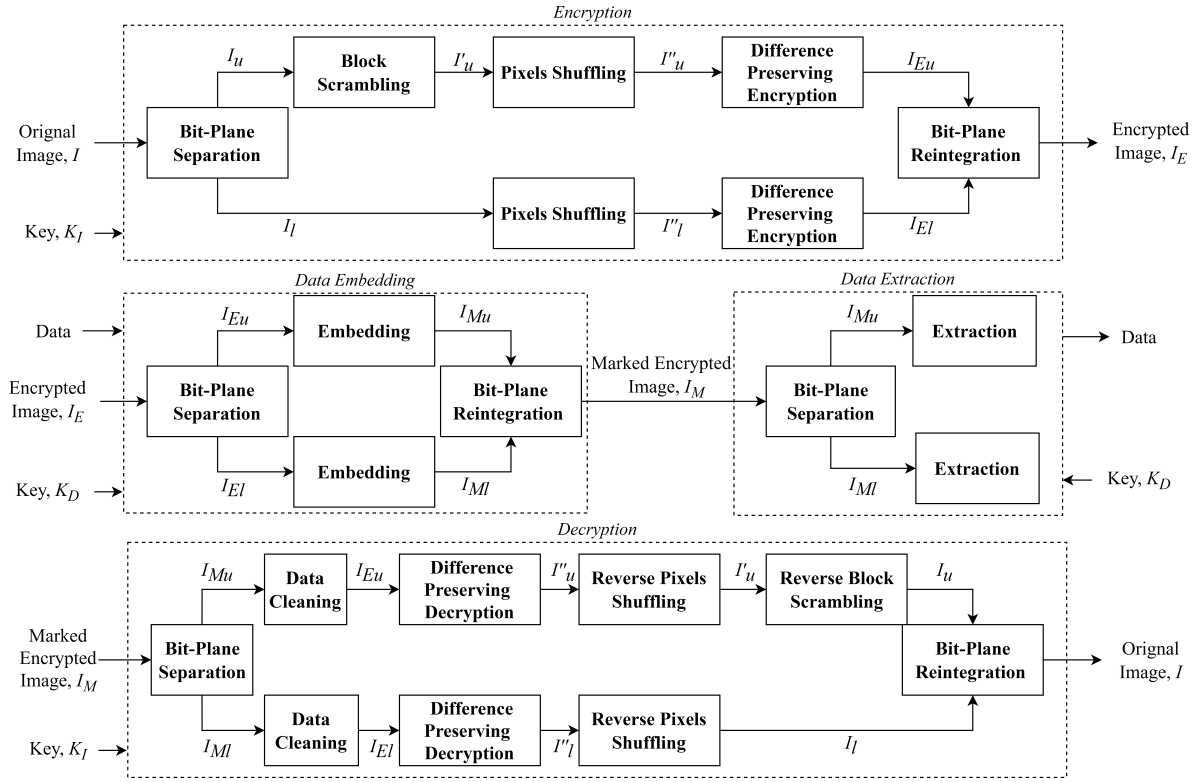


Fig. 7.1: Block diagram of the proposed method

7.1.1 Image Encryption

The main objective of image encryption is to obfuscate the image content, rendering it unrecognizable prior to the embedding process. The proposed method encompasses a four-phase in image encryption procedure, which includes bit-plane division, block scrambling, pixels shuffling, and error-based encryption. The bit-plane division is performed without the use of any key.

The method incorporates data embedding at two stages of the image processing. Therefore the gray-scale image is separated into bit-planes, after which the four MSB planes are collectively reorganized, and likewise, the four least significant bit (LSB) planes are rearranged together. Let I represent a gray-scale image with dimensions $M \times N$, and let b_8, b_7, b_6, ... ,b_1 denote the bit-planes of I, ranging from the MSB plane to LSB plane, respectively. The computation of the bit-planes can be described as follows:

$$b_k(i,j) = \left\lfloor \frac{(I(i,j)) \bmod (2^{k+1})}{2^k} \right\rfloor, \tag{7.1}$$

where b_k is the k^{th} bit-plane of pixel located at (i,j) in image I, $1 \leq i \leq M$, $1 \leq j \leq N$ and $8 \geq k \geq 1$.

In the following phase, block scrambling is carried out using random permutation. Initially, images I_l is divided into non-overlapping blocks of size 2×2 and create a total of $\frac{M \times N}{4}$ blocks. Subsequently, $\frac{M \times N}{4}$ random keys (R) are generated using a permutation key generated through

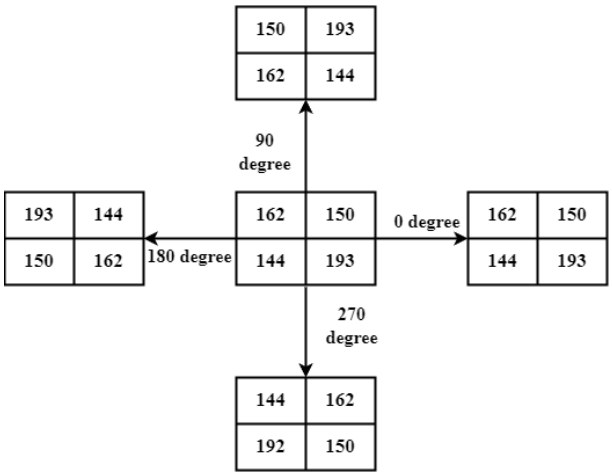


Fig. 7.2: Block scrambling using degree

an image key (K_I) and a PRNG. The image blocks are scrambled according to their corresponding index value, which is denoted as B. Thereafter, the images partially encrypted image I'_u is obtained.

In the subsequent phases of encryption, two pseudo-random matrices $\mathcal{P}_u$ and $\mathcal{P}_l$ of size $\frac{M}{2} \times \frac{N}{2}$ and $M \times \frac{N}{2}$ is generated using the PRNG, with the image key K_I serving as a seed. A key for each 2×2 size block of image I'_u is assigned from $\mathcal{P}_u$, and for image I_l, it is assigned from $\mathcal{P}_l$ after dividing the image into 1×2 blocks. Subsequently, pixel shuffling is performed on each block of both images and the final images I''_u and I''_l are obtained. The rotation angle for a block B of image I'_u is determined as follows

$$Angle = \begin{cases} 0^\circ, & \text{if } \mathcal{P}_{u(B)} \text{mod } 4 = 0 \\ 90^\circ, & \text{elseif } \mathcal{P}_{u(B)} \text{mod } 4 = 1 \\ 180^\circ, & \text{elseif } \mathcal{P}_{u(B)} \text{mod } 4 = 2 \\ 270^\circ, & \text{otherwise} \end{cases} \tag{7.2}$$

where the $^\circ$ symbol represents the degree of block scrambling and $\mathcal{P}_{u(B)}$ is the scrambling key corresponding to block B. However, the scrambling angle of the lower image block is determined as follows:

$$Angle = \begin{cases} 0^\circ, & \text{if } \mathcal{P}_{l(B)} \text{mod } 2 = 0 \\ 180^\circ, & \text{otherwise} \end{cases} \tag{7.3}$$

To further illustrate the block scrambling process, Fig. 7.2 is provided as a visual representation.

