Digital Image Watermarking: Old and New

Digital Image Watermarking: Old and New

By Wenjie Zhao,

A Thesis Submitted to the School of Graduate Studies in the Fulfillment of the Requirements for the Degree Master of Applied Science

McMaster University
Master of Applied Science (2020)
Hamilton, Ontario (Department of Electrical and Computer Engineering)

TITLE: Digital Image Watermarking: Old and New
AUTHOR: Wenjie Zhao (McMaster University)
SUPERVISOR: Dr. Jun Chen
NUMBER OF PAGES: xiii, 89

Abstract

The digital image watermarking is a process that embeds a secrete sequence into a digital image or a video segment to protect its copyright information.

There are several methods utilized to deal with the watermarking problem. There are in total two different kinds: the popular neural network and the traditional methods. And for the traditional methods, according to their working region, they can be divided into two groups: spatial domain-related algorithms (e.g., LSB SVD) and transformed domain algorithms (e.g., DCT, DWT). The spatial domain algorithms are the methods that directly work on the pixel values of the image, while the transformed domain algorithms are the methods that work on the rate of change of the image pixels.

In the thesis, first of all, we are going to propose a modified hybrid scheme, which has better results compared with the paper (Lin and Wan 2016). Then, we are going to offer a brand-new method concerning the LDPC-LDGM coding structure in the area of information theory to deal with the digital watermarking problem. Notice that this LDGM-LDPC nested code watermarking scheme only provides a particular case example, which is implemented by one of the teammates Dr. Mahdi, and there is still much space for improvement. However, according to the step tests in chapter 4.4, we have achieved a reasonably good result in imperceptibility and robustness.

Acknowledgements

First of all, I would like to appreciate the help of all committee members, Dr. Sorina Dumitrescu, Dr. Jian Kang (JK) Zhang, and Dr. Jun Chen, who help me improve my thesis.

Then, I would like to appreciate the continuous help and support from my supervisor Dr. Jun Chen of my Master's study and research. Dr. Jun Chen is very patient and warm-hearted in guiding me during my research especially when it comes to some very difficult points. I am sincerely grateful and proud to be one of his master students and his enthusiasm hard-working attitude to research inspires me a lot and will become a treasure of my life.

My sincere appreciation also extends to my lab colleagues, who work with me and share points of view in some projects and also encourage each other when meeting difficulties. Besides, I would like to appreciate the help of Dr. Mahdi Nangir, who offered a great help during my research. Thanks will also go to Cheryl Gies, who helps a lot in coordinating courses, defense schedules, and three-year daily work, especially during these difficult times.

Finally, I also would like to appreciate the spiritual and financial support of my parents. They infuse me with endless courage and confidence during these three years of study. Without their supports, it would become impossible for me to fulfill my degree.

To my beloved parents and dear friends

Contents

List of Figures

List of Tables

Abbreviations

BER	bit error rate
PSNR	peak signal to noise ratio
MSE	mean squared error
CNN	convolutional neural network
BLUR	blurring
CR	cropping
HPF	high pass filtering
JPEG	JPEG compression
LPF	low pass filtering
PN	Poisson noise
SPN	salt and pepper noise
TMP	tampering
GF	Gaussian filtering
DROP	Dropout Attack
GS	Gaussian noise
SPE	Speckle noise
LSB	least-significant bit
DCT	discrete cosine transformation
DWT	discrete wavelet transformation
LWT	Lifting wavelet transformation
SVD	Singular Value Decomposition
LDPC	Low-density parity check code
LDGM	low-density generator matrix code

$\mathbf{S}^n$	cover image with size n
$\mathbf{X}^n$	watermarked image with size n
$\mathbf{m}^k$	watermark message with length k
$\mathbf{Y}^n$	contaminated watermarked image with size n
$\hat{m}^k$	recovered watermark with length k
$\mathbf{U}^n$	auxiliary random variable with length n
$\mathbf{R}$	embedding rate
$\mathbf{R}_{LDGM}$	code rate of LDGM
$\mathbf{R}_{LDPC}$	code rate of LDPC
$\mathbf{R}_{code}$	general code rate

Declaration of Authorship

I, Wenjie Zhao, declare that this thesis titled, "Digital Image Watermarking: Old and New" and the work presented in it are my own. I confirm that:

- for all the chapters presented in this thesis except where specific reference is made are all done by myself.
- No other person's contributions (published or unpublished) has been used without notice and acknowledgements in the main text of the thesis.

Chapter 1

Introduction

1.1 Background Information

Digital watermarking is a process that embeds a secret message into a digital image or a video segment for authorization or protection needs, and this hidden message is treated as the watermark. It is a very similar concept to the steganography, while the purposes are different. Steganography focuses on secret information transmission. The use of steganography is to hide the existence of the secret message. Namely, the hidden message can only be discovered by the targeted receiver. The hidden message can be easily destroyed when being found. While for digital watermarking, it pays more attention to identifications and protections. In other words, for the digital watermarking, the watermark can be visible or invisible depending on actual needs, but this watermark message must be robust enough to be recovered after specific attacks to satisfy the need of copyright protection or identifications.

There are three essential components during this process: the encoder, the decoder, and the attacker. The cover image and the watermark message are two inputs to the process. The encoder acts to embed the watermark into the cover image. After that, a watermarked image is obtained at the end of the embedding process. Then, the watermarked image experiences attacks from the attacker, which will contaminate the watermarked image. Then, finally, the

decoder must be able to recover the watermark from the contaminated watermarked image with acceptable distortion.

There are several essential features in digital watermarking, and they are imperceptibility, robustness, and embedding rate. People utilize these three characteristics to compare different digital watermarking algorithms.

Imperceptibility

Imperceptibility is the feature to measure how invisible the watermark is. After the encoding step, the watermark is embedded into the cover image with a chosen encoding algorithm to obtain a watermarked image. Imperceptibility is determined by the distortion between the cover image and the watermarked image. The lower the distortion, the better the imperceptibility. This feature is essential for algorithms with invisible watermarks. Notice that in this thesis, we only focus on discussing the algorithms with invisible watermarks.

Robustness

Robustness is another essential feature to measure how stable the watermark is under different kinds of attack. After the encoding step, the watermarked image experiences different kinds of attack and then the contaminated watermarked image is obtained. After that, the decoding step is applied to recover the watermark from the contaminated watermarked image. The robustness is determined by the distortion between the original watermark and the recovered watermark. The smaller the distortion, the better the robustness.

Embedding Rate

The last criterion for the digital watermarking algorithm is called the embedding rate. The embedding rate indicates the length of the watermark message that can be embedded into the cover image. An efficient digital watermarking algorithm targets to obtain

a high embedding rate while maintaining good robustness and fair imperceptibility as well.

Applications

There are several essential applications related to digital watermarking.

First of all, digital watermarking is broadly used for copyright protection, especially for digital intellectual property, such as images, videos, songs, and animations. These digital products are very easy to be copied and modified without the owner's permission. Consequently, it is necessary to embed a specific message into these digital products to protect the copyrights without damaging their quality, and this particular message is the watermark.

Secondly, digital watermarking is also utilized in business transactions for anti-counterfeiting. Due to the rapid development in electronic commerce, there exist a lot of electronic documents during e-business, such as cheques, contracts, and bills. These files are very crucial that need identifications to avoid financial losses. As a result, digital watermarking is applied to embed invisible secret messages into those essential files to check the validity of the records.

Thirdly, the digital watermarking is also utilized in credentials anti-counterfeiting. Files that related to personal identifications are critical for information protection, such as passport, ID, driving license, and education certification.

Moreover, digital watermarking is also applied as a hint for digital product manipulations. For example, the original digital image is posted on a multimedia platform with a watermark inside. Then, for each time of the legal reprint, the watermark is modified to a slightly different version. Consequently, it is possible to track the manipulation during the copy.

The last application worth mentioning is covert communication. The key to covert communication is to utilize public sources to transmit secret information without attracting the attackers' attention. People use the digital watermarking technique to hide confidential information into the cover data to realize covert communication.

1.2 Contribution

In this thesis, the contribution consists of the following three parts.

First of all, I reviewed the related digital watermarking algorithms and reproduced them to get the experimental results. The experimental results are based on a fixed embedding rate to observe the differences in robustness and imperceptibility of each watermarking scheme.

Next, I proposed a modified hybrid LDPC-SVD-DWT structure to deal with the digital watermarking problem and tested for the results. The comparison is made up of two parts. Firstly, I compared the results with the method proposed in Lin's paper (Lin and Wan 2016). Then, I made a comparison between our proposed hybrid scheme and the reproduced related work to see if there is any improvement.

Lastly, I implemented the step tests for the LDPC-LDGM nested coding scheme with partial results.

1.3 Thesis Structure

Chapter 1 provides a brief introduction to the digital watermarking problem and its corresponding measure criteria. Chapter 2 will review several classical digital watermark algorithms and conclude for both their advantages and disadvantages. Besides, chapter 2 will also give a comprehensive view of the modern deep learning method on the digital watermark problem based on the HiDDeN (Zhu et al. 2018). Chapter 3 will give a background information of our proposed

methods. And then, two proposed methods are introduced in this chapter.

Chapter 4 will first implement experimental results on classical algorithms and deep learning method. Then, the result of our proposed methods:the LPDC+SVD+DWT method and the nested LDGM-LDPC coding scheme, are also provided. Besides, Chapter 4 will also provide the comparison test results for SVD, DCT, LSB and proposed hybrid scheme with same embedding rate to observe the characteristics of each scheme.

Last but not the least, Chapter 5 will conclude the present work performance and state the limitations, while in the meantime, discuss the possible future work on this topic.

Chapter 2

Classical Digital Watermarking Scheme

This chapter will give a brief introduction to several classical algorithms on the digital watermark. The digital watermarking algorithm can be mainly divided into three groups concerning the working domains. For the **spatial** domain-related algorithm, the embedding and extraction steps are all completed in bit value directly. For the **transform** domain-related algorithm, the embedding and extraction steps are accomplished in the transformed domain, such as frequency domain. Also, according to the different extraction methods, there are two kinds: blind watermarking scheme and non-blind watermarking scheme. For the **blind** watermarking scheme, the algorithm does not need the information of the cover image to recover the watermark message. For the **non-blind** scheme, the information of the cover image is required for the extraction step to obtain the watermark message.

First of all, Section 2.3 will introduce schemes that work in the spatial domain. Then, Section 2.4 will discuss algorithms that act in the transform domain. After that, the last Section 2.5 will present a modern method that utilizes the neural network.

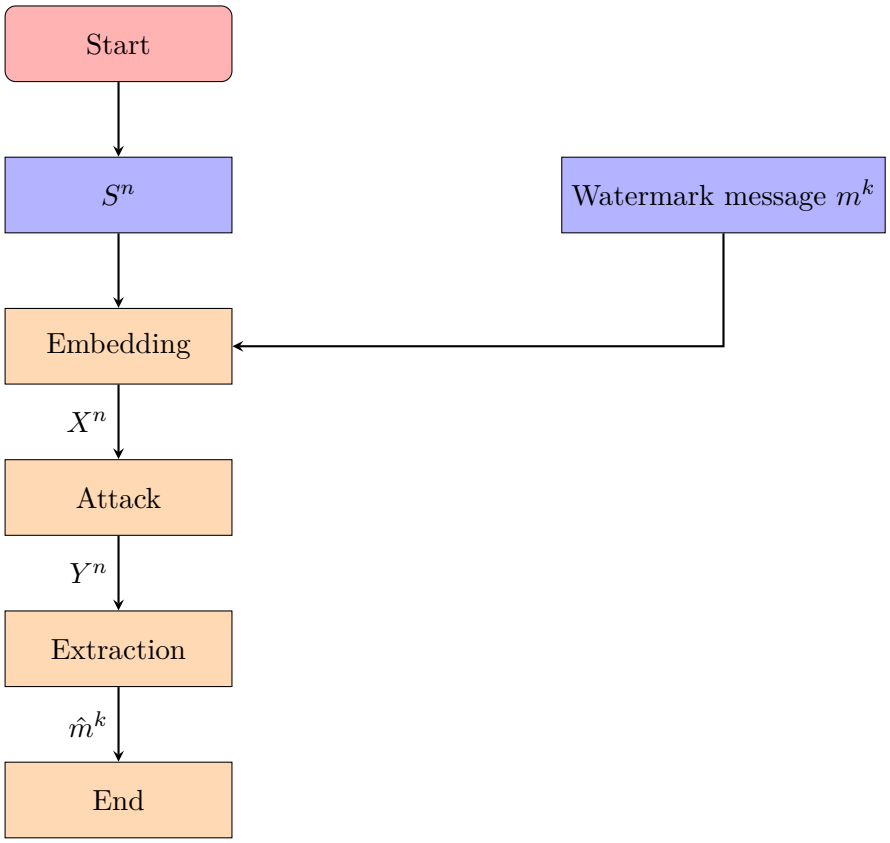

FIGURE 2.1: General Watermarking Scheme

2.1 Notation Clarification

This section will clarify the notations used in the thesis. S^n is the cover image with size n, and it can be treated as the source sequence with length n, and these two concepts can be used interchangeably. If the cover image is non-binary, then n indicates the number of pixels in the cover image. If the cover image is binary, n shows the number of bits in the cover image. X^n is the watermarked image with size n, and it can be treated as the channel input sequence with length n. Notice that the watermarked image and channel input sequence are two interchangeable concepts in this thesis. Y^n is the contaminated watermarked image with size n after the attack, and it can be treated as the channel output sequence with length n. Similarly, the contaminated watermarked image and the channel output sequence can

be used interchangeably. m^k is the watermark message with length k, while the $\hat{m}^k$ is the recovered watermark message with the same length. If the watermark message is a non-binary image, k refers to the number of pixels in the watermark message. If the watermark message is a binary sequence, k refers to the number of bits in the watermark message.

2.2 General Watermarking Scheme

The general watermarking scheme consists of three steps – embedding step, attack, and extraction step. According to the block diagram 2.1, with a particular embedding algorithm, the watermark message m^k is embedded into the cover image S^n and the watermarked image X^n is obtained after the embedding step. Then, the watermarked image X^n experiences noise attacks and the contaminated watermarked image Y^n is obtained. Then, the extraction step is aimed to get the recovered watermark message m^k from the contaminated watermarked image Y^n.

Notice that the distortion between the S^n and the X^n determines the imperceptibility while the distortion between the m^k and $\hat{m}^k$ determines the robustness.

Attack Introduction

This subsection will introduce the attacks that apply during the experiments in this thesis. In total, there are three groups of attack.

The first group is the noise-related attack. There are several kinds of noise-related attacks, and each has its features. For the Gaussian noise attack (GS), the noise follows Gaussian distribution, which is controlled by mean values and noise variance. Commonly, Gaussian noise is caused by noise from electronic components. Salt and Pepper Noise (SPN) attack is another kind of attack that randomly adds pixels with value 0 (black) or 255 (white) into the randomly selected

areas of the image. This kind of noise is often led by image segmentation. The third noise in this group is called speckle noise (SPE) attack, which is a common noise in radar transmission. SPE shares a similar pattern with the salt and pepper noise. The last kind in this group is called the Poisson attack (PN), and this noise model follows the Poisson distribution. Poisson noise is often caused by the uncertainty of measurements due to the existence of photons.

The second group of attacks is related to artificial manipulation. Cropping (CR) is a process that crops a segment of the image and makes this image segment to be the input of the decoder to recover the watermark. For example, if we apply CR attack to a watermarked image with size $r1 \times c1$. Then the resulting segment can have the size $r2 \times c2$, where $r2 \leq r1$ and $c2 \leq c1$. The cropped segment with size $r2 \times c2$ is then utilized to recover the watermark. The second attack in this group is called tampering (TMP) attack. TMP attack is applied by randomly selecting a squared area of the image and cover that area with all black pixels. TMP attack will cause the information loss of certain regions. The third attack in this group is called JPEG compression (JPEG) while it is commonly used during image transmission.

The third group of attacks is related to image filtering. High-pass filtering (HPF) and low-pass filtering (LPF) are two similar processes with different convolution kernels. HPF is applied to amplify noise and sharpen the image details while LPF is utilized to smooth out noise. Similarly, the Gaussian filtering (GF) is another filtering attack whose impulse response is a Gaussian function.

Metrics Specification

The metrics utilized in this thesis are specified in this subsection. To measure the distortion between the cover image S^n and the watermarked image X^n, we use the peak-signal-to-noise ratio (PSNR) as

the quality measurement. The PSNR can be expressed by the mean-squared error (MSE), which is shown in the following equations,

$$MSE = \frac{\sum_{i=0}^{n-1} [\, X_i^n - S_i^n]^2}{n},$$

where n refers to the size of the image and i indicates each pixel or bit in the image. Consequently, the peak-signal to noise ratio (PSNR) can be represented as

$$PSNR = 10 \times log_{10}(\frac{I_{max}^2}{MSE}),$$

where I_{max} is the maximum pixel value in the image concerning its data type. For example, if the input image is in gray-scale, I_{max} is 255; if the input image is in double-precision, I_{max} is 1.

Another metric is introduced to measure the distortion between the original watermark m^k and the recovered watermark $\hat{m}^k$. Since the watermarks used in our experiments are all binary, we introduce the bit-error-rate (BER) as the quality measurement. For binary sequence comparison, the BER can be expressed as the number of error bits divides by the total number of bits in the watermark. Namely, it is shown in the following equation,

$$BER = \frac{N_e}{N_{total}},$$

where N_e the number of error bits and N_{total} is the total number of bits in the binary watermark sequence, and in our case, N_{total} can be treated as the length of the watermark m^k, which equals to k.

The last metric that utilized in this thesis is the embedding rate $\mathcal{R}$. It is defined as the bits of the watermark m^k divide by the total number of bits in the cover image S^n. It can be expressed in the following equation,

$$\mathcal{R} = \frac{L_{watermark}}{N_{cover}},$$

where $L_{watermark}$ is the length of watermark and N_{cover} is the total number of bits in the cover image.

2.3 Spatial Domain Watermark Algorithms

2.3.1 Least Significant Bit(LSB) Scheme

The Least-Significant-Bit algorithm (LSB) is the most straightforward algorithm applied for the digital watermarking problem, as it is shown in Figure 2.2.

During LSB implementation, the binary watermark message $m^k \in \{0,1\}^k$ will substitute the bits on the least significant bit-plane of the cover image S^n bit by bit to obtain the watermarked image X^n. Then, X^n experiences attacks and the contaminated watermarked image Y^n is obtained. After that, the recover watermark message $\hat{m}^k$ is obtained by extracting the bits from the least significant bit-plane of Y^n.

LSB shows high robustness under spatial-based attacks (e.g., CR, TMP), but it has difficulties in recovering watermark messages under frequency-related attacks (e.g., JPEG, GS, GF, SPN, PN). It is reasonable since the bits in the least-significant bit-plane are the most vulnerable bits under frequency attack. Consequently, this watermarking scheme has limited applications, and it turns out to be impossible to extract watermarks with low distortion during the experiments.

2.3.2 Singular Value Decomposition(SVD) Scheme

The principle of SVD is first demonstrated by G. H. GOLUB and C. REINSCH (GOLUB and REINSCH 1970) in 1970. It is broadly utilized in image processing and signal processing areas. The SVD-related watermarking scheme is applied by adding the watermark to the singular value components of the cover image, and the watermarked image is obtained after several steps of SVD composition

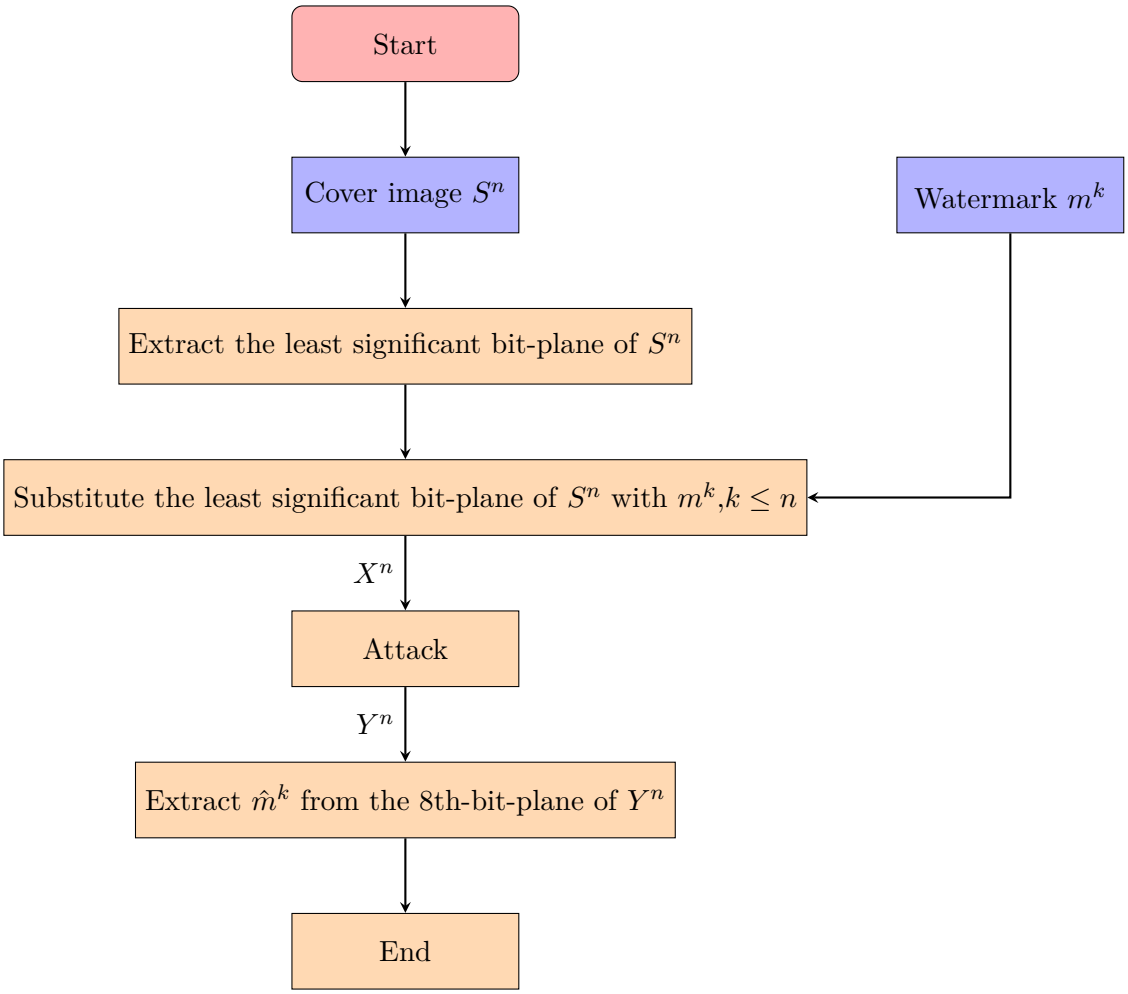

FIGURE 2.2: Least-Significant-Bit Watermarking Scheme

and decomposition. Then, the extraction step is simply the inverse of the embedding step.

