Data hiding for authentication of binary images

Yang, Huijuan

2008

Yang, H. (2008). Data hiding for authentication of binary images. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/3539>

<https://doi.org/10.32657/10356/3539>

Nanyang Technological University

Downloaded on 26 Apr 2023 16:25:30 SGT

Data Hiding For Authentication of Binary Images

Yang Huijuan

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Doctor of Philosophy

2008

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

.....

Date

.....

Yang Huijuan

To my families for their encouragements and love.

Acknowledgements

First and foremost, I would like to express my sincere thanks to my thesis advisor, Professor Alex C. Kot, for bringing the problem of binary images authentication to my attention and allowing me full freedom to explore. His confidence in my abilities to carry out the research work is greatly appreciated. His emphasis on the demonstrability of the ideas has immensely influence my research work. The tireless supports and invaluable inspiration from Prof. Kot cannot be emphasized enough. His sharp insights and excellent guidance; his encouragements in every stage of my research work; his patience and endeavor in improving every single word in papers and patents and his criticism and strict attitude towards the research work have been a resource and model for me to complete this thesis and it will continuously benefit me throughout my lifetime.

I wish to acknowledge all the panel members of this thesis for their time, valuable comments and supports. I wish to express my sincere thanks to Dr. Liu Jun for the enlightening discussions and collaborations. I wish to thank Professor Chan Choong Wah, Professor Farook Sattar and Professor Jagdish C. Patra for the encouragements. I want to thank Professor Yap Peng Tan for the helpful suggestions. I would like to thank all my subject teachers, especially, Professor Zhu Ce, Professor Xue Ping, Professor Lin Zhiping, Professor Anamitra Makur, Professor Yap Kim Hui, Professor Huang Guangbin, Professor Tan Chik How and Professor Yi Xun, the basic theory and skill that I have learned are the solid foundation of my research work. I am grateful to Dorothy Tay, Stewart Chu, Phebe Romina Tjahjono and Tee Siaw Furn, Evelyn for their kind supports. I wish to

express my appreciations to my colleagues, Lu Haiping, Zeng Jiancheng, Gao Yupeng, Cao Hong and Chen Jianping and my fellow graduate students, Cheng Jun, Fang Tao, Cui Shi, Chang Lanlan, Liu Manhua and Bappaditya Mandal for the discussions and friendship.

My dream towards a PhD can never be possible without the strong supports from my family. I am indebted to my grandmother (Zheng Guozhi) and my parents (Yang Yongzhi and Xiang Guifen) for instilling me the perseverance and hard-working from childhood which benefits me no matter what I choose to do. I am grateful to my brother, Professor Yang Guang-Hong for his inspiration and encouragements, and my sister-in-law, Zhai Ying for her care during the course of my study.

Finally, I would like to dedicate this work to my husband Zhao Jian and my dearest son Zhao Songyuan for their persistent love, endless support and sacrifice. I also wish to acknowledge the financial supports for this thesis, which are provided by Research Grant Management RGM6/02, Nanyang Technological University, Singapore.

Abstract

The advancement in digital technology and multimedia has made it easy to duplicate, manipulate and illegally distribute the multimedia content, which brings serious concerns in copyright protection and authentication. In addition, it is easy to forge an image with the powerful image editing software and digital cameras. In this thesis, we propose data hiding techniques that operate on different level (e.g., object and pixel) and on different domain (e.g., spatial and transform) for the purpose of binary images authentication and tampering localization. Issues in capacity, visual quality, system security, computational complexity and accuracy in tampering localization are addressed. The validity of the methods is demonstrated by both quantitative results and theoretical analysis.

We firstly investigate data hiding for text images in object level by integrating the inter character and word spaces. A new concept of maximum inter character space is introduced such that the relationship between the inter character and word spaces can be established. The security level of the data hiding system has been enhanced by randomly shifting the characters as well as by embedding the authenticator watermark. This method can be applied to a wide variety of text documents authentication, especially for the applications that do not allow the distortions introduced by flipping pixels in binary images.

How to identify the suitable locations to carry the information data is a key issue for data hiding techniques operating in pixel level. We propose a “Connectivity-Preserving” criterion to assess the “flippability” of a pixel to preserve the connectivity of pixels in a

3×3 pixel neighborhood, hence the visual quality of the watermarked image is ensured. The problem of how to locate an “embeddable” pixel in a block is addressed, which facilitates the incorporation of the cryptographic signature to address the security issue. Different ways of partitioning an image are investigated that achieve from moderate to large capacity and good visual quality.

Data hiding for binary images in transform domain is investigated. Directly using the detail coefficients to determine the data hiding locations is difficult to achieve blind watermark extraction. Hence, we view flipping an edge pixel in binary images as the “Edge Shifting” phenomenon. To this end, we propose an interlaced morphological binary wavelet transform such that the coefficients can be used to track the shifted edges. This facilitates blind watermark extraction and incorporation of cryptographic signature. Two processing cases that the “flippability” condition of one does not affected by flipping the candidates of another can be used for “Orthogonal Embedding”. Processing an image based on 2×2 blocks and employing “Orthogonal Embedding” solve the “un-embeddability” problem in the boundary or sharing rows or columns in block-based approach. As a result, the efficiency of utilizing “flippable” pixels has been increased.

Localization of tampering for binary images is challenging due to the uneven distribution of the “flippable” pixels. On top of the data hiding schemes proposed, we propose a two-layer authentication scheme, in which the first layer is targeted at overall authentication and the second layer is targeted at identifying the tampered locations. Multiple finer-blocks are grouped to form a macro-block and the macro-blocks are classified. The block identifier is defined adaptively for each class and embedded in those “Qualified” and “Self-Detecting” macro-blocks such that any tampering occurred either to the macro-blocks that have a lot of “flippable” pixels or the macro-blocks that have few or no “flippable” pixels, the alteration can be located.

Contents

Acknowledgements	i
Abstract	iii
List of Figures	ix
List of Tables	xii
List of Acronyms	xiii
List of Symbols	xv
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Major Contributions of the Thesis	3
1.4 Organization of the Thesis	5
2 General Overview	7
2.1 Multimedia Data Hiding	7
2.2 Binary Images Data Hiding	10
2.3 Design Models	15
2.4 A General Framework	17
2.5 Performance Measures	20
3 Integrated Inter Character and Word Spaces Data Hiding	24
3.1 The Proposed Algorithm	25
3.1.1 Embeddable Segment Determination	25

3.1.2	Data Embedding and Authentication	27
3.1.3	Data Extraction and Verification	30
3.2	Experimental Results and Comparisons	30
3.3	Discussions	35
3.4	Chapter Summary	37
4	Connectivity-Preserving Pattern-Based Data Hiding	38
4.1	Background and Objectives	38
4.2	Data Hiding Mechanism	40
4.2.1	Flippability Decision	40
4.2.2	Block Partition and Embeddability	43
4.2.3	Capacities	45
4.2.4	Watermark Embedding and Extraction	46
4.3	Authentication Mechanism	49
4.3.1	Locate Embeddable Pixels Criterion	49
4.3.2	Authentication Process	52
4.4	Experimental Results	55
4.4.1	Capacity and Visibility	55
4.4.2	Authentication Results	56
4.4.3	Comparisons	61
4.5	Discussions	65
4.5.1	System Security	65
4.5.2	Security Against Steganalysis and VQ Attack	66
4.5.3	Visual Quality	67
4.5.4	Computational Complexity	68
4.6	Chapter Summary	70
5	Morphological Transform-Based Data Hiding	71
5.1	Background and Motivation	72
5.2	Interlaced Morphological Binary Wavelet Transform	75
5.2.1	Signal Analysis and Synthesis	75
5.2.2	Interlaced Morphological Binary Wavelet Transform	79
5.3	Data Hiding Based on <i>IMBWT</i>	83
5.3.1	Single Processing	84

5.3.2	Control Sharp Corner (Isolated Corner) Flipping	87
5.3.3	Double Processing	88
5.3.4	Double Processing with Distortion Control	90
5.3.5	Authentication Mechanism	92
5.4	Lifting Scheme	92
5.4.1	Forward and Inverse <i>LMBWT</i> for 1-D Signal	93
5.4.2	Forward and Inverse <i>LMBWT</i> for 2-D Signal	97
5.5	Experimental Results	102
5.5.1	Coefficients Comparison	102
5.5.2	Visual Quality and Capacity	104
5.5.3	Comparisons	106
5.5.4	Watermark Effectiveness and Security Concerns	110
5.6	Flippability Decision in Larger Block	112
5.7	Applications	115
5.8	Relationships of <i>CPDH</i> and <i>MTDH</i>	116
5.9	Chapter Summary	119
6	Authentication With Tampering Localization	120
6.1	General Overview	121
6.2	Flippability Decision and Block Signature Generation	122
6.3	Authentication and Tampering Localization	125
6.3.1	Macro-Block Classification	125
6.3.2	Distance Computation	126
6.3.3	Block Identifier Formation	126
6.3.4	Authentication Data Generation and Embedding	128
6.3.5	Authentication and Tampering Localization	129
6.4	Experimental Results and Discussions	131
6.4.1	Tampering Localization Results	131
6.4.2	Comparisons	134
6.4.3	Discussions and Security Considerations	135
6.5	Chapter Summary	137
7	Conclusions and Future Work	140
7.1	Conclusions	140

7.2 Future Work	143
Appendix	145
A Capacity Analysis for $CPDH$	145
A.1 Capacities for Interlaced Block Scheme	145
A.2 Capacities for Non-Interlaced Block Scheme	147
Author's Publications	151
Bibliography	153

List of Figures

2.1	Classifications of Data Hiding Techniques for Binary Images.	10
2.2	Effects of Arbitrarily Flipping Pixels in Binary Images.	11
2.3	Tool Sets Involved in a Binary Image Data Hiding System.	16
2.4	A General Framework.	20
3.1	Left and Right Distances of a Character.	25
3.2	A Sample Profile.	27
3.3	Illustration of the Overlapping Windows of Size Three.	27
3.4	Four Combinations of the Left and Right Distances of a Character.	28
3.5	Hiding Effects on an English Text Image.	31
3.6	Hiding Effects on a Chinese Text Image.	31
3.7	Authentication Results.	33
3.8	Number of Pixels Shifted	34
3.9	Comparison of Data Hiding Effects on an English Text Image.	34
4.1	Designations of Pixels in a 3×3 Neighborhood.	41
4.2	The 3×3 Block Patterns	42
4.3	Example of “Flippable” and “Non-Flippable” Pixels.	43
4.4	Different Schemes of Partitioning an Image.	44
4.5	Probability of Each Block To Be “Embeddable”.	45
4.6	Previous Four and Next Four Neighbors of Current Pixel.	48
4.7	Examples of Flipping a Pixel Changes the “Flippability” of its $P-4$ Neighbors	48
4.8	Examples of Flipping a Pixel Changes the “Flippability” of its $N-4$ Neighbors	49
4.9	$P-4$ Neighbors and Those Already Processed $D-2$ Pixels	51
4.10	Illustration of Flipping a Pixel may Affect the “Embeddability” of $D-2$ Pixels.	51
4.11	Hard Authenticator Watermark Embedding Process.	53

4.12	Hard Authenticator Watermark Extraction and Verification Process.	54
4.13	Capacity of Using Different Block Sizes.	57
4.14	Comparison of Hiding Effects Using Different Types of Blocks.	58
4.15	Capacity of Imposing <i>LEP</i> Criterion versus Full Capacity for <i>NIB</i> Scheme.	60
4.16	Capacity of Imposing <i>LEP</i> Criterion versus Full Capacity for <i>IB</i> Scheme.	61
4.17	Authentication Results.	62
4.18	Comparison Results for Low Capacity.	63
4.19	Comparison Results for High Capacity.	64
4.20	List of Patterns that Satisfy the “Flippability Criterion”.	68
5.1	Illustration of “ <i>Edge Shifting</i> ” Phenomenon	75
5.2	Designations of Samples in a 2×2 Block.	78
5.3	Illustration of 1-D <i>IMBWT</i>	80
5.4	Different 2×2 Blocks in a Binary Image.	80
5.5	Four Different 2×2 Blocks in a 3×3 Block of a Binary Image.	81
5.6	Main and Subsidiary Blocks for Even-Even Processing Case.	83
5.7	Illustration of Flipping a Pixel Changes the Four Surrounding 2×2 Blocks.	84
5.8	List of Patterns that Satisfy the “Cross” Condition	86
5.9	List of Sharp Corner and Isolated Corner Patterns.	87
5.10	Flippability Conditions Checking for a “Pair Case”	89
5.11	Designated 2×2 Blocks.	90
5.12	Illustration of Capacity Increase by Employing “Orthogonal Embedding”	91
5.13	Watermark Generation and Embedding Using <i>IMBWT</i>	93
5.14	Watermark Extraction and Verification Using <i>IMBWT</i>	93
5.15	Illustration of 1-D <i>LMBWT</i>	96
5.16	Designations of Samples in a 3×3 Block.	98
5.17	Coefficients Comparison	103
5.18	Hiding Effects of Different Processing Cases	105
5.19	Comparisons of Capacity Increase for Different Processing Cases.	106
5.20	Comparisons of Total Distortion for Different Processing Cases.	106
5.21	Comparisons of Average Per Pixel Distortion for Different Processing Cases.	107
5.22	Comparison of Data Hiding Results Using Different Methods.	109
5.23	Authentication Results	111

5.24	Example of Dominant Edge and Follow Up Edges in 3×3 Blocks	113
5.25	Illustration of “Pair Flipping” and “Single Flipping”	113
5.26	Example of “Triangle Noise” Patterns.	114
5.27	Example of “End Stroke” Patterns.	114
5.28	Efficiency of Utilization of “Flippable” Pixels for <i>CPDH</i>	118
6.1	Macro-Blocks and Finer-Blocks.	123
6.2	Weights Assignment and Pixels of the Image	124
6.3	Block Identifier For Different Classes.	128
6.4	Authentication Data Generation and Embedding.	129
6.5	Data Extraction, Authentication and Tampering Localization.	130
6.6	Results of Data Hiding and Tampering Localization.	131
6.7	Comparison of Data Hiding Results.	135
6.8	Comparison of Tampering Localization Results.	139
A.1	X Versus Different Probabilities p	147
A.2	Capacity Versus Different Block Sizes for Interlaced Block Scheme.	148
A.3	T Versus Different Probabilities p	149
A.4	Capacity Versus Different Block Sizes for Non-interlaced Block Scheme.	150

List of Tables

4.1	Capacity Comparison for Different Types of Blocks	55
4.2	Total Distortion of Using Different Types of Blocks	59
4.3	Average Per Pixel Distortions ($\mathcal{D}_a$) for Different Methods	63
4.4	Scores Assigned for Different Patterns	68
4.5	Computational Load Comparison	69
5.1	Inputs to the Watermark Generation and Embedding Using <i>IMBWT</i> . . .	94
5.2	Strategy of Watermark Generation and Embedding Using <i>IMBWT</i>	95
5.3	Strategy of Watermark Extraction and Authentication Using <i>IMBWT</i> . . .	96
5.4	Computational Complexity of Using <i>IMBWT</i> and <i>LMBWT</i>	102
5.5	Entropy of Wavelet Coefficients	103
5.6	Comparisons of Capacity for Different Methods	108
5.7	Comparisons of Average Per Pixel Distortion for Different Methods	108
6.1	Macro-Blocks Classification	125