In the final step of encryption, each block of I''_u is processed as follows: let $p_{(1)}$, $p_{(2)}$, $p_{(3)}$, and $p_{(4)}$ represent the pixels of a block in raster scan order. The ascending order of the pixels is denoted as $p'_{f(1)}$, $p'_{f(2)}$, $p'_{f(3)}$, and $p'_{f(4)}$, where $p'_{f(1)} \leq p'_{f(2)} \leq p'_{f(3)} \leq p'_{f(4)}$. Furthermore, if two pixels have the same value, the pixel that appears first in the sequence is placed first in

the sorted list. Subsequently, the Prediction Error (PE) between each pair of pixels is calculated and denoted as $e_{(1,2)}$, $e_{(2,3)}$, $e_{(3,4)}$, where $e_{(x,y)} = p'_{f(y)} - p'_{f(x)}$, $1 \leq x \leq 4$ and $1 \leq y \leq 4$. The encrypted pixels of the block are then determined as follows: if $e_{(1,4)}$ is less than or equal to 12 $(2^4 - 4)$, the block uses Eq. (7.4).

$$p''_{f(x)} = \begin{cases} (p'_{f(x)} + \mathcal{P}_{(B)}, \text{ mod } (\alpha - e_{(1,n)}) + 1 & \text{if } x = 1 \\ (p''_{f(x-1)} + e_{(x-1,x)}, & \text{otherwise} \end{cases} \tag{7.4}$$

where n is the total number of pixel in block and α is threshold, for image I''_u, $n = 4$ and $\alpha = 12$. Otherwise, the encrypted pixels of the blocks are obtained using given four conditions, $p''_{f(1)} = p'_{f(2)}, p''_{f(2)} = p'_{f(1)}, p''_{f(3)} = (p'_{f(3)} + \mathcal{P}_{(B)})$ mod 8+4 and $p''_{f(4)} = (p'_{f(4)} + \mathcal{P}_{(B)})$ mod 8+4.

Conversely, let $p_{(1)}$ and $p_{(2)}$ be the pixels of any block in I'_l. Their ascending order is $p'_{f(1)}$, $p'_{f(2)}$, and the PE is $e_{(1,2)}$. If $e_{(1,2)}$ is less than or equal to 14, the encrypted pixels are determined by Eq. (7.4), where α is 14 $(2^4 - 2)$ and $n = 2$. Otherwise, the encrypted pixels of the block are obtained using four conditions.

Upon completion of the aforementioned process, two encrypted images $I_{E(u)}$ and $I_{E(l)}$ are obtained, respectively. These images are combined using Eq. (7.5) to generate the final encrypted image, denoted as I_E.

$$I_E(i,j) = 2^4 * I_{E(u)}(i,j) + I_{E(l)}(i,j). \tag{7.5}$$

7.1.2 Data Embedding

The data hider initially encrypts the data using a highly secure encryption algorithm, employing a data key K_D and generating a data stream. Thereafter, the received encrypted image I_E is initially separated into two images, upper encrypted image $I_{E(u)}$ with four MSB planes and lower encrypted image with four LSB planes as follows: $I_{E(u)}(i,j) = \left\lfloor \frac{I_E(i,j)}{2^4} \right\rfloor$ and $I_{E(l)}(i,j) = I_E(i,j) \text{mod } 2^4$.

Before proceeding with the embedding process, the images $I_{E(u)}$ and $I_{E(l)}$ are divided into non-overlapping blocks of sizes 2×2 and 1×2, respectively. Subsequently, in a manner akin to the image encryption process, the pixels within block B are arranged in ascending order. In the case of $I_{E(u)}$, the sorted sequence of encrypted pixels in block B are denoted as $p''_{f'(1)}$, $p''_{f'(2)}$, $p''_{f'(3)}$, and $p''_{f'(4)}$. Additionally, the PEs between each pair of pixels are represented by $e'_{(1,2)}$, $e'_{(2,3)}$, and $e'_{(3,4)}$, where $e'_{(x,y)} = p''_{f'(y)} - p''_{f'(x)}$.

The blocks are processed in a raster scan and the embedding process within a block is conducted recursively, as long as the condition $e'_{(1,4)} \leq 12$ holds. If this condition is not met, the block is designated as non-embeddable in the LM. The embedding of each bit is carried out

by processing Eq. 7.6 and 7.7 for x values ranging from 1 to 3 in $p''_{f'(x)}$.

$$p''_{f'(x')} = \begin{cases} p''_{f'(x)} + b, & \text{if } e'_{(x,x+1)} = 0 \\ p''_{f'(x)}, & \text{otherwise} \end{cases} \tag{7.6}$$

and

$$p''_{f'(x')} = p''_{f'(x')} + 1, \quad x < x' \leq n \tag{7.7}$$

where $\bar{p}_{f'(x)}$ is marked encrypted pixel and b is the data bit. Upon completion of the three embedding positions, the final assignment is $\bar{p}_{f'(4)} = p''_{f'(4)}$.

Similarly in the second stage, in the case of $I_{E(l)}$, the sorted sequence of encrypted pixels in block B are denoted as $p''_{f'(1)}$ and $p''_{f'(2)}$. Additionally, the error between the pixel is represented by $e'_{(1,2)}$. The embedding process within a block is conducted using Eq. 7.6 and 7.7, as long as the condition $e'_{(1,2)} \leq 14$ holds and finally $\bar{p}_{f'(4)}$ is assigned as $p''_{f'(4)}$.

Upon completion of the aforementioned embedding, two marked encrypted images $I_{M(u)}$ and $I_{M(l)}$ are obtained, respectively. These images are combined using Eq. (7.5) to generate the final encrypted image, denoted as I_M.

It is important to note that non-embeddable blocks, which may generate overflow/underflow, are marked in the location map. The two location maps are subsequently compressed using arithmetic encoding and embedded within the image alongside the data. In cases where the available embeddable space proves insufficient for accommodating the compressed location maps in any given image, the image is considered unsuitable for embedding and remains unaltered.

7.1.3 Data Extraction and Image Recovery

The proposed ATS-RDHEI method allows for separate data extraction and image recovery based on the availability of keys. As a result, three distinct cases can arise, which are detailed below.

When the receiver possesses only the K_D key, they divide the image into two images, $I_{M(u)}$ and $I_{M(l)}$, following the data hider's process, and subsequently split the images into blocks. In the image $I_{M(u)}$, let $q_{(1)}$, $q_{(2)}$, $q_{(3)}$, and $q_{(4)}$ represent the pixels of a block in raster scan order. The ascending order of the pixels is denoted as $q'_{f(1)}$, $q'_{f(2)}$, $q'_{f(3)}$, and $q'_{f(4)}$. The encrypted bits of data are determined from x = 3 to 1 using Eq. (7.8) and, simultaneously update $q'_{f(x)} = q'_{f(x)} - b$ and $q'_{f(x')} = q'_{f(x')} - 1$, where $x < x' \leq n$.

$$b = \begin{cases} 0, & \text{if } q'_{f(x)} = q'_{f(x-1)} \\ 1, & \text{else if} q'_{f(x)} = q'_{f(x-1)} - 1 \end{cases} \tag{7.8}$$

Upon completing the bit extraction process from a block, the bits are appended to the data

stream in reverse order. After extraction from all blocks of $I_{M(u)}$, following the same order as the hider used for embedding, data bits are similarly extracted from $I_{M(l)}$ and added to the same data stream. This combined data stream is then decrypted using K_D, resulting in the retrieval of the original data.