For more specific explanation, the SVD-related watermarking scheme is indicated in Figure 2.3. The cover image S^n undergoes SVD process,

$$S^n = U\mathcal{S}V$$

where the columns of U are the eigenvectors of AA^T and the columns of V are the eigenvectors of A^TA. U and V are orthnormalized to each other. The middle component $\mathcal{S}$ is a singular value matrix, which has the same size as S^n. The singular values in $\mathcal{S}$ matrix

refer to the energy of the image S^n. It is said that the majority energy of a particular image is concentrated on the large singular values. It is then demonstrated that when adding a small disturbance to the image, it will not make much change on the singular value of the cover image S^n. Consequently, this SVD-related algorithm has comparatively optimal results under frequency-related attacks. Mathematical speaking, the general SVD watermarking algorithm is shown as the following:

Step1: SVD of the cover image S^n:

$$S^n = U\mathcal{S}V^T.$$

Step2:Add the watermark m^k to the singular value matrix **S** of the source sequence to get the matrix A_1, and then apply SVD to A_1 to obtain its singular value matrix S_1.

$$A_1 = \mathcal{S} + \alpha m^k,$$

$$A_1 = U_1\mathcal{S}_1V_1^T,$$

where α is the strength factor

Step3: Obtain the watermarked image X^n

$$X^n = U\mathcal{S}_1V^T,$$

where U and V are the eigenvectors from the cover image S^n

Step4: After the attack, Y^n is obtained.

Step5: The extraction step is similar with the embedding step with singular value decomposition and its inverse process to get the recovered watermark message. $\hat{m}^k$

$$Y^n = U_2\hat{\mathcal{S}}_1V_2^T,$$

$$D = U_1\hat{\mathcal{S}}_1V_1,$$

$$\hat{m}^k = \frac{1}{\alpha}(D - \mathcal{S}),$$

where D is the intermediate value matrix. Since this SVD-related watermarking scheme requires the cover image information, which refers to the singular value matrix $\mathcal{S}$ of S^n, this is a non-blind watermarking scheme.

The SVD algorithm has comparatively good results in resisting frequency-based attacks (e.g., JPEG, GS, GF, SPN, PN). However, due to the principle of SVD, there are some limitations to such an algorithm. First of all, although the experimental results for robustness tests are better than those from LSB, they are still not that optimal. Secondly, the trade-off between imperceptibility and robustness is controlled by the strength factor α. The larger the α is, the more robust the watermark and the less transparent the watermarked image is. Sometimes, the distortion between the watermarked image and the cover image is obvious. Third, SVD can only survive JPEG with a moderate compression rate (Vivek Singh Verma 2015) and have a poor performance in geometrical attacks (e.g., CR) due to the loss of information to a specific location.

The experiments in Chapter 4 are based on the original standard form SVD algorithm, and its working scheme is shown as in Figure 2.3. There are also many derived SVD watermarking scheme. Methods (Aslantas 2009; Ahmad A.Mohammad and Shaltaf 2008) that only modify the singular values in the diagonal matrix that are non-blind. Compared with a non-blind SVD-based scheme (Aslantas 2009), some methods modify U and V components to realize the blind SVD scheme (C.-C. Chang and Lin 2005), which introduces a higher security level.

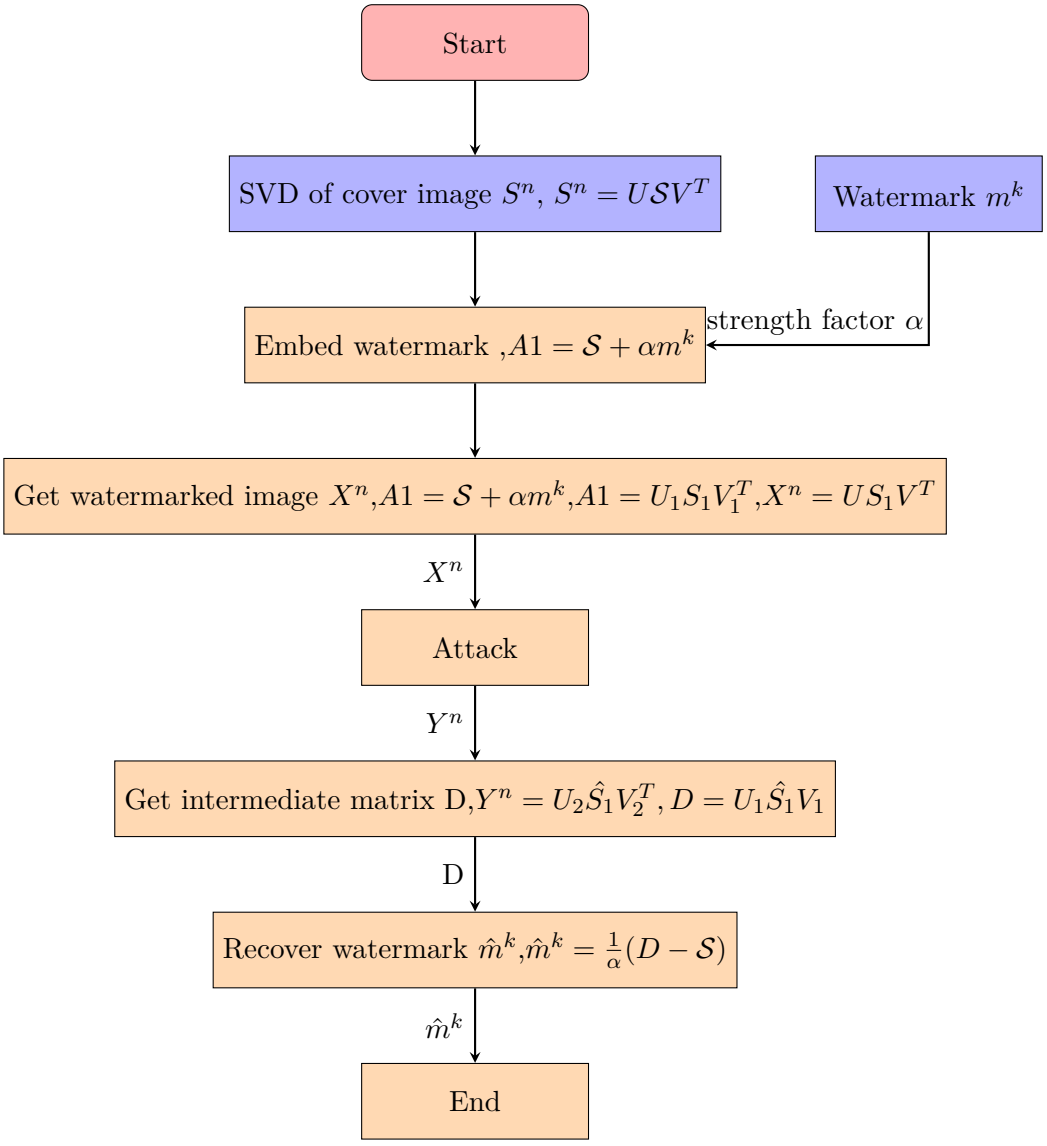

FIGURE 2.3: Singular Value Decomposition Watermarking Scheme

2.4 Transform Domain Watermark Algorithms

The second group of classical algorithms works in the transform domain. This group of algorithms does not work directly on the image pixels but first transformed into another domain (always frequency domain) and then complete the embedding and extraction step in order to achieve high robustness against frequency attacks.

The following parts of this section are going to give a brief explanation of two typical transform-domain algorithms: DCT and DWT respectively.

2.4.1 Discrete Cosine Transformation(DCT) Scheme

Discrete Cosine Transformations (DCT) is a process that transforms the digital image from the spatial domain to the frequency domain. This section will introduce a DCT-related algorithm with full detail.

The advantages and disadvantages of this algorithm are stated as the following. For most of the images, the primary visible information is located in the low-frequency area, while the edges information is located in the high-frequency area. After the DCT process, all the information with low frequency would be settled at the upper left corner of DCT. It is universally acknowledged that some actions like JPEG compression can only get rid of the high-frequency-part of the image so that the resulted image becomes blurry. Consequently, when the watermark is embedded into the low or mid-frequency-band in the DCT domain, it can efficiently survive frequency-based attacks (e.g., JPEG, LPF). However, the imperceptibility will not be optimal if there is no strength factor to control the embedding process. As a result, the parameter alpha α is introduced to improve imperceptibility. There is one significant limitation that cannot be overemphasized: the number of the non-overlapping blocks in the cover image must equal to the number of bits in the watermark for the algorithm in Figure 2.4 to work correctly.

The example DCT watermarking scheme is shown in Figure 2.4, where the strength factor α is a predefined value in the range $[0, 1]$ and the parameter β is another factor that determined by the bit value of the watermark message. If $m_i^k = 0$, then $\beta = -1$; otherwise, $\beta = 1$. The realized 8×8 block-DCT-based algorithm in MATLAB is shown in Chapter 4.

Many researchers have modified the original DCT-version to meet better performance. Hsu and Wu (Hsu and Wu 1999) embedded the watermark into the middle-frequency band instead of the low-frequency band to get better imperceptibility without sacrificing the robustness. While in another paper (Cox et al. 1997), authors compared the performance under different attacks and found out the outstanding performance of DCT-related watermarking scheme under geometrical attacks (e.g., CR). After one year, M. Barni and the other three researchers (Barni et al. 1998) enhanced the algorithm (Cox et al. 1997) performance. Besides, instead of embedding an image as watermark, in this paper (Barni et al. 1998), authors also used a randomly generated binary sequence as the watermark message. All the modified algorithms mentioned above are non-blind watermarking schemes, which indicates that the decoder has to know the input cover image to recover the watermark. In 2003, A. Noore (Noore 2003) introduced a blind DCT watermarking scheme which indicates high robustness under most frequency attacks (e.g., JPEG, SPE, PN, GS, HPF, LPF) and some geometrical attacks.

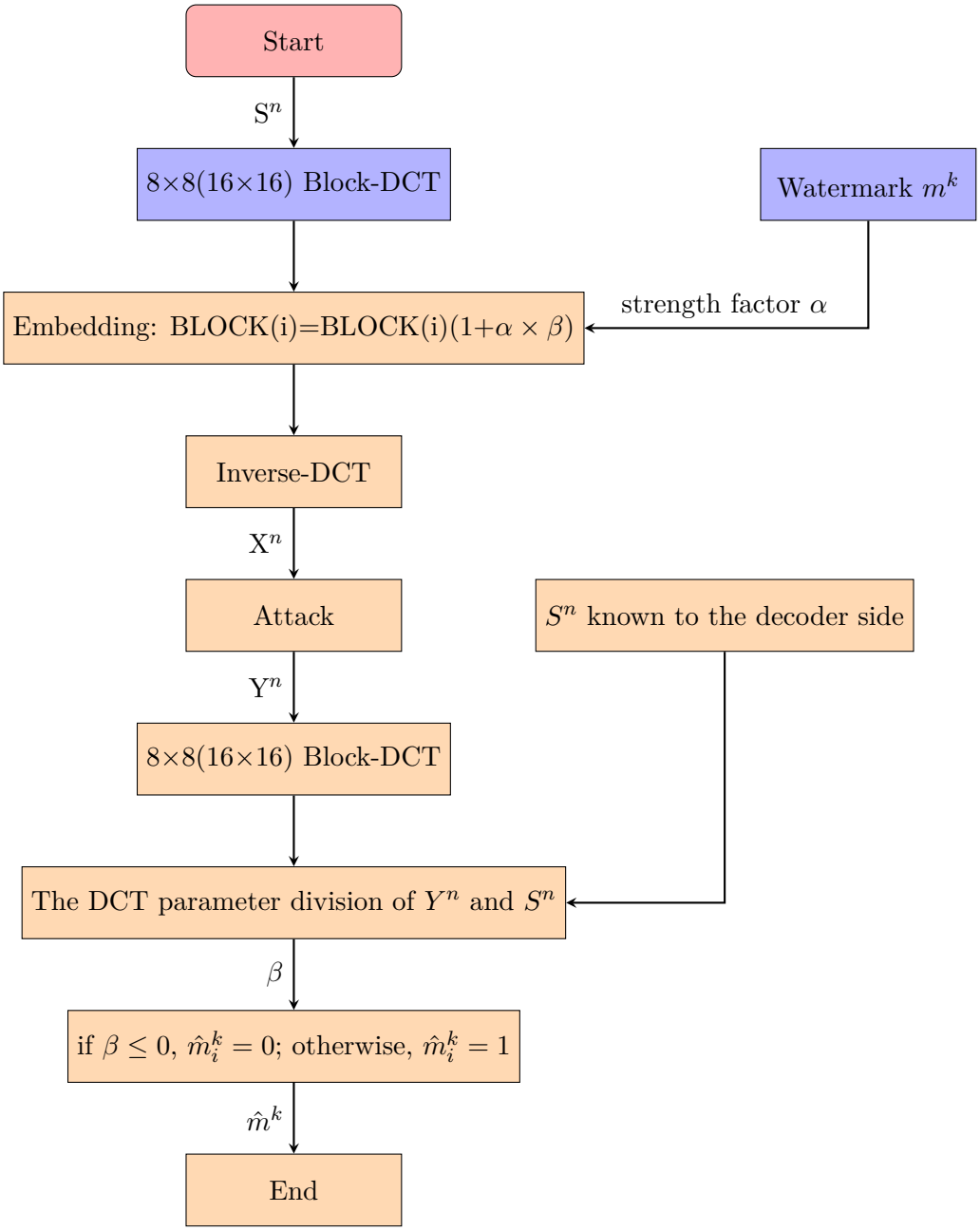

FIGURE 2.4: Discrete Cosine Transformation Watermarking Scheme

2.4.2 Discrete Wavelet Transformation(DWT) Scheme

Discrete Wavelet Transformation (DWT) watermarking scheme is an algorithm that has a similar working principle to the DCT process. It indicates excellent performance in resisting JPEG compression.

The principle of DWT is stated as the following. It decomposes the cover image S^n into several sub-images with different energy levels. Each DWT level can decompose S^n into four sub-images, LL, LH, HL, HH, where LL contains the low-frequency information and all other three sub-images contain horizontal, vertical and diagonal components of the image respectively. During the frequency-related attacks, almost all information in the LH, HL, and HH domains will be eliminated or contaminated. Because these regions specify the details of the images, and only the information in the LL sub-image can be preserved. Consequently, it can be an excellent choice to embed the watermark message into the LL domain. However, if the watermark message is directly embedded into the first level LL_1 sub-image, the imperceptibility performance will be reduced despite some strength factor α.

Some researchers find out that it is better to embed the watermark into the high-level DWT sub-bands of the image. Kundur and Hatzinakos (Kundur and Hatzinakos 1998) embedded the binary watermark into the high-frequency sub-images while W. Zhu et al. (Wenwu Zhu et al. 1999) and N.Kaewamnerd et al. (Kaewamnerd and Rao 2000) both introduced a blind watermarking scheme that embeds the Gaussian-distributed sequence as a watermark into the high-frequency sub-images with different coordinates parameters. Wang and Lin (Shih-Hao Wang and Yuan-Pei Lin 2004) proposed a more complicated DWT algorithm that uses the wavelet tree to process quantization to embed the watermark into the high-frequency sub-images of the high-level DWT. Lin together with the other five researchers (Lin et al. 2008) proposed another efficient DWT-based algorithm that embeds the watermark into the LH_3 sub-image of the

third-level DWT.

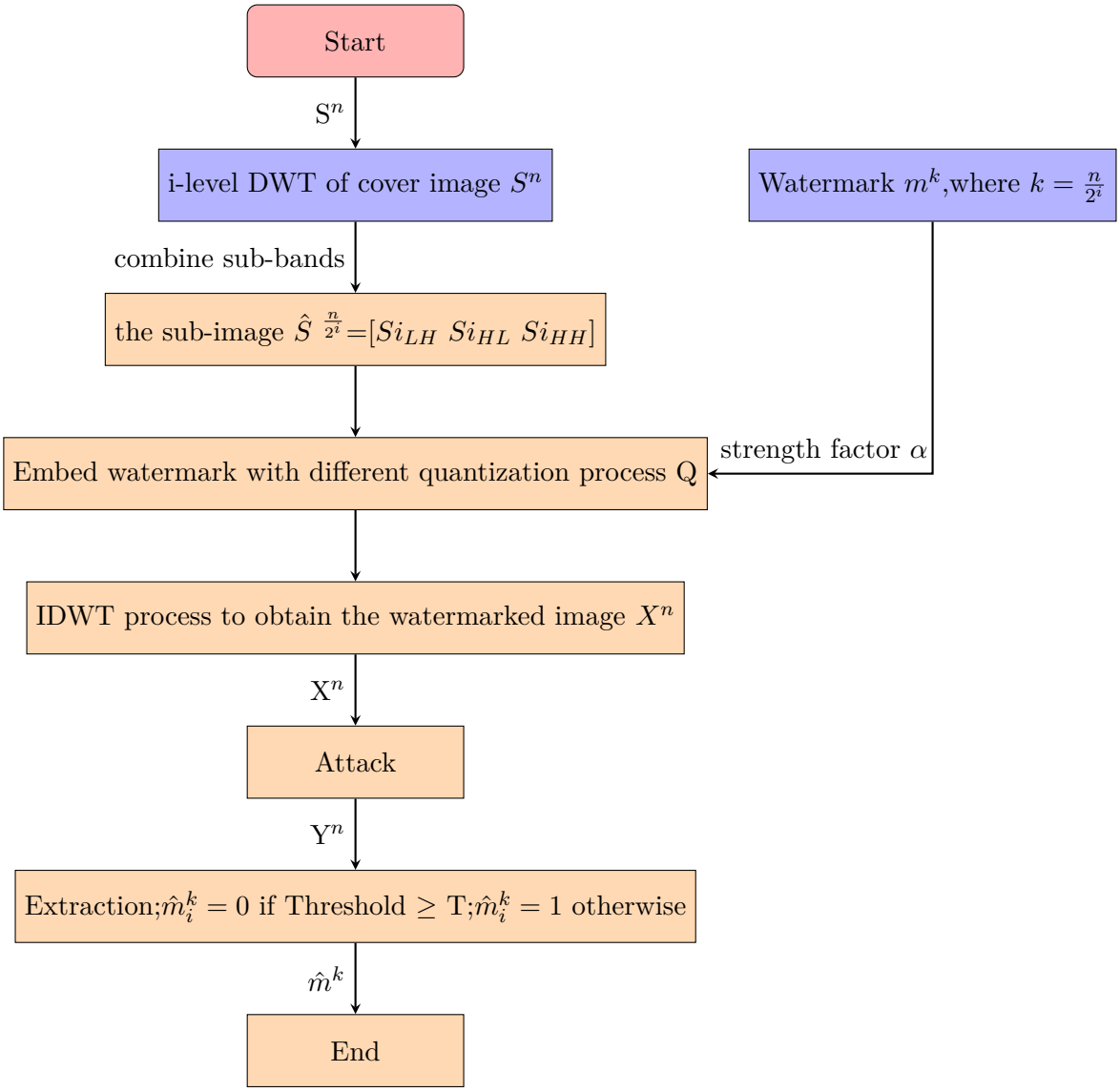

FIGURE 2.5: Discrete Wavelet Transformation Watermarking Scheme

The embedding process varies concerning different modifications, but the extraction process remains similar. The critical point is to find out a proper threshold value to determine whether the particular pixel is watermarked or not (Lin et al. 2008). This threshold value T changes concerning the different quantizations during the embedding process.

Figure 2.5 shows a general working scheme of level-one DWT and α is the controlling factor of the watermark strength. Q_1 and Q_2 are two quantization process defined by different papers. T is the threshold value only related to the quantizations Q_1 and Q_2. Quantization step in Figure 2.5 is as the following:

$$\mathrm{X}_i^n = \begin{cases} Q_1(m_i^k, S_i^{\frac{n}{2^i}}), & m_i^k = 1 \\ Q_2(m_i^k, S_i^{\frac{n}{2^i}}), & m_i^k = 0 \end{cases} \quad (2.1)$$

where $k = \frac{n}{2^i}$,

The limitation of the DWT watermarking scheme is straightforward. Although it can reach good robustness and good imperceptibility under many different frequency attacks, the embedding rate is limited during the DWT process. The better the performance, the smaller the embedding rate (Lin et al. 2008).

2.5 Deep Learning Related Watermark Algorithm

Neural Networks (NN) has become increasingly popular these days since it has the outstanding capability on data processing.

When it comes to the digital image watermarking problem, many researchers have been trying to build up an efficient neural network to solve this problem since 2007. Jin and Wang (Jin and Wang 2007) built up a non-blind watermarking network combined with the DCT process to test the embedding strength of the watermark

in certain situations. Then, several years later, Zhu et al.(Zhu et al. 2018) successfully built up a blind-watermarking system with high imperceptibility and robustness under several kinds of attack. This sub-section will give a brief introduction to HiDDeN Network created by Zhu et al. (Zhu et al. 2018). Chapter 4 will show the test results over the chosen data-set. The general architecture of HiDDeN is

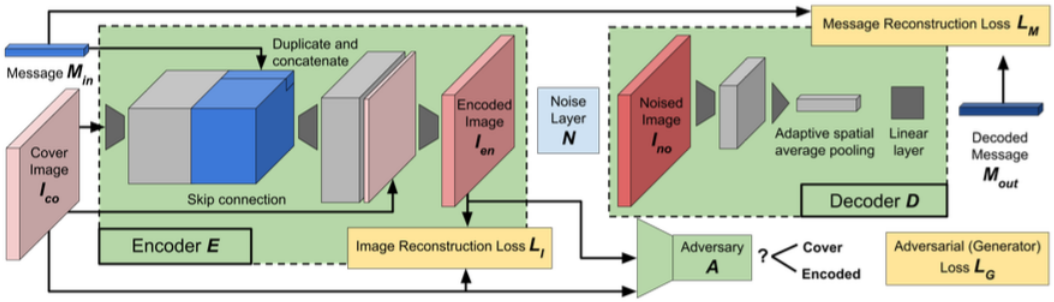

FIGURE 2.6: HiDDeN Model(Zhu et al. 2018)

shown in Figure 2.6. It consists of several parts. The two key parts are the encoder network and decoder network. The encoder network is responsible for embedding watermark messages into a given input image, while the decoder network is responsible for extracting a watermark message from a contaminated watermarked image respectively. Besides, HiDDeN introduced an attack layer in between these two networks to simulate different kinds of attacks (e.g. JPEG, BLUR, CR, DROP). The attack layer received the image from the encoder network and sent the output image to the decoder network. After that, to enhance the quality of the image, HiDDeN introduced another GAN network to make the output image more reasonable concerning Human Visual System (HVS).