List of Acronyms

APPD	Average Per Pixel Distortion
BI	Block Identifier
C	Corner
CP	Connectivity-Preserving
CPDH	Connectivity-Preserving Pattern-based Data Hiding
CS	Cryptographic Signature
CSCF	Control Sharp Corner (Isolated Corner) Flipping
DC	Direct Current
DCT	Discrete Cosine Transform
DPC	Double Processing Case
DPDC	Double Processing With Distortion Control
DS	Digital Signature
ECC	Error Correction Coding
EEB	Even-Even Block
EOB	Even-Odd Block
ER	Embedding Rule
ES	End Stroke
FB	Finer-Block
FXB	Fixed 3×3 Block
IB	Interlaced Block
ICWS	Integrated Inter Character and Word Spaces Data Hiding
IMBWT	Interlaced Morphological Binary Wavelet Transform
IR	Interior Right Angle
LEP	Locate Embeddable Pixels
LMBWT	Lifting scheme of Morphological Binary Wavelet Transform
LUT	Look Up Table

MAC	Message Authentication Code
MB	Macro-Block
MBWT	Morphological Binary Wavelet Transform
MTDH	Morphological Transform-based Data Hiding
NIB	Non-Interlaced Block
OEB	Odd-Even Block
OOB	Odd-Odd Block
QMB	Qualified Macro-Block
SPC	Single Processing Case
TC	Transition Change
TD	Total Distortion
UMB	Unqualified Macro-Block
VH	Vertical and Horizontal
VQ	Vector Quantization
XOR	Exclusive OR
XR	Extraction Rule

List of Symbols

Symbols Used in Chapter 2

Y, Y_w	original and watermarked binary images.
$\mathcal{B}_f$	block or segment feature.
$\mathcal{C}_f()$	compute block or segment features.
$\mathcal{W}_p, \mathcal{W}'_p$	original and extracted payload watermark.
$E_m(), E_x()$	embedding and extraction functions.
$\mathcal{D}_e()$	determine embeddable location in a block.
$\mathcal{X}_f()$	extract block feature or generate intermediate image.
Y_1, Y_{w1}	intermediate images of Y and Y_w .
$Hash()$	hash function.
$E_k(), D_k()$	encryption and decryption functions.
$\mathcal{K}_s, \mathcal{K}_p$	private and public keys of an authorized user or owner.
$\mathcal{C}_m()$	function to combine $\mathcal{W}_s$ with $\mathcal{W}_p$.
$\mathcal{W}_s, \mathcal{W}'_s$	original and extracted cryptographic signatures.
$\mathcal{W}_r, \mathcal{W}'_r$	original and extracted authenticator watermarks.
$\mathcal{H}_o, \mathcal{H}_w$	hash values of original and watermarked images.
$\mathcal{S}_f, \mathcal{S}_d$	flippability score and distortion score.
$\mathcal{D}_t, \mathcal{D}_a$	total distortion and average per pixel distortion.
$\mathcal{R}_d()$	normalized hamming distance.
$\mathcal{P}_{md}, \mathcal{P}_{fa}$	probability of miss detection and probability of false alarm.

Symbols Used in Chapter 3

d_l, d_r	left and right distances of a character.
$\min(), \max()$	minimum and maximum values in the set.
$h(i), v(j)$	horizontal profile of i^{th} line and vertical profile of j^{th} column.
Φ_c, Φ_w	maximum inter character space and word space.

Symbols Used in Chapter 3

N_s	maximum number of shifts.
$\mathcal{R}_s$	random sequence.
$\mathcal{B}_s, \mathcal{B}_{sw}$	bitstream of original and watermarked image.
$\mathcal{C}_\zeta$	capacity of the image.
$Len()$	length of the element.

Symbols Used in Chapter 4

$\mathcal{N}_{vhw}, \mathcal{N}_{vhb}$	numbers of uniform white and black transitions.
$\mathcal{N}_{ir}$	number of interior right angle transitions.
$\mathcal{N}_c$	number of sharp corner transitions.
$\mathcal{B}_z$	block size.
$\mathcal{P}_{eb}$	probability of a block to be embeddable.
$\mathcal{N}_k$	total number of blocks.
$\mathcal{N}_b$	number of black pixels in a block.
$\mathcal{N}_d$	number of determined pixels in a block.

Symbols Used in Chapter 5

V_l, W_l	signal space and detail space at level l .
ψ^+, ϖ^+	signal and detail analysis operators.
ψ^-, ϖ^-	signal and detail synthesis operators.
$I, \underline{0}$	identity and zero operators.
$\mathbb{Z}$	nonnegative integer set.
$s(i), d(i)$	coarse signal and detail signal at level $l + 1$ for 1-D transform.
ψ^o, ϖ^o	operators of coarse and detail signals for 1-D odd transform.
$\psi^k, \varpi_v^k, \varpi_h^k, \varpi_d^k$	operators of coarse signal, vertical, horizontal and diagonal detail signals for 2-D transform k .
$s^k(i, j), h^k(i, j)$	coarse signal, horizontal, vertical and diagonal detail signals at level $l + 1$ and indexed at (i, j) for 2-D transform k .
$v^k(i, j), d^k(i, j)$	
$\mathcal{B}_p(i, j)$	2×2 block indexed at (i, j) .
$f_k(i, j)$	flippability of the coarse signal indexed at (i, j) for transform k .
$\mathcal{N}_h, \mathcal{N}_v$	numbers of transitions in horizontal and vertical directions.
$\mathcal{P}_b(i, j)$	pair block indexed at (i, j) .
$\mathcal{D}_g(i, j)$	dominant edge block indexed (i, j) .

Symbols Used in Chapter 5

$\mathcal{D}_b(i, j)$	designated 2×2 block indexed at (i, j) .
$\mathcal{P}_{ro}, \mathcal{P}_{rw}$	original and watermarked patterns.
$\mathcal{Q}_{ee}, \mathcal{Q}_{oo}$	embedders indexed at even-even and odd-odd 2×2 blocks
ψ^p, ϖ^p	operators for coarse and detail signals of 1-D <i>LMBWT</i> .
$\psi^{pu}, \varpi_h^{pu},$ $\varpi_v^{pu}, \varpi_d^{pu}$	operators for coarse signal, horizontal, vertical and diagonal detail signals of 2-D <i>LMBWT</i> .
$s_k^{pu}(i, j), v_k^{pu}(i, j),$ $h_k^{pu}(i, j), d_k^{pu}(i, j)$	coarse signal, vertical, horizontal and diagonal detail signals at level $l + 1$ and indexed at (i, j) of <i>LMBWT</i> .
$P_k(i, j)$	flippability of signal indexed at (i, j) for processing case k .
$\mathcal{P}_f$	probability for an adversary to figure out data hiding locations.
$\mathcal{H}(r_o)$	entropy of r_o .
η_e	efficiency of utilization of the “flippable” pixels.

Symbols Used in Chapter 6

$f_c(x, y)$	feature code of a finer block centered at $f(x, y)$.
$\bar{f}_c(k, l)$	mean feature code of region (k, l) .
$\mathcal{B}_s(k, l)$	block signature of region (k, l) .
$\mathcal{N}_f$	number of flippable pixels in a macro-block.
$\mathcal{D}_x, \mathcal{D}_y$	horizontal and vertical distances between two <i>QMBs</i> for Y .
$\mathcal{D}_{xw}, \mathcal{D}_{yw}$	horizontal and vertical distances between two <i>QMBs</i> for Y_w .
$\mathcal{I}_c, \mathcal{J}_c$	indices of current <i>QMB</i> in horizontal and vertical directions.
$\mathcal{I}_p, \mathcal{J}_p$	indices of previous <i>QMB</i> in horizontal and vertical directions.
$\mathcal{S}_x, \mathcal{S}_{xw}$	signs for $\mathcal{D}_x$ and $\mathcal{D}_{xw}$.
$\mathcal{F}_{ub}$	features of <i>UMBs</i> that lie between two <i>QMBs</i> for Y .
$\mathcal{F}_{ubw}$	features of <i>UMBs</i> that lie between two <i>QMBs</i> for Y_w .
$\mathcal{N}_x$	number of <i>MBs</i> in the horizontal direction.
$N_c(k)$	number of bits consumed to embed <i>BI</i> in the k^{th} <i>MB</i> .
$\mathcal{C}_s$	number of bits required to embed <i>CS</i> .
N_a	available number of bits to embed <i>CS</i> .
P_e	probability of a bit to be wrong.
$P_B(k)$	average probability of error for the k^{th} <i>MB</i> .
P_M	total probability of error for a total of n <i>MBs</i> .

Chapter 1

Introduction

1.1 Motivation

Authentication of digital documents has aroused great interest due to the wide application areas such as bank checks, legal documents, certificates, books, maps and engineering drawings in digital form. Digital documents are usually stored in binary image format. In addition, more important documents such as fax, insurance, personal files and passports are digitized and stored to facilitate processing and data recovery with the aid of the scanner and software such as “Digital Vault” [1]. Similar to the hard-copy document processing, in which authentication is an important procedure such as endorsing on a document, ensuring the authenticity and integrity has increased the confidence level in using the digital documents from the user point of view. On the other hand, editing an image becomes easier with the powerful image editing tools and digital cameras. Authentication and detection of tampering and forgery is thus of primary concern.

In another aspect, cryptographic authentication technique such as digital signature or message authentication code has been used for document authentication for a long time. The digital signature or message authentication code is generally transmitted by appending it in the header or as a separate file, which may be easily lost during lossless file format conversion [2]. Further, cryptography protects the communication channel by encrypting the document. However, once the encrypted document is decrypted, there is

no protection of the document at all, which is so called “protection in transmission”. In addition, the size increase in the document may easily cause the adversary’s suspicion. Whereas data hiding has the advantage of integrating the authentication data with the document and assumes protection even after the watermark is extracted. Therefore, any tampering occurs to the document may destroy the watermark as well. The need for document authentication and integrity protection, and the advantages of data hiding over digital signature based techniques motivate us to design data hiding-based methods for digital document authentication.

1.2 Objectives

In the past few years, many techniques are proposed for document data hiding [3]. Among these methods, the line, word and character shifting methods [4–9] have low capacity. Typically, these methods are designed mainly for copyright protection purposes in which the security issue has not been sufficiently addressed to be used for authentication purposes. Moreover, most of these techniques maybe vulnerable to the attacks of deliberate retyping and reformatting the documents. The randomness in choosing the data hiding locations results in poor visual quality of the watermarked images despite the high embedding rate, i.e., the number of bits embedded for each flipping [10–12]. A shuffling key is employed to distribute the “flippable” pixels for hiding the information in [13–15]. Practically, it is difficult to find a proper shuffle key such that each block in the shuffled domain has a suitable pixel to flip for small blocks. Hence a large block size is required which decreases the capacity and increases the computational load. Contour tracing is employed to find suitable pixels for data embedding along the edges of the characters and words [16]. The minimum length of a contour segment is required so that the capacity is not high. Localization of tampering for binary images is seldom addressed due to the uneven distribution of the “flippable” pixels and low capacity of binary images. The low accuracy of the tam-

pering localization is achieved in [17] due to employing a larger sub-image size. Moreover, the uniform white or black areas that do not participate in the data hiding process are easily tampered. Data hiding for binary images based on real transforms is known to be difficult due to the quantization error introduced in the pre- or post-processing [18]. The visual quality of the watermarked images obtained by employing the direct current (DC) components of the discrete cosine transform (DCT) [19] is more obtrusive.

Considering the present situations in data hiding for binary images, we investigate data hiding techniques that operate in object and pixel levels. We design schemes to address the issues in computational load, capacity, visual quality and security for the purpose of binary images authentication, especially for text document images authentication. Data hiding in transform domain is investigated and the morphological binary wavelet transform is employed for data hiding to address the quantization error issue exists in DCT -based approach. A tampering localization scheme is proposed to achieve high accuracy in localization by embedding the macro-block identifier. Authentication of the electronic documents in binary images format is the focus of this thesis, whereas authentication of the documents in paper format is out of the scope of this thesis.

1.3 Major Contributions of the Thesis

The problem we address in this thesis is data hiding for binary images for authentication purposes. To this end, we address the following issues relating to this problem: 1) how to create linkage between the inter character and word spaces for data hiding; 2) how to choose suitable pixels/segments to carry the watermark so that the visual distortion of the resultant watermarked image is low; 3) how to handle the uneven “embeddability” of the host image such that the “flippable” pixels can be efficiently utilized; 4) how to utilize the invariant features in data embedding process to extract the watermark

without requiring the presence of the original image; 5) how to increase the capacity by partitioning the image in appropriate ways; 6) how to locate the “embeddable” pixels in a large block, “embeddable” segments and the “embeddable” coefficients to facilitate the incorporation of cryptographic signature and 7) how to detect and localize the tamperings occurred to the watermarked images. From Chapter 3 to Chapter 6 of this thesis, a detail description of the proposed approaches will be presented that highlights the following major contributions.

1. The proposal of introducing the concept of the maximum inter character space makes it possible to establish the relationship between the inter character and word spaces for data hiding. As a result, not only the capacity but also the system security have been improved compared with conventional line/word and character shifting methods. The proposed object-based technique is especially suitable for the applications that do not allow the changes introduced by employing the pixel-flipping techniques.

2. The proposal of using the “Connectivity-Preserving” criterion to evaluate the “flipability” of a pixel guarantees the visual quality of the watermarked image. The problem of how to locate an “embeddable” pixel in a large block is addressed. It combines with the “Connectivity-Preserving” criterion to facilitate the incorporation of cryptographic signature, which addresses the security issue in binary images data hiding. In addition, the invariant features in data hiding process help achieve blind watermark extraction and provide an effective way to handle the “uneven” embeddability of binary images. Different ways of partition an image are investigated, in which high capacity can be achieved by employing our proposed interlaced block partition scheme.

3. The proposal of employing the interlaced morphological binary wavelet transform provides a systematic way to find suitable locations to carry the watermark data, which addresses the “quantization error” issue in the *DCT DC*-based approach [19]. Using the interlaced transform to track the shifted edges helps to achieve blind watermark extraction

in morphological transform domain. Processing an image in 2×2 blocks and combining two processing cases solve the “un-embeddability” problem in the block boundary or in the sharing rows or columns of the block-based approach. As a result, the efficiency of utilization of the “flippable” pixels has significantly improved. A large capacity can be achieved while maintaining the low computational cost. Discussions on how to extend the “flippability” decision block from 3×3 to 4×4 or 5×5 are also presented.

4. Localization of tampering for binary images is seldom addressed in the literature due to the low capacity of binary images compared with that of grayscale or color images and the “uneven” embedding capacity of binary images. We address this problem by grouping multiple finer-blocks into a macro-block and embedding the dynamic block identifier and cryptographic signature, which shows good performance in terms of localization accuracy, probabilities of miss detection and false alarm. Our proposed tampering localization scheme can be incorporated with any block-based data hiding scheme that the “embeddable” locations can be determined in both the data embedding and extraction process. Determining the locations that are tampered and locations that are still credible help to analyze the objectives of the adversaries and hence counter measures can be designed.

1.4 Organization of the Thesis

This thesis is organized as follows.

- In Chapter 2, an overview on the data hiding methods for multimedia with a focus on binary images is presented. The design model, performance measure and a general framework for binary images data hiding are also discussed.
- In Chapter 3, data hiding in object level by integrating inter character and word

spaces is presented, which can be applied to a wide variety of text document images authentication.

- In Chapter 4, a pixel level “Connectivity-Preserving” pattern-based data hiding method is proposed. Discussions on how to partition an image to achieve large capacity and how to improve system security and visual quality are presented.
- In Chapter 5, data hiding in transform domain is investigated. An interlaced morphological binary wavelet transform is proposed for data hiding, in which a large capacity is achieved while maintaining the low computational cost.
- In Chapter 6, on top of the data hiding schemes proposed in Chapters 4 and 5, a two-layer authentication with tampering localization scheme is presented by embedding the cryptographic signature and macro-block identifier.
- In Chapter 7, conclusions are drawn and future research directions are suggested.

