Implementation of Zerowatermarking using Various Techniques

A Report Submitted in Partial Fulfilment of the Requirements for the

SN Bose Internship Program, 2024

Submitted by

Rajiv Kumar Sah	2112176
Jagrit Kr. Jain	2112178
Sourav Jyoti Nath	2112179
Spandan Priyam Chetia	2112188

Under the guidance of

Dr. Dalton Meitei Thounaojam
Assistant Professor
Department of Computer Science & Engineering
National Institute of Technology Silchar

Department of Computer Science & Engineering
NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR
Assam

June-July, 2024

DECLARATION

"Implementation of Zerowatermarking using Various Techniques"

We declare that the art on display is mostly comprised of our own ideas and work, expressed in our own words. Where other people's thoughts or words were used, we properly cited and noted them in the reference materials. We have followed all academic honesty and integrity principles.

Rajiv Kumar Sah	2112176	
Jagrit Kr. Jain	2112178	
Sourav Jyoti Nath	2112179	
Spandan Priyam Chetia	2112188	

Department of Computer Science and Engineering
National Institute of Technology Silchar, Assam

ACKNOWLEDGEMENT

We would like to thank our supervisor, Dr. Dalton Meitei Thounaojam, CSE, NIT Silchar, for his invaluable direction, encouragement, and assistance during this project. His helpful suggestions for this entire effort and cooperation are gratefully thanked, as they enabled us to conduct extensive investigation and learn about many new subjects.

Our sincere gratitude to Pavani Meesala for her unwavering support and patience, without which this project would not have been completed properly and on schedule. This project would not have been feasible without her, from providing the data sets to answering any doubts we had to helping us with ideas whenever we were stuck.

Rajiv Kumar Sah	2112176
Jagrit Kr. Jain	2112178
Sourav Jyoti Nath	2112179
Spandan Priyam Chetia	2112188

Department of Computer Science and Engineering
National Institute of Technology Silchar, Assam

ABSTRACT

This report presents the development and implementation of a robust zero-watermarking technique. The work is divided into two phases: replicating established watermarking methodologies and developing a novel approach for zero-watermarking.

In the first phase, advanced watermarking techniques involving feature extraction using DWT (Discrete Wavelet Transform), LWT (Lifting Wavelet Transform), SVD (Singular Value Decomposition), and chaotic sequences were implemented. The process included decomposing the color image into R, G, and B components, applying DWT/LWT, performing SVD, and extracting singular values to construct a feature matrix, which was then transformed into a binary sequence for watermark embedding.

In the second phase, a new zero-watermarking method was developed. The watermark image was encrypted using Elliptic Curve Cryptography (ECC) to ensure security. The host image was processed to extract features and generate a perceptual hash. The encrypted watermark and the perceptual hash image were combined to generate the zero-watermark. For extraction, the attacked host image underwent similar processing to retrieve and decrypt the original watermark.

The proposed method was evaluated using metrics such as Normalized Correlation (NC), Bit Error Rate (BER), Structural Similarity Index Measure (SSIM), and Peak Signal-to-Noise Ratio (PSNR), demonstrating high robustness and reliability against various attacks.

List of Algorithms

List of Figures

List of Tables

Contents

Chapter 1

An Efficient and Robust Zero-Bit Watermarking Technique for Biometric Image Protection

This paper is about a robust watermarking algorithm for biometric images. The key focus of the study is ensuring data security and maintaining the quality of data through a zero-bit watermarking methodology. The algorithm leverages Discrete Wavelet Transform (DWT) and Singular Value Decomposition (SVD) to generate a secure master share of the biometric data. This approach is designed to withstand various image processing attacks such as contrast enhancement, Gaussian and median filtering, histogram equalization, and JPEG compression, without causing distortion to the original iris image.

1.1 Introduction

In the field of biometric security, protecting sensitive data such as iris images is crucial. The paper introduces an innovative watermarking algorithm aimed at enhancing the security of biometric data. By utilizing a zero-bit watermarking methodology, the algorithm ensures that the integrity and quality of the original biometric data are preserved. The key components of this algorithm are Discrete Wavelet Transform (DWT) and Singular Value Decomposition (SVD), which together create a robust and secure master share of the biometric data. This advanced approach has been tested against a variety of image processing attacks, including contrast enhancement, Gaussian

and median filtering, histogram equalization, and JPEG compression. The results demonstrate that the watermarking algorithm is highly effective in protecting the biometric data without causing any noticeable distortion to the original iris images.

1.1.1 Watermarking and Zero Watermarking

What is Watermarking?

Watermarking is a process used to embed data (the watermark) into a digital signal (such as an image, video, or audio) in a way that is imperceptible to the human senses but can be detected or extracted by a computer. The primary purposes of watermarking are to protect intellectual property, ensure data integrity, and prevent unauthorized copying or tampering. Watermarking techniques can be classified into two categories:

1. **Visible Watermarking**: Visible Watermarking: The watermark is visible to the viewer, such as a logo or text overlay on an image.
2. **Invisible Watermarking:**: The watermark is hidden within the digital content and can only be detected or extracted using specific algorithms.

What is Zero Watermarking?

Zero watermarking is a technique where the watermark is not embedded directly into the original content. Instead, a unique feature (such as a hash or master share) is extracted from the original content and stored separately. During verification, the extracted feature is compared with the stored feature to confirm authenticity and integrity.

Key advantages of zero watermarking include:

- No modification of the original content: The original content remains unaltered as the watermark is not embedded directly into it.
- High resistance to various attacks: Since the watermark is not embedded in the content, it provides high resistance to attacks and modifications.

1.1.2 The Discrete Wavelet Transform (DWT)

The Discrete Wavelet Transform (DWT) is a powerful tool for signal processing and image analysis. DWT works by decomposing an image into different

frequency components, allowing the analysis of both spatial and frequency information simultaneously. This decomposition is achieved by breaking down an image into sub-bands at multiple resolution levels. The sub-bands typically include low-low (LL), high-low (HL), low-high (LH), and high-high (HH) components, with LL representing the approximation coefficients and the others representing the detailed coefficients. DWT is widely used in applications such as image compression, noise reduction, and feature extraction due to its efficiency in capturing localized frequency information. The ability of DWT to focus on specific frequency components makes it an excellent choice for tasks requiring detailed analysis of signal variations across different scales.

1.1.3 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a fundamental linear algebra technique used for matrix factorization. SVD decomposes a given matrix into three component matrices: U, Σ, and V^T. Here, U and V are orthogonal matrices, and Σ is a diagonal matrix containing the singular values of the original matrix. The singular values are a set of scalars that provide insight into the intrinsic properties of the matrix. SVD is widely used in various fields such as signal processing, statistics, and machine learning due to its ability to reveal important structural information. In image processing, SVD helps in tasks such as image compression, noise reduction, and watermarking by enabling the analysis of image properties in a transformed domain that highlights significant features and reduces redundancy. The robustness of SVD to noise and distortions makes it a valuable tool for applications requiring stable and reliable matrix representations.