The principle of watermark embedding and extraction in HiDDeN is straightforward. For the digital watermark embedding process, it directly embeds the watermark by convolution through three convolutional layers (Zhu et al. 2018), each completes convolution, pooling,

and non-linear activation (ReLU). For the extraction process, the decoder network receives the encoded image and works together wither the adversary network to extract the watermark message from the encoded image.

The key structure of HiDDeN is the Generator and Adversary Network (GAN), which can enhance the image quality. HiDDeN first introduces a discriminator D to distinguish between the cover image and the watermarked image. For each batch of training, GAN computes and updates one group of loss, called the adversary loss L_{adv}. Then the generator G in GAN utilizes the three losses L_{decode}, L_{encode} and L_{adv} to get a weighted sum of loss L_{total}. The converge signal would be the value of L_{total} smaller than a super-parameter threshold T.

$$L_{total} = g_{adv} \times L_{adv} + g_{encode} \times L_{encode} + g_{decode} \times L_{decode}$$

All the g values are the gradients, which are continuously updated through batch iterations. For ideal situation, only the adversary loss is comparatively large and all other losses will be close to zero, which is reasonable since $g_{adv} \times L_{adv}$ represents binary cross entropy loss with logistic of how generator G fools discriminator D to distinguish one image (e.g. D thinks the image is encoded (encoded=0) while G considers the image is not (cover=1)).

HiDDeN is a typical example of how neural networks can successfully deal with the digital watermarking problem and it indicates a lot of improvements on imperceptibility and robustness. However, there is still some space for improvement since the embedding rate $\mathcal{R}$ is pretty low compared with the classical methods mentioned before and our proposed methods which will be introduced in Chapter 3.

Chapter 3

Proposed Watermarking Scheme

3.1 Background Information

3.1.1 Shannon's Theorem

In 1948, Claude E. Shannon published an landmark paper (Shannon 1948) that proves the limitation of transmission rate concerning a given channel. He stated the fact that given a communication channel, it is only possible to transmit digital signal correctly without any error as long as the transmission rate R is not over the channel capacity C. Namely, the capacity of this channel would be the upper limit for the transmission rate. Notice that Shannon also stated in this paper (Shannon 1948) that although this upper bound is proved through strict mathematical demonstration, it does not provide the explicit coding construction.

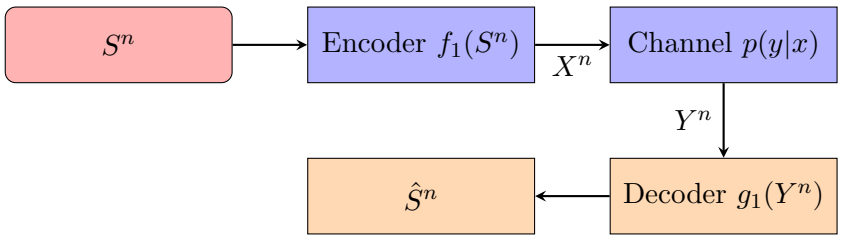

FIGURE 3.1: Communication System

The general mathematical model for the communication system is as shown in Figure 3.1. A source sequence S^n can be encoded to a proper channel input sequence X^n concerning the encoding function

$f_1(S^n)$. Then, the X^n is sent through a noise channel with transmitting probability $p(y|x)$ and get Y^n. $p(y|x)$ is a fixed characteristic that is defined by the specific noisy channel. The decoder with decoding function $g_1(Y^n)$ receives Y^n and successfully decodes to get the approximated source sequence $\hat{S}^n$. The mutual information can represent the information capacity of a particular channel with fixed characteristics, while the channel capacity C is the maximum reliable information that can be sent per channel use.

$$C = sup_{p_x(x)} I(X;Y),$$

where the *sup* considers all possible values of $P_x(x)$ Since the mutual information can also be represented by probability and entropy, the formula can then change to the following.

$$I(X;Y) = \sum_{x,y} P_{XY}(x,y) log \frac{P_{XY}(x,y)}{P_X(x)P_Y(y)},$$

where $P_X(x)$ and $P_Y(y)$ are the marginal probabilities, and $P_{XY}(x,y)$ is the joint probability of x and y. The unit of information capacity is determined by the base of the logarithm. For most of the cases, the source signal is binary. Consequently, it would be log2 computation, and the unit would be bits.

There has been many classical algorithms that perform encoding and decoding functions. In our proposed method, low-density parity-check (LDPC) code together with its dual code low-density generator matrix (LDGM) code are used to fulfill the task.

3.1.2 LDPC Codes

General Introduction and Representation

Low-density parity-check (LDPC) code is one of the highly efficient block codes, which was first introduced by Robert G. Gallager (Gallager 1962a) in his Ph.D. dissertation at M.I.T in 1962. At the time it was first introduced to the world, this code seemed to be impractical

due to the computing limitation. Consequently, It remained to be ignored by people for over 35 years. Thanks to the fast development of computational technology, LDPC code is rediscovered by a group of researchers nowadays. People find out that the LDPC code has a reasonably good performance, which allows the code rate R to get very close to the Shannon limit with very low bit error rate.

When it comes to the LDPC code, it is necessary to talk about its fundamental components. The critical parts of LDPC code are the parity check matrix H and its corresponding generator matrix G. The parity check matrix is a sparse binary matrix with only a few numbers of "1" inside, while the generator matrix is not guaranteed to be sparse, but its row-space has to be the null space to its parity check matrix. The relationship between H and G is the following:

$$GH^T = 0^T.$$

The general procedure of the LDPC code coding scheme is shown in Figure 3.2. The input message m_{input} with length k after the multiplication with the generator matrix G with size $k \times n$ would produce a codeword x^n with length n.

$$x^n = m_{input}G.$$

The codeword x^n is then the input of the channel. After x^n goes through the noisy channel, y^n is received at the decoder side. For a binary sequence, when it goes through a noisy channel, it will experience some bit flips. Consequently, it is the decoder's job to figure out which codeword this y^n sequence represents. Mathematically speaking, if a sequence $\hat{x}^n$ is a codeword, then it has to satisfy the equation as the following:

$$H\hat{x^n}^T = 0^T.$$

Notice the fact that the row space of G matrix must be orthogonal to its corresponding parity check matrix H. However, there are

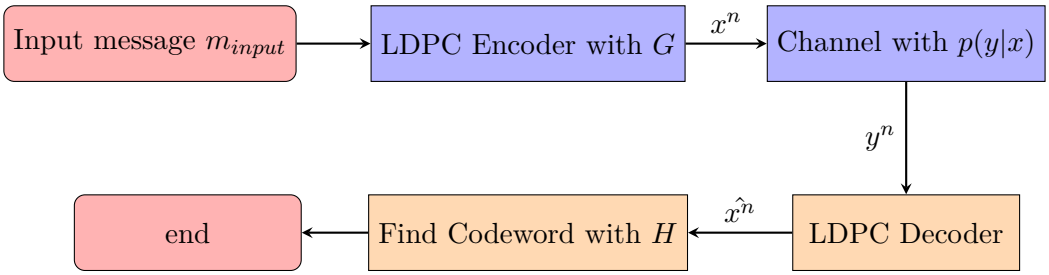

FIGURE 3.2: LDPC Scheme

more than one generator matrix G that can satisfy the relationship. Besides, for the LDPC codes, for most of the time, the size of the matrix can be extremely huge ($n \geq 10000$). Consequently, it may not be efficient to get a G matrix directly for the encoding part. The efficient method will be introduced in a later section.

Matrix Construction Methods

This section will give an introduction of two general methods to construct a functional parity check matrix.

The first method was invented by Gallager in his paper back to the early 1960s (Gallager 1962b). His approach focused on how to construct a regular parity check matrix.

Definition 1 *The **weight** indicates the number of non-zero elements in a set and since the set is a sparse binary matrix, then the **weight** is the number of ones in the matrix. Consequently, the row weights $\boldsymbol{w}_r$ refers to the number of ones in a particular row, while the column weights $\boldsymbol{w}_c$ indicates how many ones in a specific column.*

Definition 2 *Parity check matrix H with row weight $\boldsymbol{w}_r$ and column weight $\boldsymbol{w}_c$, if the value of w_r and w_c remain constant, it is called the **regular** LDPC code. The ($\boldsymbol{w}_r$,$\boldsymbol{w}_c$) pair is the **degree distribution**.*

Supposed the parity check matrix H has the size M $\times$ N (M rows and N columns in total), the column weight is set to be w_c and then H can be divided into $\frac{M}{w_c}$ number of sub-matrices and each of

size $\frac{M}{w_c}$. First of all, the first submatrix is constructed in an echelon-formed with w_r number of ones each row, respectively. After the first submatrix is completed, the rest of the submatrices are constructed with column permutation of the first submatrix. The example is shown in Figure 3.3.

$$H = \left[\begin{array}{cccccccccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right].$$

Figure 3.3: A regular(12,9) H Matrix

In this example, the column weight w_c is 3 and the row weight w_r is 4. In total, it is a regular- 9×12 parity check matrix.

Notice that when Gallager firstly introduced LDPC to the world, there were no efficient encoding and decoding algorithms that can work with the LDPC matrix to get the experimental results. When decades year later, Mackay (MacKay 1999) and some researchers rediscovered the excellent coding performance of the LDPC code, they started to consider the computational efficiency in constructing a functional parity check matrix.

The second method was introduced by D.J.C Mackay et al. (MacKay 1999) when they rediscovered the Gallager code in the late 1990s. This method aimed to construct a very sparse parity check matrix H that allows the code to be not systematic, since non-systematic code can give us more optimal results.

Definition 3 *For a parity check matrix with size $M \times N$, if the message bit can be directly read from the codeword x^n, this code is the systematic code.*

For a parity check matrix with size M×N, the source length is M, and the block code length is N. Consequently, the resulting message bit has the length K, where $K = N - M$. To construct a very sparse parity check matrix with size $M \times N$, a rectangular matrix A is first constructed. A is supposed to be uniform for the row weight w_r, and the column weight w_c is a pre-defined value t ($t \geq 3$). After that, A is partitioned into two submatrices. While C_1 is a very sparse matrix with size M×K and C_2 is an invertible square matrix with size $M \times M$. According to the standard form of H and G, where

$$H = [P|I_m],$$

$$G^T = [I_K|P^T],$$

P matrix is then defined to be $P = C_2^{-1}C_1$. Consequently, the final construction result would be:

$$H = [C_2^{-1}C_1|I_m],$$

$$G^T = [I_K|(C_2^{-1}C_1)^T],$$

Notice that the time complexity of the process mostly comes from the matrix inversion computation. Since the inversion is done for the C_2 matrix, which has the size M×M, the overall time complexity of computing $P = C_2^{-1}C_1$ matrix would be M^2N (MacKay 1999)

Encoding algorithm

This section will give an introduction of an efficient method to complete LDPC encoding without generating the G matrix. (Richardson and Urbanke 2001a)

According to the definition of LDPC coding, a source message is encoded to a valid codeword by doing matrix multiplication with the generator matrix G. However, for most of the cases, the computational cost of getting the proper G matrix from the corresponding H matrix is exceptionally high. Although the sparseness of the H

matrix saves much computational time, there is no guarantee of the sparseness of the generator matrix G. Consequently, people need to figure out a method that can maintain the sparseness and allow the encoding step to be as efficient as possible.

Notice that there is a prerequisite that needs to meet for this encoding algorithm (Richardson and Urbanke 2001a) to work correctly. The parity check matrix H is assumed to be full-rank. The working

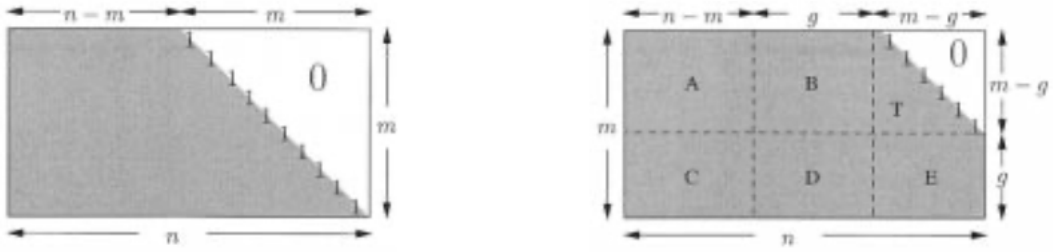

FIGURE 3.4: Example Matrix H and its Approximated form H_t (Richardson and Urbanke 2001b)

procedure is shown in the following. In Figure 3.4, assuming that there as a full-rank parity check matrix H on the left-hand-side with size $m \times n$ and its upper-right corner can form a lower triangular matrix with size $m \times m$. After doing permutations and column switches, it is possible to get an approximate lower triangular matrix H_t on the right-hand side. From the image of H_t, the value **g** is a pre-defined value called **gap**. The smaller the gap, the faster the computation. After the partition, the parity check matrix H is being divided into six sub-matrices.

Here are the requirements for the partition. First, the T matrix in the upper-right corner must be a lower triangular matrix. Besides, the value of $ET^TB + D$ must be non-singular.

The Mathematical process is shown in the following:

$$H = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix}$$

Assume that the T matrix is a lower triangular matrix with no non-zero elements above the diagonal elements. The codeword x is assumed the satisfy the equation

$$Hx^T = 0.$$

If H is systematic, the message bits can be directly read from the codeword. Consequently, let

$$x = (s|p_1|p_2),$$

for p_1 and p_2 are parity check bits and s is the source bit(message bit).

After that, set another parity check matrix $\hat{H}$ being the linear transform of H, which satisfy the equation

$$\hat{H} = \begin{bmatrix} I_{m-g} & 0_g \\ -ET^T & I_g \end{bmatrix} \times H$$

Consequently,

$$\hat{H} = \begin{bmatrix} A & B & T \\ -ET^TA + C & -ET^TB + D & 0 \end{bmatrix}$$

$$\begin{bmatrix} A & B & T \\ -ET^TA + C & -ET^TB + D & 0 \end{bmatrix} \times x^T = 0^T$$

since $x = (s|p_1|p_2)$,

$$As^T + Bp_1^T + Tp_2^T = 0$$

$$(-ET^TA + C)s^T + (-ET^TB + D)p_1^T = 0$$

Define two values α and β,

$$\alpha = -ET^TB + D$$

$$\beta = -ET^TA + C$$

If α is invertible,

$$p_1^T = -\alpha^{-1}\beta s^T$$
$$p_2^T = -T^{-1}(As^T + Bp_1^T)$$

Notice that, for the equation on calculating p_2, since T is a lower triangular matrix, a forward substitution can be applied to easily solve the equation. Finally, the codeword x^n is solved.

Moreover, as long as the proper H matrix can form a lower triangular matrix T at the right-top corner and the ϕ is invertible, it is possible to convert it to a G matrix. According to the example that stated above,

$$H = \begin{bmatrix} A & B & T \\ C & D & E \end{bmatrix}$$

where T is a lower triangular matrix and D needs to be as small as possible.

We have already defined α and β. Suppose the parity check matrix has the size $m \times n$, then the corresponding generator matrix is given by,

$$G^T = \begin{bmatrix} I_m \\ \phi^{-1}\beta \\ T^{-1}(A + B\phi^{-1}\beta) \end{bmatrix}$$

where the G matrix is a systematic matrix.

All the steps above have been tested in MATLAB and more details for this procedure can be referred to the paper (Richardson and Urbanke 2001a).

Decoding algorithm

The decoding part is more complicated than the encoding part. There are many efficient algorithms, and all the algorithms aim to find the proper codeword $\hat{x}^n$ concerning the received signal y^n, which

is contaminated by channel noise. This section will give a brief introduction to one of the message-passing algorithms called Belief Propagation (BP)(Kschischang et al. 2001).

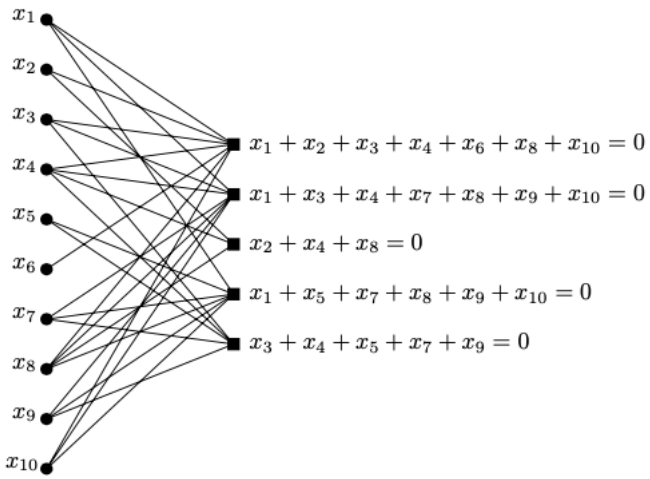

FIGURE 3.5: Tanner Graph (Shokrollahi 2003)

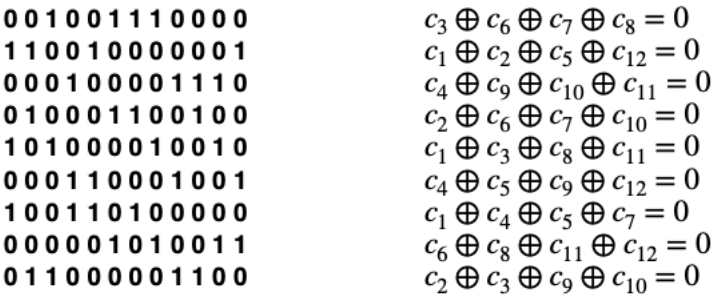

FIGURE 3.6: Relationship between constraints and H matrix

Before discussing the property and working procedure of BP, it is necessary to introduce a useful graph called the Tanner Graph 3.5, which was first invented by R. Michael Tanner (Tanner 1981) in 1981. Tanner graph is the crucial element that makes iterative decoding possible. It is much more straightforward for people to learn the functionality of BP through the Tanner Graph. The relationship between the parity check matrix H and its corresponding nodes

present in the Figure 3.6. On the left, it is a parity check matrix H with size 9×12. Each row of the H matrix represents a parity check equation, and each column indicates a codeword bit. The number of check equations represents the number of constraints that a valid codeword x^n needs to satisfy.

Then, combined with the sample Tanner graph in Figure 3.5, the circle indicates variable nodes, and the square symbolizes the check nodes. Each variable node defines a codeword bit, and each check node interprets a parity check constraint. When there is an *edge* connect the variable node i and the check node j, it indicates a **one** with coordinates (j, i) (j-th row and i-th column) in the parity check matrix H.

The message-passing algorithm plays a crucial role in the decoding part. It operates as passing the message through the *edge* iteratively backward and forward between the variable nodes and the connected check nodes. The message refers to the probability of a certain variable node is one or zero at a particular iteration. The overall probability of a single variable node is the sum of the product of all the sub-probabilities from the related check nodes, and this multiplication process has a high time-complexity. Consequently, it is more friendly to use log-ratio to represent this probability.

$$L(x) = log(\frac{p(x=0)}{p(x=1)}),$$

where the logarithm computation is processed with base e. In this equation, when $p(x = 0) > p(x = 1)$, the value of $L(x)$ will be a positive number. When $p(x = 1) > p(x = 0)$, the value of $L(x)$ will be a negative number instead. In total, when the message is a **Log-Likelihood Ratio (LLR)**, the algorithm is referred to as the Belief Propagation.

There are two kinds of probabilities that need to mentioned before

starting. The *prior probability* $\mathbf{r}_i$ of the bit means the bit's probability before the iteration, and the *posterior probability* of the bit indicates the probability values that being updated all the time through the whole iteration. Notice the fact that the *prior probability* r_i is a channel-defined value.

The program for the Sum-product decoding is shown as the following. Let set $c_{i,j}^t$ is the check node message at the t-th iteration and with all related variable node j; $v_{i,j}^t$ is the variable node message at the t-th iteration with all related check node j; L_i is set to be the LLR values; $\hat{x}$ is the decoded codeword bit. The process is as the following:

Step 1: Initialization of all the variable node with the prior probability r_i, so that

$$v_{i,j}^0 = r_i.$$

Step 2: Update the message for each check node with the sum-of-product of the LLR values,where B_j is a check set at j.

$$c_{i,j} = log(\frac{1 + \prod_{i' \subseteq B_j, i' \neq i} tanh(\frac{v_{j,i}}{2})}{1 - \prod_{i' \subseteq B_j, i' \neq i} tanh(\frac{v_{j,i}}{2})}).$$

Step 3: Calculate the log-likelihood ratio LLR_i for each variable node and decide whether the decoded codeword $\hat{x}_i$ is zero or one, where A_i is a variable set at i.

$$LLR_i = \sum_{j \subseteq A_i} c_{i,j} + r_i.$$

$$\hat{x}_i = \begin{cases} 1, & LLR_i \leq 0 \\ 0, & LLR_i > 0 \end{cases} \tag{3.1}$$

Step 4: Check if the process reach the maximum iteration I_{max} or $\hat{x}$ satisfy all the constraints in H, that is $H\hat{x}^T = 0$. If yes, process finishes. Otherwise, go to step 5.

Step 5: Update the variable bit state $v_{i,j}^{t}$ with the previous state $c_{i,j}^{t-1}$ and the *prior probability* r_i.Then, continue step 2 and 3 until the whole process meet the converge condition.

$$v_{i,j}^{t} = \sum_{j' \subseteq A_i, j^i \neq j} c_{i,j}^{t-1} + r_i.$$

In conclusion, all the presented steps refer to the paper (Johnson 2000), and they have been realized in the preparation step of this thesis to demonstrate its functionality. For short-length source vector and small size H matrix, the maximum iteration I_{max} can always be small ($I_{max} \leq 100$). However, if the size of the H reaches 10,000 for the row number, I_{max} ($I_{max} \geq 300$) needs to be enlarged.