Chapter 2

General Overview

In this Chapter, a brief literature survey on data hiding for multimedia with a focus on binary images is conducted. A general design framework for binary images data hiding for the purpose of authentication is presented.

2.1 Multimedia Data Hiding

The history of secret communication can be traced back to ancient Greek [20,21]. To describe the secret communication by hiding one information into another data, “Steganography” and “Data Hiding” are commonly used. “Steganography” stands for the techniques used for secret communication [22,23], which generally relies on the assumption that the existence of the covert communication is unknown to the third parties and the methods are generally not robust. Data hiding and watermarking are different from cryptography which protects the content of the messages, as well as steganography which conceals the very existence of the messages [24]. “Watermarking” refers to the techniques to embed a visible or preferably invisible identification code permanently in the cover data and remains present within the cover data after any decryption process [25]. Hence, watermarking techniques usually have robustness requirements. The existence of the hidden information may not need to be kept secret. Data hiding refers to the technique for hiding

one information into the other information, making the hidden information imperceptible and keeping the existence of the information secret [25].

Due to the lacking of consensus in the literature on the definition of digital watermarking and data hiding, in this thesis, we ignore the small differences and mainly use *data hiding*. The cover data refers to audio such as speech and music, text images, images such as photographs and graphics and video. Comprehensive reviews of the data hiding techniques for multimedia are presented in [20–28]. Some fundamental issues that need to be addressed for a data hiding system are

- **Visual quality.** The perceptual similarity between the original and watermarked version of the cover work, in other words, the visual distortion incurred by embedding the watermark.
- **Capacity or payload data.** The number of bits that a watermark encodes within a unit of time or within a work, which may vary from a few bits to a large amount of data for different applications.
- **Blind or informed detector.** Whether the original is needed for the watermark detection. Having the original image available for the watermark detection or verification will improve the performance of the detector. However, it is not practical to distribute the original image to every customer in most applications.
- **Robustness.** The ability to detect the watermark regardless of whatever signal processing has happened to the host data, which is application dependent.
- **Security.** The ability to resist the hostile attacks such as unauthorized removal, embedding and detection [25], which is generally achieved by incorporating cryptographic signature (*CS*) or Message Authentication Codes (*MACs*).

Data hiding has many application areas, e.g., broadcast monitoring, ownership identification, transaction tracking, copy control, fingerprinting, authentication and annotation. The design requirements for a data hiding system are different catering for different applications. For example, high robustness is required for a system designed for ownership protection applications, whereas robustness is not a major concern for a system designed for authentication applications. Since the adversary would rather try every attempt to embed his own watermark and get the forged watermark verified than removing the original watermark.

Watermarking techniques for digital multimedia can be classified into two categories: robust and fragile/semi-fragile depending on different applications. A good robust watermark is designed to survive most signal processings such as rotation, scaling and translation [29, 30]. Fragile and semi-fragile watermarks, i.e., “hard authentication” and “soft authentication” watermarks, are generally designed for authentication and tampering detection and localization purposes. Any tampering occurred to the watermarked image may render the authentication fail for fragile watermarking, which is usually achieved through the use of the message authentication codes and cryptographic signatures [31, 32]. While semi-fragile watermarking differentiates the lenient manipulations, e.g., compression, from malicious ones, which can be performed on the salient features of the images [33–38]. The capacity limits of a data hiding system under the specified distortion measures are discussed in [39–42]. Embedding watermark into images unavoidably introduces artifacts and distortions to the host images, which may raise serious concerns in applications such as medical imaging, military imaging and law enforcement. Reversible data hiding methods proposed in [43–47] can be used to recover the original image losslessly once the payload watermark is extracted. A general quantization index modulation-based method [48] embeds the information by modulating the indices with the embedded information and then quantizes the host signal with the associated quantizers. Efficient tradeoffs can be

achieved in maximizing information-embedding rate, minimizing distortion between the host signal and watermarked signal, and maximizing the robustness of the embedding.

2.2 Binary Images Data Hiding

Similar to data hiding techniques for gray or color images, data hiding on binary images can be classified as *fragile* and *robust* watermarks depending on different applications. It can be classified as *spatial* and *transform* domain techniques based on the domain where data hiding occurs. In addition, it can be classified as *pixel-flipping* and *object*-based techniques depending on whether the watermark embedding is done in pixel level or object level. Specifically, a pixel-flipping technique encodes a message bit by flipping a pixel from white to black or vice versa, whereas an object-based technique encodes a message bit by modulating the spaces or features of characters and words. The proposed methods in Chapters 3 to 6 fall into different categories as shown in Figure 2.1.

Figure 2.1: Classifications of Data Hiding Techniques for Binary Images.

Most data hiding techniques are for gray or color images due to the wide range of pixel values. Slightly perturbing the pixel values does not cause obtrusive distortions under normal viewing conditions. On the contrary, binary images only have one bit-

plane, whereby each pixel takes one of the two possible colors, e.g., “1” represents black and “0” represents white. Arbitrarily flipping pixels in the non-edge regions in binary images creates dramatic difference as illustrated in Figure 2.2. Note that throughout

Figure 2.2: Effects of Arbitrarily Flipping Pixels in Binary Images. (a) the original letter “D”, (b) the letter “D” with 24 pixels being flipped and (c) the enlarged difference image.

this thesis, in the difference image, the foreground of the original image is colored in light gray, pixels in black represent the pixels that are flipped from white to black and pixels in dark gray represent the pixels that are flipped from black to white. Data hiding for binary images can be performed either in spatial domain or transform domain. Methods based on transform domain generally involve a pre/post-thresholding and binarization process which renders the visual quality of the resultant watermarked image more obtrusive. An early investigation on watermarking in transform domain is made in [18], in which the watermark is embedded in spatial domain by using the line and word shifting method to

achieve good visual quality while the watermark detection is done in frequency domain to achieve high robustness. A further development in [19] is to embed the watermark on DC components of DCT with the aid of a pre-blurring and post-biased binarization process.

Generally, the watermark in spatial domain can be embedded in binary images by 1) incorporating the odd-even feature of a group of pixels [13–15, 49–52]; 2) employing pairs of contour patterns [16]; 3) mapping the block features to the message bit [53] or enforcing the modulo value of the element-wise computation of the image, key and weight matrixes [10–12, 54]; 4) enforcing the ratios of black versus white pixels [55] or enforcing the neighborhood pixel ratios [56], and 5) enforcing the relationship or modulating the white spaces in neighboring lines, words, characters and segments [4–9, 57]. Several object-based methods shift the line, word and character to encode the watermark information in text documents. The line and word shifting methods [4, 5] have good robustness such that the watermark can survive the printing-photocopying-scanning process. The inter-word spaces coding [8, 9, 57] and inter character spaces coding [6, 7] methods achieve larger capacity compared with that of the line shifting method, but the robustness has decreased. The feature-based method [58] encodes the watermark by modifying the length of selected runs of a stroke. Generally speaking, the capacities of these methods are rather low and they are not target for authentication purposes.

Recently, significant efforts have been put in finding good “flippable” locations in binary images for watermark embedding to minimize the visual distortion [11–16, 49–53]. These methods include: defining a visual distortion table to assess the “flippability” of a pixel [13–15, 50–52]; defining pairs of contour edge patterns [16] and just using edge locations [11, 12]. Employing random shuffling technique [13–15, 50, 51] to equalize the uneven embedding capacity of binary images generally requires a large block size which results in decrease in capacity and increase in computational load. For the pattern-based approach proposed in [59], flipping the center pixels in some patterns may create

distortions such as a protrusion or a hole in a straight line. Employing the denoise patterns [49,53] achieves good visual effects of the watermarked image due to the denoising effects of the patterns. A low capacity can be expected for the images of high resolutions due to the decrease of the number of denoise patterns. Pairs of contour edge patterns that are dual to each other are employed to find suitable locations for data hiding in [16], a minimum length of contour segments is required such that the capacity is not high.

A random key matrix and a weight matrix are used to protect the hidden data in [10] to achieve high embedding rate. However, the randomness in choosing data hiding locations creates poor visual results. Further improvements [11,12] set a constraint such that the edge pixels are chosen to hide the data. Traditional distortion measures such as mean square error, signal to noise ratio and peak signal to noise ratio are no longer suitable for measuring distortion in binary images since the relationships among pixels in binary images such as connectivity and smoothness are not considered. Flipping the same amount of pixels in different locations of the same binary image may result in visually quite different watermarked images even if mean square error, signal to noise ratio and peak signal to noise ratio are same. Methods for evaluating the distortions caused by flipping pixels in binary images are proposed in [60–62] and the method proposed in [60] is subsequently employed in determining the “embeddable” locations for binary images data hiding [63].

The edge portions are chosen as the data hiding locations in the run-length based method [64]. Choosing the edge locations alone does not guarantee the good visual quality of the resultant watermarked image. Watermarking for binary message under the attack of modulo two addition noise is considered in [65], the probabilities of watermark missing and false alarm as a function of the distortion constraints and the length of the watermark are derived. Employing the code replacement based on the minimum distance of two codes [66] results in visible distortions for large number of authentication bits or

small number of card holders. The changeable pixels in a block are chosen by computing the distance and weight matrixes in [67]. Errors in one row may be propagated to its subsequent rows due to the row by row processing.

To gain high security, public/private key encryption algorithms are incorporated for binary images authentication [17, 50–52, 68, 69], e.g., for binary images [17, 50–52] and for halftone images [68, 69]. Localization of tampering for binary images is discussed in [17], in which the accuracy is limited to the sub-image size, e.g., 128×128 . How to counter against the “Parity Attack” is a key problem for the algorithms that employ the odd-even enforcement to embed one bit of data [50–52]. Chaining the blocks in the shuffled domain and embedding the image fingerprint computed in one block into the next block [50] help alleviate the attack. However the last several blocks still suffer the “Parity Attack”. Watermarking on JBIG2 text images [51] is done by embedding watermark in one of the instances, namely the data-bearing symbol using the pattern-based method proposed in [50]. Recently, a list of 3×3 patterns with symmetrical center pixels are employed to choose the data hiding localizations in [52]. Flipping the center pixels in many patterns may break the “connectivity” between pixels or create an erosion and protrusion [52]. Hence the visual quality of the watermarked image is difficult to control. In addition, employing the fixed 3×3 block scheme to partition the image leads to small embedding capacity. The random locations that are known both to the embedder and receiver are used to carry the message authentication codes for halftone images [68, 69].

More recent works are presented in [70–72]. A watermark of 7 bits is used to represent each alphanumeric character in [70]. The watermark for restoring or detecting tampering for each character is embedded in another character determined by a random or cyclic key. Accurate segmentation and recognition of each character play an important role in implementing the scheme. Embedding information on a selection channel of the cover object while keeping the receiver with no information on the selection rule is discussed in

[71,72]. The wet paper codes allow a high utilization of pixels with high flippability score, thus improve the embedding capacity significantly. Extensive research has been carried out on data hiding for halftone images [73,74]. In general, the embedding techniques designed for halftone images cannot be directly applied to text document images.

Steganalysis methods for binary images are presented in [75–80]. The quantitative relation between compression rate and embedding rate is utilized for steganalysis in [75]. While the original image pattern is estimated by averaging similar patterns in the watermarked image in [76–78]. The halftone image is converted into grayscale image and the statistical features of the sub-band coefficients are employed for steganalysis in [79]. Similarly, the statistical features such as boarder complexity, block noisiness and coarseness are employed for steganalysis in [80]. To summarize, these steganalysis methods rely on how accurately the original patterns or the changes in the statistical features by embedding the hidden data can be estimated, which can be used for subsequent classification decision.

2.3 Design Models

The design of a system on data hiding for binary images borrows concepts from multiple disciplines: steganography, spread spectrum communication, human visual perception, authentication/cryptography, image forensics, compression, binary image segmentation and thinning, and digital topology. Figure 2.3 shows the basic elements involved in a binary image data hiding system.

Steganography. Data hiding is analogous to steganography, however, steganography generally requires to keep the existence of the hidden information secret. Nevertheless, in data hiding applications, the existence of the hidden data can be publicly known [23].

Figure 2.3: Tool Sets Involved in a Binary Image Data Hiding System.

Spread Spectrum Communication. Embedding a watermark is analogous to transmitting a narrow band signal over a wide band carrier (the cover signal) such that the signal energy present in any frequency bins is small and thus difficult to be detected [81,82].

Human Visual Perception. Most data hiding algorithms employ the human visual perception models to ensure the imperceptibility of the watermark signal. Using such a model ensures efficient utilization of the “flippable” pixels while keeping good visual quality of the watermarked image.

Authentication/Cryptography. Most data hiding algorithms borrow tools from cryptography systems such as cryptographic signatures, message authentication codes and hash functions to ensure the integrity and authenticity.

Image Forensics. Data hiding methods designed for authentication purposes normally have the capability of detecting and localizing the tamperings, which thus can borrow tools from image forensics such as the statistical analysis methods.

Compression. Data hiding is analogous to data compression in the sense that both utilize the imperfection of human visual perception. The major difference lies in that compression eliminates the redundancy while data hiding may add in redundancy under

certain distortion criterion.

Binary image segmentation and thinning. Many object-based binary image watermarking techniques employ the segmentation technique to extract characters, words and features [4–9, 57]. Thinning is a technique that attempts to extract the medial line of an object while preserving the connectivity of the object or among the pixels [83–86]. In this context, data hiding technique is analogous to thinning technique.

Digital topology. One always tries to maintain the overall shape of an object and preserves the connectivity among pixels in binary image data hiding. The concepts in digital topology can be easily modified so that notions such as connected component can be efficiently employed.

2.4 A General Framework

Many data hiding techniques for binary images have been proposed in recent years. To design hard authenticator watermark-based data hiding schemes for binary images authentication is the main focus of this thesis. Towards this goal, we propose a typical framework for watermark embedding, extraction and authentication for binary images, which is depicted in Figure 2.4. Noticeably, our proposed “Connectivity-Preserving” pattern-based data hiding (*CPDH*) scheme presented in Chapter 4 is also included in the framework for completeness.

The watermark embedding process starts with processing a binary image Y based on blocks, e.g., in [10–15, 49, 51–54, 59, 63, 66, 67] and *CPDH*, or pixels scanned in raster scan order, i.e., in row by row and column by column sequence [16, 64]. The features of the block or segment ($\mathcal{B}_f$) computed by $\mathcal{C}_f(Y)$ can be the odd-even features (modulo 2) [13–15, 49, 51, 52, 63, 64], other modulo features [10–12, 54], block signatures [53], “add” and

“delete” patterns [16], etc. The k th payload watermark $\mathcal{W}_p$ is embedded in $E_m(Y, \mathcal{B}_f, \mathcal{W})$ by flipping the “embeddable” pixels (if needed) to enforce $\mathcal{W}(k) = \mathcal{B}_f(m)$, where $\mathcal{W} = \mathcal{W}_p$ and m is the block index, $m, k \in \mathbb{Z}$ and $\mathbb{Z}$ denotes the nonnegative integer set. Finally, the watermarked image Y_w is obtained. The “embeddable” blocks or segments and “embeddable” pixels are determined by $\mathcal{D}_e(Y, \mathcal{R})$, where $\mathcal{R}$ is the determination rules. Specifically, $\mathcal{R}$ for determining the “embeddable” blocks or segments can be: a block having at least one “embeddable” pixel, e.g., in *CPDH*, or having a pixel with low visual impact [52]; each block in the shuffled domain [13–15]; a segment containing “add” or “delete” patterns [16]; each non-completely black or white block after embedding the data [10, 63]; and each non-completely black or white block after embedding the data that contains the edge pixels [11, 12], etc.. While $\mathcal{R}$ for assessing the “flippability” of a pixel can be: visual distortion table [13–15, 51, 52]; “Connectivity-Preserving” criterion, e.g., in *CPDH*; contour edge pattern pairs [16], distortion measure [63] and pixels [10] or edge pixels [11, 12] chosen by a random key matrix, etc.. Finally, $\mathcal{R}$ for determining the “embeddable” pixel in a block or segment can be: a pixel with the highest flippability score [13–15] or low visual impact [51, 52, 63]; first “embeddable” pixel in the block, e.g., in *CPDH*; changeable pixel in “add” or “delete” patterns [16], and pixels [10] or edge pixels [11, 12] chosen by a random key matrix, etc..