When the receiver possesses only the K_I image key, the receiver first removes the embedded data using Eq. (7.8) and four conditions, after slicing the image. The restoration of the images is then accomplished through a series of steps performed in reverse order: error decryption, pixel reverse shuffling, block reverse rotation, and image recombination. Initially, the pseudo-random matrices are generated in a manner similar to the encryption process, and a key is assigned to each block. By applying the key to the corresponding block, the pixels within the block are decrypted, and the original pixel arrangement is restored. Following this, the blocks are rotated back to their initial positions, and the images are recombined to obtain the original, decrypted image.

If the receiver has both the keys(K_D and K_I), he/she can extract the secret data as case 1 and can also recover the image as in case 2.

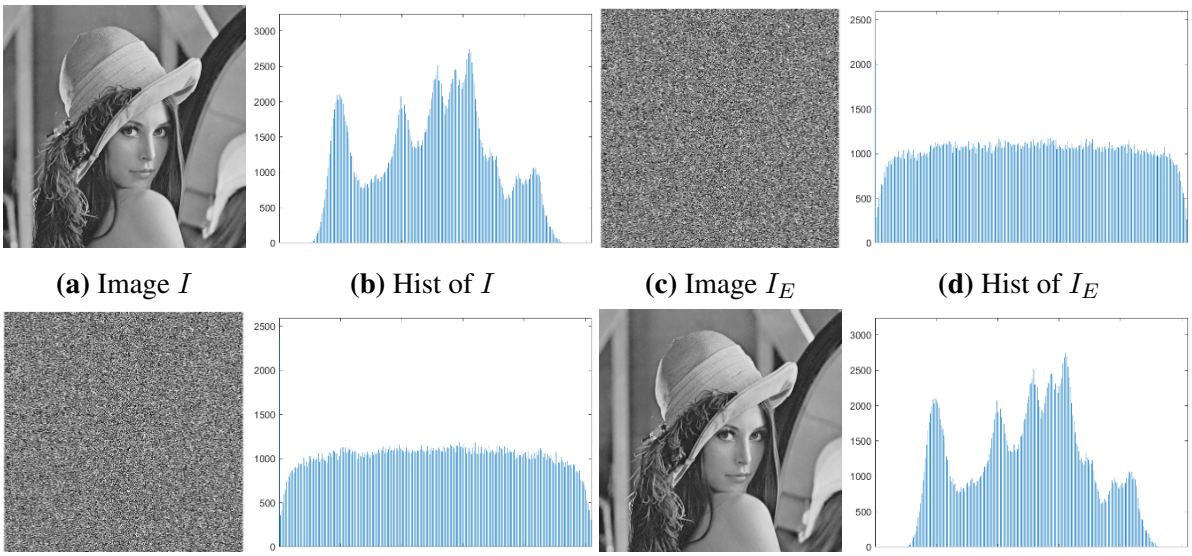
7.2 Experimental Analysis

This section unveils the experimental results and analysis of the novel ATS-RDHEI method. Through a rigorous comparison with SOTA methods, the efficacy and superiority of the proposed method is revealed. The discussion and analysis are divided into two parts, as detailed below. The first part delves into the security aspects of the method under consideration. The second part presents the performance analysis of the proposed method on four test images as depicted in Fig. 1.3 and datasets, such as BOSSbase [38] and BOWS-2 [39] and also compares with the SOTA methods.

7.2.1 Security Analysis

To assess the security of the proposed method, we have considered widely-used metrics such as histogram distribution, horizontal and vertical correlation, Shannon entropy, NPCR, UACI, PSNR, and SSIM. As a use case, the $Lena$ image shown in Fig. 7.3a has been utilized. The encrypted (I_E) and marked image (I_M) for the original image $Lena$ are displayed in Figs. 7.3c and 7.3e, respectively. Both images appear as noise-like and entirely scrambled. The histogram of I, I_E and I_M are depicted in Figs. 7.3b, 7.3d and 7.3f, respectively. With unwavering consistency across all intensities, the histograms of I_E and I_M bear witness to the exceptional perceptual security offered by our proposed method.

The experimental results for statistical metrics are given in Table 7.1. In the case of I_E and I_M, both horizontal and vertical correlations are close to 0, indicating no association between adjacent pixels. In contrast, for the original image I, the correlation is close to 1. The en-



(a) Image I **(b)** Hist of I **(c)** Image I_E **(d)** Hist of I_E

(e) Image I_M **(f)** Hist of I_M **(g)** Image I_R **(h)** Hist of I_R

Fig. 7.3: Processed image $Lena$

Table 7.1: Statistical analysis on image $Lena$

Image	Original image (I)	Encrypted image (I_E)	Marked encrypted image (I_M)
Horizontal Correlation	0.9719	0.0031	0.0041
Vertical Correlation	0.9850	0.0011	-0.0001
Diagonal Correlation	0.9593	-0.0021	0.0002
Information Entropy	7.4451	7.9993	7.9970
NPCR (I and I_E/I_M)	-	98.78	98.70
UACI (I and I_E/I_M)	-	74.12	81.11
PSNR (I and I_E/I_M)	-	9.1982	9.3520
SSIM (I and I_E/I_M)	-	0.0211	0.0142

tropy of I_E and I_M is slightly higher than that of I, signifying that the probability distribution of different pixel values in images I_E and I_M is higher than in I. Likewise, higher values of NPCR and UACI, approximately 100% and 75% respectively, indicate strong resistance against differential attacks. Low PSNR and SSIM values, around 9.2 and 0.02 respectively, demonstrate the dissimilarity of both I_E and I_M with I. Through a series of rigorous security tests, the proposed method emerges victorious, demonstrating remarkable resilience against visual, differential, and statistical security attacks.

7.2.2 Embedding Performance Evaluation and Compression with State-of-the-Art Methods

In this section, we first analyze the efficiency of the ATS-RDHEI method using various test images. Initially, four images are considered from Fig. 1.3. We examine the EC and total com-

Table 7.2: Performance of ATS-RDHEI on test images

Images	Upper image		Lower image		Pure EC ($bits$)	Pure ER (bpp)
	EC ($bits$)	LM ($bits$)	EC ($bits$)	LM ($bits$)		
$Lena$	89949	4875	28975	7395	104654	0.3992
$Baboon$	52528	3481	14629	8798	54878	0.2093
$Barbara$	75257	3529	25101	5589	91240	0.3481
$Boat$	74348	2348	22375	6798	87577	0.3341

Table 7.3: Performance of ATS-RDHEI on Two Data Sets

	Average ER (bpp)	Best ER (bpp)	Worst ER (bpp)	PSNR (dB)	SSIM
BOSSBase	0.3524	1.0911	0.0322	∞	1
BOWS-2	0.3387	0.9748	0.0309	∞	1

pressed location map for both the upper and lower images. Table 7.2 presents various metrics for each test image, including the image name, EC for the upper image, compressed location map size of the upper image, EC for the lower image, compressed location map size of the lower part, total pure EC, and pure ER. This analysis offers insight into the performance of the ATS-RDHEI method. To further substantiate the proposed method's performance, larger datasets such as BOSSbase and BOWS-2 have been employed. First, the best, average, and worst-case ERs are identified. The proposed method attains 1.2278, 0.5393, and 0.1577 bpp for BOSSbase, and 1.1186, 0.5134 and 0.1612 bpp for BOWS-2, respectively. In the next part of this section, we will discuss a comparison between the ATS-RDHEI method and the SOTA RDHEI methods to further assess its efficiency. To unleash the full potential of the proposed method, a rigorous performance analysis is conducted against SOTA methods. Each method's parameters are meticulously fine-tuned to achieve the pinnacle of ER. Since the proposed method consistently delivers lossless image reconstruction, resulting in an infinitely PSNR and SSIM of 1, it stands out from the competition. Table. 7.4 illustrates the ER comparison, which include [26], [42], [60], and [61], for four standard test images from the SIPI dataset. The