3.1.3 LDGM Codes

LDGM Code Representations

This section will provide a brief introduction to the general representations of LDGM code. The low-density generator matrix (LDGM) code is fashionable and highly practical in the area of data hiding and data compression due to its excellent performance on the rate-distortion result. LDGM code is a dual code of the LDPC code, but it has a similar working scheme. It is also practical to use the Tanner graph to represent and show its functionality.

Assume that there is a generator matrix G for a particular LDGM code, and it has the size $m \times n$. The size of the G matrix specifies the number of information bits and the number of check bits. For the example above, it indicates that this G matrix has m information bits and n check bits. Each row of G represents an information bit, while each column represents a constraint. It is similar to the LDPC code.

Mathematically speaking, for a generator matrix $G \subseteq \{0,1\}^{m \times n}$, where $n > m$, there are three sets of variables that need to be updated and keep tracking through the whole process, let set $\mathcal{G}=(\mathcal{V}, \mathcal{C}, \mathcal{E})$.

$\mathcal{V}$ represents the a set of information bit where $\mathcal{V} = \{1, 2, 3, ..., n\}$, while $\mathcal{C}$ represents a set of check bits where $\mathcal{C} = \{1, 2, 3, ..., m\}$. There is also another group of bits that is not presented here called the **source bit**. The source bit and the check bit has a one-to-one correspondence relationship. The last group $\mathcal{E}$ represents the edge, which indicates the connections between certain check bit and information bit. Suppose $\{a, b, c\} \subseteq \mathcal{C}$ and $\{i, j, k\} \subseteq \mathcal{V}$, if $(b, j) \subseteq \mathcal{E}$, then it is clear that $G(b, j) = 1$.

LDGM Coding Scheme

This section will introduce the LDGM coding scheme concerning data hiding. Since the LDGM code is the dual code of LDPC code, the working scheme appears to be similar. According to Figure 3.7, the source bit node is presented on the top of the graph and each connects to its unique check node $\{a, b, c, d, e, f\} \subseteq \mathcal{C}$. Then, the information bit is presented at the bottom of the figure, where $\{w_i, w_j, w_k\} \subseteq \mathcal{V}$.

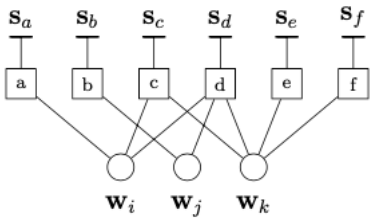

FIGURE 3.7: factor graph of LDGM
(Filler and Fridrich 2007)

Mathematically speaking, suppose there is a binary input message $\mathbf{m}_{input}$ with length m and it can be encoded with the generator matrix G, where $G \subseteq \{0, 1\}^{m \times n}$. Namely, the codeword x^n is defined as,

$$x^n = m_{input} \times G.$$

After that, the decoding process is simply the inverse of the encoding step.

Notice the fact that the most challenging part of the LDGM code is the encoding process. Since the LDGM code is the dual code of the LDPC code, people may think straight ahead to apply Belief propagation (BP) to find the closest codeword to represent the source sequence. However, it will lead to terrible results. It is explained in paper (Fridrich and Filler 2007) that since the input binary message is randomly generated, it is nearly impossible for the encoder to find a codeword x within proper error range or within the maximum iteration time. As the Belief Propagation (BP) has one crucial prerequisite that the input sequence S must be encoded very close to a present codeword x_i^n within the codebook $\mathcal{C}$ concerning the parity check matrix H. Consequently, the LDGM code needs to introduce another practical and highly efficient propagation algorithm called Survey Propagation (SP).

Survey Propagation (SP)

Survey propagation (SP) is another highly efficient algorithm to find the proper codeword, especially for the situation that the input sequence s^m is randomly generated. According to the statement in paper (Fridrich and Filler 2007), for a randomly generated vector s^m, it is nearly impossible to find an optimal codeword x^n within an acceptable Hamming Distance. It can result in the failure of converging or no proper codeword, which implies that it is not practical to employ Belief Propagation (BP) as the appropriate algorithm to complete the encoding step in the LDGM code. Since there is an essential prerequisite for BP to work appropriately – the input sequence must be already very close to one of the valid codewords. Consequently, another propagation algorithm requires to be offered in the embedding step.

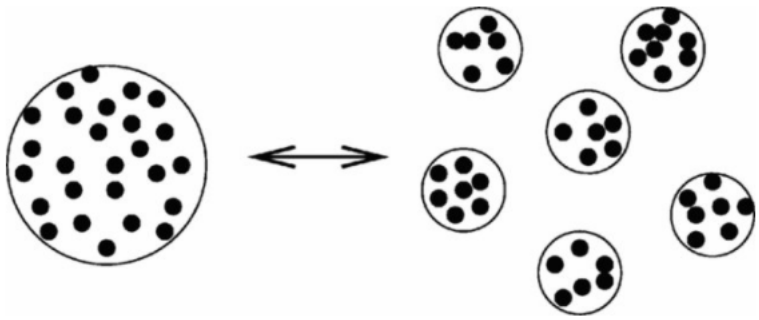

Figure 3.8: Survey Propagation with Cluster
(Weigt and Zhou 2006)

To solve this problem, first of all, the codewords are divided into several non-overlapping clusters as shown in Figure 3.8, and each has several codewords. The target of the Survey Propagation (SP) is to find the proper *cluster* that has the highest possibility to contain the target codeword (Fridrich and Filler 2007). After that, it is then possible for BP to find the closest codeword within that cluster.

The overall working procedure is similar to that of Belief Propagation (BP). The log-likelihood ratio (LLR) is utilized as the *message* to transmit and update among source nodes $\mathbf{s}$, check nodes $\mathbf{c}$ and the information (variable) nodes $\mathbf{v}$. First of all, the check nodes and information nodes process the message and send it to their connected check nodes. Then, the check nodes process the message and send the LLR values back to the information nodes. These two steps iterate continuously until their converge condition is satisfied, and they are called the message-passing step. After it converges, to simplify the original factor graph $\mathcal{F}$, the decimation step is implemented so that the information nodes with the highest bias value and its corresponding edges will be removed from the graph. Notice that if the related check nodes have the check degree decreases to zero, they will also be removed from the graph. Then, the new factor graph

$\mathcal{F}$ [1] can start the next iteration for both message-passing and decimation. After r iterations with factor graph $\mathcal{F}^r$, if the bit value of all the information nodes are fixed or the algorithm reaches the predefined maximum iterations, the process end. In total, the critical part of the process is to fix the bit value of information nodes as fast as possible. The step that transmits message among nodes is called message-passing, while the step that simplifies the factor graph is called decimation (Fridrich and Filler 2007).

The survey propagation passes five variables as the message in total, which can be presented as

$$\mathbf{M}_{i\to a}^{(l)} = (M_{i\to a}^{0f(l)}, M_{i\to a}^{1f(l)}, M_{i\to a}^{0w(l)}, M_{i\to a}^{1w(l)}, M_{i\to a}^{*(l)}),$$

Mathematically speaking, the overall process can be presented as the following. Assume that $\{a, b, c\} \subseteq \mathcal{C}$ and $\{i, j, k\} \subseteq \mathcal{V}$,$M_{i\to a}^{(l)}$ indicates the message passing from information node i to check node a in the l_{th} iteration. The meanings of the five messages are discussed in paper (Wainwright and Maneva 2005).

Step1: Initialization of all the check nodes $\mathcal{C}$ with the corresponding source nodes $\mathcal{S}$, for $\mathcal{S} \subseteq \{0,1\}^n$. The number of check nodes equal to the number of source nodes.

$$M_{s_a\to a}^{(l)} = (\phi_a(0), \phi_a(1), 0, 0, \omega_{sou}),$$

$$\phi_a(1) = s_a e^{\gamma} + (1 - s_a)e^{-\gamma},$$

$$\phi_a(1)\phi_a(0) = 1,$$

where w_{sou} and γ are experimental values that need to be decided. (Fridrich and Filler 2007)

Step2: Information nodes to check nodes updates

$$M_{i\to a}^{0f(l)} = \prod_{b\subseteq C_i\setminus\{a\}} [\, M_{b\to i}^{0f(l-1)} + M_{b\to i}^{0w(l-1)}] - \prod_{b\subseteq C_i\setminus\{a\}} M_{b\to i}^{0w(l-1)} \tag{3.2}$$

$$M_{i\to a}^{1f(l)} = \prod_{b\subseteq C_i\backslash\{a\}} [\, M_{b\to i}^{1f(l-1)} + M_{b\to i}^{1w(l-1)}] - \prod_{b\subseteq C_i\backslash\{a\}} M_{b\to i}^{1w(l-1)} \tag{3.3}$$

$$M_{i\to a}^{0w(l)} = \prod_{b\subseteq C_i\backslash\{a\}} [\, M_{b\to i}^{0f(l-1)} + M_{b\to i}^{0w(l-1)}] - \prod_{b\subseteq C_i\backslash\{a\}} M_{b\to i}^{0w(l-1)} - \sum_{c\subseteq C(i)\backslash\{a\}} M_{c\to i}^{0f(l-1)} \prod_{b\subseteq C(i)\backslash\{a,c\}} M_{b\to i}^{0w(l-1)} \tag{3.4}$$

$$M_{i\to a}^{1w(l)} = \prod_{b\subseteq C_i\backslash\{a\}} [\, M_{b\to i}^{1f(l-1)} + M_{b\to i}^{1w(l-1)}] - \prod_{b\subseteq C_i\backslash\{a\}} M_{b\to i}^{1w(l-1)} - \sum_{c\subseteq C(i)\backslash\{a\}} M_{c\to i}^{1f(l-1)} \prod_{b\subseteq C(i)\backslash\{a,c\}} M_{b\to i}^{1w(l-1)} \tag{3.5}$$

$$M_{i\to a}^{*(l)} = \omega_{info} \prod_{b\subseteq C(i)\backslash\{a\}} M_{b\to i}^{*(l-1)} \tag{3.6}$$

Step3: Check nodes to information nodes update

$$M_{a\to i}^{0f(l)} = \frac{1}{2}\Big(\prod_{j\subseteq \hat{V}(a)} [\, M_{j\to a}^{0f(l)} + M_{j\to a}^{1f(l)}] + \prod_{j\subseteq \hat{V}(a)\backslash\{i\}} [\, M_{j\to a}^{0f(l)} - M_{j\to a}^{1f(l)}]\Big) \tag{3.7}$$

$$M_{a\to i}^{1f(l)} = \frac{1}{2}\Big(\prod_{j\subseteq \hat{V}(a)} [\, M_{j\to a}^{0f(l)} + M_{j\to a}^{1f(l)}] - \prod_{j\subseteq \hat{V}(a)\backslash\{i\}} [\, M_{j\to a}^{0f(l)} - M_{j\to a}^{1f(l)}]\Big) \tag{3.8}$$

$$M_{a\to i}^{0w(l)} = \prod_{j\subseteq \hat{V}(a)\backslash\{i\}} [\, M_{j\to a}^{*(l)} + M_{j\to a}^{1w(l)} + M_{j\to a}^{0w(l)}] - \omega_{sou} \prod_{j\subseteq V(a)\backslash\{i\}} [\, M_{j\to a}^{1w(l)} + M_{j\to a}^{0w(l)}] \tag{3.9}$$

$$M_{a\to i}^{1w(l)} = M_{a\to i}^{0w(l)} \tag{3.10}$$

$$M_{a\to i}^{*(l)} = \prod_{j\subseteq V\hat{(a)}\backslash\{i\}} [\, M_{j\to a}^{*(l)} + M_{j\to a}^{0w(l)} + M_{j\to a}^{*(l)}] \tag{3.11}$$

Step4: Iterate step 2 and 3 to meet the converge condition. And step 2 and step 3 are called **message-passing**.

Step5: After the converges, select the information nodes with the highest *bias* to be fixed their bit values and remove these nodes from the factor graph F to reach simplification, this step is called

Decimation. When Step 4 converges, step 5 will calculate the sum-of-product pseudo-marginals B_i (Wainwright and Maneva 2005). The bias values B_i indicates the tendency of a non-setting (not zero or one) information node bit value can be fixed to one or zero. This step then sorts all the non-setting information nodes concerning their B_i values from great to small and then select n_{fixed} number of information nodes to fix their bit values if their B_i values are greater than a threshold value t. Notice that this n_{fixed} should not be larger than a maximum threshold value n_{max}, while it also should be larger than a n_{min} lower bound.

$$\mu_i(0) = \prod_{a \subseteq C(i)} [M_{a\to i}^{0f(l)} + M_{a\to i}^{0w(l)}] - \prod_{a \subseteq C(i)} M_{a\to i}^{0w(l)} - \sum_{b \subseteq C(i)} M_{a\to i} \prod_{a \subseteq C(i)\backslash b} M_{a\to i}^{0w(l)},$$

$$\mu_i(1) = \prod_{a \subseteq C(i)} [M_{a\to i}^{1f(l)} + M_{a\to i}^{1w(l)}] - \prod_{a \subseteq C(i)} M_{a\to i}^{1w(l)} - \sum_{b \subseteq C(i)} M_{a\to i} \prod_{a \subseteq C(i)\backslash b} M_{a\to i}^{1w(l)},$$

$$\mu_i(*) = \omega_{info} \prod_{a \subseteq C(i)} M_{a\to i}^{*(l)},$$

$$B_i = |\mu_i(1) - \mu_i(0)|,$$

where ω_{info}, t, n_{max} and n_{min} are predefined variables.

Step6: Repeat step 2 $\sim$ 5 until the bit values of all the information nodes are fixed. End of the process.

3.2 Proposed Method

3.2.1 SVD-DWT-LDPC Hybrid Scheme

Motivation and Innovation

After the review section, it turns out that there is a the shortage of applying a single classical algorithm to deal with the digital watermarking problem. If a particular algorithm can have high robustness

over frequency-related attacks, then it would not have good performance over spatial-related attacks. And since the LDPC code shows its great capability on error correction, our first proposed method is to combine SVD, DWT, and the LDPC code to deal with digital watermarking problem.

Similar to the method mentioned in Lin's paper (Lin and Wan 2016), our proposed hybrid scheme also utilizes these three algorithms to handle the digital watermarking problem.

However, there are overall three design differences concerning Lin's design. First of all, in Lin's method, they utilized the three sub-band images LL_3, HL_3 and HH_3 from the third level DWT during the process, while in our proposed method, we apply $LL3$ for the procedure. Secondly, in Lin's approach, after decomposing the grayscaled watermark message into eight bit-planes, they used LDPC code to encode the upper four bit-planes. while in our proposed method, we apply the LDPC encoding to all eight bit-planes. The last difference is, in Lin's approach, they utilized the LDPC code with code rate 0.5 with uniformly distributed check node degree and optimal variable node degree distributions that shown below (Lin and Wan 2016)

$$\lambda(x) = 0.30780x + 0.27287x^2 + 0.41933x^6.$$

while in our proposed method, we also utilize the LDPC code with rate 0.5, but the degree distributions are both uniformly distributed.

After comparing the experimental results, we observe that our proposed hybrid method has a better performance on both imperceptibility and robustness compared with Lin's method.

LDPC-SVD-DWT watermarking scheme

The general working scheme is shown in Figure 4.12. First of all, the grayscaled cover image S^n experiences a 3-level DWT process, and the LL_3 components are prepared for the embedding step. In the

meanwhile, we start to process the watermark message. if the watermark message is a grayscaled image, we decompose the watermark message m^k into eight corresponding bit-planes and each with only zero and one inside. After that,m^k experiences the LDPC coding of all the bits in eight bit-planes to get $m^k_{encoded}$. If the watermark message is binary, the LDPC encoding step is similar. Instead of encoding eight bit planes, we just need to encode one bit plane. The output $m^k_{encoded}$ together with LL_3 goes to the next step, which applies SVD with strength factor α to embed the watermark message into the cover image. After that, the inverse-DWT process is applied the get the watermarked image X^n, which is also the attack channel input. After the noise attack, Y^n is obtained. The extraction process is simply the inverse of the embedding process, similar to the process of the SVD watermarking algorithm. Finally, the recovered watermark message $\hat{m}^k$ is obtained at the end.

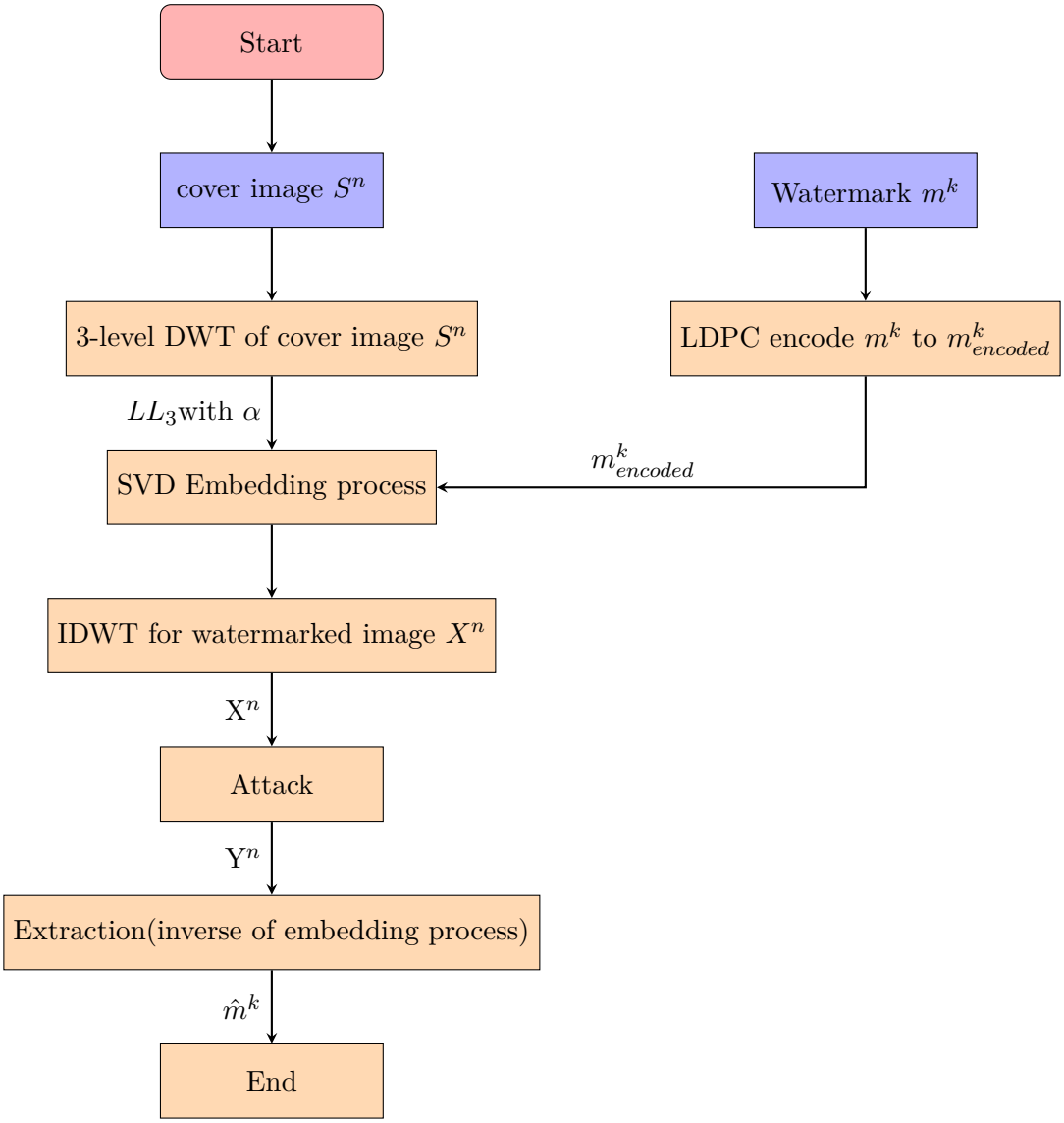

Figure 3.9: Hybrid Watermarking Scheme

The experimental results are shown in Chapter 4.3 . Notice that in the paper (Lin and Wan 2016), it only provides the plots concerning the noise strength (variance) and PSNR values. Compared with the paper, our proposed method modifies the number of bit-planes that are under LDPC encoding and some predefined parameters. Besides, to test the robustness of our proposed hybrid method, two groups of BER test results are also provided in Chapter 4.3.

3.2.2 LDGM-LDPC Nested Coding Scheme

Notation Clarification

This subsection is the extension of notations with respect to the notation clarifications at the very beginning of the thesis. The meaning of S^n, X^n, Y^n, m^k and $\hat{m}^k$ stays the same, which can refer to the clarifications in Section 2.1. Besides, there are some new notations that need to be specified. The auxiliary random variable U^n with length n is a ternary sequence in our case. $\mathcal{C}_{m^k}$ refers to the codebook with respect to the watermark message m^k and u^n is the codeword in codebook $\mathcal{C}_{m^k}$. G_m is the generator matrix with respect to the watermark message m^k.

LDGM-LDPC Nested Coding Scheme

According to Figure 3.10, the source sequence S^n where $S^n \subseteq \{0,1\}^n$ processes an LDGM quantization with generator matrix G_m and followed by decimation step and Gallagher mapping to get a quantized version of S^n, called U^n. In our proposed method, U^n is a ternary code and G_m is the LDGM generator matrix concerning the watermark message m^k. After that, a proper deterministic mapping function f maps S^n and U^n to the jointly typical binary sequence X^n with proper probability distribution. X^n is the channel input sequence and also can be treated as the watermarked image. After channel transmission, the channel output sequence Y^n is received

at the other end, which can be treated as the contaminated watermarked image. Finally, the LDPC decoding applies to find out the proper codebook that contains the proper codeword. Consequently, we can finally recover the watermark message $\hat{m}^k$.

Notice that the number of watermark message equals to the number of the LDGM codes,that is m_i^k for $i \subseteq \{0, 1, 2, ..., 2^{nR} - 1\}$. Since the overall code rate $R = I(U;Y) - I(U;S)$, the length of a certain watermark message m^k is $l_{m^k} = nR$, where n is the length of the source sequence S^n. As a result, for a watermarking system, there will be 2^{nR} number of LDGM codes.