To ensure the authenticity and integrity of Y , the salient features of the image or intermediate image, e.g., Y_1 , can be generated by $\mathcal{X}_f(Y)$. For the methods whose “embeddable” locations can be determined for both embedding and extraction processes, e.g., in *CPDH* and [52], one way to generate the intermediate image Y_1 is to clear out the “embeddable” locations of Y . Y_1 is subsequently fed to a hash function such as *SHA-1* to generate the hash value $\mathcal{H}_o = \text{Hash}(Y_1)$. $\mathcal{H}_o$ is encrypted with an encryption algorithm $E_k()$ to obtain the message authentication code or cryptographic signature of Y , i.e., $\mathcal{W}_s = E_k(\mathcal{H}_o, K_s)$, where K_s is the private key of the authorized user or owner. $\mathcal{W}_s$ is subsequently com-

binized with $\mathcal{W}_p$ to generate the authenticator watermark $\mathcal{W}_r = \mathcal{C}_m(\mathcal{W}_s, \mathcal{W}_p)$. One choice of the operation $\mathcal{C}_m$ is concatenating $\mathcal{W}_s$ with $\mathcal{W}_p$, i.e., $\mathcal{W}_r = \mathcal{W}_s \parallel \mathcal{W}_p$, where “ $\parallel$ ” denotes “concatenation” operation. $\mathcal{W}_r$ is embedded in the “embeddable” locations of Y by $E_m(Y, \mathcal{B}_f, \mathcal{W})$ to enforce $\mathcal{W}(k) = \mathcal{B}_f(m)$, where $\mathcal{W} = \mathcal{W}_r$, finally the watermarked image Y_w is obtained. $\mathcal{W}_p$ can be a visual pattern such as a binary logo image, a random sequence, a fingerprint or a handwritten signature image obtained from the authorized user or owner. Bitwise accuracy is required in the verification of the watermark.

For the watermark extraction, Y_w will go through similar process to determine the extraction locations by $\mathcal{D}_e(Y_w, \mathcal{R}')$. $\mathcal{R}'$ can be the same rules used in the embedding process, e.g., in *CPDH* and [16, 52] or some pre-defined rules, e.g., each block of Y_w in the shuffled domain [13–15]; each non-completely black or white block [10, 63] and each even, non-completely black or white block [11, 12]. Compute the features of blocks $\mathcal{B}'_f$ by $\mathcal{C}_f(Y_w)$, thereafter, the watermark data $\mathcal{W}'_p$ or $\mathcal{W}'_r$ can be extracted by $E_x(\mathcal{B}'_f, Y_w)$ from the extraction locations of Y_w . Comparison of $\mathcal{W}'_p$ with $\mathcal{W}_p$ gives the verification results when $\mathcal{W}_p$ is embedded. Otherwise, if message authentication code or cryptographic signature is employed, the same $\mathcal{C}_m(\mathcal{W}'_r, \mathcal{W}_p)$ is used to retrieve $\mathcal{W}'_s$. $\mathcal{W}'_s$ is subsequently decrypted via the decryption algorithm $D_k(\mathcal{W}'_s, \mathcal{K}_p)$ to obtain $\mathcal{H}'_o$ of Y , where $\mathcal{K}_p$ is the public key of the authorized user or owner. The salient features of Y_w or the intermediate images Y_{w1} is computed by $\mathcal{X}_f(Y_w)$, which are subsequently fed to the hash function to generate the hash value $\mathcal{H}_w = \text{Hash}(Y_{w1})$. Comparison of $\mathcal{H}'_o$ with $\mathcal{H}_w$ gives the authentication results. Computing message authentication code or cryptographic signature based on salient features can achieve certain robustness for the authenticator watermark, which is not explored in this thesis. In designing a data hiding scheme for authentication, certain robustness against random noise is desirable. For this purpose, Error Correction Codes (*ECC*) can be employed to encode the watermark, the encoded messages are embedded in the image.

Figure 2.4: A General Framework for Watermark Embedding, Watermark Extraction and Authentication.

2.5 Performance Measures

Performance of the proposed data hiding schemes is evaluated based on a series of measures. Imperceptibility is the basic requirement for a data hiding algorithm, which is

achieved by utilizing the imperfection of Human Visual System. It can be evaluated by subjective and objective measures.

2.5.1 Distortion Measure

Several distortion measures to evaluate the amount of distortion caused by flipping pixels in binary images are available in literature [14, 15, 60–62]. A weight matrix is employed to calculate the distortion caused by flipping pixels in [60]. This method works well when salt and pepper noise is introduced by flipping pixels, which is reasonable since flipping a non-edge pixel usually causes larger visual distortion than flipping an edge pixel. However, lacking consideration of connectivity between pixels makes it difficult to evaluate accurately when edge pixels are flipped. Smoothness and connectivity are taken into consideration in [61]. However, it has limitations in evaluating flipping pixels for sharp corner patterns and pixels along straight lines. Tracing the contour to extract the contour segment for evaluation is required in [62], which is time-consuming. Hence, in this thesis, the visual distortion table proposed by Wu *et al.* [13–15] is employed. Specifically, the flippability score ($\mathcal{S}_f$) is obtained by matching the 3×3 pattern of the original image centered at the flipped pixel in the look up table (*LUT*). The complement of $\mathcal{S}_f$ with reference to value “1”, namely “distortion score ($\mathcal{S}_d$)” is used and given by $\mathcal{S}_d(k) = 1 - \mathcal{S}_f(k)$, where k represents the k th flipped pixel. Obviously, the lower the $\mathcal{S}_d$ is, the less the distortion will be, i.e., the better the visual quality of the resultant watermarked image compared with the original image. The total distortion (*TD*) and average per pixel distortion (*APPD*), denoted as $\mathcal{D}_t$ and $\mathcal{D}_a$, introduced by flipping pixels in binary images are given by

$$\mathcal{D}_t = \sum_{k=1}^n \mathcal{S}_d(k) \quad (2.1)$$

$$\mathcal{D}_a = \mathcal{D}_t / n \quad (2.2)$$

where n is the total number of pixels being flipped to embed the watermark.

2.5.2 Normalized Hamming Distance

The bitwise accuracy of the extracted watermark compared with the original watermark is required for exact authentication. To this end, the normalized hamming distance is employed [87], which is given by

$$\mathcal{R}_d(\mathcal{W}, \mathcal{W}') = \frac{1}{N_w} \sum_{k=1}^{N_w} (\mathcal{W}(k) \oplus \mathcal{W}'(k)), \quad (2.3)$$

where, N_w is the length of the watermark and $\mathcal{W}$ and $\mathcal{W}'$ are the original and extracted watermarks, respectively.

2.5.3 Probabilities of Miss Detection and False Alarm

Probability of miss detection ($\mathcal{P}_{md}$) and probability of false alarm ($\mathcal{P}_{fa}$) are used to evaluate the performance of the proposed tampering localization scheme. In terms of percentage, $\mathcal{P}_{md}$ and $\mathcal{P}_{fa}$ are given by

$$\mathcal{P}_{md} = \frac{N_{md}}{N_t} \times 100\% \quad (2.4)$$

$$\mathcal{P}_{fa} = \frac{N_{fa}}{N_t} \times 100\% \quad (2.5)$$

where N_{md} , N_{fa} and N_t denote the number of miss-detected, the number of false-identified and the total number of tamperings, respectively.

2.5.4 Computational Complexity

Computational complexity is one of the major criteria to evaluate the systems designed for authentication applications. The watermark embedding process can be done either online or off-line to generate the watermarked image. It is preferably to be online for real-time application. However, the watermark extraction and authentication process should be performed online for most applications. The computational complexity is evaluated by the types and numbers of computations required. Due to the variations in implementation, we do not compare *CPU* time.

2.5.5 Security Concerns

Assume that the attacker has no knowledge of any secret keys, the following two problems are to be considered for the algorithms designed for authentication purposes [14, 15, 25]. They are: (1) the probability of making the content alterations while keeping the authentication data authentic, (2) the possibility for an adversary to hide specific data in a watermarked image. These issues will be addressed in Chapter 4.

Chapter 3

Integrated Inter Character and Word Spaces Data Hiding

In this Chapter, we propose an integrated inter character and word spaces data hiding scheme (*ICWS*) for text document images authentication, which can be applied to a wide variety of text document images, especially for those text documents that do not have obvious word spaces. In addition, this scheme is suitable for the applications that do not allow the distortions introduced by flipping pixels in binary images.

A text document can be represented in many formats such as bitmap and page description language formats. In this thesis, we assume that the text document is in bitmap image format and all the text are black in a white background. A text document can be aligned to the left, right, center and justified depends on different ways of formatting. The most commonly used format in text documents is “justified”, in which the white spaces are distributed among all the word spaces. In this Chapter, we propose a method that is applicable for all the formats. The basic idea of the scheme is based on the fact that generally the inter character spaces are much smaller than that of the inter word spaces, e.g., 1-3 pixels and 4-18 pixels for a text document of resolution 150dpi. Integrating the two types of spaces not only achieves capacity improvements, but also the inter word spaces could compensate for the insufficient inter character spaces, which makes it possible to

watermark text documents in any formats and languages. The basic requirement for the integrated inter character and word spaces coding is that shifting a character should not join two characters or separate one character into two. Note that in this Chapter, the characters not only refer to letters, but also include numerals, ligatures, punctuation and other symbols.

3.1 The Proposed Algorithm

3.1.1 Embeddable Segment Determination

The left and right distances of a character are the minimum horizontal distances from the peak edge points of the current character to that of its preceding or following characters, which are denoted as d_l and d_r and shown in Figure 3.1. To calculate $d_l(k)$ and $d_r(k)$ of

Figure 3.1: Left and Right Distances of a Character.

the k th character, let us denote the edge points of the current, its preceding and following characters as $c(i_2, j_2)$, $p(i_1, j_1)$ and $f(i_3, j_3)$, respectively, e.g., G , A and C shown in Figure 3.1, where $i_1, i_2, i_3 = t, t + 1, \dots, e$, t and e denote the starting and ending indices of a text line, respectively. j_2, j_1, j_3 denote the column indices of the edge points of the current, its preceding and following characters, respectively, which form the sets $\{J2\}$, $\{J1\}$ and $\{J3\}$. $d_l(k)$ and $d_r(k)$ are calculated by

$$d_l(k) = \min_{j_2 \in \{J2\}} (j_2) - \max_{j_1 \in \{J1\}} (j_1) \quad (3.1)$$

$$d_r(k) = \min_{j_3 \in \{J3\}} (j_3) - \max_{j_2 \in \{J2\}} (j_2) \quad (3.2)$$

For text documents in PostScript format, various PostScript commands or equivalent word shifter command can be used to adjust the x-coordinate of a word or character or y-coordinate of a text line to perform the watermark embedding [5]. In this study, we shall focus on text documents in image format. Assume a text document image $Y(i, j) \in \{0, 1\}$, $i = 0, 1, 2, \dots, M$ and $j = 0, 1, 2, \dots, N$, where M and N are the height and width of the image. The strategy to calculate the inter character and word spaces and determine the “embeddable” data hiding segments is detailed as follows. Calculate the horizontal profile of the image to locate a text line by

$$h(i) = \sum_{j=0}^N Y(i, j) \quad (3.3)$$

Locate the white spaces between text lines by examining consecutive runs of “0s” of $h(i)$. Calculate the vertical profile for each text line by

$$v(j) = \sum_{i=t}^e Y(i, j) \quad (3.4)$$

Consecutive runs of “0s” of $v(j)$ indicate the presence of white spaces between two characters or two words. A character lies between the two white spaces and several characters form a character segment. The data embedding is based on the character segments rather than each single character. In this way, the relationship between the character and word spaces within a character segment can be easily enforced and the changes caused by shifting a character will not affect other character segments. A sample profile of a text line is shown in Figure 3.2.

In the proposed scheme, an overlapping window of size three characters or symbols is employed to group the characters to form the character segments, as shown in Figure 3.3. Each character segment consists of three consecutive characters and every two neighboring character segments share one common character that is not used for shifting. If shifting a character does not change the number of character segments as well as the number of

Figure 3.2: A Sample Profile (b) for the 4th Text Line of a French Text Image (a), which consists of 1 character, 5 characters, 1 sign and 8 characters from left to right.

Figure 3.3: Illustration of the Overlapping Windows of Size Three.

characters, this character segment is identified as “embeddable”. The watermark is embedded in the “embeddable” character segments based on the “Embedding Rules (*ERs*)”, which will be discussed in the following Section.

3.1.2 Data Embedding and Authentication

The inter character space refers to the space between two neighboring characters. Let us denote the maximum inter character space as Φ_c and define the white space between two characters that is larger than Φ_c as the word space, denoted by Φ_w . While $\{\Phi_w\}$ denotes the set containing all the word spaces. The four combinations of the left and right distances (d_l , d_r) of a character shown in Figure 3.4 are: (Word, Character), (Word, Word), (Character, Character) and (Character, Word). The embedding rules (*ERs*) are

defined as follows

Figure 3.4: Four Combinations of the Left and Right Distances of a Character.

1. Excluding Rule (*ER1*).

Exclude a character from embedding if $(d_l = 1) \wedge (d_r = 1)$. In such a case, shifting the character will join two separate characters, which changes the number of characters as well as the number of character segments, where “ $\wedge$ ” is the logical “And” operation.

2. Basic Rule (*ER2*).

To embed bit “0”, ensure that $d_l < d_r$ or $d_r = \Phi_c$ if $d_l \in \{\Phi_w\}$. To embed bit “1”, ensure that $d_l \geq d_r$ or $d_l = \Phi_c$ if $d_r \in \{\Phi_w\}$.

3. Supplement Rule (*ER3*).

To embed bit “0”, left shift the character one pixel if $(d_l = \Phi_c) \wedge (d_r \in \{\Phi_w\})$, such shifting is not required if $(d_r = \Phi_c) \wedge (d_l \in \{\Phi_w\})$. To embed bit “1”, right shift the character one pixel if $(d_r = \Phi_c) \wedge (d_l \in \{\Phi_w\})$, such shifting is not required if $(d_l = \Phi_c) \wedge (d_r \in \{\Phi_w\})$.

4. Special Rule (*ER4*).

If $(d_l \in \{\Phi_w\}) \wedge (d_r \in \{\Phi_w\})$ and after the character is left or right shifted, $d_l = \Phi_c$ or $d_r = \Phi_c$. In such a case, we just continue shifting the character to the left one more pixel to embed bit “0” or to the right to embed bit “1”.

It can be noticed that the maximum number of shifts (N_s) for a character whose (d_l, d_r) to be (Character, Character), (Character, Word) and (Word, Character) are limited to

Φ_c , i.e., $N_s \leq \Phi_c$, which will be discussed further in Section 3.3. In defining the embedding rules above, we make use of Φ_c such that the relationship between the inter character space and word space can be established. In general, we require that both d_l and d_r should be greater than “0”. If $d_l = 0$ or $d_r = 0$, the current character is considered to be connected with its preceding or following characters, which are treated as one character.