1.1.4 Arnold Cat Map

The Arnold Cat Map is a method used for image encryption, scrambling image pixels in a seemingly random manner while preserving pixel count and color values. This transformation is reversible, allowing the original image to be restored by applying the inverse transformation the same number of times. The Arnold Cat Map is useful for lossless encryption, ensuring the confidentiality and integrity of transmitted data. For instance, it can encrypt a master share 'k' containing patient details, which can then be decrypted at the receiver's end to retrieve the needed information accurately.

1.2 Workflow of the Algorithm

The algorithm consists of two main parts: the zero-watermarking construction and the watermark image extraction.

1.2.1 The Zero-Watermarking Construction

The watermarking algorithm involves two main phases: the construction process and the extraction process. In the construction process, the original biometric iris image is loaded and decomposed into four sub-bands using Discrete Wavelet Transform (DWT), focusing on the LL sub-band. This sub-band is then divided into non-overlapping blocks, and Singular Value Decomposition (SVD) is applied to each block to obtain singular values. A unique identification code is encrypted using the Arnold Cat Map to create the master share, which serves as the watermark. This watermark is embedded by comparing the first singular values of the blocks, generating a matrix 'tem'. The encrypted watermark is then stored securely.

1.2.2 The Zero-Watermarking Extraction

In the extraction process, the watermarked image is loaded and decomposed into sub-bands using DWT, focusing again on the LL sub-band. The LL sub-band is divided into non-overlapping blocks, and SVD is applied to obtain singular values. The matrix 'tem' is generated by comparing the first singular values of the blocks. The stored master share is decrypted using the Arnold Cat Map, and an XOR operation between the 'tem' matrix and the decrypted master share extracts the encrypted watermark. The extracted watermark is then decrypted and validated against the original watermark.

1.3 Algorithm

1.3.1 Embedding Process

Algorithm 1 Zero Watermark Embedding Process.

Require: $irisImage$ (input image), wm_Image (watermark image)
Ensure: $master\_share\_ency$ (encrypted master share)

```
function ZERO-BIT-WATERMARK_EMB(irisImage, wm_Image)
    I ← image_Resize(irisImage)                          ▷ resize to 128 × 128
    I_gray ← color2gray(I)
    [LL, LH, HL, HH] ← DWT(I_gray)
    BB ← block_division(LL)                              ▷ non-overlapping 4 × 4 blocks
    for each block in BB do
        [U, S, V] ← svd(block)                           ▷ Computing SVD
        tem ← diagonal(S)
    end for
    z ← generate_binary_watermark_bits(tem)
    wm_bin ← image_resize(imbinarize(im2gray(wm_Image)), [16, 16])
    X ← XOR(z, wm_bin)
    master_share ← X
    iterations ← 1
    while iterations ≤ 10 do
        master_share_ency ← Arnold_Cat_Map(master_share)
        iterations ← iterations + 1
    end while
    return master_share_ency
end function
```

1.3.2 Extraction Process

Algorithm 2 Zero Watermark Extraction Process.

Require: $aImage$ (attacked image), $master\_share\_encrypted$ (encrypted master share), wm_Image (original watermark image)
Ensure: $recovered\_watermark$ (recovered watermark)
function ZERO-BIT-WATERMARK_EXTR($aImage$, $master\_share\_encrypted$, wm_Image)
$I \leftarrow image\_Resize(aImage)$
$I\_gray \leftarrow color2gray(I)$
$[LL, LH, HL, HH] \leftarrow DWT(I\_gray)$
$BB \leftarrow block\_division(LL)$
for each block in BB **do**
$[U, S, V] \leftarrow \text{svd}(block)$
$tem \leftarrow diagonal(S)$
end for
$z \leftarrow generate\_binary\_watermark\_bits(tem)$
$original\_watermark\_bin \leftarrow image\_resize(imbinarize(im2gray(original\_watermark))),$
$itr \leftarrow 1$
while $itr \leq 500$ **do**
$master\_share\_decrypted \leftarrow Arnold\_Cat\_Map(master\_share\_encrypted)$
$itr \leftarrow itr + 1$
end while
$master\_share \leftarrow master\_share\_decrypted$
$recovered\_watermark \leftarrow XOR(master\_share, z)$
return $recovered\_watermark$
end function

1.4 Performance Analysis

The images and data presented below showcase the processes of embedding and extraction for the proposed zero-watermarking technique. This evaluation includes results from multiple experiments aimed at testing the robustness of the technique against various types of attacks.

Figure 1.1: Image 1

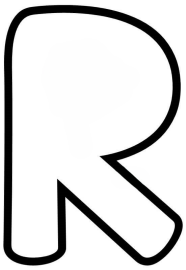


Figure 1.2: Image 2

1.4.1 Embedding Process

The embedding process involves integrating the encrypted matrix of watermark image into the featured matrix of host image. This technique ensures that the watermark is not easily visible but can still be detected using specific algorithms. Below are the results showing how the watermark integrates into the host image.

Figure 1.3: Master Share

1.4.2 Extraction Process

This section contains the details of the extraction process, including the type of attack on the original image, the attacked image, and the extracted watermark. Additionally, it provides NC and BER values for different attacks.