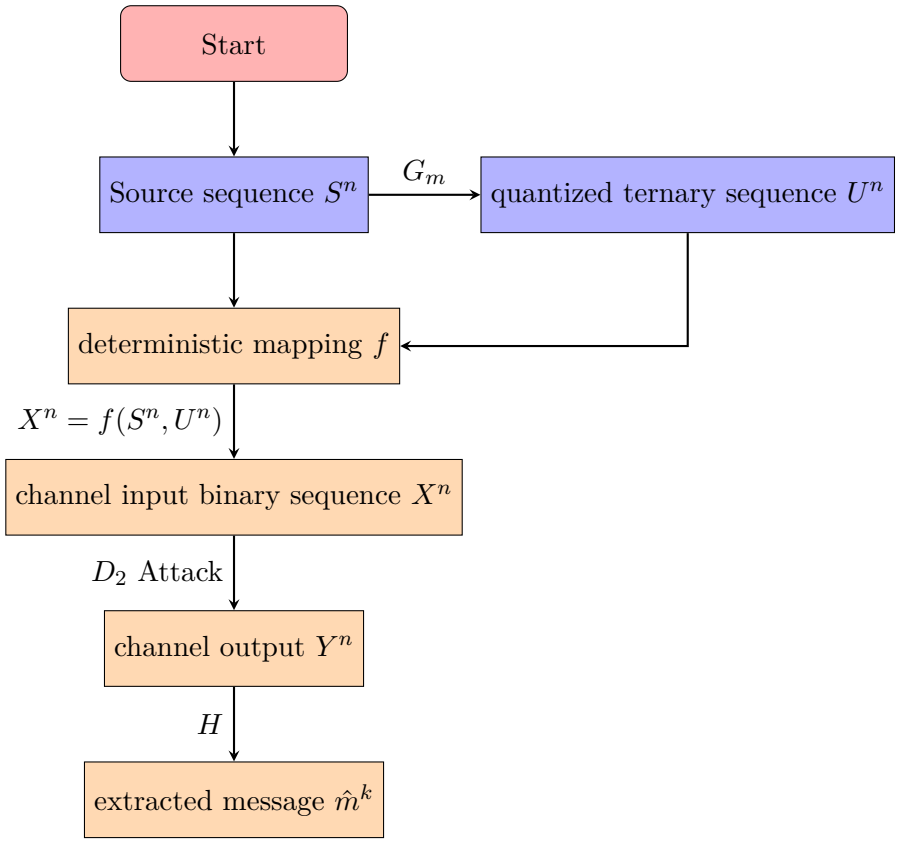

FIGURE 3.10: LDGM-LDPC nested Coding Scheme

The first target of the watermarking problem is that the watermarked sequence X^n must be as close to the source sequence S^n as possible. Namely, the target is to minimize the Hamming Distance between S^n and X^n. We use D_1 to define the upper bound of the distortion between S^n and X^n. Due to this demand, our proposed method introduces a deterministic mapping function f, so that S^n, X^n and U^n are all jointly typical $\langle S^n, X^n, U^n \rangle$ with certain probability distribution and this distribution will be discussed in the deterministic mapping subsection. Notice the fact that S^n and X^n are binary sequences while U^n is a ternary sequence.

$$X_i = f_i(S_i, U_i),$$

where $i \subseteq \{1, 2, 3, ..., n\}$. The result of this deterministic mapping will produce a binary sequence X^n with length n, and this is the channel input, while the purpose of this deterministic mapping is to minimize the hamming distance between S^n and X^n.

The next step is how the LDPC code acts to get the proper codeword for the received sequence Y^n. First of all, channel input sequence X^n is going to pass through a selected noise channel with transmitting probability $p(y|x)$, and the received sequence at the other end is denoted as Y^n. The distortion between the input sequence X^n and the received sequence Y^n is denoted as D_2. If the decoder finds out a codeword that corresponding to the channel output Y^n and also is a component in the codebook $\mathcal{C}_{m^k}$, the extracted watermark massage $\hat{m}^k$ is found successfully.

General statement

The watermarking problem involves five random variables, and they are source sequence S^n, channel input X^n, channel output Y^n, auxiliary random variable U^n and the watermark message m^k. Among these five variables, X^n can be treated as the watermarked image and Y^n can be treated as the contaminated watermarked image. Each

watermark message m^k corresponds to an unique LDGM code with codebook $\mathcal{C}_{m^k}$. The auxiliary random variable U^n has the joint distribution, which can be presented as the following,

$$P(s)p(u,x|s)p(y|x)$$

Moreover, it is possible to define X^n as the function of S^n and U^n, denoted as

$$X^n = f(S^n, U^n)$$

And this deterministic mapping function will be discussed in detail in the next sub-section.

The code rate of the watermarking system is given by

$$R = I(U;Y) - I(U;S),$$

where $I(\bullet)$ is the mutual information.

While the capacity of this watermarking system is given by

$$C = \max_{p(x,u|s)} [\, I(U;Y) - I(U;S)]$$

Coding Scheme

Encoding

1. Given the watermark message m^k and the source sequence S^n, it is possible to find a codeword u^n in $\mathcal{C}_{m^k}$ that is jointly typical with S^n

2. Set the channel input $X^n = f(S^n, U^n)$

Decoding

1. Given the channel output Y^n, the decoder searches for the unique codeword $\hat{u}^n$ in codebook $\mathcal{C}_{m^k}$, which is supposed to be jointly typical with Y^n

2. The decoder then finds the codebook $\mathcal{C}_{\hat{m}^k}$ that contains $\hat{u}^n$ and sets $\hat{m}^k$ as the decoded watermark message.

Deterministic Mapping f

Deterministic mapping function f is an essential part to find the jointly typical pair of (S^n, U^n, X^n), the programming implementation is according to the information hiding statement from the paper (Moulin and O'Sullivan 1999). This section not only works to find the jointly typical pair but also works to find the information capacity with respect to the two distortion values D_1 and D_2, where D_1 is the distortion between X^n and S^n and D_2 is the distortion between X^n and Y^n.

The working procedure is shown step by step as the following:

Step1. Since S^n is binary and U^n is ternary, there are overall six possible combinations. Each combination has the existing possibility from zero to one. The constraint is that the sum of the existing possibility for each valid group must equal to one. Table $\mathcal{T}$ is generated.

Step2. Since D_1 has something to do with X^n and X^n is binary, we need to extend Table $\mathcal{T}$ with another column to represent the probability of X^n. Another constraint is that the overall distortion between S^n and X^n must be less or equal to D_1. Namely,

$$\sum_{i=1}^{n} d_i(s_i, x_i) \leq D_1$$

Step3. Calculate the mutual information $I(U;S)$, which will be

$$I(U;S) = \sum_{(s,u)} p(s;u) \log_2 \frac{p(s;u)}{p(s)p(u)}$$

Step4. Extend the Table $\mathcal{T}$ with another column for the probability value that concerning Y^n

Step5. Calculate the mutual information between U^n and Y^n, and D_2 is applied in this step to calculate the distributions.

$$I(U;Y) = \sum_{(y,u)} p(y;u) \log_2 \frac{p(y;u)}{p(y)p(u)}.$$

Step6. Calculate the maximum difference between $I(U;Y)$ and $I(U;S)$, which would be the information capacity of the system

$$C = \max_{p(x,u|s)} \left[I(U;Y) - I(U;S) \right].$$

In general, this deterministic mapping step is going to figure out the information capacity of the system and the corresponding jointly typical pair with optimized distribution.

Matrix Partition

In order to make this watermarking system work properly, one the most essential parts is to get the properly parity check matrix H and the corresponding generator matrices $G_{m_i^k}$, for $i \subseteq \{0, 1, 2, ..., 2^{nR} - 1\}$.

Initially, we consider the possibility of partitioning the LDGM code into 2^{nR} number of distinct LDPC codes. However, when it comes to the all-zero codeword, what LDGM code should it belong to? Then, we are thinking about the idea of partition matrix as a linear shifted version of the main LDPC code.

Suppose we have a parity check matrix with size $(n-k) \times n$, which has the optimized degree distributions with respect to the binary symmetric channel (BSC). The general matrix partition procedure is shown as the following:

Step1. According to the parity check matrix H, we find the generator matrix G, where $GH^T = 0$

Step2. Select k_1 rows of G with the lowest Hamming Weights and put them together to form a smaller generator matrix $G_{m_i^k}$. Notice that there is no guarantee that the $G_{m_i^k}$ matrix is sparse. If it is not sparse, move back to step 1. Otherwise, move to step 3.

Step3. Determine 2^{k_1} number of codewords that generated by the generator matrix $G_{m_i^k}$ and put them all together into the codebook $\mathcal{C}_1$.

Step4. For a binary watermark message m^k with length nR, there will be 2^{nR} different watermark messages. For each watermark message m_i^k construct the codebook $\mathcal{C}_{m_i^k}$ with the components in $\mathcal{C}_1$

Step5. Now, the overall codebook $\mathcal{C}_{m^k}$ is constructed.

Chapter 4

Implementation and Experimental Results

This chapter will provide all the experimental results from the previous chapters. All the classical algorithms are in standard forms, which are not optimized by other researchers or papers. Besides, this chapter will also provide the implementation results for the proposed hybrid scheme. For the proposed hybrid scheme, the comparisons are made in two aspects. On the one hand, we will compare the proposed hybrid method with Lin's methods. On the other hand, we will compare the performance between the related watermarking algorithms and our proposed hybrid method. Finally, the step test results of the nest LDPC-LDGM watermarking scheme will also be provided in this chapter.

4.1 Attack Specification

This section will give a specification on all the attacks that apply during the experiments. It is shown in Table 4.1.

Notice that the kernel functions in LPF and HPF are shown as the following,

$$K_{lpf} = \frac{1}{6}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$K_{hpf} = \frac{1}{6}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Attack	Specification
GS	variance is 0.1 and mean is 0
SPN	noise density=0.02
TMP	5×5 block with zero value to cover part of X^n with randomly picked location
SPE	variance is 0.05 and mean value is 0
LPF	define the kernel K_{lpf} and apply MATLAB imfilter inner function
HPF	define the kernel K_{hpf} and apply MATLAB imfilter inner function
GF	standard deviation is 2
JPEG	quality factor set to be 50
CR	randomly crop out a valid image segment with size 20×20 for further extraction step

TABLE 4.1: Attack Specification

4.2 Classical Algorithms and HiDDeN

4.2.1 LSB-Based Scheme

This section provides the results in the LSB-based algorithm. Figure 4.1 shows the cover image Baboon with size 25×40 in gray-scaled and the watermark message m^k with size 4×4 in binary. This section uses peak signal-to-noise ratio (PSNR) to measure the performance of imperceptibility and the bit-error-rate (BER) value to measure the robustness through several kinds of attack. The test results are shown in Table 4.2. The PSNR value reaches 51.4098dB over 20 tests shown in Figure 4.2. The LSB algorithm is tested over nine attacks, which are specified in the previous sub-section. It turns out that the LSB-based algorithms have relatively good results in salt and pepper noise attack (SPN), Tampering Attack (TMP) and JPEG compression (JPEG). The image results are shown in Figure 4.3.

In general, the LSB-based algorithm shows good results in imperceptibility, while the test results over nine kinds of attack appear

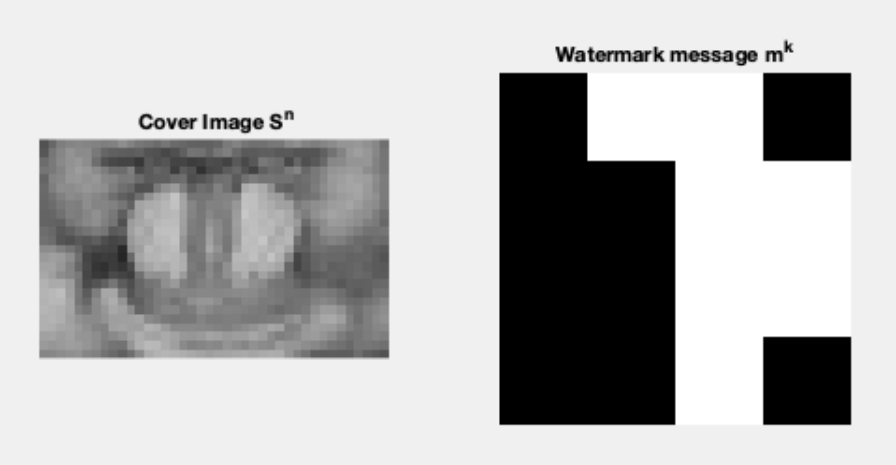

FIGURE 4.1: LSB cover image and watermark

to be not that optimal. In conclusion, a watermark that embedded into the least-significant bit-plane of the image is susceptible to frequency-related attacks.

The embedding rate $\mathcal{R}$ of the LSB-based algorithm is limited to the size of the cover image. This experiment used a gray-scaled image as the cover image and a binary sequence with size 4×4 as the watermark. Consequently, $\mathcal{R}$ is 0.001953 in general. It is common to embed the watermark message m^k with size that equal to n, however, for the comparison convenience, we just embedded 125 bits into the cover image S^n.

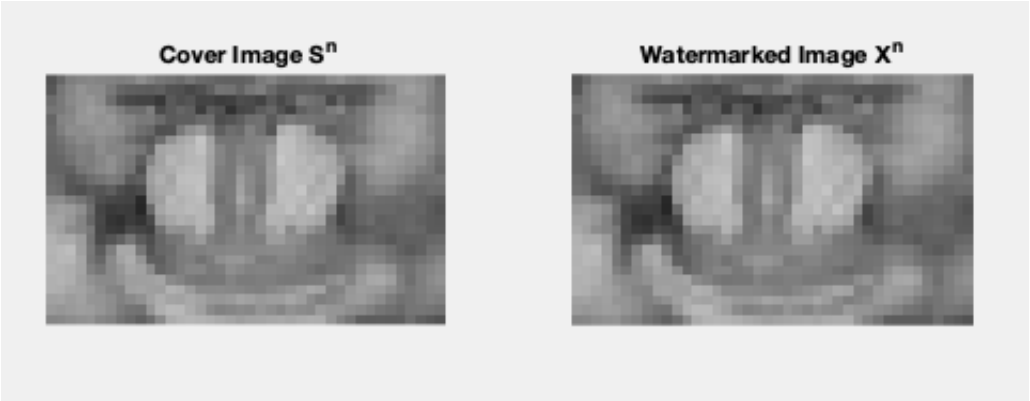

FIGURE 4.2: LSB cover image and watermarked image

Attack	BER
GS	44.6875%
SPN	0.6250%
TMP(5×5)	0.6250 %
SPE	49.3750%
LPF	56.2500%
HPF	56.2500%
GF	62.5000%
JPEG	7.0000%
CR(20×20)	39.6875%

TABLE 4.2: LSB test result for embedding rate $\mathcal{R} = 0.001953$ over 20 tests

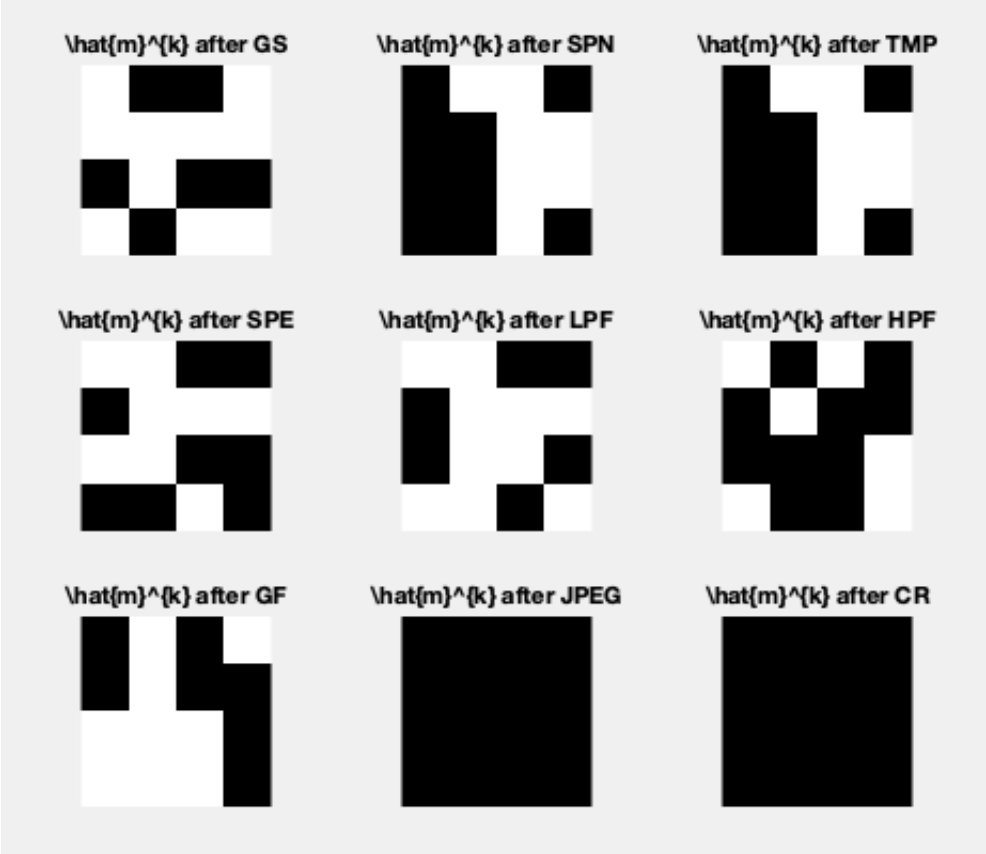

FIGURE 4.3: LSB test result

4.2.2 DCT-Based Scheme

This section will provide the results for DCT non-blind watermark scheme. As shown in Figure 4.4, the gray-scaled image Baboon with size 256×256 is the cover image S^n and the binary image Lena with size 32×32 is the watermark message m^k. The PSNR is utilized to measure the imperceptibility while the BER is used to measure the robustness. The cover image S^n and the watermark message m^k are shown in Figure 4.4, and the watermarked image X^n is shown in Figure 4.5. The average PSNR value is 33.4308dB over 20 tests. The overall embedding rate $\mathcal{R}$ is 0.001953, which is set to be the same with the LSB test for comparison convenience.

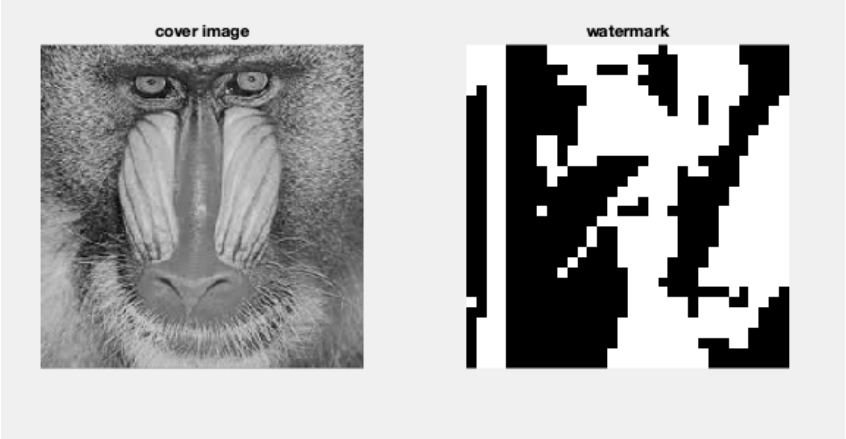

Figure 4.4: Cover Image and Watermark for the DCT

According the experimental results, the DCT-related algorithm has a comparatively good performance over most of the attacks except for high-pass filtering (HPF) and cropping attack (CR). The analytical results are presented in Table 4.3 while the image results of robustness tests over several attacks are shown in Figure 4.6.

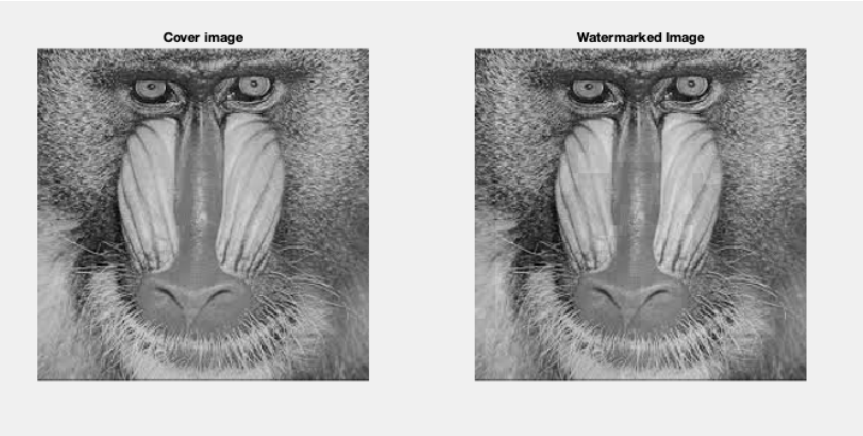

Figure 4.5: Cover Image and Watermarked Image for DCT

Attack	BER
GS	9.7656%
SPN	3.7109%
TMP	11.7188%
SPE	9.6680%
LPF	0.0977%
HPF	50.3906%
GF	8.6914%
JPEG	0.001%
CR(20×20)	Not realized

Table 4.3: DCT Test Results with $\mathcal{R} = 0.001953$ over 20 Tests

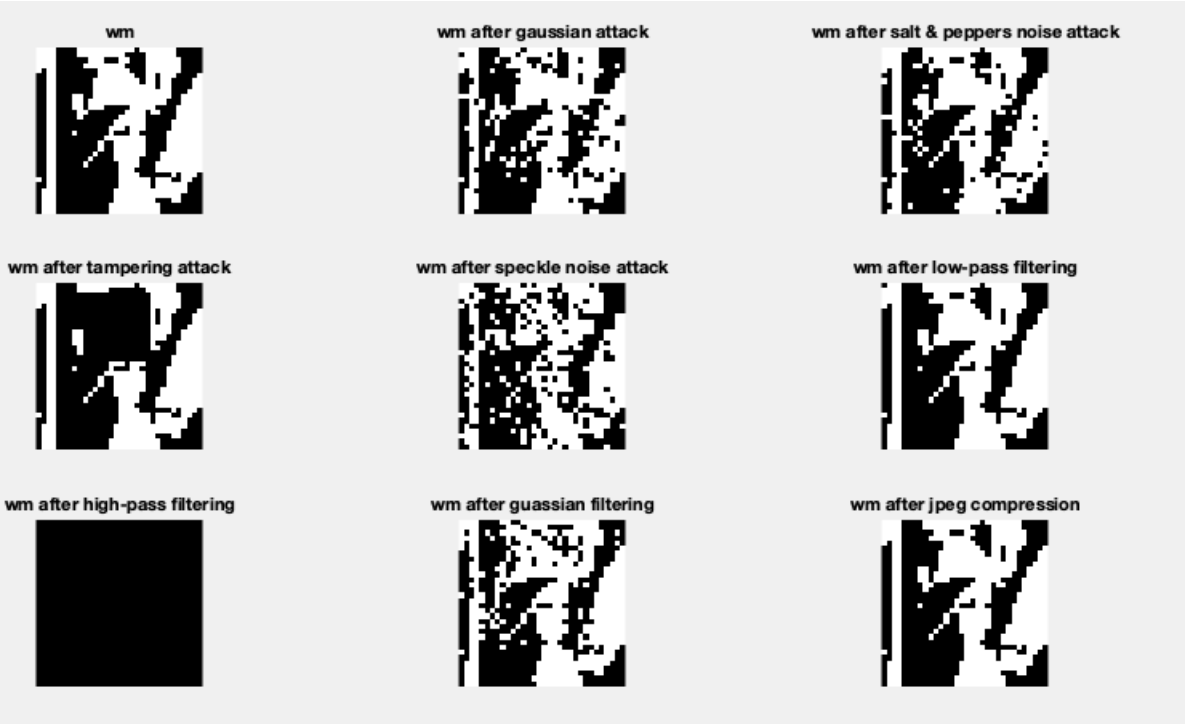

Figure 4.6: DCT Robustness Test Result with $\mathcal{R} = 0.001953$

Comparison between LSB And DCT

This sub-section will compare the robustness results with BER values of LSB and DCT under the same embedding rate.