Table 7.4: ER (bpp) compassion of ATS-RDHEI with SOTA methods

Test Image	Proposed	[60]	[61]	[26]	[42]
$Lena$	0.3992	0.0651	0.0570	0.0124	0.0009
$Baboon$	0.2093	0.0453	0.0050	0.0057	0.0010
$Barbara$	0.3481	0.1130	0.0911	0.0149	0.0058
$Boat$	0.3341	0.0940	0.0989	0.0124	0.0050

proposed method achieves ERs of 0.3992, 0.2093, 0.3481, and 0.3341 bpp for $Lena$, $Baboon$, $Barbara$, and $Boat$, respectively, surpassing SOTA methods. Thus, it can be concluded that the proposed method delivers the highest ER regardless of the images.

7.3 Summary

The proposed ATS-RDHEI method offers a robust solution for secure image transmission and data hiding. Its unique combination of image encryption and RDH techniques provides enhanced security and adaptability, catering to a wide range of applications. The proposed encryption maintains the correlation of pixels in two types of blocks while increasing visual security using block rotation. Since the correlation among the pixels of each block is maintained and embedding of secret data bits is done in two stages, the ATS-RDHEI achieves enlarged EC in comparison to related existing works, as demonstrated by the experimental results. The method's capability to separately extract data and recover images based on the availability of keys further ensures the flexibility and confidentiality of the system.

CHAPTER 8

CONCLUSION AND FUTURE SCOPE

With the ever-increasing volume of digital data in modern communication, ensuring the privacy and security of such data has become a daunting challenge for data owners. To address this concern, RDHEI appears to be one of the most viable options as it enables the information owner to hide their information behind a cover image without being aware of the image's content. Various RDHEI methods have been proposed, aiming to achieve larger EC, reduced distortion, and enhanced security. These methods can be categorized as VRAE and RRBE from the encoder's perspective. VRAE methods vacate room after encryption, while RRBE methods reserve room before encryption by pre-processing the image, utilizing spatial correlation.

It was observed that traditional RDHEI methods achieve low EC or result in lossy reconstruction of the original image, in other words, exhibiting a trade-off between EC and reconstructed image quality. To mitigate this trade-off, this thesis has presented several novel RDHEI methods that significantly advance the SOTA by achieving higher EC, improved visual quality, and enhanced security. The contributions of the thesis are outlined in the subsequent subsections.

8.1 Research Contribution 1

Reversible Data Hiding in Encrypted Images: A Bidirectional Analysis with Agenda for Future Research: This dynamic area of study necessitates a comprehensive review and analysis to map out the intellectual development of the field, identify emerging trends, and highlight promising research directions. Consequently, this work presents a detailed investigation into the domain of RDHEI, an area that has experienced remarkable growth in recent years. This work not only provides a comprehensive review that showcases the advancements in the domain with a critical assessment of the employed methodologies but also explores quantitative perspectives by uncovering trends, collaborations, and impactful contributions. Thus, this review is designed to offer a complete overview of the RDHEI domain by undertaking a two-fold analysis of the

current state of RDHEI and illuminating promising avenues for future research.

This exploration delves into the theoretical frameworks, practical implications, and methodological understanding of RDHEI methods. It presents an in-depth analysis of the existing literature's strengths and weaknesses, offering insights into the various approaches categorized by their operational strategies, VRAE and RRBE. Under quantitative analysis, this study illuminates the trends, collaborative efforts, and significant contributions within the RDHEI area over the recent five years. Employing bibliometric methods, it shows the evolutionary path of RDHEI, highlights the key researchers, institutions, and nations in this domain. Additionally, it investigates co-authorship, institutional, and national collaborations to uncover patterns and networks, while also utilizing keyword co-occurrence and citation analysis to identify central themes, seminal works, and the intellectual structure of the RDHEI field.

8.2 Research Contribution 2

LDT-RDHEI: Link Chain Driven Reversible Data Hiding in Encrypted Images for High Payload: In the research, a method LCD-RDHEI presents a new way to the field of RDHEI. The main objective of the method is the innovative use of pixel neighborhood similarities, which are organized into link chains of variable sizes. This organization plays a critical role in generating substantial space within the image, which is then utilized for the effective embedding of secret data. One of the feature of this method the methodological estimation of the pixels within each link chain. Based on the length of the chain, pixels are represented either by their average or by employing two distinct quantization value. The method is designed in a way that ensure an optimal balance between maximizing embedding capacity and maintaining the integrity of the original image. Furthermore, the method extends its capacity by incorporating pixels not associated with any link chain into the data embedding process. This is achieved using the AM2PHC method, thereby enhance the overall EC of the method.

Experimental analysis demonstrate the superior EC of the LCD-RDHEI method in comparison to SOTA methods. This increased capacity is a direct outcome of the methodologically employment of link chains, which make the LCD-RDHEI one of prominent technique for RDHEI. Moreover, the adaptability and effectiveness of the LCD-RDHEI algorithm are significantly influenced by the implementation of a variability coefficient. This coefficient determines the maximum permissible intensity variation within a link chain, which can be decided by the content owner that how much distortion can be afforded. This work not only shows the methods technical superiority but also represents its potential for wide application in various digital communication contexts that demand improved data security.

8.3 Research Contribution 3

DIT-RDHEI: High-Capacity Reversible Data Hiding in Encrypted Images based on Difference Image Transfiguration: The work DIT-RDHEI introduces a novel approach to enhance the EC and ensure full reversibility of images in the context of RDHEI. This novel method based on the DIT, marks a significant advancement in addressing the limitations of current SOTA RDHEI methods. This method utilize the difference and MED predictor for pixel values prediction and subsequently derived a difference image. This process combined with the strategic cleaning of the MSB planes of the difference image to creates substantial space for secret data embedding, and generate the transfigured image. An innovative aspect of this approach is the transfiguration of the cleaning process byproduct, a weight matrix, into a compressed form using a novel location difference technique and Huffman encoding. This not only reduces the overhead significantly but also enhances but also improves the visual security of the transfigured image, ensuring that the encrypted image after the embedded with secret data, remains indistinguishable from its original.

Through comparative experiments, the proposed DIT-RDHEI method has been shown to achieve the highest average payload of 3.5784 and 3.7010 bpp for the BOSSbase and BOWS-2 datasets, respectively. This represents a marked improvement over existing techniques, establishing new benchmarks in the domain of RDHEI. The robustness of method and efficiency are also evidenced by various standard test images, where it consistently outperforms existing RDHEI methods in terms of EC. For instance, from the BOSSbase dataset, the method reached an ER of up to 6.8181 bpp for certain images, showing its potential for high-capacity data embedding in practical applications. This work contributes significantly to the advancement of RDHEI by providing a highly effective method that combines increased EC with the ability to fully recover the original image.