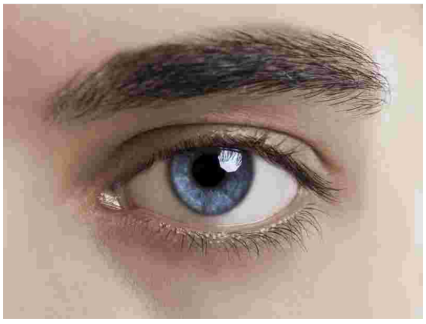


Fig. 4. JPEG Compression

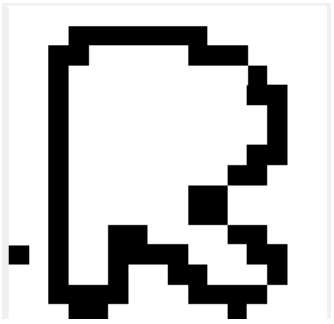


Fig. 5. Extracted Watermark 1

Fig. 6. Median Filter

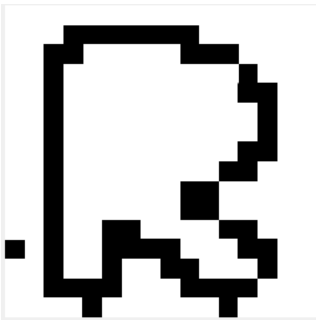


Fig. 7. Extracted Watermark 2

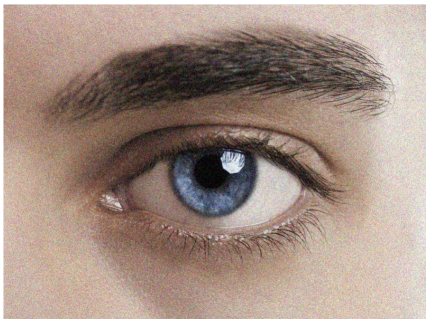


Fig. 8. Gaussian Filter

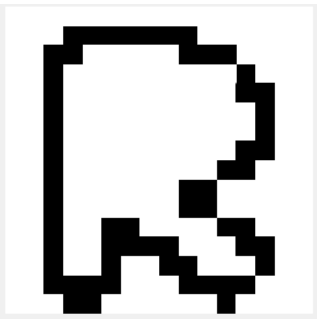


Fig. 9. Extracted Watermark 3

Fig. 10. Sharpening

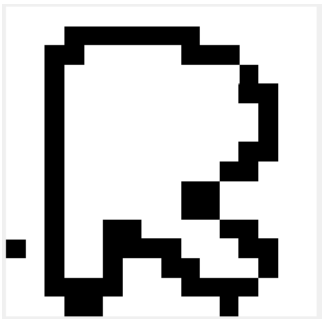


Fig. 11. Extracted Watermark 4

Fig. 12. Histogram Equalization

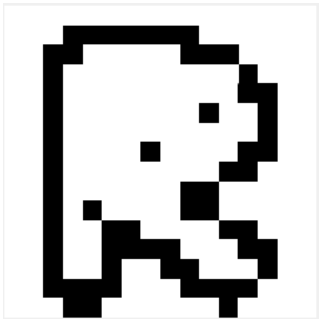


Fig. 13. Extracted Watermark 5

Fig. 14. Contrast Enhancement

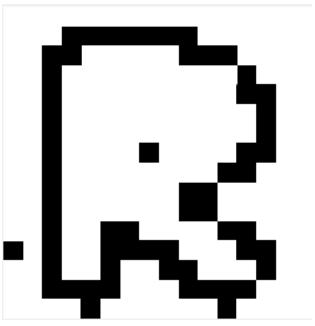


Fig. 15. Extracted Watermark 6

Fig. 16. Blurring

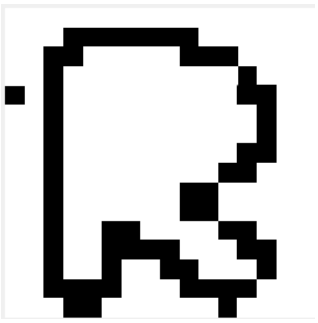


Fig. 17. Extracted Watermark 7

Fig. 18. No Attack

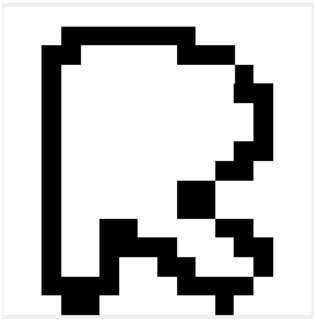


Fig. 19. Extracted Watermark 8

Table 1.1: NC and BER values for different attacks

Attack Type	NC Value	BER Value
JPEG Compression	0.99487	0.0039062
Median Filter	0.99487	0.0078125
Gaussian Filter	1	0
Sharpening	0.99487	0.0039062
Histogram Equalization	0.98462	0.011719
Contrast Enhancement	0.98974	0.011719
Blurring	0.99487	0.0039062
No Attack	1	0

Chapter 2

A zero-watermarking for color image based on LWT-SVD and chaotic system

The paper presents a robust zero-watermarking algorithm for color images, focusing on data security and image quality preservation. Utilizing Lifting Wavelet Transform (LWT) and Singular Value Decomposition (SVD), the algorithm extracts and encrypts feature information from the low-frequency region of the image. A chaos correction sequence ensures unique and secure zero-watermarking, with keys varying based on the watermark image's hash value. Rotation correction improves the accuracy of copyright determination. Experiments show the algorithm's robustness against attacks like median filtering, JPEG compression, scaling, rotation, Gaussian noise, and cropping. Future work aims to enhance the security and efficiency of the encryption system.

2.1 Introduction

With the advancement of marine communication, the storage, copying, and transmission of color images over the Internet have become widespread. This necessitates effective copyright protection and authentication, making watermarking technology essential. Various techniques have been developed to balance image quality and robustness against attacks.

2.1.1 Traditional Watermarking Techniques

Traditional watermarking involves embedding copyright information into the invisible domain of the host image. This method faces significant challenges:

1. **Image Distortion**: Embedding information in low-frequency parts of the image can cause visible distortions.
2. **Compression Attacks**: High-frequency embedding makes the copyright information vulnerable to compression attacks, compromising integrity.

2.1.2 Zero-Watermarking Technology

Zero-watermarking technology ensures image quality by not embedding the watermark into the host image. Instead, it constructs a watermark from the feature matrix of the host image and registers it for copyright protection. This method addresses the issues of image distortion and compression attacks inherent in traditional techniques.

1. **Spatial Domain Methods**: Embed watermarks directly into some coefficients but are less robust to geometric attacks.
2. **Frequency Domain Methods**: Use transformations like DCT and DWT to embed watermarks into feature matrices, improving invisibility but facing computational challenges.

2.1.3 Integration of Chaos Theory

Chaos theory enhances the security of watermarking algorithms. Chaotic systems generate pseudo-random sequences with strong ergodicity, high randomness, and sensitivity to initial conditions, improving encryption security. However, some methods face issues with complexity and periodicity.

2.1.4 Color Image Zero-Watermarking

Most existing zero-watermarking algorithms focus on grey-scale images. Recent methods aim to extend these techniques to color images, addressing limitations in robustness and visibility. The development of robust algorithms for color images involves:

- Converting RGB to other color spaces (e.g., YCbCr) to better utilize image characteristics.

- Ensuring that the feature matrix fully represents the color image without reducing it to grey-scale.

2.2 Chaotic system

Recently, a 5-D chaotic system has been proposed, which can be described as

$$\begin{cases} \dot{x} = -ax + yz + bw + u \\ \dot{y} = cy - xz + k \\ \dot{z} = xy - dz + xz - ew \\ \dot{u} = gy \end{cases} \tag{2.1}$$

A 5-D chaotic system is described by a set of differential equations involving the variables x, y, z, w, u and system parameters a, b, c, d, k, e, g. Specific values for these parameters are given: $a = 11$, $b = 2$, $c = 10$, $d = 21$, $k = 3$, $e = 10$, and $g = 1.15$.

In this system, the Lyapunov exponent spectrums and bifurcation diagrams indicate that the system exhibits chaotic behavior. When the parameters a, b, and c are varied within specific ranges, the system shows a wide range of chaotic states. The largest Lyapunov exponents in these conditions are always positive, confirming the presence of chaos.

This chaotic system has high randomness and sensitivity to initial conditions and parameters. It can generate random chaotic sequences, making it suitable for applications like image encryption to enhance security.

2.3 Basic Theory

Since the watermarks that can identify copyright are not embedded in the host image, the superiority of the zero-watermarking algorithm mainly depends on whether the selected feature has strong robustness.

2.3.1 Lifting wavelet transform(LWT)

The lifting wavelet transform (LWT) is an efficient and simple method for analyzing signals by transforming an image into four bands: high-low (HL), low-high (LH), high-high (HH), and low-low (LL). The LL band contains a rough approximation of the image and most of its energy, making it stable and ideal for feature extraction and zero-watermarking. The high-frequency bands (HL, LH, HH) capture detailed fringe information.

2.3.2 Singular value decomposition(SVD)

Singular Value Decomposition (SVD) is a method that decomposes data into three matrices: The left singular matrix U, the singular matrix S, and the right singular matrix V^T.

The orthogonal matrices U and V represent the image's geometric structure, while the diagonal matrix S represents the image's brightness information. S contains important and stable features of the image, but since singular values of different images can be identical, false alarms can occur when detecting watermark information.