Attack	LSB	DCT
GS	44.6875%	9.7656%
SPN	0.6250%	3.7109%
TMP	0.6250 %	11.7188%
SPE	49.3750%	9.6680%
LPF	56.2500%	0.0977%
HPF	56.2500%	50.3906%
GF	62.5000%	8.6914%
JPEG	7.0000%	0.001%
CR(20×20)	39.6875%	Not realized

TABLE 4.4: Comparison between LSB and DCT for the BER Value with $\mathcal{R} = 0.001953$

According to Table 4.4, it is clear to see that under the same attack condition and at the same embedding rate, the LSB-related algorithm has a comparatively better results in SPN and TMP. Due to the reason that the DCT algorithm will result in the loss of local information of the image, the cropping attack has not been realized yet. For the other attacks, the DCT-related scheme has a better performance comparing with the LSB-related scheme.

4.2.3 SVD-Based Scheme

This section will provide the experimental results for the SVD-based watermarking algorithm. The cover image S^n is grayscaled with size 225×225 and the watermark is binary with size 7×113. Consequently, the embedding rate is set to be the same, where $\mathcal{R} = 0.001953$. The robustness tests are over eight kinds of attack. The watermark strength factor α is set to be 0.07. The embedding results is shown in Figure 4.7. The robustness test results are shown in Figure 4.8, with the original watermark message m^k at the left top corner.

FIGURE 4.7: Cover Image, Watermark, and Watermarked Image in SVD

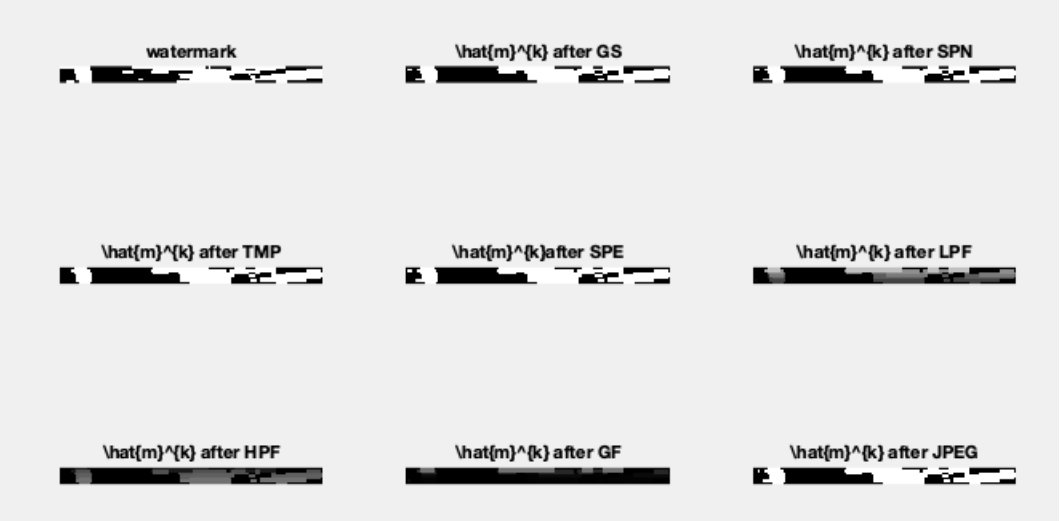

FIGURE 4.8: SVD Robustness Test Result with $\mathcal{R} = 0.001953$

Since the watermark message is binary, we can analyze the robustness performance with the BER values, and the results for the eight attacks are shown in Table 4.5 over 20 tests.

Comparison for SVD DCT LSB

Since the embedding rate is set to be consistent, in this sub-section, we will compare the robustness test results for these three algorithms. Here are the results in Table 4.6, which analyzes the overall robustness performance of the three schemes.

Attack	BER
GS	11.5803%
SPN	11.5992%
TMP	10.9987%
SPE	11.5676%
LPF	33.8812%
HPF	51.2010 %
GF	51.2010%
JPEG	11.5044 %
CR(20×20)	Not realized

TABLE 4.5: SVD Test Results with $\mathcal{R}$ =0.001953 over 20 tests

Attack	LSB	DCT	SVD
GS	44.6875%	9.7656%	11.5803%
SPN	0.6250%	3.7109%	11.5992%
TMP	0.6250 %	11.7188%	10.9987%
SPE	49.3750%	9.6680%	11.5676%
LPF	56.2500%	0.0977%	33.8812%
HPF	56.2500%	50.3906%	51.201%
GF	62.5000%	8.6914%	51.201%
JPEG	7.0000%	0.001%	11.5044%
CR(20×20)	39.6875%	Not realized	Not realized

TABLE 4.6: Comparison LSB, DCT and SVD for the BER Value with $\mathcal{R}$ =0.001953

4.2.4 Deep Learning Method:HiDDeN

HiDDeN (Zhu et al. 2018) is the first end-to-end trainable framework for data hiding, which works well in both steganography and watermarking at low embedding rates.

Notation Clarifications

The subsection is the extension of the notation in 2.1, while X^n, S^n, m^k, Y^n and $\hat{m}^k$ have the same meanings with what we clarified in 2.1. Here are some supplementary notations for HiDDeN network.

The small letter d is the distortion value, while the capital letter L indicates the length of the sequence. The encoder E_θ has a trainable

parameter θ, and the Noise Layer is denoted as $\mathcal{N}$ without any trainable parameters. The decoder D_ϕ has another trainable parameter ϕ. The Generator and Adversary Network (GAN) is denoted as A_γ with a trainable parameter γ.

Loss Function and Metrics Specifications

There are in general four kinds of distortions and losses that applied during the stochastic gradient descent. The distortion value d_1 that introduced during the encoding step is denoted as

$$d_1 = \frac{\|S^n - X^n\|_2^2}{L_{S^n}},$$

where L_{S^n} is the total pixel number in the source sequence and $\| \bullet \|_2^2$ is the l_2 distance.

The second group of distortions comes from the GAN network, where the Adversary network aims to distinguish between the cover image S_n and the watermarked image X^n with a classification loss d_A, and the Generator network intends to cheat the Adversary network with a loss d_G. They can be denoted as the following,

$$d_A = log(1 - A(S^n)) + log(A(S^n)),$$

$$d_G = log(1 - A(X^n)).$$

The last distortion that introduced into the system is the message recovery loss d_4, which can represented as the following,

$$d_4 = \frac{\|m^k - \hat{m}^k\|_2^2}{L_{m^k}},$$

where L_{m^k} is the length of the watermark message m^k and $\| \bullet \|_2^2$ is the l_2 distance.

Consequently, during the stochastic gradient descent, the neural network needs to minimize the general loss over the distribution of

m^k and S^n and X^n

$$E_{S^n,m^k}[\, d_4 + \lambda_1 d_1 + \lambda_G d_G]\,,$$

where λ_1 and λ_G are two super parameters that set before the training.

Besides, for the GAN network, the loss for the discriminator A_γ also needs to be minimized.

$$E_{S^n,m^k}[\, d_A]\,.$$

The metrics that apply in the HiDDeN network will be offered as the following. In general, there are three values utilized to measure the features of the watermarking system. Peak signal-to-noise ratio (PSNR) is applied to measure the imperceptibility of the watermarking scheme, while bit-error-rate (BER) is used to measure the robustness of the system. Besides, unlike the other related algorithms shown in the previous section, in the HiDDeN network, they used bit-per-pixel (BPP) to measure the embedding rate of the watermark rather that bit-per-bit value.

General Statement

The general neural network architecture is shown in Figure 4.9. An encoder network is responsible for hiding a secret input message into a cover image by direct matrix convolution. A decoder network is responsible for recovering the secrete message signal through proper linear regression. The third network is GAN, which is responsible for checking the output image to see if it is reasonable concerning the Human Visual System (HVS).

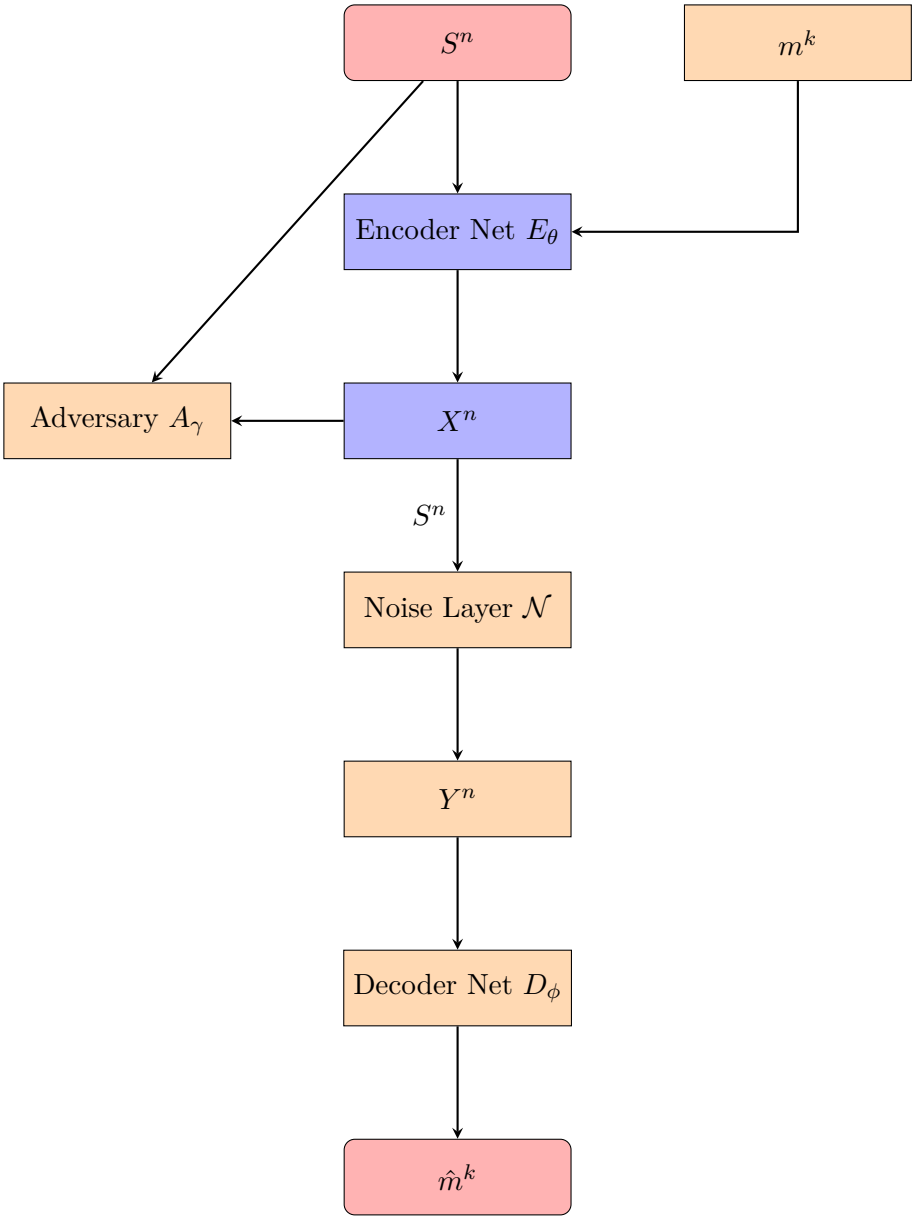

Figure 4.9: HiDDeN Neural Network

The highlight of HiDDeN is the noise layer added between the encoder and the decoder. To simulate different kinds of attacks, Zhu et al. (Zhu et al. 2018) figured out that it is necessary to add one layer that can introduce noise into the image. However, the difficulty is that the neural network cannot deal with data that is not differentiable. For some of the attacks, especially the one that related to quantization, it is not possible to embed it into the neural network directly. As a result, they stimulated some differentiable methods which have a similar effect on images.

Attack Layers Specifications

There are, in general, four groups of attacks and five different attack layers. The attacks are as the following: cropping(CR), dropout attack (DROP), Gaussian Blur (GF), and the JPEG compression (JPEG).

For the cropping attack (CR), there are two sub-versions: crop attack and cropout attack. For crop attack with ratio 3.5%, it indicates that the ratio of the Y^n size over X^n size is 3.5%. While for cropout attack with percentage p, it indicates that p percentage of pixels are from X^n combined with rest of the pixels from S^n Notice that, for cropout attack, the p percentage of pixels are not independent but from a cropped squared part of X^n, where $p = 30\%$

For dropout attack with percentage p, it indicates a random drop of the pixels in X^n and substitute them with the pixels in S^n, where $p = 30\%$

For Gaussian blur attack, the filter kernel with standard deviation value $\sigma = 2$

For JPEG compression, the quality factor is close to 50.

Experimental Results

There are totally two groups of experiments, and each focuses on different components. Notice that there are generally two methods to train the network. For the watermarking system purpose, we applied the combined training that trains the model by introducing a different noise layer for each mini-patch. There are in total five different attack layers (crop layer, cropout layer, Gaussian blur layer, drop layer and JPEG layer).

The cover image is in YUV colored form with size 128×128 and the watermarked message m^k is binary with length k. This k value varies in the second experiments, where $k \subseteq \{30, 50, 70\}$, while in the first experiment, the k is set to be 30. The super parameter are set to be $\lambda_1 = 0.7$ and $\lambda_G = 0.001$.

Experiment one is aiming to know how the noise layer that affects the final result. The result is shown in Figure 4.10. This experiment sets 300 epoch. 20,000 images from the COCO dataset with size 128×128 are set to be the training set and 2,000 not-repeatable images with size 128×128 from the COCO dataset are set to be the validation set. The group without noise has the BER value around 0.0037, while the BER of the group with combined noise reaches 0.1410.

Notice that for the combined noises attacks, the model is trained by introducing a different noise layer for each mini-batch.

Experiment two is targeted to find the relationship between the length of the secret message and the final BER values. According to the result shown in Figure 4.11, when the message length is 50-bit, the decoding error rate is around 0.2238, and if the message length increases to 70, it is almost impossible for the decoder to recover the proper message, and the BER value is over 0.5. Namely, the result is no better than a random guess. During the experiment, with $L_{m^k} = 70$, the network sometimes cannot converge to the end.

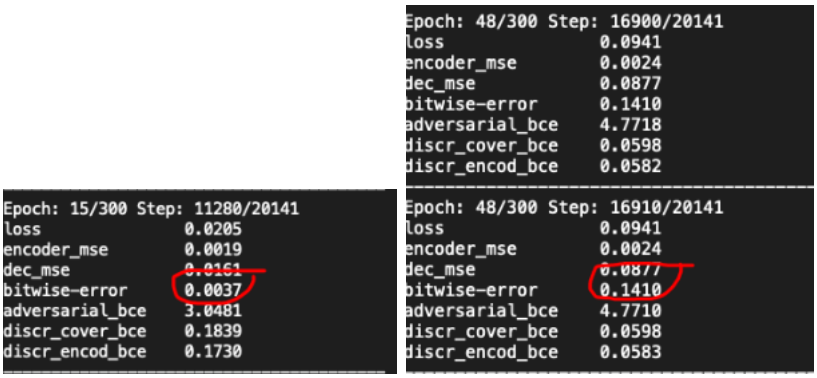

FIGURE 4.10: HiDDeN:Experiment on Noise Layer, where the left-hand-side results is the experiments without noise, and the right-hand-side results are the experiments with noise.

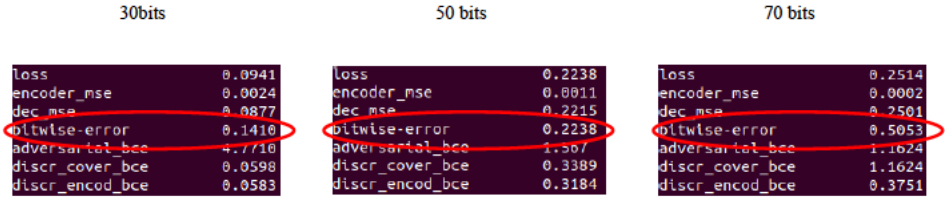

FIGURE 4.11: HiDDeN:Experiment on different message length

Besides, with the 50-bit length of the message, since the cover image S^n is a 128×128 YUV image, the embedding rate at bit-level is only 1.271×10^{-4}.

Overall, the HiDDeN neural network indicates the possibility for the neural network to recover the watermark message. However, with combined noise attacks, even with length 30 of the watermark message m^k, the HiDDeN network has the BER value 0.1410, which is not that optimal. Besides, when we increase the watermark message length to 50 bits or 70 bits, the results turn out to be even worse.

4.3 Proposed Hybrid Scheme

This section will introduce a hybrid structure that utilizes SVD, Discrete Wavelet Transformation (DWT), and LDPC code to complete the digital watermarking task. This new structure applies SVD as

the main embedding and extraction algorithm, while the LDPC code is used to encode the watermark message m^k, and DWT is for cover image S^n transformation.

General Scheme

The general working scheme is shown in Figure 4.12. The embedding step is shown as the following, First of all, the watermark message m^k is encoded with proper LDPC code to $m^k_{encoded}$ to enhance the capability of resisting noise, while at the same time, the cover image S^n applies 3-level DWT transformation. After that, SVD transformation is used to complete the embedding step with the low-frequency sub-band component at the third level LL_3 and the encoded watermark $m^k_{encoded}$. Then, the watermarked image X^n can be obtained through the inverse DWT process. After the attack channel, the extraction step is simply the inverse of the embedding step. Finally, the recovered watermark message $\hat{m}^k$ is obtained.

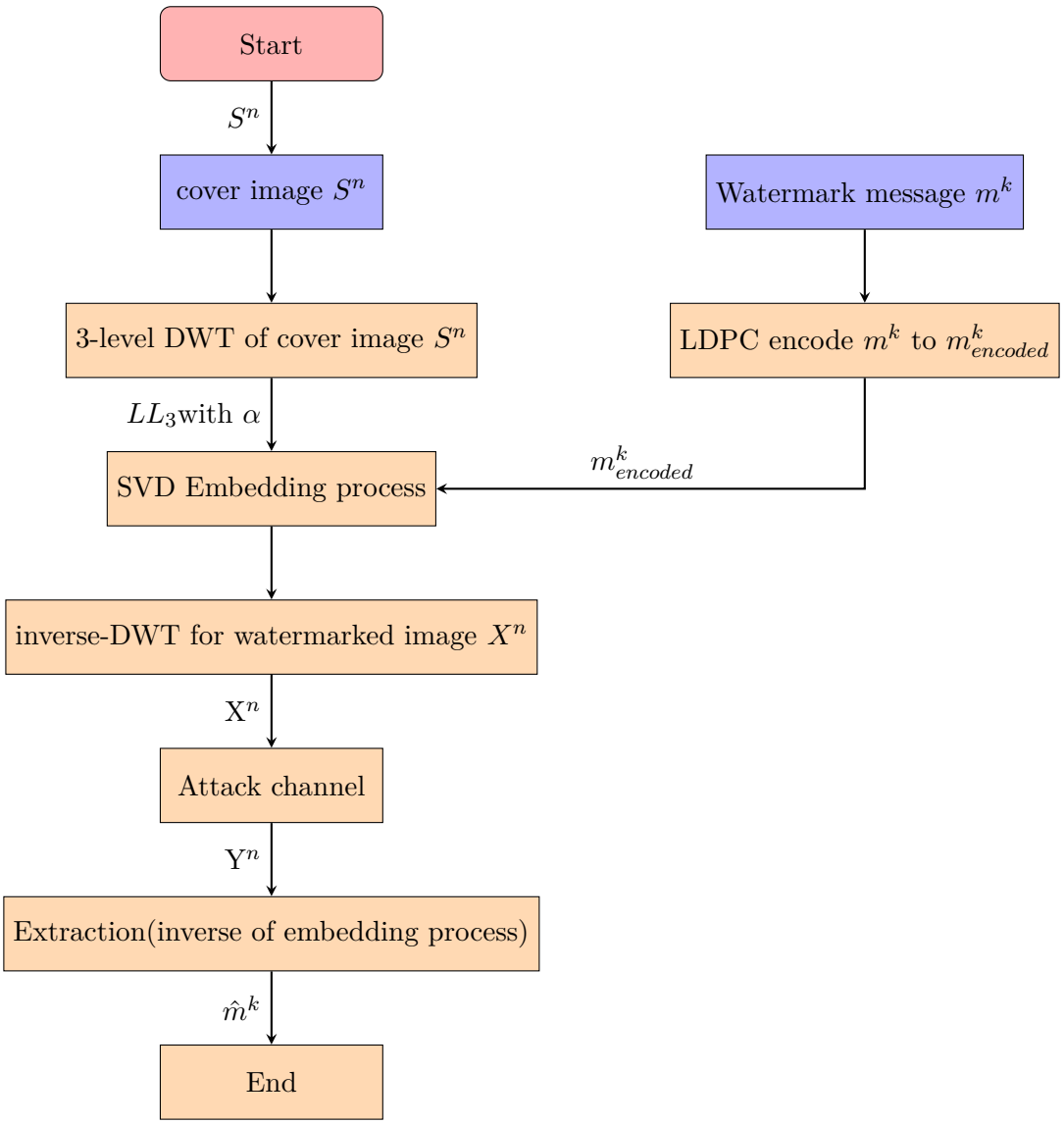

FIGURE 4.12: Hybrid Watermarking Scheme, where α refers to the watermark strength factor

Implementations NO.1

The target of this implementation is to test whether our proposed method can get a better result compared with Lin's methods.