8.4 Research Contribution 4

MBQ-RDHEI: Bit-plane based Reversible Data Hiding in Encrypted Images using Multi-Level Blocking with Quad-Tree: The study titled MBQ-RDHEI proposed a novel methodology designed to augment the high EC for RDHEI, while maintaining the important principles of image reversibility and security. It incorporates a MED with Difference predictor to determine the prediction error, a potent combination that transforms the original image into a low-magnitude difference matrix. The bit-planes of the image are encoded by employing a novel quad-tree based bit-plane representation technique to condense their size by exploiting the sparsity. This process leads to the quad-tree generation for each bit-plane, where each node of the tree is associated with a variable-sized block of the image. Next, the nodes are mapped with binary bits and traversed in label order to generate the compressed bit-stream. Additionally, an Inter Bit-plane Redundancy Elimination strategy is proposed, which further compacts

the size of bit-plane representations by mitigating redundancy between subsequent bit-planes. By exploiting both intra and inter bit-plane correlations, the proposed method effectively creates a larger embedding space within the image, thus providing a significant advancement in the domain.

Experimental analysis demonstrate the superiority of the method over existing RDHEI techniques, particularly in achieving higher EC while maintaining image reversibility and security. For instance, the proposed method demonstrated an impressive embedding rate of 3.997 bpp on the BOSSbase dataset and 3.9161 bpp on the BOWS-2 dataset, outperforming several SOTA methods. The proposed method unequivocally demonstrates superior performance, with the average embedding rate for six test images consistently exceeding the ER achieved by the other fourteen SOTA methods. The ability of the method to achieve high data embedding rates with mitigation of redundancies, not only redefines excellence in RDHEI but also charts a path to new possibilities for preservation of secrat data.

8.5 Research Contribution 5

HCRDHEI-CS: High Capacity Reversible Data Hiding with Contiguous Space in Encrypted Images: The work introduces a RDHEI method that addresses the issue of SI security and provides a continuous space for data hiding. Similar to the DIT-RDHEI method, prediction error is generated in this research. The difference image is created using the prediction error, and block-level division is implemented, where the image is divided into 2×2 blocks. This block division facilitates the generation of a block label map, which is encoded to minimize its size, thereby maximizing the EC compared to previous works. Additionally, the introduction of a block-shifting mechanism provides a continuous space for the data hider, allowing the SI to be encrypted and the data to be embedded without any obstruction, while also minimizing the data embedding time. To synchronize the image content with the data content, an index-based data synchronization strategy is proposed, which further refines the process by enhancing security.

Experimental analysis demonstrate the superiority of the proposed method in terms of ER and achieve a remarkable increment while ensuring complete reversibility and lossless extraction of the hidden data. Notably, the method achieves an ER of 3.2684 bpp for the Lena image, demonstrating a significant improvement over SOTA methods. Furthermore, the method is comprehensively compared with existing methods, and it consistently outperforms them across various standard test images and large datasets, such as BOSSbase and BOWS-2. By achieving exceptional ER without sacrificing image reversibility or integrity, this novel method redefines standards in the RDHEI domain.

8.6 Research Contribution 6

ATS-RDHEI: Adaptive Two-Stage Reversible Data Hiding in Encrypted Images using Prediction Error Expansion: The ATS-RDHEI research introduces an innovative adaptive two-stage approach that employs prediction error expansion technique, along with PVO techniques for RDHEI. This method strikes a balance between enhancing the EC and preserving the original image's integrity and reversibility, setting new standards in the VRAE category of RDHEI domain. The ATS-RDHEI method utilizes a two-fold encryption and data embedding process that maintains spatial correlation within encrypted images, enabling high embedding rates without compromising visual quality or reversibility. It employs bit-plane division, block scrambling, pixel shuffling, and encryption to obfuscate content while preserving block correlation, ensuring high entropy and resistance against attacks. The novel two-stage embedding process manages overflow/underflow issues and expands prediction errors to embed data efficiently, achieving an impressive 0.3992 bits per pixel for test images. Experimental evaluations demonstrate the method's superiority over existing works in terms of EC while maintaining full reversibility and elevated visual security. For instance, the improved EC for the Lena image and near-perfect NPCR and UACI scores confirm its robustness against differential attacks, validating its effectiveness in enhancing data security without sacrificing image quality or accurate data recovery.

8.7 Future Scope

The field of RDHEI holds immense promise for further advancements and applications. Building upon the foundational work established in methods such as LCD-RDHEI, DIT-RDHEI, MBQ-RDHEI, and HCRDHEI-CS, future research efforts have a productive ground for exploration. Specifically, the method LCD-RDHEI is organizing pixel neighborhood similarities into link chains demonstrates a novel way of generating substantial embedding space within an image. However, an observed limitation, where the number of unused pixels increases if the variability coefficient is small, suggests an avenue for refinement. For instance, the LCD-RDHEI method could potentially benefit from exploring link chains in eight directions instead of four, potentially increasing the link chain size and capturing unused pixels, although the associated overhead would need to be addressed. Similarly, both DIT-RDHEI and MBQ-RDHEI methods shows the advancements in exploiting redundancies and employing novel compression techniques to maximize EC. Yet, a small amount of redundancy left in the LSB planes points towards an opportunity for even greater optimization. Future research could explore into this unexplored redundancy, aiming to further enhance the embedding capacity of RDHEI methods. Furthermore, these methods, including HCRDHEI-CS, label a 2×2 block for embedding without considering the potential redundancy within such small blocks. This oversight presents another research opportunity, where investigating the redundancy within 2×2 blocks could lead

to more efficient data embedding strategies, potentially increasing the overall EC.

Transitioning from these method-specific enhancements, it can also be considered the broader horizon of RDHEI research, which is equally productive. The integration of cutting-edge technologies, such as Artificial Intelligence, blockchain and advanced cryptographic techniques, to enhance the security and efficiency of RDHEI methods. Additionally, extending the application of RDHEI techniques to new and diverse domains, such as secure document storage, IoT device security, quantum-safe cryptography and beyond, could open up exciting opportunities. However, adapting RDHEI methods to these new domains will require addressing their unique challenges and constraints. Furthermore, continuous optimization and security enhancements should be a top priority, focusing on reducing computational complexity, improving embedding rates, and exploring new security paradigms to stay ahead of evolving cyber threats. Ongoing research efforts in these directions will not only advance the SOTA in RDHEI but also contribute to the broader goal of ensuring the privacy and security of digital data in modern communication systems.