Due to the fact that only the white spaces are changed in data embedding process, the characters that are not touched in the embedding process can be used to compute the hash value to ensure the authenticity and integrity of the document. The secret key of the owner is firstly fed to a pseudo random sequence generator to generate a random sequence $\mathcal{R}_s$ [88]. The pixels of the characters are scanned in raster scan order and the resulting bitstream $\mathcal{B}_s$ is fed to a hash function $Hash()$ such as *MD5* to generate hash value $\mathcal{H}_o$. Subsequently, the authenticator watermark $\mathcal{W}_r$ is generated by $\mathcal{W}_r = \mathcal{H}_o \oplus \mathcal{R}_s$, where “ $\oplus$ ” is the “Xor (Exclusive OR)” operation. In the implementation, if the capacity of the image $\mathcal{C}_\zeta \neq Len(\mathcal{H}_o)$, logical “Or”, “And” and “Xor” operations can be used, where $Len(x)$ gives the length of the element x . Other symmetric or asymmetric cryptographic algorithms such as *DES* and *RSA* may be employed based on the capacity of the image.

To combat malicious attacks such as retyping or reformatting the watermarked document, the starting index (I_x) of the sharing characters, e.g., 1, 34, 66,..., could be converted to a binary string $\mathcal{B}_{ix}$, to generate $\mathcal{B}_{so} = \mathcal{B}_s \parallel \mathcal{B}_{ix}$. $\mathcal{B}_{so}$ is subsequently fed to $Hash()$ to generate $\mathcal{H}_o$, where “ $\parallel$ ” is the “concatenation” operation. Assume the adversary tries to reformat the text document to erase the watermark, I_x will change, hence the re-computed hash value of the watermarked image $\mathcal{H}_w$ will vary from $\mathcal{H}_o$. Generally, retyping and reformatting a text document results in changes in the bounding box of the character, which also renders $\mathcal{H}_w$ different from $\mathcal{H}_o$.

3.1.3 Data Extraction and Verification

The watermarked image Y_w has to go through similar process as that in the embedding process to compute d_l and d_r of a character. Thereafter, the watermark can be extracted based on the extraction rules (XR s) without requiring the presence of the original image. Let us first define the following two conditions.

$$\xi_1 = (d_l = \Phi_c) \wedge (d_r \in \{\Phi_w\}) \quad (3.5)$$

$$\xi_2 = (d_r = \Phi_c) \wedge (d_l \in \{\Phi_w\}) \quad (3.6)$$

The extraction rules are defined as follows

1. $XR1$. If $(d_l = 1) \wedge (d_r = 1)$, no bits can be extracted from the segment.
2. $XR2$. If $(d_l < d_r) \wedge (\xi_1 = 0)$, the extracted watermark bit is “0”. Otherwise, if $(d_l < d_r) \wedge (\xi_1 = 1)$, the extracted watermark bit is “1”.
3. $XR3$. If $(d_l \geq d_r) \wedge (\xi_2 = 0)$, the extracted watermark bit is “1”. Otherwise, if $(d_l \geq d_r) \wedge (\xi_2 = 1)$, the extracted watermark bit is “0”.

During watermark verification, the secret key of the owner is used to generate $\mathcal{R}_s$. The hash value of original image $\mathcal{H}'_o$ is obtained by $\mathcal{H}'_o = \mathcal{R}_s \oplus \mathcal{W}'_r$, where $\mathcal{W}'_r$ is the extracted authenticator watermark by applying the extraction rules. The bitstream of the characters $\mathcal{B}_{sw}$ of Y_w are concatenated with the index of the sharing characters $\mathcal{B}_{ixw}$ to generate $\mathcal{B}_{swo} = \mathcal{B}_{sw} \parallel \mathcal{B}_{ixw}$, which is subsequently fed to $Hash()$ to generate hash value $\mathcal{H}_w$ of Y_w . Authenticity and integrity of Y_w is given by comparing $\mathcal{H}_w$ and $\mathcal{H}'_o$.

3.2 Experimental Results and Comparisons

Experiments are conducted to test the data hiding effects of the proposed scheme. Two text document images in English and Chinese and of resolution 150dpi and 300dpi are

used. Experimentally, Φ_c is taken as the mean of all the character and word spaces, which is close to the actual value of maximum inter character spaces. The segment size is chosen to be three, which not only can maximize the capacity but also can easily enforce the relationship between the inter character and word spaces. The data hiding results are shown in Figures 3.5 and 3.6, which demonstrate that shifting the characters does not cause annoying effects to human eyes, the shifting is imperceptible.

Figure 3.5: Hiding Effects on an English Text Image. (a) the original image of size 250×250 , (b) the watermarked image with 35 bits embedded and (c) the difference image.

Figure 3.6: Hiding Effects on a Chinese Text Image. (a) the original image of size 640×180 , (b) the watermarked image with 25 bits embedded and (c) the difference image.

Experiments are also conducted to test the authentication mechanism. First, the watermarked image is tested with correct and incorrect secret key as the seed value to generate the random sequence. As expected, the verification gives “yes” and “no” to

indicate that the watermarked image is “authentic” and “not authentic”. A “NTU” logo image is used as the payload watermark to visually show the tamperings, which is subsequently *Xored* with the hash value of the image to produce the authenticator watermark $\mathcal{W}_r$. The authentication results are shown in Figure 3.7. When no tampering occurs to the image, the retrieved logo image is intact. When any tampering occurs, the retrieved logo image looks like random noise. The cutting and pasting attack is also simulated. The results show that the hash value of the modified watermarked image is always different from that of the original, which reveals the effectiveness of the proposed authenticator watermark. The number of pixels shifted N_s for the image shown in Figure 3.7 is plotted in Figure 3.8 for the shifting number from 50 to 100, which confirms that $N_s \leq \Phi_c$ holds, where $\Phi_c = 3$ in the experiments.

Comparisons are made between our proposed method and the line and word shifting [4,5] and character and group of characters shifting [6,7] methods. For the line and word shifting method, every alternate line or word is shifted such that the watermark can be extracted blindly if the original text is formatted in uniform spaces. The character shifting method shifts every individual character to the left or right to make the inter character spaces deviate a small amount from the average character space. However, arbitrarily shifting two consecutive characters to impose the space deviation constraint may render two neighboring characters overlapping with each other. Hence, the shifting is performed in each character so that its previous (for left shifting) or subsequent (for right shifting) characters in the same text line are shifted the same number of pixels. As a result, the text is easily expanded to the white margin. The comparison results shown in Figure 3.9 reveal that both line and word shifting methods have low capacity. While the visual quality of the watermarked image obtained by employing our proposed method are better than that obtained by employing the character shifting and group of character shifting methods. The distortion at the margin is quite obvious for the character shifting method.

The main function of the parser is to recognize the statements of the source program as language constructs described by the grammar. The parser will use tokens recognized by the scanner to build a parse tree for the program.

In this assignment, a top-down parsing technique known as recursive descent is used. The basic idea of recursive-descent parsing is that a parsing procedure is associated with each nonterminal of the grammar. Parsing begins by calling the procedure associated with the start symbol. During parsing, each terminal in the right-hand side of a production is matched with the look-ahead symbol, and each nonterminal will result in a call of its associated procedure. An error is generated when there is no match for one terminal. The

(a)

The main function of the parser is to recognize the statements of the source program as language constructs described by the grammar. The parser will use tokens recognized by the scanner to build a parse tree for the program.

In this assignment, a top-down parsing technique known as recursive descent is used. The basic idea of recursive-descent parsing is that a parsing procedure is associated with each nonterminal of the grammar. Parsing begins by calling the procedure associated with the start symbol. During parsing, each terminal in the right-hand side of a production is matched with the look-ahead symbol, and each nonterminal will result in a call of its associated procedure. An error is generated when there is no match for one terminal. The

(b)

The major function of the parser is to recognize the statements of the source program as language constructs described by the grammar. The parser will use tokens recognized by the scanner to build a parse tree for the program.

In this assignment, a top-down parsing technique known as recursive descent is used. The basic idea of recursive-descent parsing is that a parsing procedure is associated with each nonterminal of the grammar. Parsing begins by calling the procedure associated with the start symbol. During parsing, every terminal in the Left-hand side of a production is matched with the look-ahead symbol, and each nonterminal will result in a call of its associated procedure. An error is generated when there is no match for one terminal. The

Tampered location

(c)

(d)

(e)

(f)

Figure 3.7: Authentication Results. (a) original image of size 908×371 , (b) watermarked image with 162 bits embedded and (c) watermarked image that has been tampered. (d) original logo image, (e) extracted logo from the watermarked image without being tampered and (f) extracted logo image from the tampered watermarked image.

Figure 3.8: Number of Pixels Shifted for the Text Image Shown in Figure 3.7.

While for group of character shifting method, it is difficult to differentiate the character and word spaces in the watermarked image.

Figure 3.9: Comparison of Hiding Effects on an English Text Image. (a) the original image of 300dpi and of size 660×160 ; (b) and (c) hide 2 and 12 bits by Brassil *et al.*'s line shifting and word shifting methods, respectively; (d) and (e) hide 68 and 14 bits by Chotikakamthorn's character shifting and group of character shifting methods, respectively and (f) hide 58 bits by our proposed *ICWS*.

3.3 Discussions

We now show that the number of shifts N_s when (d_l, d_r) are (Character, Character), (Character, Word) and (Word, Character) is limited to the maximum inter character space Φ_c , i.e., $N_s \leq \Phi_c$. To embed bit “0”, we have the following cases.

1. For $(d_l = 1) \wedge (d_r = 1)$, the character cannot be shifted. Hence, $N_s = 0 < \Phi_c$ holds.
2. For $d_l < d_r$, only when $(d_l = \Phi_c) \wedge (d_r > \Phi_c)$, the character needs to be left shifted 1 pixel. Obviously, $N_s = 1 \leq \Phi_c$. Other cases such as $(d_l < \Phi_c) \wedge (d_r < \Phi_c)$, $(d_l > \Phi_c) \wedge (d_r > \Phi_c)$, $(d_l < \Phi_c) \wedge (d_r > \Phi_c)$ and $(d_l < \Phi_c) \wedge (d_r = \Phi_c)$, shifting is not required.
3. For $d_l > d_r$
 - If $(d_l > \Phi_c) \wedge (d_r = \Phi_c)$, no shifting is required.
 - If $(d_l > \Phi_c) \wedge (d_r < \Phi_c)$, we only need to left shift $N_s = \Phi_c - d_r$. Since $d_r \geq 1$, so $N_s \leq \Phi_c$ is always true. If $d_l = \Phi_c$ after left shifting, the character needs to be continuously left shifted one more pixel, i.e., $N_s = \Phi_c - d_r + 1$. Since $d_r \geq 1$, we have $N_s \leq \Phi_c$ always hold.
 - If $(d_l > \Phi_c) \wedge (d_r > \Phi_c)$, $N_s = \lceil (d_l - d_r + 1)/2 \rceil$, where $\lceil x \rceil$ is the ceiling function which gives the smallest integer not less than x . In this case, $N_s > \Phi_c$ may be true. However, shifting a singular character between two word spaces is not noticeable. If after left shifting, $d_l = \Phi_c$, continuously left shifting one more pixel is required.
 - If $(d_l = \Phi_c) \wedge (d_r < \Phi_c)$, $N_s = \lceil (d_l - d_r + 1)/2 \rceil$. Since $(d_r \geq 1) \wedge (d_l = \Phi_c)$, so, $N_s \leq \Phi_c$ always true.
 - If $(d_l < \Phi_c) \wedge (d_r < \Phi_c)$, $N_s = \lceil (d_l - d_r + 1)/2 \rceil$. Since $(d_l < \Phi_c) \wedge (d_r \geq 1)$, so, $N_s \leq \Phi_c$ always holds.

4. For $d_l = d_r$

- If $(d_l = \Phi_c) \wedge (d_r = \Phi_c)$, we only need to left shift the character one pixel, $N_s = 1 \leq \Phi_c$ holds.
- If $(d_l < \Phi_c) \wedge (d_r < \Phi_c)$, we only need to left shift the character one pixel, $N_s = 1 \leq \Phi_c$ holds.
- If $(d_l > \Phi_c) \wedge (d_r > \Phi_c)$, we need to left shift the character one pixel, $N_s = 1 \leq \Phi_c$ holds. If after left shifting, $d_l = \Phi_c$, we just need to continuously left shift one more pixel, obviously, $N_s = 2 \leq \Phi_c$ holds.

The above analysis confirms that $N_s \leq \Phi_c$ is true for shifting between two characters, or between a character and a word. Only when the character is a singular character, i.e., $d_l, d_r \in \{\Phi_w\}$, $N_s > \Phi_c$ may be true. However, shifting a singular character between two word spaces is not easily noticed. Similar conclusions can be drawn for embedding watermark bit “1”.

Compared with conventional line, word and character shifting methods [4–9], the pros and cons of our proposed method are summarized below. Integrating the inter character and word spaces for watermark embedding not only improves the capacity but also achieves good visual quality of the watermarked image. The amount of shifting between two characters, and between a character and a word is limited to the maximum inter character space. Randomly distributing the shifting enhances system security in the sense that an attacker is difficult to figure out the shifting patterns. Using the authenticator watermark addresses the security issue, which has not been addressed by other object-based methods. Extracting watermark without requiring the control line or word makes the method more immune to the errors of page segmentation. The proposed scheme has difficulty in surviving the printing-photocopying-scanning process, which is not a major concern for a method designed for authentication purposes. Our proposed method can be

applied to a wide variety of text images, including text images that do not have obvious word spaces such as Chinese and Thai writing.

3.4 Chapter Summary

In this Chapter, an integrated inter character and word spaces data hiding scheme is presented, which has a moderate capacity while preserving good visual quality of the watermarked image. The security level has enhanced by randomly shifting the characters as well as by embedding the authenticator watermark. The number of shifting between two characters, or between a character and a word is limited to the maximum inter character spaces. By not considering the robustness against printing, photocopying and scanning for electronic binary document images authentication, comparisons with conventional line, word and character shifting methods demonstrate the superiority of the proposed scheme.

Chapter 4

Connectivity-Preserving Pattern-Based Data Hiding

In Chapter 3, we present a data hiding scheme that operates on object-level by enforcing the relationship of the inter character and word spaces to encode the data. To achieve a larger capacity, we shall study data hiding techniques that operate in low level such as flipping a pixel from black to white or vice versa to encode the data, namely, the pixel-flipping technique.

4.1 Background and Objectives

In this chapter, we present a novel spatial domain “Connectivity-Preserving” pattern-based data hiding (*CPDH*) approach for binary images authentication. Specifically, we define a “Connectivity-Preserving” criterion to assess the “flippability” of a pixel to address the problem of how to choose the suitable pixels to carry the watermark data. The criterion is defined based on the observation that connectivity among pixels in binary images plays an important role to their visual qualities. If flipping a pixel does not destroy the connectivity among pixels, the visual quality is ensured. The connectivity and smoothness have been considered in designing the visual distortion table for binary im-

ages in the approaches proposed by Wu *et al.* [13], Wu [14] and Wu and Liu [15]. Here we emphasize the 4-connectivity and 8-connectivity in such a way that the good edge patterns will be chosen for data hiding.