For the experiment setting, the cover image S^n is set to be 512×512 in gray-scale, and the watermark message m^k is a 64×64 gray-scaled image. The LDPC is used to encode all bit-planes and all channels of the watermark message. Consequently, the resulting embedding rate would be 0.0156, which is consistent with the embedding rate in Lin's method.

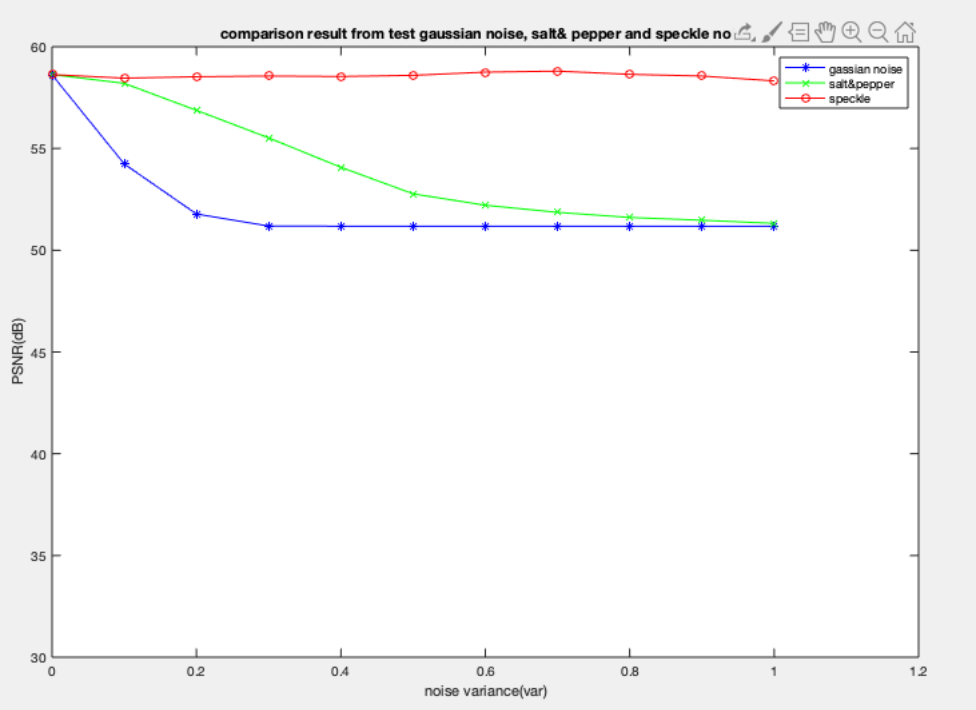

FIGURE 4.13: Proposed Hybrid Scheme:PSNR v.s Noise Variance over GS, SPN and SPE with $\mathcal{R} = 0.0156$

The experiment results are shown as the following. Figure 4.14 (Lin and Wan 2016) presents the curve results for PSNR concerning noise variances of Lin's method, and our proposed hybrid scheme also provides the same curve, which is shown in Figure 4.13. According to the figures above, it is clear to see that our proposed method with

modified structures and parameters has a relatively good result in PSNR values. Besides, the watermark embedding result is shown in Figure 4.15, and the extraction results are also provided in Figure 4.16. This section shows some of the robustness test results in Table 4.7.

Attack	MSE	BER	PSNR(dB)
GS	1.9191e+04	0.09058	51.1816
SPN	8.7856e+03	0.08716	51.2921
TMP	255.1360	0.08862	58.5025
SPE	262.2102	0.11353	58.3215
LPF	1.7278e+04	0.08594	58.6315
HPF	1.7428e+04	0.50513	51.0968
GF	1.7279e+04	0.09302	58.5724
JPEG	1.7278e+04	0.08545	58.6553
CR (20×20)	Not available	Not available	Not available

Table 4.7: Hybrid Robustness Test Table with $\mathcal{R} = 0.0156$

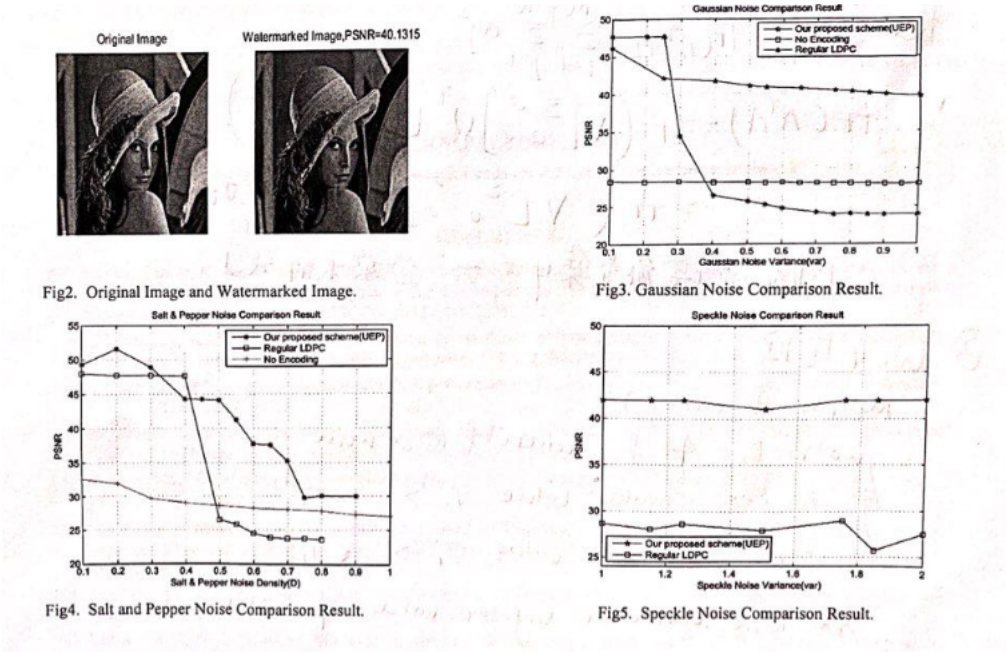

Figure 4.14: Lin's Hybrid Scheme: PSNR v.s Noise Variance (Lin and Wan 2016)

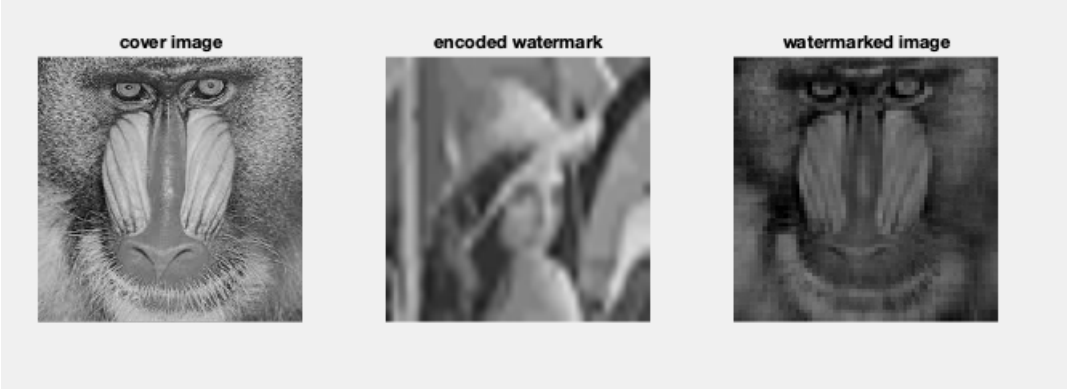

Figure 4.15: Proposed Hybrid Scheme: Embedding Result with $\mathcal{R} = 0.0156$

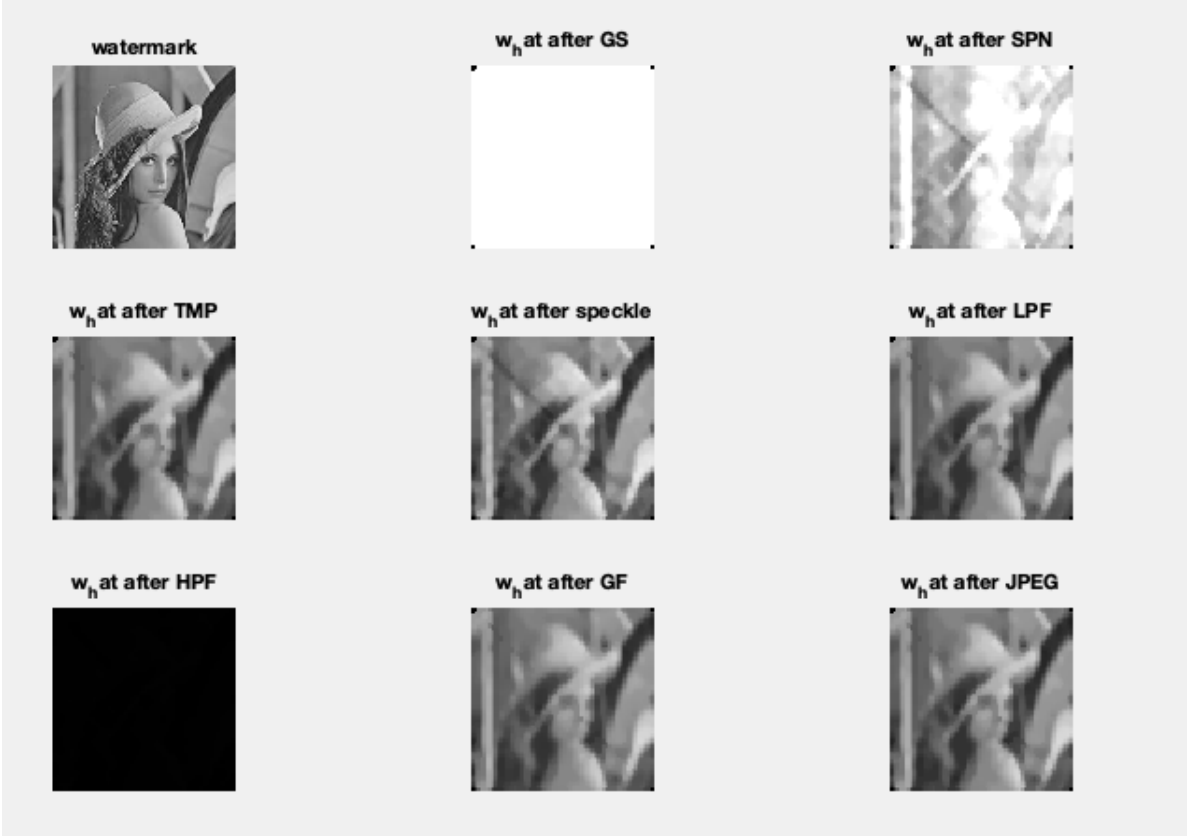

Figure 4.16: Proposed Hybrid Scheme: Extraction Result with $\mathcal{R} = 0.0156$

Implementation No.2

The second implementation aims to control the embedding rate to be the same as that in the related work. This group test is aimed to find out whether our proposed method can get a better performance with respect to the related watermarking algorithms.

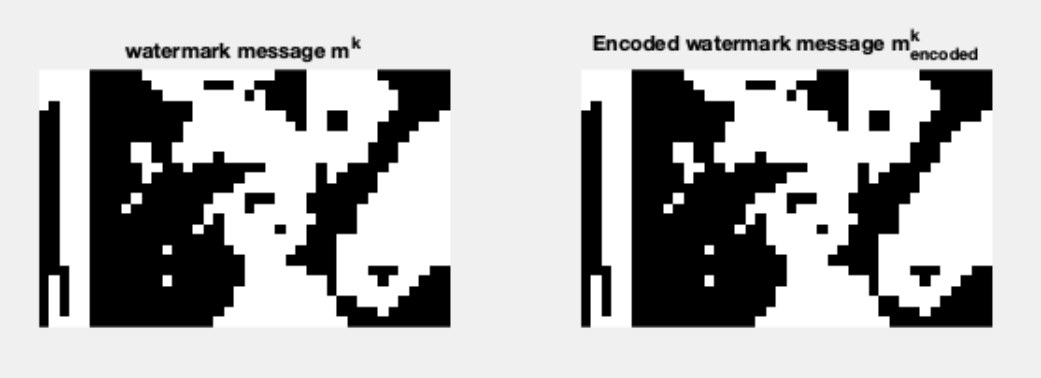

FIGURE 4.17: Proposed Hybrid Scheme: LDPC encoding Result of m^k to $m^k_{encoded}$

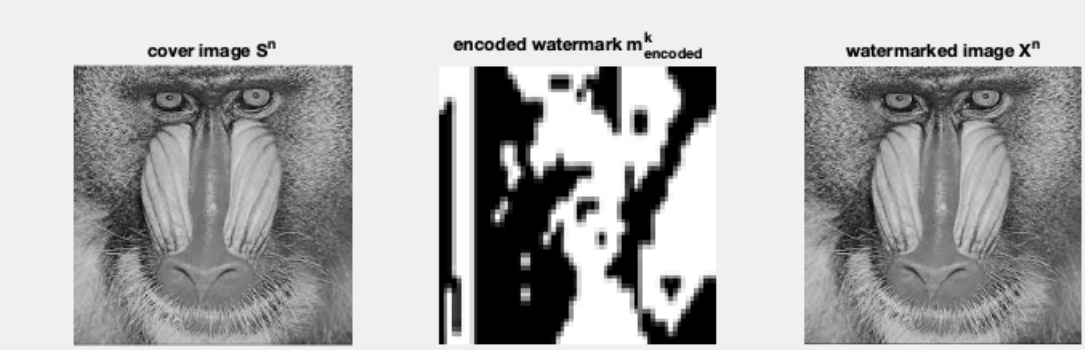

FIGURE 4.18: Proposed Hybrid Scheme: Embedding Result with $\mathcal{R} = 0.001953$

During the implementation, the cover image S^n is a grayscaled image with a size 512×512 while the watermark message is a binary image with size 64×64. Consequently, the $\mathcal{R}$ is 0.001953, which is consistent with the embedding rate used in the implementations of the related work.

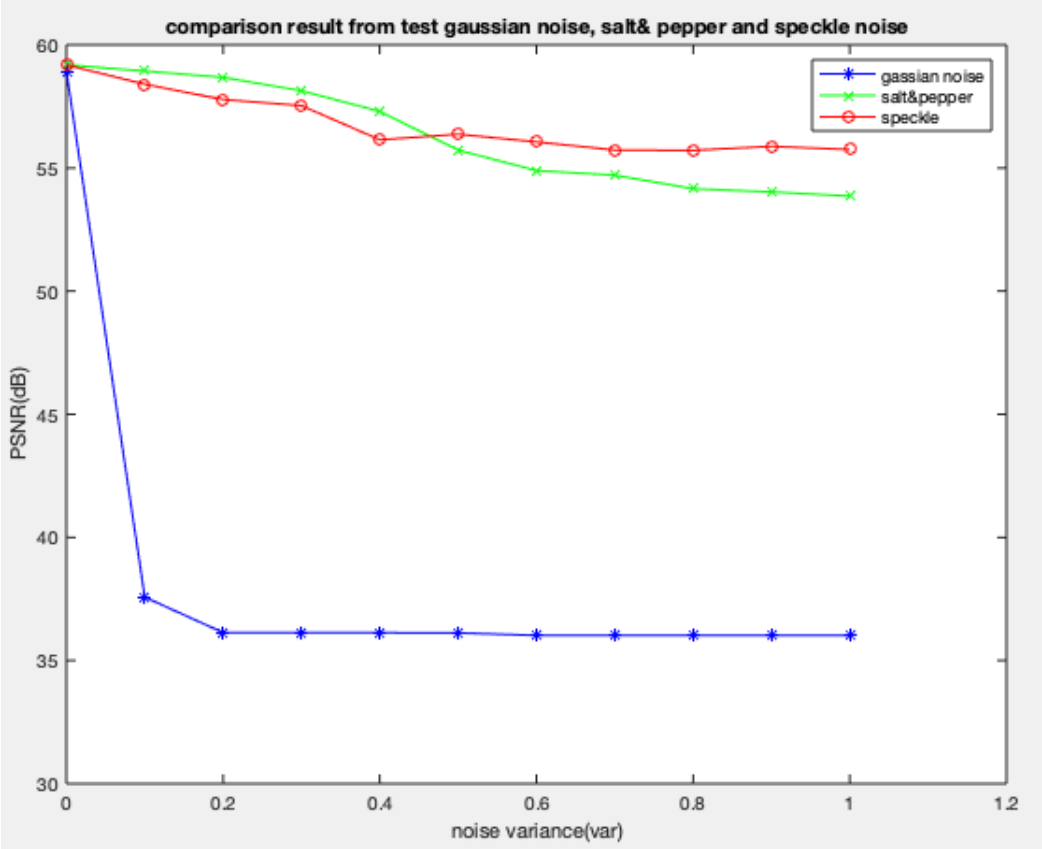

Figure 4.19: Proposed Hybrid Scheme: PSNR v.s Noise Variance over Gaussian (GS), salt and pepper (SPN) and speckle (SPE) attack with $\mathcal{R}$ = 0.001953

The metrics utilized in this implementation are also consistent with the measurement in the related work part. We use the PSNR value to measure the imperceptibility and use the BER value to measure the robustness performance over seven attacks.

The robustness tests are over these following attacks: Gaussian noise attack (GS), salt and pepper noise attack(SPN), speckle noise attack (SPE), tampering attack (TMP), low-pass-filtering(LPF),high-pass-filtering(HPF), Gaussian filtering(GF) and JPEG compression (JPEG). The specification details of each attack are mentioned at the beginning of this chapter.

The embedding results are shown in Figure 4.18, and the LDPC

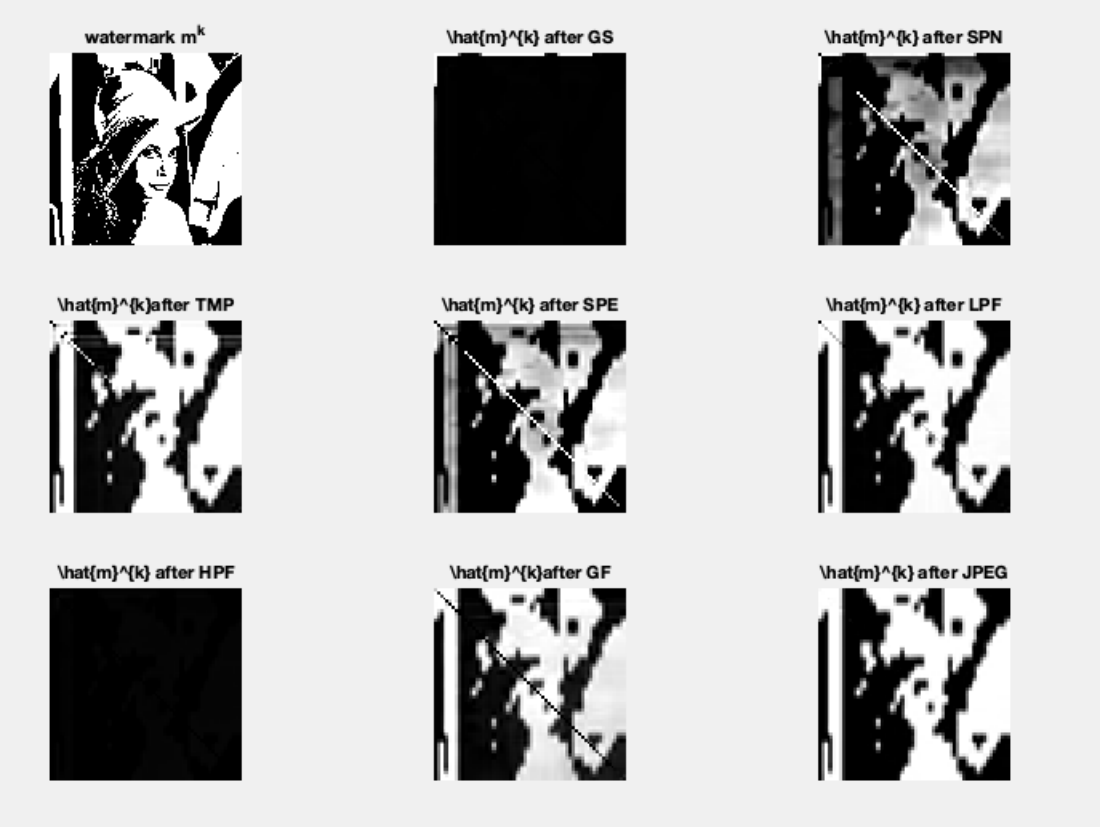

FIGURE 4.20: Proposed Hybrid Scheme: Extraction results with $\mathcal{R} = 0.001953$

encoded results are shown in Figure 4.17. For the imperceptibility test, the PSNR value reaches 46.4657 dB. While for the robustness test, we analyzes the performance with BER values, which is shown in Table 4.8. Also, we provide similar curve results that show the relationship between the PSNR value and the noise variance concerning three kinds of attacks, and it is shown in Figure 4.19.

Comparison with Related Work

This sub-section aims to compare the robustness results between our proposed methods and the related works.

For the imperceptibility results, our proposed method has the PSNR value reaches 46.4657 dB, while for the PSNR value in test

Attack	BER
GS	6.467%
SPN	6.877%
TMP	2.271%
SPE	3.963%
LPF	7.959%
HPF	50.513%
GF	8.545%
JPEG	7.837%
CR (20×20)	Not available

TABLE 4.8: Hybrid Robustness Test Table with $\mathcal{R} = 0.001953$

LSB, DCT, and SVD, the PSNR values are 51.4098dB, 33.4308dB, and infinity respectively.

Attack	LSB	DCT	SVD	Proposed Hybrid
GS	44.6875%	9.7656%	11.5803%	6.467%
SPN	0.6250%	3.7109%	11.5992%	6.877%
TMP	0.6250 %	11.7188%	10.9987%	2.271%
SPE	49.3750%	9.6680%	11.5676%	3.963%
LPF	56.2500%	0.0977%	33.8812%	7.959%
HPF	56.2500%	50.3906%	51.201%	50.513%
GF	62.5000%	8.6914%	51.201%	8.545%
JPEG	7.0000%	0.001%	11.5044%	7.837%
CR(20×20)	39.6875%	Not realized	Not realized	Not realized

TABLE 4.9: Comparison LSB, DCT, SVD and Hybrid Scheme for the BER value with $\mathcal{R} = 0.001953$

For the robustness test results, the analytical results are shown in Table 4.9, and all the results are measured with BER values.