To address this, a correction factor β is introduced to modify SVD, making the singular values less sensitive to attacks.

2.4 Workflow of the Algorithm

The algorithm consists of two main parts: the zero-watermarking construction and the watermark image extraction.

2.4.1 The Zero-Watermarking Construction

The zero-watermarking construction process involves several steps. First, the watermark image is preprocessed by combining individual watermark images (W1, W2, W3) into a single watermark (W) and using SHA-256 hashing and a 5-D chaotic mapping for enhanced security. Chaotic sequences are generated from the hash value, affecting the initial conditions of the chaotic system, with initial iterations discarded to obtain a stable chaotic sequence. Next, the watermark image is encrypted by sorting the chaotic sequence in descending order, recording indices, scrambling the watermark image using these indices, and diffusing it with the chaotic sequence to obtain three encrypted images of size M $\times$ N $\times$ 3.

In the feature extraction step, the color host image is decomposed into R, G, and B components, and the Lifting Wavelet Transform (LWT) is applied to each component to obtain low-frequency coefficient matrices (LLR, LLG, LLB). Improved SVD is then applied by dividing each low-frequency coefficient matrix into K $\times$ K blocks, performing SVD on each block with a chaotic sequence as the correction factor, and extracting the maximum singular value from each block to construct a feature matrix T_k of size M $\times$ N $\times$ 3. The feature matrix T_k is binarized to form a binary sequence T, which is then converted into a binary feature matrix. Finally, the zero-watermarking is generated by XORing the encrypted watermark information with the binary feature matrix, scrambling and diffusing the result using a chaotic sequence,

and registering the zero-watermarking, along with the key and timestamp, in the Intellectual Property Database (IPR).

2.4.2 The Watermark Image Extraction

The watermark image extraction process begins with correcting the attacked host image using the Radon algorithm to detect and correct rotation and resizing the image to match the original dimensions. Next, image features are extracted by applying the feature extraction procedure to obtain the feature matrix T'.

The zero-watermarking is then decrypted by retrieving and decrypting it from the IPR database. Finally, the watermark image W' is extracted by XORing the feature matrix T' with the decrypted zero-watermarking.

2.5 Algorithm

Algorithm 3 Zero Watermark Embedding Process

Require: wm (watermark image), $host$ (host image)
Ensure: zw_data (data structure containing the zero-watermark and metadata)
function ZEROWATERMARK_CONS(wm, $host$)
$wm \leftarrow imresize(wm, [64, 64])$
$wm \leftarrow im2gray(wm)$
$wm \leftarrow imbinarize(wm)$
$[M, N, \ ] \leftarrow size(wm)$
$hash \leftarrow DataHash(wm,' SHA - 256')$
$k\_blks \leftarrow reshape(hash, 8, 8)$
$key\_strm \leftarrow zeros(1, 8)$
for $i \leftarrow 1$ to 8 **do**
$key\_strm(i) \leftarrow (10^{-15}) \times mean(k\_blks(i, :))$
end for
$x0 \leftarrow 1 + key\_strm(1) + key\_strm(5)$
$y0 \leftarrow 1 + key\_strm(2) + key\_strm(6)$
$z0 \leftarrow 1 + key\_strm(3) + key\_strm(7)$
$w0 \leftarrow 1 + key\_strm(4) + key\_strm(8)$
$u0 \leftarrow 1 + key\_strm(5) + key\_strm(8)$

$m \leftarrow 5000$
$num\_iter \leftarrow m + M \times N$
$[X, Y, Z, W, U] \leftarrow zeros(1, num\_iter)$
$x \leftarrow x0$
$y \leftarrow y0$
$z \leftarrow z0$
$w \leftarrow w0$
$u \leftarrow u0$
for $i \leftarrow 1$ to num_iter **do**
    $X(i) \leftarrow x$
    $Y(i) \leftarrow y$
    $Z(i) \leftarrow z$
    $W(i) \leftarrow w$
    $U(i) \leftarrow u$
    $dx \leftarrow -11 \times x + y \times z + 2 \times w + u$
    $dy \leftarrow 10 \times y - x \times z + 3$
    $dz \leftarrow x \times y - 21 \times z \times w$
    $dw \leftarrow x \times z - 10 \times w$
    $du \leftarrow 1.15 \times y$
    $dt \leftarrow 0.0005$
    $x \leftarrow x + dx \times dt$
    $y \leftarrow y + dy \times dt$
    $z \leftarrow z + dz \times dt$
    $w \leftarrow w + dw \times dt$
    $u \leftarrow u + du \times dt$
end for
$X \leftarrow X(m + 1 : end)$
$Y \leftarrow Y(m + 1 : end)$
$Z \leftarrow Z(m + 1 : end)$
$W \leftarrow W(m + 1 : end)$
$U \leftarrow U(m + 1 : end)$
$[X\_sort, idx] \leftarrow sort(X, 'descend')$
$wm\_flat \leftarrow wm(:)'$
$scr\_flat \leftarrow wm\_flat(idx)$
$scr \leftarrow reshape(scr\_flat, size(wm))$
$Y\_flat \leftarrow Y$
$cipher\_flat \leftarrow bitxor(uint8(scr\_flat), uint8(Y\_flat))$
$cipher \leftarrow reshape(cipher\_flat, size(scr))$ =0

$host \leftarrow imresize(host, [1024, 1024])$
$[R, G, B] \leftarrow host(:, :, 1), host(:, :, 2), host(:, :, 3)$
$wave\_type \leftarrow' db2'$
$lvl \leftarrow 2$
$[LLR, \ ] \leftarrow lwt2(R,' Wavelet', wave\_type, Level = lvl, Int2Int = true)$
$[LLG, \ ] \leftarrow lwt2(G,' Wavelet', wave\_type, Level = lvl, Int2Int = true)$
$[LLB, \ ] \leftarrow lwt2(B,' Wavelet', wave\_type, Level = lvl, Int2Int = true)$
$K \leftarrow 4$
$num\_blkR \leftarrow size(LLR, 1)/K$
$num\_blkC \leftarrow size(LLR, 2)/K$
$Tk \leftarrow zeros(M, N, 3)$
for $i \leftarrow 1$ to num_blkR **do**
  for $j \leftarrow 1$ to num_blkC **do**
    $blkR \leftarrow LLR((i-1) \times K + 1 : i \times K, (j-1) \times K + 1 : j \times K)$
    $blkG \leftarrow LLG((i-1) \times K + 1 : i \times K, (j-1) \times K + 1 : j \times K)$
    $blkB \leftarrow LLB((i-1) \times K + 1 : i \times K, (j-1) \times K + 1 : j \times K)$
    $[UR, SR, VR] \leftarrow svd(double(blkR))$
    $[UG, SG, VG] \leftarrow svd(double(blkG))$
    $[UB, SB, VB] \leftarrow svd(double(blkB))$
    $z\_idx \leftarrow (i-1) \times num\_blkC + j$
    $betaR \leftarrow Z(z\_idx)$
    $betaG \leftarrow Z(z\_idx)$
    $betaB \leftarrow Z(z\_idx)$
    $SR\_corr \leftarrow SR^{betaR}$
    $SG\_corr \leftarrow SG^{betaG}$
    $SB\_corr \leftarrow SB^{betaB}$
    $max\_singR \leftarrow max(diag(SR\_corr))$
    $max\_singG \leftarrow max(diag(SG\_corr))$
    $max\_singB \leftarrow max(diag(SB\_corr))$
    $Tk(i, j, 1) \leftarrow max\_singR$
    $Tk(i, j, 2) \leftarrow max\_singG$
    $Tk(i, j, 3) \leftarrow max\_singB$
  end for
end for