Observations have made that the “embedding capacity” of binary images is unevenly distributed. Some regions have more “flippable” pixels to be used to carry the watermark, whereas other regions have few or no “flippable” pixels. The uneven “embeddability” problem has been addressed by Wu *et al.* [13], Wu [14] and Wu and Liu [15] earlier, in which a random shuffling key is used to shuffle the image so that the “flippable” pixels are distributed uniformly throughout the image. However, it is noticed that practically it is difficult to find a good shuffle key to ensure that each block in the shuffled domain has a suitable pixel to flip for small blocks. Hence a larger block size such as 12×12 is generally required. As a result, the capacity has decreased and the computational load has increased. This problem is revisited by employing the “Wet Paper Coding” technique to improve the capacity in [72]. In this Chapter, our main objectives are summarized as follows.

1. Assess the “flippability” of a pixel using “Connectivity-Preserving” criterion to achieve good visual quality of the watermarked image.
2. Handle the uneven “embeddability” of the image by adaptively embedding the watermark only in those “embeddable” blocks.
3. Study the invariant features in flipping pixels in binary images and employ these features to achieve blind watermark extraction.
4. Explore different ways of partitioning an image to achieve larger capacity.
5. Investigate on how to locate an “embeddable” pixel in a larger block of the image so as to incorporate the cryptographic signature of the image to achieve higher security.

This Chapter is organized as follows. In Section 4.2, the “flippability” decision criterion, the capacities of employing different block schemes to partition the image and the watermark embedding and extraction process are presented. The authentication mechanism and the criterion for locating the “embeddable” pixels for the non-interlaced and interlaced block schemes are discussed in Section 4.3. Experimental results and comparisons are presented in Section 4.4. Further discussions and security considerations are discussed in Section 4.5, followed by a summary of the Chapter in Section 4.6. The proposed scheme can be applied for text documents, cartoons, engineering drawings, handwritten signatures and even halftone images authentication.

4.2 Data Hiding Mechanism

The data hiding mechanism consists of three parts, i.e., how to assess the flippability of a pixel, how to partition an image into blocks and determine the “embeddability” of each block, and how to embed one bit of data in each block.

4.2.1 Flippability Decision

The flippability of a pixel relies on the transitions between the current pixel and its eight neighbors in a 3×3 block, in particular, the 4-connectivity and 8-connectivity among pixels. In this Chapter, the transition refers to the relationship between the current pixel and its 4- or 8-neighbors in a 3×3 block. This definition is slightly different from the conventional definition of transition, which generally refers to abrupt changes of pixel values between two neighboring pixels. The eight neighbors of the center pixel p_c shown in Figure 4.1 in a 3×3 neighborhood are denoted as $w_1, w_2, w_3, w_4, w_5, w_6, w_7$ and w_8 , where (i, j) denotes the pixel located at row i and column j of the image. In this thesis, we use “1” and “0” to represent black and white pixels, respectively.

$(i-1, j-1)$ w_6	$(i-1, j)$ w_7	$(i-1, j+1)$ w_8
$(i, j-1)$ w_5	(i, j) p_c	$(i, j+1)$ w_1
$(i+1, j-1)$ w_4	$(i+1, j)$ w_3	$(i+1, j+1)$ w_2

Figure 4.1: Designations of Pixels in a 3×3 Neighborhood.

Definition 1. The number of uniform white transition $\mathcal{N}_{vhw}$ and the number of uniform black transitions $\mathcal{N}_{vhb}$ in a 3×3 block along the vertical and horizontal (VH) directions are collectively defined as “ VH Transition” and given by

$$\mathcal{N}_{vhw} = \sum_{k=1,3} \bar{p}_c \cdot \bar{w}_k \cdot \bar{w}_{k+4} \quad \text{and} \quad \mathcal{N}_{vhb} = \sum_{k=1,3} p_c \cdot w_k \cdot w_{k+4} \quad (4.1)$$

where $\bar{w}$ denotes logical “Not w ”, “ $\cdot$ ” is the logical “And” operation for binary data and $\mathcal{N}_{vhw}, \mathcal{N}_{vhb} \in \{0, 1, 2\}$.

Definition 2. The number of interior right angle (IR) transitions $\mathcal{N}_{ir}$ in a 3×3 block is defined as “ IR Transition”, which is given by

$$\mathcal{N}_{ir} = \sum_{k=1}^4 \bar{p}_c \cdot \bar{w}_{2k-1} \cdot w_{2k} \cdot \bar{w}_{2k+1} \quad (4.2)$$

where $\bar{w}_{2k+1} = \bar{w}_1$, for $2k+1 > 8$ and $\mathcal{N}_{ir} \in \{0, 1, 2, 3, 4\}$.

Definition 3. The number of transitions from the center pixel to the sharp corners (C) $\mathcal{N}_c$ in a 3×3 block is defined as “ C Transition”, which is given by

$$\mathcal{N}_c = \sum_{k=1}^4 p_c \cdot w_{2k} \cdot w_{2k+1} \cdot w_{2k+2} \cdot w_{2k+3} \cdot w_{2k+4} \quad (4.3)$$

where $w_9 = w_1$, $w_{10} = w_2$, $w_{11} = w_3$, $w_{12} = w_4$ and $\mathcal{N}_c \in \{0, 1, 2, 3, 4\}$.

We define the “**Flippability Criterion**” such that the center pixel in a 3×3 block is “flippable” if the number of VH transition $\mathcal{N}_{vhw}$ and $\mathcal{N}_{vhb}$, the number of IR transition $\mathcal{N}_{ir}$ and the number of C transition $\mathcal{N}_c$ remain the same before and after flipping the center pixel. $\mathcal{N}_{vhw}$, $\mathcal{N}_{vhb}$ and $\mathcal{N}_{ir}$ are calculated before and after flipping the center pixel. If the transition numbers do not change, it implies that flipping the pixel will not destroy the connectivity between pixels in the neighborhood and does not create extra clusters or isolated corner pixels as well. While $\mathcal{N}_c$ is used to control not to flip pixels in sharp corners so that the 8-connectivity among pixels is not destroyed. These conditions are collectively named as “**Connectivity-Preserving**” criterion, which are illustrated in Figure 4.2 for different 3×3 block patterns.

Figure 4.2: The 3×3 Block Patterns that (a) satisfy VH transition, (b) satisfy VH transition and C transition but do not satisfy IR transition and (c) satisfy VH transition and IR transition but do not satisfy C transition, excluding the patterns that differ only by rotation, complement and mirroring. Pixels in grid represent “don’t care” pixels and the arrows show the checking directions.

The reason why the connectivity among the pixels in a 3×3 neighborhood is preserved can be explained as follows. The pixel that meets the condition, i.e., $\mathcal{N}_{vhw}$ and $\mathcal{N}_{vhb}$ in Equation (4.1) remain the same before and after flipping the center pixel, would have one white 4-neighbor in both horizontal and vertical directions. Hence, it is an edge pixel both horizontally and vertically, as shown in Figure 4.2(a). The edge pixel is defined as the black pixels in a binary image with at least one white 4-neighbor [86]. The condition, i.e., $\mathcal{N}_{ir}$ in Equation (4.2) remains the same before and after flipping the center pixel, is to ensure that flipping the center pixel does not create another cluster or in another aspect, an isolated

corner pixel. The isolated corner pixel is the corner pixel that has three white neighbors in a 3×3 block, which can be seen from the lower right corner of Figure 4.2(b) when the center pixel is white. Furthermore, by satisfying the conditions that $\mathcal{N}_{vhw}$ and $\mathcal{N}_{vhb}$ in Equation (4.1) and $\mathcal{N}_{ir}$ in Equation (4.2) remain the same before and after flipping the center pixel, at least one corner has three white pixels. This further ensures that flipping the center pixel will not destroy the local connectivity of the pattern. Finally, by meeting the condition that $\mathcal{N}_c$ in Equation (4.3) remains the same before and after flipping the center pixel, it implies that flipping the center pixel will not destroy the 8-connectivity of the center pixel and its neighbors. Hence, by satisfying the “Connectivity-Preserving” criterion, the local connectivity is preserved. An example of the “flippable” and “non-flippable” pixels identified by “Connectivity-Preserving” criterion shown in Figure 4.3 further enforces our arguments.

Figure 4.3: Example of “Flippable” and “Non-Flippable” Pixels Identified by “Connectivity-Preserving” Criterion.

4.2.2 Block Partition and Embeddability

There are different ways to partition an image, which results in different types of blocks employed in the data hiding process. We propose the fixed 3×3 block (*FXB*), non-

interlaced block (*NIB*) and interlaced block (*IB*) schemes as depicted in Figure 4.4. For

Figure 4.4: Different Schemes of Partitioning an Image, (a) fixed 3×3 block scheme, an image of size 6×6 is partitioned into four 3×3 blocks, (b) non-interlaced block scheme, an image of size 8×8 is partitioned into four non-interlaced 4×4 blocks and (c) interlaced block scheme, an image of size 7×7 is partitioned into four interlaced 4×4 blocks.

the interlaced block scheme, any two vertically or horizontally neighboring blocks share one common row or column that is not used in data hiding process. The “embeddability” of a block depends on the “flippability” of determined pixels in the block, which are the center pixels of the blocks for the fixed 3×3 block scheme, all the pixels except the boundary pixels for the non-interlaced block scheme and all pixels except those lie in the sharing rows and columns for the interlaced block scheme. In computing the “flippability” of a determined pixel, a moving window shown in Figure 4.1 is employed to be centered at those determined pixels.

4.2.3 Capacities

Let us assume the probability that a pixel satisfies the “Flippability Criterion” is p , since only one pixel is flipped (if needed) in each block, so, we could assume that the probability for a pixel to be “flippable” in a block is independent to other pixels. The probability for a block to be “embeddable” $\mathcal{P}_{eb} = p$ for fixed 3×3 block scheme since only the center pixels are the determined pixels. For the non-interlaced and interlaced block schemes, if any of the determined pixels in the block is “flippable”, the block is identified as “embeddable”. For a non-interlaced block with block size $\mathcal{B}_z = n \times n$, the probability of none of the determined pixels is “flippable” is $(1 - p)^{(n-2)^2}$, which shows that the probability for a non-interlaced block to be “embeddable” is $\mathcal{P}_{eb} = 1 - (1 - p)^{(n-2)^2}$. Similarly, the probability for an interlaced block with block size $\mathcal{B}_z = n \times n$ to be “embeddable” is $\mathcal{P}_{eb} = 1 - (1 - p)^{(n-2)^2}$. The plot of $\mathcal{P}_{eb}$ for different p and n is shown in Figure 4.5. It

Figure 4.5: Probability of Each Block To Be “Embeddable”.

is not difficult to see from the figure that $\mathcal{P}_{eb}$ has increased for the non-interlaced and interlaced schemes compared with that of the fixed 3×3 block. The probability for a block to be “embeddable” $\mathcal{P}_{eb} = p$ when $\mathcal{B}_z = 3 \times 3$, i.e., the non-interlaced block becomes

fixed 3×3 block. The total block number $\mathcal{N}_k$ is given by

$$\mathcal{N}_k = \begin{cases} \lfloor M/3 \rfloor \times \lfloor N/3 \rfloor & \text{for a FXB with } \mathcal{B}_z = 3 \times 3 \\ \lfloor M/n \rfloor \times \lfloor N/n \rfloor & \text{for a NIB with } \mathcal{B}_z = n \times n \\ \lfloor M/(n-1) \rfloor \times \lfloor N/(n-1) \rfloor & \text{for an IB with } \mathcal{B}_z = n \times n \end{cases}$$

where M and N are height and width of the image, while $\lfloor x \rfloor$ is the floor function which gives the largest integer less than or equal to x . It is obvious that $\mathcal{N}_k$ has increased for the interlaced block scheme compared with that of the non-interlaced block scheme. More pixels can be chosen using the moving window to increase the probability of a block to be “embeddable” for the non-interlaced and interlaced block schemes. A larger $\mathcal{B}_z$ definitely will increase $\mathcal{P}_{eb}$. However, $\mathcal{N}_k$ will be decreased.

4.2.4 Watermark Embedding and Extraction

The watermark embedding process is described as follows.

1. Partition the image into equal size blocks. For convenience, we choose equal size square blocks.
2. Determine the flippability of determined pixels based on “Flippability Criterion”.
3. Once a pixel is identified as “flippable”, the block is marked as “embeddable”, the current “flippable” pixel is identified as the “embeddable” pixel, i.e., “embeddable” location of the block.
4. Proceed to the next block.
5. Repeat steps 2 to 4 until all the blocks are processed.
6. Embed the watermark in the “embeddable” blocks by flipping the “embeddable” pixel (if needed) to enforce the odd-even feature of the number of black or white pixels in the block.

The “embeddability” of a pixel refers to whether a “flippable” pixel will be chosen as the “embeddable” location in a block. In order to guarantee correct watermark extraction,

the “embeddability” of a block should be invariant in the watermark embedding process. Recall from the “Flippability Criterion” that the “flippability” of a pixel is invariant in the watermark embedding process and only one bit watermark is embedded in each block, i.e., at most one pixel will be flipped in each block. So, a “flippable” pixel which is identified as “embeddable” is still “flippable”, hence an “embeddable” block remains “embeddable”.

Now we shall show that flipping a pixel in a block may affect the “flippability” of the pixels in the same block but does not affect that of the pixels in its neighboring blocks. Let us divide the pixels in the “embeddable” block $\mathcal{F}_{eb}$ into two sets, $\mathcal{F}_1$ and $\mathcal{F}_2$. Set $\mathcal{F}_1$ consists of the determined pixels and set $\mathcal{F}_2$ consists of the non-determined pixels such as the boundary pixels for the non-interlaced block scheme or pixels in the sharing rows or columns for the interlaced block scheme. Flipping a pixel in the “embeddable” block which is in set $\mathcal{F}_1$ may affect the “flippability” of its eight neighbors. However, the farthest pixels being affected will be located in the boundary for the non-interlaced block scheme or in the sharing rows or columns for the interlaced block scheme, i.e., pixels in set $\mathcal{F}_2$. This is due to the reason that the pixels in set $\mathcal{F}_2$, i.e., the non-determined pixels, will not be flipped. Since $\mathcal{F}_{eb} = \mathcal{F}_1 \cup \mathcal{F}_2$, so these affected pixels still fall in the same block. Hence, flipping a pixel may change the “embeddable” locations within a block without affecting the “embeddability” of the block. Whereas it does not affect the “flippability” of pixels in its neighboring blocks. Now that the “embeddability” of a block is invariant in the watermark embedding process, the watermark can be extracted from the “embeddable” blocks by computing the odd-even feature of the number of black or white pixels. In the following, we shall elaborate on how the flipping of a pixel may affect the “flippability” of its 8-neighbors in a block of size larger than 3×3 .

The eight neighbors of a pixel can be divided into previous four ($p-4$) and next four ($n-4$) neighbors, which are processed prior to or after the current pixel, respectively, as shown in Figure 4.6. Flipping a pixel may change the “flippability” of its $p-4$ neighbors

Figure 4.6: Previous Four and Next Four Neighbors of Current Pixel.

Figure 4.7: Examples of Flipping a Pixel Changes the “Flippability” of its $P-4$ Neighbors and the “Embeddable” Location Has Changed.

as illustrated in Figure 4.7. In Figures 4.7 and 4.8, pixels A and B are both black pixels and lie in the same block. Before flipping A, B is not “flippable” due to not satisfying VH transition and C transition as shown in Figure 4.7(a) and (b), and Figure 4.7(c) and (d), respectively. Obviously, flipping pixel A changes the “flippability” of one of its $p-4$ neighbors such as B. Since pixel B is processed prior to pixel A, hence the “embeddable” location of the block has changed. On the other hand, flipping pixel A changes the flippability of its $n-4$ neighbors such as pixel B, as shown in Figure 4.8. Before flipping A, B is not “flippable” due to not satisfying VH transition and C transition as shown in Figure 4.8(a) and (b), and Figure 4.8(c) and (d), respectively. However since pixel A is processed prior to pixel B and pixel A is still “flippable” after being flipped. Hence, the

Figure 4.8: Examples of Flipping a Pixel Changes the “Flippability” of its $N-4$ Neighbors and the “Embeddable” Location Remains Unchanged.