According to Table 4.9, it is clear that our proposed method has comparatively better results under GS, GF, and SPE attacks and also it shows a good performance under JPEG compression which is practical in reality.

Attack	Our proposed Method(PSNR)	Lin's Method(PSNR)
GS	$51.8 \sim 57.3$	$41 \sim 46$
SPN	$52.1 \sim 56.3$	43.5
SPE	$52.6 \sim 57.4$	$30 \sim 52$

TABLE 4.10: Comparison between our proposed method and Lin's method with $\mathcal{R} = 0.0156$

Comparison with Lin's Methods

For the imperceptibility test, we utilize PSNR value as the metric. In Lin's method, the PSNR for the imperceptibility reaches 40.13dB, while our proposed method reaches 50.0792dB with strength factor α set to be 60.

For the robustness test, in Lin's paper, it provides the robustness tests results over three attacks with varying noise variance and to observe the changes in PSNR value. The curve provided in Figure 4.14. While during our experiment, we also provide a similar curve that indicates the relationship between the PSNR value and the noise variance for those three attacks, which is shown in Figure 4.13. The comparison of these three attacks is shown in Table 4.10.

4.4 Proposed LDPC-LDGM Nested Watermarking Scheme

4.4.1 General Overview

This section will show some experimental results for each part of the coding scheme. First of all, the LDGM encoding part is going to provide the rate-distortion curve $R(D)$ results for different kinds of sources, including uniform/non-uniform binary source and uniform ternary source. Also, this step will provide image results for watermark embedding. Secondly, the LDPC decoding part will provide the BER values for different parity check matrices with various degree distributions.

Notice that this section only provide a simple case by Dr. Mahdi. Since for the general LDPC-LDGM watermarking system, for the watermark messagem_i^k with length nR, where $i \subseteq \{0, 1, 2, ..., 2^{nR} - 1\}$. Consequently, there will be 2^{nR} different watermark messages, while each watermark message corresponds to a different LDGM code. In our example, we only realize the simplest case with a single LDGM code and its corresponding LDPC code to give us a brief view of how the LDPC-LDGM nested coding system to deal with the watermarking problem.

4.4.2 LDGM Experimental Result

This section will provide the experimental results of the LDGM coding with decimation and followed by deterministic mapping. The source bit node is denoted as x^n,where $x^n \subseteq \{0, 1\}^n$, and the network node N^n has the same length with the source bit node. While the check node C^{mn}, where $C^{mn} \subseteq \{0, 1\}^{mn}$. The check node length is fixed for convenience,$n \times m = 40000$, and the number of variable nodes V varies concerning different code rates R_{code}.

```
List Bias: 1481 0.018344 0
List: 1 remaining
Solution Converged in 1 steps.
List Bias: 1351 0.000804 0
List: 0 remaining
NO of 0= 8723       NO of 1= 616        NO of 2= 661

DeltaDistortion: 0.245600


Final distortion value is: 0.245600
```

FIGURE 4.21: Test Screenshot: Uniform Ternary with rate R_{code}=0.05

```
Solution Converged in 0 steps.
List Bias: 1111 0.048223 0
List: 1 remaining
Solution Converged in 0 steps.
List Bias: 1073 0.039321 0
List: 0 remaining
NO of 0= 3188       NO of 1= 3112       NO of 2= 3700

Distortion1: 0.312000
```

FIGURE 4.22: Test Screenshot: Uniform Ternary with rate R_{code}=0.1

In general,there are totally three kinds of experiments are provided, including uniform binary source with $Ber(0.5, 0.5)$, non-uniform binary source with $Ber(0.75, 0.25)$ and uniform ternary source. Some of the testing screenshot is provided in Figure 4.21 and Figure 4.22. Since a good LDGM code is capable of being closed to the theoretical R(D) bound, the $R(D)$ curves of all three groups of tests are provided in Figure 4.25, Figure 4.23 and Figure 4.24. The bottom blue curve is the theoretical bounds.

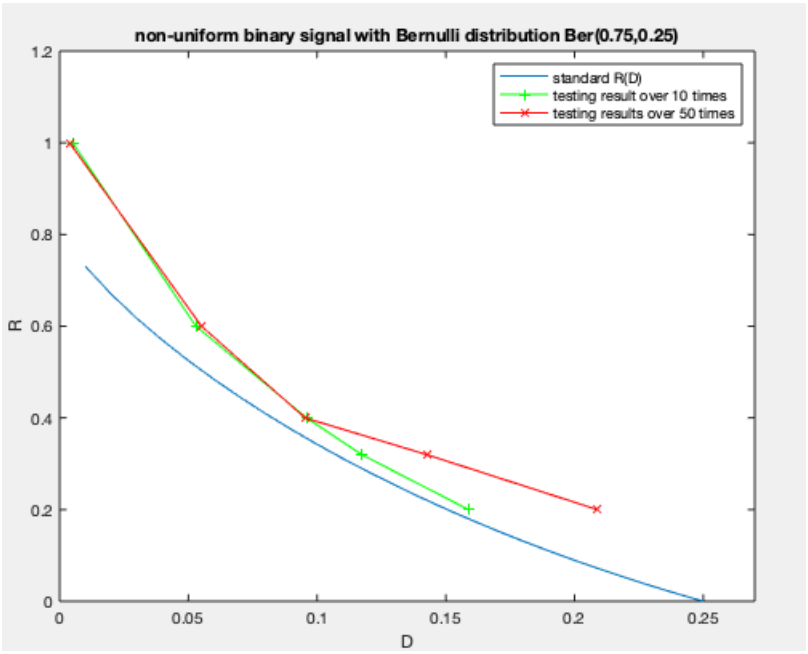

FIGURE 4.23: Non-uniform Binary Source Ber(0.75,0.25) R(D) curve

After that, to test the encoding efficiency of the LDGM code, an image test is applied based on the group of uniform binary source with $Ber(0.5, 0.5)$. The size of the binary image is 100×100 and it is supposed to have nearly equal-weight of zeros and ones inside. According to the experiment, the more uniform distributed , the better the performance. The result of some image is show in Figure 4.26. Notice that it might be not that optimal since the input image is not optimally uniform distributed.

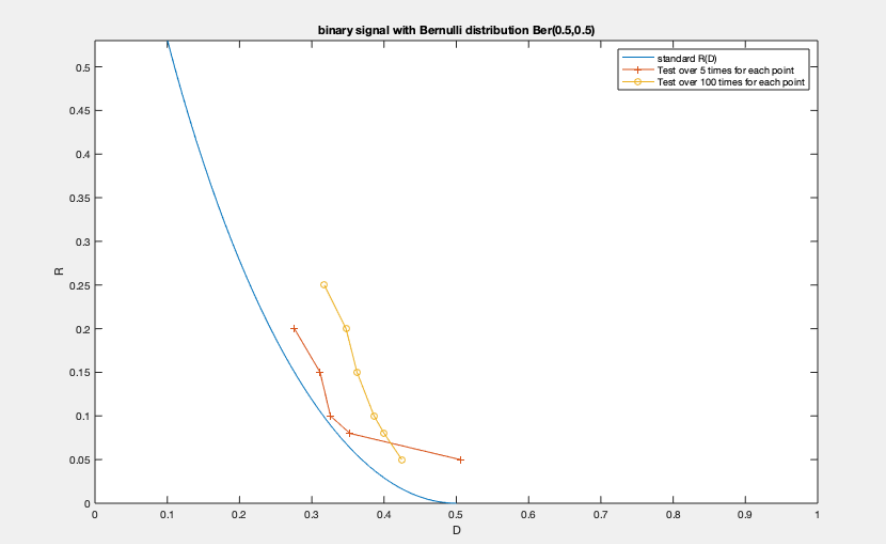

FIGURE 4.24: Uniform Binary Source R(D) curve

4.4.3 LDPC Experimental Result

This section will provide LDPC experimental results. All the test results are coming from regular-parity check matrices with relatively small size for computational convenience. In general, four groups of tests present in this section, and all of them are optimized based on the AWGN channel model. According to the introduction from the previous section, the log-likelihood ratio LLR is closely related to the chosen channel model. For the AWGN channel, the LLR value can be denoted by

$$LLR = 4 \times \frac{y}{N_0},$$

where y is the channel output.

$$N_0 = \frac{E_0}{10^{\frac{SNR}{10}} \times R_{code}},$$

where E_0 is set to be 1, and R_{code} is the code rate that related to the size of the parity check matrix, which is 0.5 for all the tests.

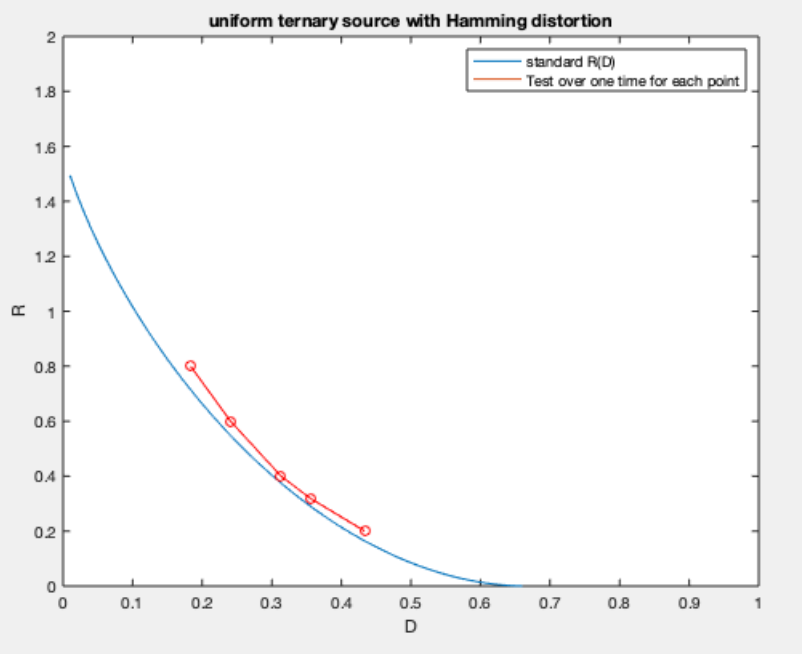

FIGURE 4.25: Uniform Ternary Source R(D) curve

During the experiments, the channel input x is a randomly generated binary sequence, which is uniformed-distributed. The signal-to-noise ratio (SNR) varies from 1.4 to 4.0, and the test measurement is based on the bit-error-rate (BER). The experiments are based on the parity check matrices with size 1250×2500, 512×1024, 256×512 and 128×256. The code rate is set to be 0.5 for all tests. Notice that some groups of test do not provide the results for high SNR values because the error becomes very small ($e < 10^{-6}$).

SNR	BER	SNR	BER
1.4	0.1196	2.6	0.0782
1.6	0.1032	2.8	0.0545
1.8	0.0987	3.0	0.0352
2.0	0.0974	3.2	0.0128
2.2	0.0943	3.4	0.003082
2.4	0.0796	3.6	0.0004117

TABLE 4.11: 1250×2500 with $R_{code} = 0.5$ (w_r, w_c)=(16, 7)

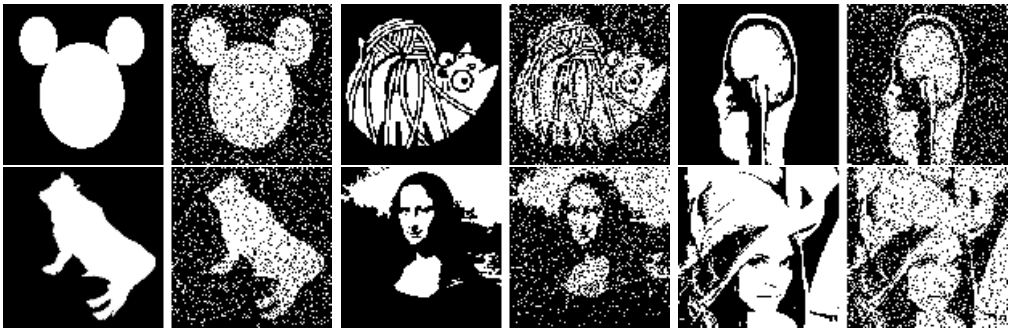

FIGURE 4.26: encoded image test result

SNR	BER	SNR	BER
1.4	0.0432	2.0	0.000489
1.6	0.0079	2.2	0.000113
1.8	0.00148	2.4	4.328×10^{-5}

TABLE 4.12: 512×1024 with $R_{code} = 0.5$ (w_r, w_c)=$(6, 3)$

SNR	BER	SNR	BER
1.4	0.01269	2.6	0.0003602
1.6	0.01575	2.8	4.733×10^{-5}
1.8	0.0085	3.0	2.7915×10^{-5}
2.0	0.00343	3.2	1.327×10^{-5}
2.2	0.00244	3.4	8.857×10^{-6}
2.4	0.000756	3.6	0

TABLE 4.13: 256×512 with $R_{code} = 0.5$ (w_r, w_c)=$(6, 3)$

SNR	BER	SNR	BER
1.4	0.033	2.8	0.00114
1.6	0.0243	3.0	0.0004413
1.8	0.0225	3.2	9.292×10^{-5}
2.0	0.0102	3.4	0.0001096
2.2	0.00893	3.6	2.051×10^{-5}
2.4	0.00327	3.8	6.319×10^{-6}
2.6	0.002125	4.0	0

TABLE 4.14: 125×256 with $R_{code} = 0.5$ (w_r, w_c)=$(6, 3)$

4.4.4 Nested-LDGM-LDPC watermarking Sample Results

Deterministic Mapping

s	$u=a$	$u=b$	$u=c$
0	0.12	0	0.78
1	0.12	0	0

TABLE 4.15: The deterministic mapping for x=0

s	$u=a$	$u=b$	$u=c$
0	0	0.12	0
1	0	0.12	0.78

TABLE 4.16: The deterministic mapping for x=1

This subsection will show two tables which provides the deterministic mapping of the jointly typical pair $\langle S^n, U^n, X^n \rangle$, where the small letter u,s x are the bit components in the corresponding sets, and a, b and c are ternary symbols.

Implementation

The section will introduce a simple case example that demonstrated by Dr. Mahdi. During him demonstration, the source sequence S^n with $n = 10000$ and information hiding capacity is around 0.0025 which is demonstrated from previous section 3.2.2, then the message m_i^k, where $i \subseteq \{0, 1, 2, ..., 2^{nR}-1\} = \{0, 1, 2, ..., 2^{250}-1\}$. A multilevel LDGM code (Nangir et al. 2018) is constructed with $n = 10000$ $w_r = 20$ and $t = 100$. Then, the LDGM code rate is indicated as

$$R_{LDGM} = \frac{t}{n} = 0.01$$

Since the overall information capacity is around 0.025, the code rate of the LDPC code is then given by,

$$R_{LDPC} = \frac{n-t}{n} = 0.01 + 0.025 = 0.035$$

According to a similar code design in the paper (Nangir et al. 2018), Dr. Mahdi successfully partitioned the matrix and get an example result.

Again, the target of the process is generally two. First of all, to minimize the distortion between the source sequence and the watermarked sequence, where the watermarked sequence is also the channel input in our proposed method. Second, to minimize the bit-error-rate between the watermark message m^k and the recovered watermark message $\hat{m}^k$. According to the experimental results, the Hamming distortion between X^n and S^n is given by 0.0735.

After getting the proper value of X^n, the Y^n is given by

$$Y^n = X^n \oplus Ber(0.3)$$

where the cross-over probability of the channel is set to be 0.3.

For the last step, the decoding process is done with the corresponding well-designed parity check matrix by the Sum-Product message-passing algorithm to find the proper codeword.

Finally, the decoding BER is defined to be

$$P_{error} = 1 - P_{correct} = 0.0053,$$

which indicates that in this example, there is a 0.53% chance the decoding result is not a proper codeword. Consequently, this demonstrates the robustness of the watermarking scheme, and the information hiding capacity C reaches 0.025 with comparatively high robustness and low bit-error-rate.

Chapter 5

Conclusion and Future Work

In general, this thesis provides a review of several general watermarking algorithms, including the algorithms work in the spatial domain, as well as those that work in the transformed domain. We proposed an implementation comparison among the classical algorithms LSB, DCT, and SVD with the same embedding rate to see their advantages and disadvantages.

After the review part, two proposed method is stated to see whether it is possible to improve both robustness and imperceptibility while keep good embedding rate at the same time.

For the first proposed method, which is a hybrid structure with the LDPC+SVD+DWT watermarking scheme, we completed two groups of tests. Firstly, we compared its performance with Lin's method (Lin and Wan 2016) and obtained comparatively better results in both imperceptibility and robustness. Secondly, we compared the proposed hybrid scheme with LSB, SVD, and DCT by setting the same embedding rate. According to the experimental results, our proposed hybrid scheme has a better imperceptibility result than the LSB. For the robustness test, our proposed hybrid scheme has the best performance over GS, GF, and SPE attacks. Besides, it also achieves a relatively good result in resisting JPEG compression, which is practical in reality.

While for the second part, the simple case results are based on experimental trials to get the proper parity check matrix and the

corresponding generator matrix. In order to make it into a general case, future work is mainly focused on how to find an efficient way to partition huge matrices and directly find out the proper generator matrix G concerning the given well-designed parity check matrix H.

Bibliography

Ahmad A.Mohammad, A. A. and Shaltaf, S. (2008). An improved SVD-based watermarking scheme for protecting rightful ownership. *Signal Processing* 88-9.

Aslantas, V. (2009). An optimal robust digital image watermarking based on SVD using differential evolution algorithm. *Optics Communications* 282-5.

Barni, M., Bartolini, F., Cappellini, V., and Piva, A. (1998). *Copyright Protection of Digital Images by Embedded Unperceivable Marks.*

C.-C. Chang, P. T. and Lin, C.-C. (2005). Svd-based digital image watermarking scheme. *Pattern Recognit. Lett* 26.

Cox, I. J., Kilian, J., Leighton, F. T., and Shamoon, T. (Dec. 1997). Secure Spread Spectrum Watermarking for Multimedia. *Trans. Img. Proc.* 6(12), 1673–1687. ISSN: 1057-7149.

Filler, T. and Fridrich, J. J. (2007). Binary quantization using Belief Propagation with decimation over factor graphs of LDGM codes. *CoRR* abs/0710.0192.

Fridrich, J. and Filler, T. (2007). Practical methods for minimizing embedding impact in steganography. In: *Security, Steganography, and Watermarking of Multimedia Contents IX*. Ed. by E. J. D. III and P. W. Wong. Vol. 6505. International Society for Optics and Photonics. SPIE, 13–27.

Gallager, R. (1962a). Low-density parity-check codes. *IRE Transactions on Information Theory* 8(1), 21–28.

Gallager, R. (1962b). Low-density parity-check codes. *IRE Transactions on Information Theory* 8(1), 21–28.

GOLUB, G. H. and REINSCH, C. (1970). Singular Value Decomposition and Least Squares Solutions. *Numerische Mathematik* 14.

Hsu, C.-T. and Wu, J.-L. (1999). Hidden digital watermarks in images. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 8 1, 58–68.

Jin, C. and Wang, S. (2007). Applications of a Neural Network to Estimate Watermark Embedding Strength, 68–68.

Johnson, S. J. (2000). Introducing Low-Density Parity-Check Codes.

Kaewamnerd, N. and Rao, K. R. (2000). Wavelet based image adaptive watermarking scheme. *Electronics Letters* 36(4), 312–313.

Kschischang, F. R., Frey, B. J., and Loeliger, H. .-.-. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 47(2), 498–519.

Kundur, D. and Hatzinakos, D. (1998). Digital watermarking using multiresolution wavelet decomposition. 5, 2969–2972 vol.5.

Lin, W., Horng, S., Kao, T., Fan, P., Lee, C., and Pan, Y. (2008). An Efficient Watermarking Method Based on Significant Difference of Wavelet Coefficient Quantization. *IEEE Transactions on Multimedia* 10(5), 746–757.

Lin, Y. and Wan, Y. (2016). A new gray-scale watermark method based on irregular LDPC codes with Unequal Error Protection (UEP). *Eighth International Conference on Digital Image Processing (ICDIP 2016).*

MacKay, D. J. C. (1999). Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory* 45(2), 399–431.

Moulin, P. and O'Sullivan, J. (1999). Information-Theoretic Analysis of Information Hiding. *IEEE Transactions on Information Theory* 49, 563–593.

Nangir, M., Ahmadian-Attari, M., and Asvadi, R. (2018). Binary Wyner–Ziv code design based on compound LDGM–LDPC structures. *IET Communications* 12(4), 375–383.

Noore, A. (2003). An improved digital watermarking technique for protecting JPEG images, 222–223.

Richardson, T. J. and Urbanke, R. L. (2001a). Efficient encoding of low-density parity-check codes. *IEEE Transactions on Information Theory* 47(2), 638–656.

Richardson, T. J. and Urbanke, R. L. (2001b). Efficient encoding of low-density parity-check codes. *IEEE Transactions on Information Theory* 47(2), 638–656.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell System Technical Journal* 27(3), 379–423.

Shih-Hao Wang and Yuan-Pei Lin (2004). Wavelet tree quantization for copyright protection watermarking. *IEEE Transactions on Image Processing* 13(2), 154–165.

Shokrollahi, A. (2003). LDPC Codes: An Introduction.

Tanner, R. (1981). A recursive approach to low complexity codes. *IEEE Transactions on Information Theory* 27(5), 533–547.

Vivek Singh Verma, R. K. J. (2015). An Overview of Robust Digital Image Watermarking. *IETE* 32:6.

Wainwright, M. J. and Maneva, E. N. (2005). Lossy source encoding via message-passing and decimation over generalized codewords of LDGM codes. *CoRR* abs/cs/0508068.

Weigt, M. and Zhou, H.-J. (Nov. 2006). Message passing for vertex covers. *Physical review. E, Statistical, nonlinear, and soft matter physics* 74, 046110.

Wenwu Zhu, Zixiang Xiong, and Ya-Qin Zhang (1999). Multiresolution watermarking for images and video. *IEEE Transactions on Circuits and Systems for Video Technology* 9(4), 545–550.

Zhu, J., Kaplan, R., Johnson, J., and Fei-Fei, L. (2018). HiDDeN: Hiding Data With Deep Networks. *CoRR* abs/1807.09937.