$T \leftarrow zeros(1, M \times N)$
for $i \leftarrow 1$ to $(M \times N - 1)$ **do**
    if $Tk(i) > Tk(i+1)$ **then**
        $T(i) \leftarrow 1$
    else
        $T(i) \leftarrow 0$
    end if
end for
if $Tk(M \times N) > Tk(1)$ **then**
    $T(M \times N) \leftarrow 1$
else
    $T(M \times N) \leftarrow 0$
end if
$T \leftarrow reshape(T, [M, N])$
$zw \leftarrow bitxor(uint8(cipher(:)), uint8(T(:)))'$
$scr\_zw \leftarrow bitxor(uint8(zw), uint8(Z))$
$scr\_zw\_mat \leftarrow reshape(scr\_zw, [M, N])$
$key \leftarrow \{x0, y0, z0, w0, u0\}$
$timestamp \leftarrow datetime('now')$
$zw\_data \leftarrow struct('Key', key, 'Timestamp', timestamp, 'ZeroWatermark', scr\_zw\_mat)$
return zw_data

Algorithm 4 Zero Watermark Extraction Process

Require: wm (watermark image), $attacked\_host$ (attacked host image), zw_data (precomputed zero-watermark data)
Ensure: $final\_wm$ (extracted watermark image)

function ZeroWatermark_Extr(wm, $attacked\_host$, zw_data)

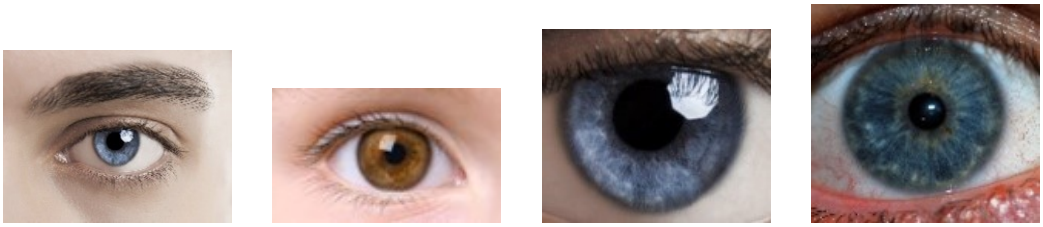
$wm \leftarrow imresize(wm, [64, 64])$
$wm \leftarrow im2gray(wm)$
$wm \leftarrow imbinarize(wm)$
$[M, N, \ ] \leftarrow size(wm)$
$hash \leftarrow DataHash(wm,' SHA - 256')$
$k\_blk \leftarrow reshape(hash, 8, 8)$
$key\_strm \leftarrow zeros(1, 8)$
for $i \leftarrow 1$ to 8 **do**
$key\_strm(i) \leftarrow (10^{-15}) \times mean(k\_blk(i, :))$
end for
$x0 \leftarrow 1 + key\_strm(1) + key\_strm(5)$
$y0 \leftarrow 1 + key\_strm(2) + key\_strm(6)$
$z0 \leftarrow 1 + key\_strm(3) + key\_strm(7)$
$w0 \leftarrow 1 + key\_strm(4) + key\_strm(8)$
$u0 \leftarrow 1 + key\_strm(5) + key\_strm(8)$
$m \leftarrow 5000$
$num\_iter \leftarrow m + M \times N$
$[X, Y, Z, W, U] \leftarrow zeros(1, num\_iter)$
$x \leftarrow x0$
$y \leftarrow y0$
$z \leftarrow z0$
$w \leftarrow w0$
$u \leftarrow u0$

for $i \leftarrow 1$ to num_iter **do**
$X(i) \leftarrow x$
$Y(i) \leftarrow y$
$Z(i) \leftarrow z$
$W(i) \leftarrow w$
$U(i) \leftarrow u$
$dx \leftarrow -11 \times x + y \times z + 2 \times w + u$
$dy \leftarrow 10 \times y - x \times z + 3$
$dz \leftarrow x \times y - 21 \times z \times w$
$dw \leftarrow x \times z - 10 \times w$
$du \leftarrow 1.15 \times y$
$dt \leftarrow 0.0005$
$x \leftarrow x + dx \times dt$
$y \leftarrow y + dy \times dt$
$z \leftarrow z + dz \times dt$
$w \leftarrow w + dw \times dt$
$u \leftarrow u + du \times dt$
end for
$X \leftarrow X(m + 1 : end)$
$Y \leftarrow Y(m + 1 : end)$
$Z \leftarrow Z(m + 1 : end)$
$W \leftarrow W(m + 1 : end)$
$U \leftarrow U(m + 1 : end)$
$attacked\_host\_gray \leftarrow rgb2gray(attacked\_host)$
$corrected\_host \leftarrow attacked\_host$
$original\_size \leftarrow [1024, 1024]$
$resized\_host \leftarrow imresize(corrected\_host, original\_size)$
$host \leftarrow resized\_host$
$R \leftarrow host(:, :, 1)$
$G \leftarrow host(:, :, 2)$
$B \leftarrow host(:, :, 3)$
$wavelet \leftarrow' db2'$
$lvl \leftarrow 2$
$=0$

$[LLR,\ ] \leftarrow lwt2(R,'Wavelet', wavelet, Level = lvl, Int2Int = true)$
$[LLG,\ ] \leftarrow lwt2(G,'Wavelet', wavelet, Level = lvl, Int2Int = true)$
$[LLB,\ ] \leftarrow lwt2(B,'Wavelet', wavelet, Level = lvl, Int2Int = true)$
$K \leftarrow 4$
$num\_blkR \leftarrow size(LLR, 1)/K$
$num\_blkC \leftarrow size(LLR, 2)/K$
$Tk \leftarrow zeros(M, N, 3)$
for $i \leftarrow 1$ to num_blkR **do**
  for $j \leftarrow 1$ to num_blkC **do**
    $blkR \leftarrow LLR((i-1) \times K + 1 : i \times K, (j-1) \times K + 1 : j \times K)$
    $blkG \leftarrow LLG((i-1) \times K + 1 : i \times K, (j-1) \times K + 1 : j \times K)$
    $blkB \leftarrow LLB((i-1) \times K + 1 : i \times K, (j-1) \times K + 1 : j \times K)$
    $[UR, SR, VR] \leftarrow svd(double(blkR))$
    $[UG, SG, VG] \leftarrow svd(double(blkG))$
    $[UB, SB, VB] \leftarrow svd(double(blkB))$
    $Z\_idx \leftarrow (i-1) \times num\_blkC + j$
    $beta\_R \leftarrow Z(Z\_idx)$
    $beta\_G \leftarrow Z(Z\_idx)$
    $beta\_B \leftarrow Z(Z\_idx)$
    $SR\_corr \leftarrow SR^{beta\_R}$
    $SG\_corr \leftarrow SG^{beta\_G}$
    $SB\_corr \leftarrow SB^{beta\_B}$
    $max\_singR \leftarrow max(diag(SR\_corr))$
    $max\_singG \leftarrow max(diag(SG\_corr))$
    $max\_singB \leftarrow max(diag(SB\_corr))$
    $Tk(i, j, 1) \leftarrow max\_singR$
    $Tk(i, j, 2) \leftarrow max\_singG$
    $Tk(i, j, 3) \leftarrow max\_singB$
  end for
end for
$T \leftarrow zeros(1, M \times N)$
for $i \leftarrow 1$ to $(M \times N - 1)$ **do**
  if $Tk(i) > Tk(i + 1)$ **then**
    $T(i) \leftarrow 1$
  else
    $T(i) \leftarrow 0$
  end if
end for