“embeddable” location in the block remains unchanged.

4.3 Authentication Mechanism

Enforcing the odd-even feature of a block to embed the watermark is vulnerable to “parity attack”, e.g., an adversary can carefully flip two pixels in a block while keeping the odd-even feature of the block unchanged. To tackle this problem, we propose a hard authenticator watermark-based authentication scheme described below.

4.3.1 Locate Embeddable Pixels Criterion

In order to generate the hard authenticator watermark, the key issue is how to locate the “embeddable” pixels given the watermarked image. When the block size $\mathcal{B}_z = 3$ (for both x and y directions), the non-interlaced 3×3 block becomes fixed 3×3 block. While the patterns of the determined pixels for the interlaced 3×3 block becomes the chess board

pattern. The “embeddable” pixels are the determined pixels located at the centers of the 3×3 blocks for these two cases, which are easily located by checking the “Flippability Criterion”. The effects of flipping a pixel may affect its $p-4$ neighbors in a block larger than 3×3 and hence may change the “embeddable” locations in a block are shown in Figure 4.7. Therefore, we define **Locate Embeddable Pixels (LEP)** criterion for the non-interlaced block and interlaced block schemes for $\mathcal{B}_z \geq 4$ as follows, which, of course, can be used to locate the “embeddable” pixels for $\mathcal{B}_z = 3$ as well.

- a) Flipping the pixel that does not change the “flippability” of its previous four neighbors which lie in the same 3×3 block ($p-4$ condition).
- b) Flipping the pixel that does not affect the “embeddability” of those $d-2$ pixels (determined pixel) that have already been processed in the same block ($d-2$ condition).

The $d-2$ pixels refer to those pixels that are located within a distance in the range of $[2, 2\sqrt{2}]$ pixel apart from the current processing pixel. The $d-2$ pixels are “flippable” but are not “embeddable” because of not satisfying the $p-4$ condition. Noticeably, the $d-2$ pixel condition only needs to be imposed when $\mathcal{B}_z > 4$ for the non-interlaced and interlaced block schemes. This is due to the fact that the number of determined pixels in a block is four when $\mathcal{B}_z = 4$, no $d-2$ pixel exists. Therefore, only $p-4$ condition is sufficient to locate the “embeddable” pixels. An illustration of the $p-4$ neighbors and those already processed $d-2$ pixels of the current pixel is shown in Figure 4.9. By imposing the *LEP* criterion, the “embeddable” pixel now refers to the “flippable” pixel that is selected and used in data embedding in the block by satisfying both $p-4$ and $d-2$ conditions. The “flippable” pixel refers to the pixel that satisfies the “Flippability Criterion” discussed in Section 4.2.1. An illustration in Figure 4.10 shows the effect of flipping a pixel on the “embeddability” of the $d-2$ pixels.

Figure 4.9: P -4 Neighbors and Those Already Processed D -2 Pixels. Figure 4.10: An Illustration of Flipping a Pixel may Affect the “Embeddability” of its D -2 Pixels. (a) a non-interlaced block of size 6×6 and (b) one block of data of (a) to illustrate that flipping pixel P changes the “embeddability” of pixel Q .

For the non-interlaced block scheme, pixels in each block are processed in a raster scan order. For example, pixels in the 3×3 block which is centered at pixel P shown in Figure 4.10(a) are processed in the sequence of $n_1, n_2, n_3, n_4, P, n_5, n_6, n_7$ and n_8 . Q is a “flippable” pixel, however, flipping Q changes the flippability of one of its p -4 neighbors such as pixel n_3 , so Q is identified as “non-embeddable”. P is “flippable” and flipping P does not change the flippability of its p -4 neighbors, so, P is tentatively identified as the first “embeddable” pixel in the block. P may be flipped to embed the watermark data. If P needs to be flipped, the embeddability of those d -2 pixels of P , e.g., Q needs to be checked again to avoid changing the “embeddable” location in the extraction process. If Q becomes “embeddable”, P cannot be flipped. Otherwise, P can be flipped. For example, in Figure 4.10(b), flipping P has changed one of the transition conditions such

as IR transition of n_3 . Now, flipping Q does not change the flippability its $p-4$ neighbors such as n_3 . Since Q is the first “flippable” pixel in the block, there is no $d-2$ pixel for Q . So, Q becomes “embeddable”. During the watermark extraction, since Q is processed prior to P , the “embeddable” location has changed to Q . Hence, P cannot be flipped.

The reason why the interference exists lies in the fact that the tentatively identified “embeddable” pixels by satisfying the $p-4$ condition and those affected $d-2$ pixels share the same $p-4$ neighbors. Hence, the $d-2$ pixel condition is needed to ensure that no interference exists between two determined “flippable” pixels within a block. In this way, each “embeddable” pixel is isolated from other neighboring “flippable” candidates such that it can be located. Excluding boundary pixels from flipping renders the minimum distance between any two “determined” pixels in two neighboring blocks is three. Hence, changes caused by flipping a determined pixel in one block will not affect the “flippability” of the determined pixels in its neighboring blocks. For the interlaced block scheme, pixels located in the sharing rows or columns are always treated as “non-embeddable” and their “flippability” status is always treated as “not changed”. This is due to the fact that these pixels are not used for data embedding, which is similar to the boundary pixels for the non-interlaced block scheme. Hence, flipping a determined pixel in one block does not affect the “flippability” and hence “embeddability” of determined pixels in its neighboring blocks. So, the same criterion for locating the “embeddable” pixel for non-interlaced block scheme can be applied to locate the “embeddable” pixels for interlaced block scheme as well.

4.3.2 Authentication Process

The strategy for the hard authenticator watermark generation and embedding process shown in Figure 4.11 is described below.

1. Find the “embeddable” locations based on Steps 1-5 discussed in Section 4.2.4. Criterion for locating the “embeddable” pixels is also imposed.

2. Similar to clearing least significant bits to generate hash values for grayscale images [89], clear the “embeddable” locations by setting them to fixed values such as “0s” to generate the intermediate image Y_1 .

3. Feed Y_1 into a hash function such as *MD5* to generate the hash value of original image $\mathcal{H}_o = Hash(Y_1)$, e.g., 128 bits are generated using *MD5*.

4. Encrypt $\mathcal{H}_o$ using the private key $\mathcal{K}_s$ of the owner or issuer, e.g., employ *RSA* encryption algorithm $E_k()$ to generate the content signature of the image $\mathcal{W}_s = E_k(\mathcal{H}_o, \mathcal{K}_s)$.

5. Perform *Xor* (Exclusive Or) or concatenation of $\mathcal{W}_s$ with the payload watermark $\mathcal{W}_p$ to generate hard authenticator watermark $\mathcal{W}_r$, e.g., $\mathcal{W}_r = \mathcal{W}_s \parallel \mathcal{W}_p$, where “ $\parallel$ ” represents the “concatenation” operation.

6. Embed the m th $\mathcal{W}_r$ in the “embeddable” blocks by enforcing $\mathcal{W}_r(m) = \text{mod}(\mathcal{N}_b, 2)$, where $\mathcal{N}_b$ is the number of black or white pixels in the block and $\text{mod}()$ is the modulo 2 function.

Figure 4.11: Hard Authenticator Watermark Embedding Process.

The hard authenticator watermark extraction and verification process shown in Figure 4.12 is summarized below.

1. The first three steps, i.e., find the “embeddable” locations, generate the intermediate image Y_{w1} and generate the hash value $\mathcal{H}_w$ of the watermarked image Y_w are the same as that of steps 1-3 in the embedding process.
2. Extract the watermark $\mathcal{W}'_r$ by $\mathcal{W}'_r(m) = \text{mod}(\mathcal{N}'_b, 2)$ from the “embeddable” blocks of Y_w , where $\mathcal{N}'_b$ is the number of black or white pixels in the block of Y_w . Split $\mathcal{W}'_r$ into two parts: the content signature $\mathcal{W}'_s$ and payload watermark $\mathcal{W}'_p$.
3. Decrypt $\mathcal{W}'_s$ by providing the public key $\mathcal{K}_p$ to obtain the $\mathcal{H}'_o$, e.g., employ *RSA* decryption algorithm $D_k()$ to decrypt the first 1024 bits of $\mathcal{W}'_r$ to obtain $\mathcal{H}'_o = D_k(\mathcal{W}'_s, \mathcal{K}_p)$.
4. Compare $\mathcal{W}'_p$ with $\mathcal{W}_p$ and $\mathcal{H}_w$ with $\mathcal{H}'_o$. If $\mathcal{H}'_o$ match $\mathcal{H}_w$ and $\mathcal{W}'_p$ is the same as $\mathcal{W}_p$, the authenticity and integrity of the image can be ensured.

Figure 4.12: Hard Authenticator Watermark Extraction and Verification Process.

4.4 Experimental Results

4.4.1 Capacity and Visibility

A wide varieties of images, including cartoon images and text images in French, English, Chinese and Japanese and a handwritten image are used to test the capacity of using different types of blocks to partition the image. The results shown in Table 4.1 reveal that by employing the non-interlaced block scheme of size 4×4 , the capacity increases compared with that of the fixed 3×3 block scheme. Employing the interlaced block scheme of size 3×3 , the capacity increases further. Experimentally, the use of interlaced block scheme with size 4×4 gives the largest capacity for most text images. Different types and sizes of the blocks should be chosen for different applications so that a good compromise between visual quality and capacity can be made. In general, from the fixed 3×3 block to the non-interlaced and interlaced block schemes which are of the same size, both computational load and capacity increase.

Table 4.1: Capacity Comparison for Different Types of Blocks

File Name (.tif)	File Size (pixels)	Capacity (bits)					
		<i>FXB</i> 3×3	<i>NIB</i> 4×4	<i>IB</i> 3×3	<i>IB</i> 4×4	<i>IB</i> 5×5	<i>IB</i> 6×6
Woman	416×691	417	665	854	1169	1129	1028
PGirl	207×207	179	261	396	478	462	383
FrenchText	512×512	1509	2448	3383	4389	4282	3822
EnglishText	534×534	956	1518	2267	2725	2368	1934
EnglishText	204×204	173	291	426	517	474	424
ChineseText	540×540	1462	2180	3229	3905	3522	3054
ChineseText	336×336	482	733	1052	1261	1159	984
HandwrittenText	336×336	313	554	741	972	941	858
JapaneseText	336×336	526	822	1180	1488	1364	1188

Experiments are also conducted to compare the capacity of using the same type of blocks with different block sizes for interlaced and non-interlaced block schemes. The

images used in the experiments are three text images in Chinese, English and Japanese, one handwritten image and one cartoon image. All the images are of size 336×336 . The capacities of using different block sizes for the non-interlaced and interlaced block schemes are plotted in Figure 4.13(a) and (b). It can be observed from the figure, significant increase in capacity can be achieved by using the interlaced block scheme. Generally, the increase is more significant when the block size is small, revealing the high efficiency in utilizing the “flippable” pixels. For the non-interlaced and interlaced block schemes, the capacity varies with the block size and it reaches its maximum when the block size takes certain values such as 4×4 , 5×5 and 6×6 . How to achieve the largest capacity depends on the block size, the “flippability” of each pixel and the distribution of “flippable” pixels, which varies for different types of images.

Comparisons of the visual quality in Figure 4.14 are performed by using different types of blocks such as the fixed 3×3 block scheme, non-interlaced block scheme of size 4×4 and interlaced block scheme of size 4×4 . It can be observed from the results that from the fixed 3×3 block scheme to the non-interlaced block scheme of size 4×4 and the interlaced block scheme of size 4×4 , the capacity increases at the cost of the degradation of visual quality of the watermarked images. Applying Equation (2.1) defined in Chapter 2 to evaluate the visual quality of the resultant watermarked images shown in Figure 4.14, the total distortion (D_t) is shown in Table 4.2. It is not difficult to see from the table that the results agree with our visual observations, i.e., from the fixed 3×3 block scheme to the non-interlaced block scheme of size 4×4 and the interlaced block scheme of size 4×4 , the total distortion increases in exchange for higher capacity.

4.4.2 Authentication Results

Extensive experiments are conducted to test the correctness of the locate “embeddable” pixels (LEP) criterion. Fifty random images of size 500×500 are used, which are gen-

(a)

(b)

Figure 4.13: Capacity of Using Different Block Sizes for (a) Non-interlaced and (b) Interlaced Block Schemes. The numbers in x-axis represent the block sizes, e.g., n represents block size $(n + 2) \times (n + 2)$.

erated by randomly assigning “1” or “0” to the pixels. These images are used for testing due to the fact that random images consist of different types of patterns, which are more complex than natural images. When $\mathcal{B}_z = 4$, imposing the $p-4$ condition is sufficient to

Figure 4.14: Comparison of Hiding Effects Using Different Types of Blocks. (a)-(c) original text image of size 336×336 , (a) Chinese, (b) English and (c) Handwritten. (d)-(l) the watermarked images. (d) hide 482 bits by $FXB 3 \times 3$, (e) hide 733 bits by $NIB 4 \times 4$ and (f) hide 1261 bits by $IB 4 \times 4$. (g) hide 447 bits by $FXB 3 \times 3$, (h) hide 672 bits by $NIB 4 \times 4$ and (i) hide 1237 bits by $IB 4 \times 4$. (j) hide 313 bits by $FXB 3 \times 3$, (k) hide 554 bits by $NIB 4 \times 4$ and (l) hide 972 bits by $IB 4 \times 4$.

Table 4.2: Total Distortion of Using Different Types of Blocks

File Name	$FXB\ 3\times 3$	$NIB\ 4\times 4$	$IB\ 4\times 4$
ChineseText	143.38	189.38	351.83
TypeText	98.13	146.25	318.59
HandwrittenText	74.13	139.88	242.38

locate the “embeddable” pixels. However, when $\mathcal{B}_z \geq 5$, both $p-4$ and $d-2$ conditions are required so that the “embeddable” pixels can be located. This further enforces our arguments. It is worthwhile noticing that by imposing LEP criterion for the non-interlaced and interlaced block schemes of size $\mathcal{B}_z \geq 4$, the capacity has decreased compared with the full capacity achieved without imposing any constraint (see Table 4.1). The capacity of four images obtained by imposing LEP criterion versus full capacity for the non-interlaced and interlaced block schemes are plotted in Figures 4.15 and 4.16, respectively.

It can be easily observed that by imposing a tighter constraint to locate the “embeddable” pixels, the capacity has decreased. A test on 40 images using different block sizes for the non-interlaced and interlaced block schemes shows that on average the capacity decreases 23% for $\mathcal{B}_z = 4$ and it slowly increases to 29% when $\mathcal{B}_z = 5, 6, 7, 8, 9$. For large block size, e.g., $\mathcal{B}_z = 10, 11, 12$, the capacity decreases 26% on average. This agrees with our intuitive observation that with the increase of $\mathcal{B}_z$, the interference among $d-2$ pixels becomes more severe. Whereas the distance between pixels in a block becomes large when $\mathcal{B}_z$ is very large, hence the interference reduced.