LIST OF PUBLICATIONS

JOURNAL PUBLICATIONS

1. Ankur, Rajeev Kumar, and Ajay K. Sharma; Bit-Plane Based Reversible Data Hiding in Encrypted Images Using Multi-Level Blocking With Quad-Tree, **IEEE Transactions on Multimedia**, IEEE, Vol. 26, pp. 4722-4735, 2024, DOI:10.1109/TMM.2023.3325993. (IF 7.3)

2. Ankur, Rajeev Kumar, and Ajay K. Sharma; High capacity reversible data hiding with contiguous space in encrypted images, **Computers and Electrical Engineering** , Elsevier, Vol. 112, p. 109017, 2023, DOI:https://doi.org/10.1016/j.compeleceng.2023.109017. (IF 4.3)

3. Ankur, Rajeev Kumar, and Ajay K. Sharma; Link Chain Driven Reversible Data Hiding in Encrypted Images for High Payload, **Signal, Image and Video Processing**. Springer, DOI: 10.1007/s11760-024-03275-1 (Accepted). (IF 2.3)

4. Ankur, Rajeev Kumar, and Ajay K. Sharma; High-Capacity Reversible Data Hiding in Encrypted Images based on Difference Image Transfiguration, **Information Sciences**, Elsevier. (Under Review) (IF 8.1)

5. Ankur, Rajeev Kumar, and Ajay K. Sharma; Reversible data hiding with Sparsity-aware Image Encryption and Bit Replacement, **IEEE Transactions on Multimedia**, IEEE. (Under Review) (IF 7.3)

6. Ankur, Rajeev Kumar, and Ajay K. Sharma; Reversible Data Hiding in Encrypted Image Using Optimal Size Block and Bit-stream Encoding. **Journal of Information Security and Applications**, Elsevier. (Under Review) (IF 5.6)

7. Ankur, Rajeev Kumar, and Ajay K. Sharma; Leveraging rANS for Synchronized High Capacity Reversible data hiding in Encrypted Image, **IEEE Transactions on Circuits and Systems for Video Technology**, IEEE. (Under Review) (IF 8.4)

8. Ankur, Rajeev Kumar, and Ajay K. Sharma; Reversible Data Hiding in Encrypted Images: A Bidirectional Analysis with Agenda for Future Research, **Archives of Computational Methods in Engineering**, Springer. (Under Review) (IF 9.7)

CONFERENCE PUBLICATIONS

1. Ankur, Rajeev Kumar, and Ajay K. Sharma; Adaptive Two-Stage Reversible data hiding in Encrypted Images Using Prediction Error Expansion. Third International Conference on Secure Cyber Computing and Communication (ICSCCC), 2023. IEEE. Venue of the conference-Dr. B. R. Ambedkar National Institute of Technology Jalandhar, Punjab, India. DOI: 10.1109/ICSCCC58608.2023.10176366. (Published July 14, 2023).

2. Ankur, Rajeev Kumar, and Ajay K. Sharma; A Systematic Review of RDHEI with Consistent Experimental Evaluation. International Conference on Recent Innovations in Computing Vol-2 (ICRIC-2023). Springer. Venue of the conference- ELTE, Hungary. (Accepted and Presented December 21, 2023).

BIBLIOGRAPHY

[1] M. Dachyar, T. Y. M. Zagloel, and L. R. Saragih, "Knowledge growth and development: internet of things (iot) research, 2006–2018," *Heliyon*, vol. 5, no. 8, 2019.

[2] S. Akter, K. Michael, M. R. Uddin, G. McCarthy, and M. Rahman, "Transforming business using digital innovations: The application of ai, blockchain, cloud and data analytics," *Annals of Operations Research*, pp. 1–33, 2022.

[3] R. Bansal, A. J. Obaid, A. Gupta, R. Singh, and S. Pramanik, "Impact of big data on digital transformation in 5g era," in *Journal of Physics: Conference Series*, vol. 1963, no. 1. IOP Publishing, 2021, p. 012170.

[4] S. S. Muttagi, S. D. Biradar, and D. Kushnure, "5g: A digital society," in *Electrical, Electronics, Signals, Communication and Optimization (EESCO), 2015 International Conference on*, 2015, pp. 1–5.

[5] S. Kemp. (2024) Key internet statistics in 2024. [Online]. Available: https://www.broadbandsearch.net/blog/internet-statistics

[6] K. WONG. (2023) 16 visual content marketing statistics to know for 2024 [infographic]. [Online]. Available: https://venngage.com/blog/visual-content-marketing-statistics/

[7] B. Dean. (2024) X (twitter) statistics: How many people use x? (2024). [Online]. Available: https://backlinko.com/twitter-users

[8] R. Shewale. (2024) Twitter statistics for 2024 — (facts after "x" rebranding). [Online]. Available: https://www.demandsage.com/twitter-statistics/

[9] M. Iqbal. (2024) Instagram revenue and usage statistics (2024). [Online]. Available: https://www.businessofapps.com/data/instagram-statistics/

[10] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[11] C.-L. Hsu and J. C.-C. Lin, "Factors affecting the adoption of cloud services in enterprises," *Information Systems and e-Business Management*, vol. 14, pp. 791–822, 2016.

[12] W. Y. Chang, H. Abu-Amara, and J. F. Sanford, *Transforming enterprise cloud services*. Springer Science & Business Media, 2010.

[13] V. Chang and M. Ramachandran, "Towards achieving data security with the cloud computing adoption framework," *IEEE Transactions on services computing*, vol. 9, no. 1, pp. 138–151, 2015.

[14] M. Wu and B. Liu, "Watermarking for image authentication," in *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No. 98CB36269)*, vol. 2. IEEE, 1998, pp. 437–441.

[15] G. Langelaar, I. Setyawan, and R. Lagendijk, "Watermarking digital image and video data. a state-of-the-art overview," *IEEE Signal Processing Magazine*, vol. 17, no. 5, pp. 20–46, 2000.

[16] P. Hu, D. Peng, Z. Yi, and Y. Xiang, "Robust time-spread echo watermarking using characteristics of host signals," *Electronics Letters*, vol. 52, no. 1, pp. 5–6, 2016.

[17] D. Kersten, "Predictability and redundancy of natural images," *JOSA A*, vol. 4, no. 12, pp. 2395–2400, 1987.

[18] R. Kumar, S. Chand, and S. Singh, "An optimal high capacity reversible data hiding scheme using move to front coding for lzw codes," *Multimedia Tools and Applications*, vol. 78, no. 16, pp. 22 977–23 001, 2019.

[19] M. Sahu, N. Padhy, S. S. Gantayat, and A. K. Sahu, "Local binary pattern-based reversible data hiding," *CAAI Transactions on Intelligence Technology*, vol. 7, no. 4, pp. 695–709, 2022.

[20] Z. Yin, A. Abel, X. Zhang, and B. Luo, "Reversible data hiding in encrypted image based on block histogram shifting," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 2129–2133.

[21] F. Ren, Z. Wu, Y. Xue, and Y. Hao, "Reversible data hiding in encrypted image based on bit-plane redundancy of prediction error," *Mathematics*, vol. 11, no. 11, p. 2537, 2023.

[22] Y. Zhaoxia, W. Huabin, Z. Haifeng, L. Bin, and Z. Xinpeng, "Complete separable reversible data hiding in encrypted image," in *Cloud Computing and Security: First International Conference, ICCCS 2015, Nanjing, China, August 13-15, 2015. Revised Selected Papers 1*. Springer, 2015, pp. 101–110.

[23] W. Bender, W. Butera, D. Gruhl, R. Hwang, F. J. Paiz, and S. Pogreb, "Applications for data hiding," *IBM Systems Journal*, vol. 39, no. 3.4, pp. 547–568, 2000.

[24] J. Fridrich, "Applications of data hiding in digital images," in *ISSPA'99. Proceedings of the Fifth International Symposium on Signal Processing and its Applications (IEEE Cat. No. 99EX359)*, vol. 1. IEEE, 1999, pp. 9–vol.