if $Tk(M \times N) > Tk(1)$ **then**
    $T(M \times N) \leftarrow 1$
else
    $T(M \times N) \leftarrow 0$
end if
$T \leftarrow reshape(T, [M, N])$
$load('zero\_watermark\_data.mat', 'zw\_data')$
$enc\_zw \leftarrow zw\_data.ZeroWatermark$
$num\_elem \leftarrow numel(enc\_zw)$
$scrambled\_zw \leftarrow zeros(size(enc\_zw), 'like', enc\_zw)$
for $i \leftarrow 1$ to num_elem **do**
    $scrambled\_zw(i) \leftarrow bitxor(enc\_zw(i), uint8(Z(i)))$
end for
$M \leftarrow size(enc\_zw, 1)$
$N \leftarrow size(enc\_zw, 2)$
$scrambled\_zw \leftarrow reshape(scrambled\_zw, [M, N])$
$Y\_flat \leftarrow Y$
$decrypted\_zw \leftarrow scrambled\_zw$
$T\_flat \leftarrow reshape(T, [], 1)$
$W\_prime\_flat \leftarrow bitxor(uint8(T\_flat), uint8(decrypted\_zw(:)))$
$W\_prime \leftarrow reshape(W\_prime\_flat, [size(T, 1), size(T, 2)])$
$W\_prime\_flat \leftarrow W\_prime(:)$
$[X\_sorted, idx] \leftarrow sort(X, 'descend')$
$rev\_scrambled\_W\_prime\_flat \leftarrow zeros(size(W\_prime\_flat))$
$rev\_scrambled\_W\_prime\_flat(idx) \leftarrow W\_prime\_flat$
$rev\_scrambled\_W\_prime \leftarrow reshape(rev\_scrambled\_W\_prime\_flat, size(W\_prime))$
$rev\_scrambled\_W\_prime\_flat \leftarrow rev\_scrambled\_W\_prime(:)'$
$rev\_scrambled\_W\_prime\_flat \leftarrow bitxor(uint8(rev\_scrambled\_W\_prime\_flat), uint8(Y\_$
$final\_wm \leftarrow reshape(rev\_scrambled\_W\_prime\_flat, size(rev\_scrambled\_W\_prime))$
Display $final\_wm$

2.6 Performance Analysis



Host Image 1 Host Image 2 Host Image 3 Host Image 4

Figure 2.1: Host Images

Figure 2.2: Binary Watermark

2.6.1 Equalization of zero-watermarking:-

Image	Equilibrium Parameter (E_N)
Host Image 1	0.0625
Host Image 2	0.0625
Host Image 3	0.0625
Host Image 4	0.0103

Table 2.1: Equalization of zero-watermarking

2.6.2 Attack Analysis:-

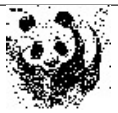
Attack	Median Filtering	JPEG Compression	Scaling
Original Watermark			
Extracted Watermark			

Table 2.2: Extracted watermarks under median filter, JPEG compress and scaling attack

2.6.3 Experiment results under various attacks:-

Attack	Median Filtering	JPEG Compression	Scaling
NC	0.96749	0.90916	0.97077
PSNR(dB)	13.6194	9.1951	14.0824
SSIM	0.6459	0.4091	0.6556
BER	0.0435	0.1204	0.0391

Table 2.3: The experiment results under median filter, JPEG compress and scaling attack

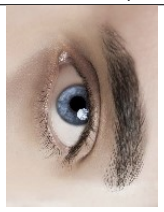
Attack	Rotation (90°)
Rotated_Image	
Original Watermark	
Extracted Image	

Table 2.4: Rotated image and extracted watermark under rotation attacks

Attack	Rotation (90°)
NC	1.0000
PSNR (dB)	Inf
SSIM	1.0000
BER	0.0000

Table 2.5: The experiment results under rotating attack

2.6.4 Gaussian noise and salt and pepper noise attack:-

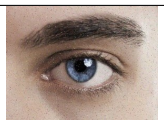
Attack	Gaussian Noise	Salt & Pepper Noise
Image		
Original Watermark		
Extracted Watermark		

Table 2.6: Disturbed images and extracted watermarks under Gaussian noise and salt and pepper noise attack

Attack	Gaussian Noise	Salt & Pepper Noise
NC	0.8958	0.90778
PSNR (dB)	8.6573	9.1688
SSIM	0.3399	0.3665
BER	0.1362	0.1211

Table 2.7: The experiment results under noise attack

Chapter 3

Zero-Watermarking Technique Using Perceptual Hash and ECC Image Encryption

This chapter outlines the Zero-Watermarking technique, which integrates Perceptual Hashing with Elliptic Curve Cryptography (ECC) for secure watermarking. The process begins with the encryption of the watermark image using ECC, ensuring its security and confidentiality. Next, features from the host image are extracted using a specified method[1, 2], creating a feature matrix. From this matrix, a perceptual hash[3] is generated, capturing the essential characteristics of the host image.

To create the zero-watermark, an ADD operation is performed between the encrypted watermark and the perceptual hash image. This operation combines the encrypted watermark with the perceptual hash to produce the final zero-watermark. During the extraction phase, the attacked host image is processed using the same feature extraction technique to generate a new feature matrix, from which a new perceptual hash is derived.

A SUBTRACT operation is then conducted between the perceptual hash and the zero-watermark to retrieve the encrypted watermark. Finally, this extracted watermark is decrypted using ECC to recover the original watermark image. This method effectively embeds and retrieves the watermark while preserving the integrity of the host image.

3.1 Basic Theory

3.1.1 Perceptual Hash

Perceptual Hashing is a technique specifically designed for generating hash values that are resilient to minor modifications in images. Unlike traditional cryptographic hash functions, which produce a unique hash value for every distinct input and are highly sensitive to even the smallest changes, perceptual hashes are tailored to handle slight alterations that do not significantly impact the visual appearance of the image. The primary goal of perceptual hashing is to create a hash value that reflects the perceptual content of the image, meaning that images with similar visual content will yield similar hash values, even if they have undergone minor transformations such as compression, resizing, or color adjustments.

The process of generating a perceptual hash typically involves several steps. First, the image is processed to extract key visual features, such as color histograms, texture patterns, or frequency components. These features are then encoded into a compact hash value, which represents the image's visual characteristics. This hash value is designed to be robust against common image manipulations, ensuring that the perceptual hash remains consistent for visually similar images. Perceptual hashes are valuable in various applications, including image comparison, copyright protection, and watermarking, as they provide a means to identify and verify images based on their visual content rather than their exact binary representation.

3.1.2 Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) is an advanced form of public-key cryptography that leverages the mathematical properties of elliptic curves over finite fields. ECC is distinguished by its ability to achieve high levels of security with relatively small key sizes compared to other cryptographic methods, such as RSA or DSA. The security of ECC is based on the difficulty of solving the elliptic curve discrete logarithm problem, which involves finding the integer that represents a point on the curve given another point and a scalar multiplication.

The key advantage of ECC is its efficiency in terms of both computational resources and key size. For a given level of security, ECC requires much shorter key lengths than traditional cryptographic algorithms. For example, a 256-bit key in ECC can provide security comparable to a 3072-bit key in RSA. This efficiency makes ECC particularly suitable for environments with constrained resources, such as mobile devices and embedded systems.

In the context of watermarking, ECC is used to encrypt the watermark image, ensuring its protection and confidentiality. The encryption process involves transforming the watermark into a secure format that can only be decrypted by authorized parties possessing the appropriate ECC key. This ensures that the watermark remains secure even if the host image is subjected to attacks or manipulations. ECC's robustness and efficiency make it an ideal choice for securing digital watermarks and protecting intellectual property.

3.2 Proposed Zero-Watermarking Algorithm

This section outlines the proposed zero-watermarking algorithm, which consists of two main parts: Zero-Watermark Construction and Zero-Watermark Extraction. The algorithm utilizes Perceptual Hashing and Elliptic Curve Cryptography (ECC)[4] to securely embed and retrieve watermarks from images.

3.2.1 Zero-Watermark Construction

The process of constructing the zero-watermark involves the following steps:

1. **Encryption of Watermark**: The watermark image is encrypted using Elliptic Curve Cryptography (ECC). ECC is employed to ensure the security and confidentiality of the watermark, making it resistant to unauthorized access and tampering.

2. **Feature Extraction from Host Image**: The host image is processed to extract its features using a specified method, resulting in a feature matrix. This matrix represents the essential characteristics of the host image and is used to generate a perceptual hash.

3. **Generation of Perceptual Hash**: A perceptual hash is computed from the feature matrix of the host image. This hash captures the visual content of the image and remains consistent despite minor modifications, ensuring robustness in the watermarking process.

4. **Creation of Zero-Watermark**: An ADD operation is performed between the encrypted watermark and the perceptual hash image. This operation combines the encrypted watermark with the perceptual hash to produce the final zero-watermark. The zero-watermark is then embedded in the host image, preserving its visual content while embedding the watermark information.

3.2.2 Zero-Watermark Extraction

The extraction process for retrieving the zero-watermark involves the following steps:

1. **Feature Extraction from Attacked Host Image**: The attacked host image is subjected to the same feature extraction technique used in the construction phase, generating a new feature matrix that reflects the image's altered state.

2. **Generation of Perceptual Hash from Attacked Image**: A perceptual hash is created from the new feature matrix of the attacked host image. This hash serves as a comparison point for extracting the watermark.

3. **Retrieval of Encrypted Watermark:** A SUBTRACT operation is performed between the perceptual hash of the attacked image and the zero-watermark. This operation isolates the encrypted watermark, which is then extracted from the zero-watermark.

4. **Decryption of Watermark**: The extracted watermark is decrypted using ECC to recover the original watermark image. This step restores the watermark to its original form, verifying the integrity and authenticity of the host image.

This algorithm effectively embeds and retrieves watermarks while maintaining the visual integrity of the host image and ensuring the security of the watermark through ECC.

3.3 Algorithm Overview

This section provides an overview of the zero-watermarking algorithm, detailing the steps involved in both the construction and extraction processes.

Algorithm 5 Zero Watermark Embedding Process.

Require: img (input image), $watermarkImage$ (watermark image)
Ensure: $ZeroWatermark$ (zero watermark image), $C2$, r, R_pub

function ZEROWATERMARK_CONS(img, $watermarkImage$)

$img \leftarrow imresize(img, [512, 512])$
$img \leftarrow rgb2gray(img)$
$img \leftarrow wiener2(img)$
$[LDN\_hist, imgDesc] \leftarrow desc\_LDN(img)$
$mean\_val \leftarrow mean(LDN\_hist)$
$perceptual\_hash \leftarrow zeros(size(LDN\_hist))$
for $i \leftarrow 1$ to length(LDN_hist) **do**
  if $LDN\_hist(i) \geq mean\_val$ **then**
    $perceptual\_hash(i) \leftarrow 1$
  else
    $perceptual\_hash(i) \leftarrow 0$
  end if
end for
$watermarkImage \leftarrow imresize(watermarkImage, [64, 64])$
$watermarkImage \leftarrow im2gray(watermarkImage)$
$inputImage \leftarrow double(watermarkImage)$
$r \leftarrow pso(inputImage)$
$a \leftarrow -7$
$b \leftarrow 10$
$p \leftarrow 487$
$G \leftarrow [13, 46]$
$[R\_pubX, R\_pubY] \leftarrow ECC\_NP(a, b, p, r, G(1), G(2))$
$R\_pub \leftarrow [R\_pubX, R\_pubY]$
$[cipherImage, C2] \leftarrow ECC\_Encryption(inputImage, r, R\_pub, [a, b, p], G)$
$perceptual\_hash \leftarrow repmat(perceptual\_hash, size(cipherImage, 1), 1)$
$ZeroWatermark \leftarrow (cipherImage + perceptual\_hash)$
return $ZeroWatermark$, $C2$, r, R_pub

end function

Algorithm 6 Zero Watermark Extraction Process.

Require: img (input image), $ZeroWatermark$ (zero watermark image), $C2$, r, R_pub

Ensure: $ExtractedWatermark$ (decrypted watermark)

function ZeroWatermark_Extr(img, $ZeroWatermark$, $C2$, r, R_pub)

$img \leftarrow imresize(img, [512, 512])$

$img \leftarrow im2gray(img)$

$img \leftarrow wiener2(img)$

$[LDN\_hist, imgDesc] \leftarrow desc\_LDN(img)$

$mean\_val \leftarrow mean(LDN\_hist)$

$perceptual\_hash \leftarrow zeros(size(LDN\_hist))$

for $i \leftarrow 1$ to length(LDN_hist) **do**

if $LDN\_hist(i) \geq mean\_val$ **then**

$perceptual\_hash(i) \leftarrow 1$

else

$perceptual\_hash(i) \leftarrow 0$

end if

end for

$[ZeroWatermark, C2, r, R\_pub] \leftarrow load('ZeroWatermark.mat')$

$watermarkImage \leftarrow imresize(watermarkImage, [64, 64])$

$watermarkImage \leftarrow im2gray(watermarkImage)$

$perceptual\_hash \leftarrow repmat(perceptual\_hash, size(ZeroWatermark, 1), 1)$

$encyWM \leftarrow (ZeroWatermark - perceptual\_hash)$

$a \leftarrow -7$

$b \leftarrow 10$

$p \leftarrow 487$

$G \leftarrow [13, 46]$

$ExtractedWatermark \leftarrow ECC\_Decryption(encyWM, r, C2, [a, b, p], G)$

return $ExtractedWatermark$

end function

3.4 Performance Analysis

The performance analysis of the proposed Zero-Watermarking technique using Perceptual Hash and Elliptic Curve Cryptography (ECC) is conducted by evaluating the robustness of the watermark against various attacks. The key metrics used in this analysis are:

- **Normalized Correlation (NC):** Measures the similarity between the original and extracted watermarks. A higher NC value indicates a more accurate extraction.
- **Peak Signal-to-Noise Ratio (PSNR):** Indicates the quality of the extracted watermark in relation to the original. A higher PSNR value signifies better quality.
- **Structural Similarity Index (SSIM):** Measures the perceived quality of the extracted watermark compared to the original.
- **Bit Error Rate (BER):** Represents the percentage of bit errors in the extracted watermark compared to the original.