[25] W. Puech, M. Chaumont, and O. Strauss, "A reversible data hiding method for encrypted images," in *Security, Forensics, Steganography, and Watermarking of Multimedia Contents X*, vol. 6819. SPIE, 2008, pp. 534–542.

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

[27] J. Zhou, W. Sun, L. Dong, X. Liu, O. C. Au, and Y. Y. Tang, "Secure reversible image data hiding over encrypted domain via key modulation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 3, pp. 441–452, 2015.

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

[29] H. Gao, X. Zhang, and T. Gao, "Hierarchical reversible data hiding in encrypted images based on multiple linear regressions and multiple bits prediction," *Multimedia Tools and Applications*, pp. 1–27, 2023.

[30] P. Puteaux, S. Ong, K. Wong, and W. Puech, "A survey of reversible data hiding in encrypted images–the first 12 years," *Journal of Visual Communication and Image Representation*, vol. 77, p. 103085, 2021.

[31] R. Abbasi, A. K. Bashir, A. O. Almagrabi, M. B. B. Heyat, and G. Yuan, "Efficient lossless based secure communication in 6g internet-of-things environments," *Sustainable Energy Technologies and Assessments*, vol. 57, p. 103218, 2023.

[32] Y. Wang, G. Xiong, and W. He, "High-capacity reversible data hiding in encrypted images based on pixel-value-ordering and histogram shifting," *Expert Systems with Applications*, vol. 211, p. 118600, 2023.

[33] J.-H. Horng, C.-C. Chang, G.-L. Li, W.-K. Lee, S. O. Hwang *et al.*, "Blockchain-based reversible data hiding for securing medical images," *Journal of Healthcare Engineering*, vol. 2021, 2021.

[34] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[35] T. Chu, W. Ranson, and M. A. Sutton, "Applications of digital-image-correlation techniques to experimental mechanics," *Experimental mechanics*, vol. 25, no. 3, pp. 232–244, 1985.

[36] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[37] Y. Wu, J. P. Noonan, S. Agaian *et al.*, "Npcr and uaci randomness tests for image encryption," *Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT)*, vol. 1, no. 2, pp. 31–38, 2011.

[38] P. Bas, T. Filler, and T. Pevnỳ, "" break our steganographic system": the ins and outs of organizing boss," in *International Workshop on Information Hiding*. Springer, 2011, pp. 59–70.

[39] P. Bas and T. Furon, "Image database of bows-2," *Accessed: Jun*, vol. 20, pp. 2016–2017, 2017.

[40] G. Schaefer and M. Stich, "Ucid: An uncompressed color image database," in *Storage and Retrieval Methods and Applications for Multimedia 2004*, vol. 5307. SPIE, 2003, pp. 472–480.

[41] A. G. Weber, "The usc-sipi image database: Version 5," *http://sipi. usc. edu/database/*, 2006.

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

[43] X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 826–832, 2011.

[44] X. Wu and W. Sun, "High-capacity reversible data hiding in encrypted images by prediction error," *Signal Processing*, vol. 104, pp. 387–400, 2014.

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

[46] K.-Y. Song, S. Kim *et al.*, "A modified reversible data hiding in encrypted image using enhanced measurement functions," in *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2016, pp. 869–872.

[47] Z. Qian, X. Zhang, and G. Feng, "Reversible data hiding in encrypted images based on progressive recovery," *IEEE Signal Processing Letters*, vol. 23, no. 11, pp. 1672–1676, 2016.

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

[49] S. Agrawal and M. Kumar, "Mean value based reversible data hiding in encrypted images," *Optik*, vol. 130, pp. 922–934, 2017.

[50] Z.-L. Liu and C.-M. Pun, "Reversible data-hiding in encrypted images by redundant space transfer," *Information Sciences*, vol. 433, pp. 188–203, 2018.

[51] C. Qin, X. Qian, W. Hong, and X. Zhang, "An efficient coding scheme for reversible data hiding in encrypted image with redundancy transfer," *Information Sciences*, vol. 487, pp. 176–192, 2019.

[52] H. Ren, S. Niu, and X. Wang, "Reversible data hiding in encrypted images using pob number system," *IEEE Access*, vol. 7, pp. 149 527–149 541, 2019.

[53] Z. Tang, S. Xu, H. Yao, C. Qin, and X. Zhang, "Reversible data hiding with differential compression in encrypted image," *Multimedia Tools and Applications*, vol. 78, pp. 9691–9715, 2019.

[54] Y. Fu, P. Kong, H. Yao, Z. Tang, and C. Qin, "Effective reversible data hiding in encrypted image with adaptive encoding strategy," *Information Sciences*, vol. 494, pp. 21–36, 2019.

[55] R. Bhardwaj and A. Aggarwal, "An improved block based joint reversible data hiding in encrypted images by symmetric cryptosystem," *Pattern Recognition Letters*, vol. 139, pp. 60–68, 2020.

[56] A. Malik, P. He, H. Wang, A. N. Khan, S. Pirasteh, and S. M. Abdullahi, "High-capacity reversible data hiding in encrypted images using multi-layer embedding," *IEEE Access*, vol. 8, pp. 148 997–149 010, 2020.

[57] K.-M. Chen, "High capacity reversible data hiding based on the compression of pixel differences," *Mathematics*, vol. 8, no. 9, p. 1435, 2020.

[58] C. Yu, X. Zhang, G. Li, S. Zhan, and Z. Tang, "Reversible data hiding with adaptive difference recovery for encrypted images," *Information Sciences*, vol. 584, pp. 89–110, 2022.

[59] K. Gao, J.-H. Horng, and C.-C. Chang, "High-capacity reversible data hiding in encrypted images based on adaptive block encoding," *Journal of Visual Communication and Image Representation*, vol. 84, p. 103481, 2022.

[60] C.-H. Yang, C.-Y. Weng, and J.-Y. Chen, "High-fidelity reversible data hiding in encrypted image based on difference-preserving encryption," *Soft Computing*, vol. 26, no. 4, pp. 1727–1742, 2022.

[61] R. Anushiadevi and R. Amirtharajan, "Separable reversible data hiding in an encrypted image using the adjacency pixel difference histogram," *Journal of Information Security and Applications*, vol. 72, p. 103407, 2023.

[62] A. K. Rai, H. Om, S. Chand, and S. Agarwal, "Reversible data hiding in encrypted image using two-pass pixel value ordering," *Journal of Information Security and Applications*, vol. 76, p. 103545, 2023.

[63] Z. Han and C. Guohua, "Reversible data hiding in encrypted image with local-correlation-based classification and adaptive encoding strategy," *Signal Processing*, vol. 205, p. 108847, 2023.

[64] Y. Wang, Z. Cai, and W. He, "High capacity reversible data hiding in encrypted image based on intra-block lossless compression," *IEEE Transactions on Multimedia*, vol. 23, pp. 1466–1473, 2021.

[65] X. Zhang, J. Long, Z. Wang, and H. Cheng, "Lossless and reversible data hiding in encrypted images with public-key cryptography," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 9, pp. 1622–1631, 2015.

[66] X. Cao, L. Du, X. Wei, D. Meng, and X. Guo, "High capacity reversible data hiding in encrypted images by patch-level sparse representation," *IEEE Transactions on Cybernetics*, vol. 46, no. 5, pp. 1132–1143, 2015.

[67] S. Yi and Y. Zhou, "Binary-block embedding for reversible data hiding in encrypted images," *Signal Processing*, vol. 133, pp. 40–51, 2017.