3.4.1 Experimental Results for Image 1

Figure 3.1: Host Image 1

Figure 3.2: Watermark

Attack	Extracted Watermark
Median Filtering	
Gaussian Noise	
Salt and Pepper Noise	
JPEG Compression	
Scaling	
Brightness Adjustment	

Table 3.1: Extracted watermarks under various attacks on Image 1.

Attack	NC	PSNR (dB)	SSIM	BER (%)
Median Filtering	0.85034	11.9877	0.7017	0.0347
Gaussian Noise	0.67123	6.5443	0.3626	0.0913
Salt and Pepper Noise	0.67123	6.5443	0.3626	0.0913
JPEG Compression	0.79623	8.5059	0.3685	0.1192
Scaling	0.85034	11.9877	0.7017	0.0347
Brightness Adjustment	1	99.0000	1.0000	0.0000

Table 3.2: Performance metrics for various attacks on Image 1.

3.4.2 Experimental Results for Image 2

Figure 3.3: Host Image 2

Figure 3.4: Watermark

Attack	Extracted Watermark
Median Filtering	
Gaussian Noise	
Salt and Pepper Noise	
JPEG Compression	
Scaling	
Brightness Adjustment	

Table 3.3: Extracted watermarks under various attacks on Image 2.

3.4.3 Experimental Results for Image 3

Figure 3.5: Host Image 3

Figure 3.6: Watermark

Attack	NC	PSNR (dB)	SSIM	BER (%)
Median Filtering	0.94861	18.4410	0.8477	0.0161
Gaussian Noise	0.84406	11.9953	0.5860	0.0581
Salt and Pepper Noise	0.84406	11.9953	0.5860	0.0581
JPEG Compression	0.80884	9.6595	0.4704	0.0925
Scaling	0.94861	18.4410	0.8477	0.0161
Brightness Adjustment	0.83745	10.9649	0.7402	0.0623

Table 3.4: Performance metrics for various attacks on Image 2.

Attack	Extracted Watermark
Median Filtering	
Gaussian Noise	
Salt and Pepper Noise	
JPEG Compression	
Scaling	
Brightness Adjustment	

Table 3.5: Extracted watermarks under various attacks on Image 3.

Attack	NC	PSNR (dB)	SSIM	BER (%)
Median Filtering	0.86154	13.1244	0.7136	0.0297
Gaussian Noise	0.70998	6.8433	0.3723	0.1066
Salt and Pepper Noise	0.71325	7.1549	0.4126	0.0904
JPEG Compression	0.89269	14.3818	0.7039	0.0417
Scaling	0.86154	13.1244	0.7136	0.0297
Brightness Adjustment	0.93811	17.6315	0.8496	0.0144

Table 3.6: Performance metrics for various attacks on Image 3.

Chapter 4

Conclusion

On our internship period, we embarked on an extensive exploration of advanced image watermarking techniques, delving into both theoretical and practical aspects. The journey began with an in-depth study of two significant research papers that laid the foundation for our understanding of contemporary watermarking methodologies. These papers focused on the integration of Lifting Wavelet Transform (LWT), Singular Value Decomposition (SVD), and chaotic sequences to enhance the security and robustness of watermarking processes.

Our initial phase involved replicating the methodologies detailed in the [5, 6]. This process included the decomposition of color images into R, G, and B components, followed by the application of LWT to each component to obtain low-frequency coefficient matrices. We then performed SVD on blocks of these matrices, extracting the maximum singular values to construct a feature matrix. This matrix was transformed into a binary sequence, which was subsequently combined with encrypted watermark information using chaotic sequences. This rigorous implementation not only reinforced our understanding of the techniques but also provided a robust platform to build upon.

In the subsequent phase, we leveraged the insights gained from the research papers to develop a novel zero-watermarking method. The watermark image was encrypted using Elliptic Curve Cryptography (ECC) to ensure its security. For the host image, we implemented a feature extraction process to form a feature matrix, from which a perceptual hash was generated. An ADD operation was then performed between the encrypted watermark and the perceptual hash image to generate the zero-watermark.

During the watermark extraction phase, the attacked host image underwent the same feature extraction technique to generate a new feature matrix and perceptual hash. A SUBTRACT operation was performed between this perceptual hash and the zero-watermark to retrieve the encrypted watermark,

which was then decrypted using ECC to obtain the original watermark image. This method proved to be robust, efficient, and secure.

To evaluate the performance of our zero-watermarking technique, we computed several metrics, including Normalized Correlation (NC), Bit Error Rate (BER), Structural Similarity Index Measure (SSIM), and Peak Signal-to-Noise Ratio (PSNR). These metrics confirmed the high robustness and reliability of our method against various attacks, validating the efficacy of our approach.

This internship has been an invaluable learning experience, providing us with a comprehensive understanding of advanced watermarking techniques and the opportunity to contribute to the field through our innovative implementation. The integration of theoretical knowledge with practical application has been immensely rewarding, and the skills and insights gained will undoubtedly serve as a solid foundation for future endeavors in the realm of digital image processing and security.

Overall, this project not only fulfilled its initial objectives but also opened new avenues for further research and development, demonstrating the potential for creating highly secure and resilient watermarking techniques in the digital age.

References

[1] C. Turan and K.-M. Lam, "Histogram-based local descriptors for facial expression recognition (fer): A comprehensive study," *Journal of Visual Communication and Image Representation*, 2018.

[2] A. Ramirez Rivera, R. Castillo, and O. Chae, "Local directional number pattern for face analysis: Face and expression recognition," *IEEE Transactions on Image Processing*, vol. 22, no. 5, pp. 1740–1752, 2013.

[3] M. Roy, D. Thounaojam, and S. Pal, "Various approaches to perceptual image hashing systems-a survey," pp. 1–9, 02 2023.

[4] M. Elhoseny and et al., "Hybrid optimization with cryptography encryption for medical image security in internet of things," *Neural Computing and Applications*, pp. 1–15, 2018.

[5] A. Dwivedi, A. Kumar, M. K. Dutta, R. Burget, and V. Myska, "An efficient and robust zero-bit watermarking technique for biometric image protection," in *2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*, pp. 236–240, 2019.

[6] R. Chu, S. Zhang, J. Mou, and X. Gao *Multimedia Tools and Applications*, vol. 82, pp. 1–24, 03 2023.