[68] S. Borse and B. Jadhav, "Data hiding in encrypted images using transpose based reserving room before encryption and discrete wavelet transform," in *2017 International Conference on Intelligent Sustainable Systems (ICISS)*, 2017, pp. 211–217.

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

[70] P. Puteaux and W. Puech, "An efficient msb prediction-based method for high-capacity reversible data hiding in encrypted images," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1670–1681, 2018.

[71] Y. Puyang, Z. Yin, and Z. Qian, "Reversible data hiding in encrypted images with two-msb prediction," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2018, pp. 1–7.

[72] A. Mohammadi, M. Nakhkash, and M. A. Akhaee, "A high-capacity reversible data hiding in encrypted images employing local difference predictor," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 8, pp. 2366–2376, 2020.

[73] P.-F. Shiu, W.-L. Tai, J.-K. Jan, C.-C. Chang, and C.-C. Lin, "An interpolative ambtc-based high-payload rdh scheme for encrypted images," *Signal Processing: Image Communication*, vol. 74, pp. 64–77, 2019.

[74] C. Yu, X. Zhang, X. Zhang, G. Li, and Z. Tang, "Reversible data hiding with hierarchical embedding for encrypted images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 2, pp. 451–466, 2022.

[75] Z. Hua, X. Liu, Y. Zheng, S. Yi, and Y. Zhang, "Reversible data hiding over encrypted images via preprocessing-free matrix secret sharing," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[76] Z. Yin, Y. Xiang, and X. Zhang, "Reversible data hiding in encrypted images based on multi-msb prediction and huffman coding," *IEEE Transactions on Multimedia*, vol. 22, no. 4, pp. 874–884, 2020.

[77] G. Gao, L. Zhang, Y. Lin, S. Tong, and C. Yuan, "High-performance reversible data hiding in encrypted images with adaptive huffman code," *Digital Signal Processing*, vol. 133, p. 103870, 2023.

[78] P. Puteaux and W. Puech, "A recursive reversible data hiding in encrypted images method with a very high payload," *IEEE Transactions on Multimedia*, vol. 23, pp. 636–650, 2021.

[79] F. Li, H. Zhu, and C. Qin, "Reversible data hiding in encrypted images using median prediction and bit plane cycling-xor," *Multimedia Tools and Applications*, vol. 82, no. 4, pp. 6013–6032, 2023.

[80] L. Sui, H. Li, J. Liu, Z. Xiao, and A. Tian, "Reversible data hiding in encrypted images based on hybrid prediction and huffman coding," *Symmetry*, vol. 15, no. 6, p. 1222, 2023.

[81] Y. Liu, G. Feng, C. Qin, H. Lu, and C.-C. Chang, "High-capacity reversible data hiding in encrypted images based on hierarchical quad-tree coding and multi-msb prediction," *Electronics*, vol. 10, no. 6, p. 664, 2021.

[82] P. Ping, J. Huo, and B. Guo, "Novel asymmetric cnn-based and adaptive mean predictors for reversible data hiding in encrypted images," *Expert Systems with Applications*, p. 123270, 2024.

[83] S. Yi and Y. Zhou, "Separable and reversible data hiding in encrypted images using parametric binary tree labeling," *IEEE Transactions on Multimedia*, vol. 21, no. 1, pp. 51–64, 2019.

[84] K. Chen and C.-C. Chang, "High-capacity reversible data hiding in encrypted images based on extended run-length coding and block-based msb plane rearrangement," *Journal of Visual Communication and Image Representation*, vol. 58, pp. 334–344, 2019.

[85] Y. Wu, Y. Xiang, Y. Guo, J. Tang, and Z. Yin, "An improved reversible data hiding in encrypted images using parametric binary tree labeling," *IEEE Transactions on Multimedia*, vol. 22, no. 8, pp. 1929–1938, 2020.

[86] Y. Wang and P. Li, "Secure reversible data hiding in encrypted images based on adaptive Huffman coding and pixel rearrangement," *Journal of Electronic Imaging*, vol. 31, no. 6, p. 063052, 2022.

[87] S. Xu, J.-H. Horng, C.-C. Chang, and C.-C. Chang, "Reversible data hiding with hierarchical block variable length coding for cloud security," *IEEE Transactions on Dependable and Secure Computing*, 2022.

[88] Y. Yao, K. Wang, Q. Chang, and S. Weng, "Reversible data hiding in encrypted images using global compression of zero-valued high bit-planes and block rearrangement," *IEEE Transactions on Multimedia*, 2023.

[89] Z. Fu, X. Chai, Z. Tang, X. He, Z. Gan, and G. Cao, "Adaptive embedding combining lbe and ibbe for high-capacity reversible data hiding in encrypted images," *Signal Processing*, vol. 216, p. 109299, 2024.

[90] Z. Yin, Y. Peng, and Y. Xiang, "Reversible data hiding in encrypted images based on pixel prediction and bit-plane compression," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 2, pp. 992–1002, 2022.

[91] M. Lema and O. Mitchell, "Absolute moment block truncation coding and its application to color images," *IEEE Transactions on Communications*, vol. 32, no. 10, pp. 1148–1157, 1984.

[92] W. F. de la Vega and M. Karpinski, "On the approximation hardness of dense tsp and other path problems," *Information Processing Letters*, vol. 70, no. 2, pp. 53–55, 1999.

[93] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[94] V. Rijmen and J. Daemen, "Advanced encryption standard," *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, vol. 19, p. 22, 2001.

[95] M. Liu, K. Wang, and T. Gao, "High-capacity reversible data hiding in encrypted images based on adaptive arithmetic coding and static huffman coding," *Cluster Computing*, vol. 26, no. 6, pp. 3627–3645, 2023.

[96] B. Qin, H. Zheng, and D. Xiao, "High-capacity data hiding in encrypted images based on compressive sensing and intra-block difference coding," *Applied Intelligence*, vol. 53, no. 21, pp. 26 240–26 254, 2023.

[97] G. Gao, S. Tong, Z. Xia, and Y. Shi, "A universal reversible data hiding method in encrypted image based on msb prediction and error embedding," *IEEE Transactions on Cloud Computing*, 2022.

[98] S. Weng, C. Zhang, T. Zhang, and K. Chen, "High capacity reversible data hiding in encrypted images using sibrw and gcc," *Journal of Visual Communication and Image Representation*, vol. 75, p. 102932, 2021.

[99] D. Xiao, F. Li, M. Wang, and H. Zheng, "A novel high-capacity data hiding in encrypted images based on compressive sensing progressive recovery," *IEEE Signal Processing Letters*, vol. 27, pp. 296–300, 2020.

[100] F. Chen, Y. Yuan, H. He, M. Tian, and H.-M. Tai, "Multi-msb compression based reversible data hiding scheme in encrypted images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 3, pp. 905–916, 2021.

[101] T. Markas and J. Reif, "Quad tree structures for image compression applications," *Information Processing Management*, vol. 28, no. 6, pp. 707–721, 1992, special Issue: Data compression for images and texts.

[102] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The loco-i lossless image compression algorithm: Principles and standardization into jpeg-ls," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309–1324, 2000.

[103] C. A. Hoare, "Quicksort," *The Computer Journal*, vol. 5, no. 1, pp. 10–16, 1962.