Experiments are also conducted to verify effectiveness of the proposed hard authenticator watermark. A logo image is used as the payload watermark in order to visually show the tampering occurred to the watermarked image. The results are shown in Figure 4.17. It can be observed from the results that the proposed hard authenticator watermark is effective in detecting any tampering occurred to the watermarked image. The logo image can be reconstructed successfully when no tampering occurs. However, when tampering

Figure 4.15: Capacity of Imposing *LEP* Criterion versus Full Capacity for Non-interlaced Block Scheme, (a) ChineseText, (b) TypeText, (c) JapaneseText and (d) Handwritten-Text. The numbers in x-axis represent the block sizes, e.g., n represents block size $(n + 2) \times (n + 2)$.

occurs, even if the tampering is minor, e.g., only one word is shifted slightly up, the computed hash value varies significantly.

Figure 4.16: Capacity of Imposing *LEP* Criterion versus Full Capacity for Interlaced Block Scheme, (a) ChineseText, (b) TypeText, (c) JapaneseText and (d) HandwrittenText. The numbers in x-axis represent the block sizes, e.g., n represents block size $(n + 2) \times (n + 2)$.

4.4.3 Comparisons

The visual effects of the watermarked image obtained by our proposed method are compared with other block-based methods proposed by Wu *et al.* [13], Wu [14] and Wu and Liu [15] (which are abbreviated as Wu *et al.* for subsequent discussions), Tseng *et al.* [12], Lu *et al.* [63] and Yang and Kot [49]. The same amount of data varying from low, medium

of the author's watermark, *coefficient and quantization selection keys* (which we describe later) and the quantization parameter Δ are necessary to embed and to extract the mark. The watermark can be an encrypted version of the author identification which is

(a)

of the author's watermark, *coefficient and quantization selection keys* (which we describe later) and the quantization parameter Δ are necessary to embed and to extract the mark. The watermark can be an encrypted version of the author identification which is

(b)

of the author's watermark, *coefficient and quantization selection keys* (which we describe later) and the quantization parameter Δ are necessary to embed and to extract the mark. The watermark can be an encrypted version of the author identification which is

(c)

(d)

(e)

(f)

Figure 4.17: Authentication Results. (a) the original image of size 920×230 , (b) hide 1056 bits by the fixed 3×3 block scheme and (c) the watermarked image which is tampered, “embed” in the 3rd line is shifted slightly up. (d) the original logo image, (e) the reconstructed logo image when no tampering occurs and (f) the reconstructed logo image when the watermarked image has been tampered.

to high (111, 180 and 260 bits) is embedded in an image using different methods. For low payload, block size 12×12 is chosen for Wu *et al.*'s method to ensure that each block has a suitable pixel to flip in shuffled domain. Block size of 16×16 , 13×13 , 7×7 and 9×9 are chosen for Tseng *et al.*'s method, Lu *et al.*'s method, Yang and Kot's method and our proposed method, respectively. These block sizes are chosen such that good visual quality of the watermarked image can be obtained while still satisfy the capacity requirement. The results are shown in Figure 4.18. Whereas block sizes 16×16 , 8×8 , 6×6

and 4×4 are chosen for Tseng *et al.*'s method, Wu *et al.*'s method, Lu *et al.*'s method and our proposed method to embed high payload with the results shown in Figure 4.19. Noticeably, The capacity for Yang and Kot's approach [49] is not enough to embed the medium payload and high payload, hence it is not shown in Figure 4.19. The incurred average per pixel distortion ($\mathcal{D}_a$) for different methods computed using Equation (2.2) is shown in Table 4.3.

Figure 4.18: Comparison Results for Low Capacity. (a) original image of size 173×115 , (b)-(f) watermarked image with 111 bits embedded by using (b) proposed method, (c) Wu *et al.*'s method, (d) Tseng *et al.*'s method, (e) Lu *et al.*'s method and (f) Yang and Kot's method.

Table 4.3: Average Per Pixel Distortions ($\mathcal{D}_a$) for Different Methods

Capacity	Proposed	Wu <i>et al.</i>	Tseng <i>et al.</i>	Lu <i>et al.</i>	Yang & Kot
Low	0.44	0.39	0.86	0.73	0.43
Medium	0.45	0.44	0.86	0.72	*
High	0.46	0.52	0.87	0.74	*

* denotes that there is not enough capacity to embed the payload data.

It can be observed from Figures 4.18 and 4.19, and Table 4.3 that the visual quality of the watermarked images obtained using our proposed method is comparable to that of Wu *et al.*'s method by embedding the same amount of data at low and medium capacity.

Figure 4.19: Comparison Results for High Capacity. (a) original image of size 173×115 , (b)-(f) watermarked image with 260 bits embedded by using (b) proposed method, (c) Wu *et al.*'s method, (d) Tseng *et al.*'s method and (e) Lu *et al.*'s method.

For high capacity embedding, our method outperforms other methods in terms of visual quality as shown in Table 4.3 and Figure 4.19. $\mathcal{D}_a$ is high for Tseng *et al.*'s method due to the randomness in choosing the edge pixels such as pixels near straight lines. Similarly, $\mathcal{D}_a$ is high for Lu *et al.*'s method due to the center pixels of some thin strokes are flipped. $\mathcal{D}_a$ of Yang and Kot's method is low due to the denoising effects of the patterns, however, the capacity of this method is rather low. Similar results are obtained when applying to other images.

Note that 20 experiments are carried out for different methods with the mean values of average per pixel distortion ($\mathcal{D}_a$) are shown in Table 4.3. In these experiments, we choose different shuffle keys and block sizes for Wu *et al.*'s method; choose different block sizes and embed different number of data for each block for Tseng *et al.*'s method; choose different block sizes for Lu *et al.*'s method, Yang and Kot's method and our proposed method. The low average per pixel distortion is resulted for low capacity using Wu *et al.*'s method in Table 4.3. This is due to the facts that a good pattern with low distortion score can be chosen in each shuffled block of size 12×12 . It should be noted that the

watermark may lose its synchronization if some changes occurred to the images change the “embeddability” of the block. To provide counter measure against the desynchronization problem, Error Correction Coding can be applied.

4.5 Discussions

4.5.1 System Security

We now study the following two attacks that may be mounted to our proposed authentication scheme. The possibilities of making the content alterations while keeping the images still authentic. Recall from Section 4.3.2, the content signature $\mathcal{W}_s$ is concatenated with $\mathcal{W}_p$ or appended to the same length as $\mathcal{W}_p$ and then Xored with it to generate the final watermark, i.e., $\mathcal{W}_r = \mathcal{W}_s \oplus \mathcal{W}_p = E_k(\mathcal{H}_o, \mathcal{K}_s) \oplus \mathcal{W}_p$. In order to compute the hash value, the images are divided into two regions: region I consisting of the major contents of the image which is used to compute $\mathcal{H}_o$ and region II consisting of the “embeddable” edge locations which is used to carry the information. Now, assume the adversary has made modifications in region I, the hash value of the watermarked image $\mathcal{H}_w$ will vary significantly from the retrieved hash value of the original image $\mathcal{H}'_o$. Therefore the tampering can be easily detected upon comparison. On the other hand, if the tampering occurred to region II, there exist two cases due to the watermark is the concatenation or *Xor* of $\mathcal{W}_s$ and $\mathcal{W}_p$. The tamperings can result in changes in either $\mathcal{W}'_s$ or $\mathcal{W}'_p$, or both of them. If $\mathcal{W}'_s$ changes, it will render the retrieved $\mathcal{H}'_o$ different from $\mathcal{H}_w$. On the other hand, if $\mathcal{W}'_p$ changes, the watermark verification will fail. Hence, we claim that security level of the proposed scheme is high.

The possibility for an adversary to hide his own data while keeping the images still authentic. For the worst case, assume that multiple copies of the watermarked images generated from the same original image is available to the adversary and the content

signature $\mathcal{W}_s$ of the same image is generated using the same private key. Under this assumption, the adversary may obtain $\mathcal{W}_s$ of the image by comparing the different extracted watermarks from the watermark verification software. In this case, he could embed his own watermark and use the same $\mathcal{W}_s$ to get his watermark verified. Therefore, to counter against this attack, one pair of keys can only be applied to one image.

4.5.2 Security Against Steganalysis and VQ Attack

The locations that are suitable for data hiding in the proposed method can be found by checking the “Flippability Criterion”. Assume the data hiding scheme is open to the public domain, the adversary can easily figure out the data hiding locations and thus reveal the existence of the watermark by checking the same conditions. In order to provide counter measures against the steganalysis attack, the proposed scheme can be modified as follows. Instead of always choosing the first pixel in a block that satisfies the “Flippability Criterion”, the location can be chosen according to the output from a random sequence generator. A priority score can be assigned to the list of patterns that satisfy the “Flippability Criterion” based on the visual distortions caused by flipping the pixel and the patterns can be sorted. Depending on the output of the pseudo random generator, one among several “flippable” pixels can be chosen. In this case, the secret key used to generate the random sequence needs to be conveyed to the receiver side. The larger the block size, the more secure the locations that are used for the data hiding are.

For the fixed 3×3 block scheme, only the center pixels of the blocks are the candidate pixels, so the probability that the adversary can figure out the embedding locations is 1. Therefore, the fixed 3×3 block scheme is not a good choice from security point of view. On the other hand, for the non-interlaced and interlaced block schemes, given $\mathcal{B}_z = n \times n$, the least probability that the adversary can figure out the embedding locations in a block

is $\frac{1}{(n-2)^2}$, which reversely means that probability for the embedding locations to be safe for a total of $\mathcal{N}_k$ blocks is $1 - \frac{1}{((n-2)^2)^{\mathcal{N}_k}}$. However, $\mathcal{N}_k$ for a non-interlaced block is smaller than that of the interlaced block as discussed in Section 4.2.3. Obviously, the interlaced block scheme is more secure from steganalysis point of view.

One important malicious attack for fragile watermarking schemes is the “cutting and pasting” attack, which is designed to defeat the block-wise independent embedding the information into the host image [90]. The success of mounting a vector quantization (VQ) attack is based on the assumption that the attacker has a partial knowledge of the possible watermark patterns and block size parameters. The VQ attack starts by collecting a set of watermarked images $\{Y_{w1}, Y_{w2}, \dots, Y_{wk}\}$, each of the same size $M \times N$ and with the same watermark pattern embedded. Given an image Y of size $M \times N$, to generate a faked watermarked image, it proceeds as follows. Find the set of blocks B_{w_r} in the set of watermarked images and the one with the least visual difference will be chosen. In this way, the attacker achieves his goal of counterfeiting the block without being detected. Our proposed scheme is immune to this attack due to the hash value is computed on the whole image by setting those “embeddable” locations to fixed values, the block independence is not valid in this case.

4.5.3 Visual Quality

To evaluate the visual quality of the resultant watermarked image of our proposed scheme, the visual distortion table proposed by Wu [15] and Wu and Liu [15]. A list of patterns that satisfy the “Flippability Criterion” is shown in Figure 4.20 and the scores assigned for different patterns are listed in Table 4.4. It can be seen from the table, all the patterns are assigned high scores except the pattern shown in Figure 4.20(b), which occurs very rare in thin stroke. Special rules could be set to avoid flipping pixels for this pattern.

The “ L ” pattern shown in Figure 4.20(a) and (f) is the most commonly used pattern for binary image data hiding [13–15, 49, 59] due to its shape invariance in flipping pixels and good matching with human visual perception.

Figure 4.20: A List of Patterns that Satisfy the “Flippability Criterion”, excluding patterns that differ only by rotation, complement and mirroring.

Table 4.4: Scores Assigned for Different Patterns

Pattern	Score	Pattern	Score	Pattern	Score	Pattern	Score	Pattern	Score
(a)	0.625	(b)	0	(c)	0.25	(d)	0.125	(e)	0.25
(f)	0.625	(g)	0.375	(h)	0.25	(i)	0.25	(j)	0.125

4.5.4 Computational Complexity

Comparisons of computational complexity of the proposed method for the fixed 3×3 block, non-interlaced and interlaced block schemes with that of Wu *et al.*’s method [13–15] and Tseng *et al.*’s method [12] are made. To compute the VH transitions in Equation (4.1), it needs 3 logical “Not”, 2 logical “And” and 2 additions for $\mathcal{N}_{vhw}$ and 2 logical “And”, 2 additions for $\mathcal{N}_{vhb}$. To compute the IR transitions in Equation (4.2), it requires 3 logical “Not”, 3 logical “And” and 4 additions. To compute the C transitions in Equation (4.3), it requires 5 logical “And” and 4 additions. Hence, totally it needs 12 logical “And”, 6 logical “Not” operations and 12 additions. The same operations are also required for

the watermarked image in the extraction process. The main computations required for each method are listed in Table 4.5. In the table, M and N are the height and width of the image and $\mathcal{N}_k$ is the total block number discussed in Section 4.2.3 for our proposed method. While the numbers of determined pixels in each block $\mathcal{N}_d$ are 1, $(n-2)^2$ and $(n-2)^2$ for FXB , NIB and IB schemes, respectively. Let us assume the non-interlaced block is used for Wu *et al.*'s method and Tseng *et al.*'s method (not excluding block boundary) and assume r is the number of bits embedded in each block for Tseng *et al.*'s method. It can be easily observed from Table 4.5 that the computational load for

Table 4.5: Computational Load Comparison

Methods	Operations	Required Computations
Proposed	Calculate transitions	$\mathcal{N}_k \times \mathcal{N}_d \times (24 \text{ And; } 12 \text{ Not; } 24 \text{ Add})$
	Calculate odd-even feature	$M \times N$ increment
Wu <i>et al.</i>	Pattern matching	$\mathcal{N}_k \times \mathcal{N}_d \times (9 \text{ Ass; } 8 \text{ Add; } 8 \text{ Mul})$
		$\mathcal{N}_k \times (\mathcal{N}_d \log_2 \mathcal{N}_d)$ Com
	Generate shuffle key	$M \times N$ assignment
	Calculate odd-even feature	$M \times N$ increment
Tseng <i>et al.</i>	Generate key/image matrix	$M \times N$ assignment
	Generate weight matrix	$M \times N$ (Mod; Ass; Bit shift)
	Calculate bit value	$\mathcal{N}_k \times \frac{r(r-1)}{2}$ bit shift
	Check priority	$\mathcal{N}_k \times (\mathcal{N}_d \log_2 \mathcal{N}_d)$ Com
	Calculate distance matrix	$\mathcal{N}_k \times \mathcal{N}_d \times (2 \text{ Mul; } 3 \text{ Add})$
	Calculate block sum	$M \times N$ (Xor; Mul; Add)

Add: Addition; Mul: Multiplication; Com: Comparison; Mod: Modulo 2.

FXB scheme is much shorter than that of NIB and IB schemes. The IB scheme is the most time-consuming among the three types of blocks due to its largest $\mathcal{N}_k$. However, when $\mathcal{B}_z = 3$, the NIB scheme is equivalent to FXB scheme and the determined pixels in IB 3×3 scheme become the chess board pattern, $\mathcal{N}_d = 1$ for all three cases. The basic computations required for our proposed schemes are in the order of $O(M \times N)$ and the basic computations are logic operations and additions. Note that in order to locate the “embeddable” pixels to incorporate the cryptographic signature for $\mathcal{B}_z \geq 4$,

re-checking of the $p-4$ conditions are required for the current “flippable” pixel and those already processed “flippable” $d-2$ pixels, which increases the computational load. The most time-consuming computations for Wu *et al.*’s method are the pattern matching and comparisons required in order to find the best patterns in each block. Multiplication is involved in the pattern matching process. The lower bound of the comparison is of order $O(\mathcal{N}_d \log_2 \mathcal{N}_d)$ for each block [91]. The computational load increases significantly when the block size increases. The most time-consuming computations for Tseng *et al.*’s method are the calculation of the distance matrix for each pixel and comparisons of the distance matrix for any two pixels in each block, in which the latter needs a lot of time especially for a large block size. Multiplication is needed for distance calculation for any two pixels within a block. For the same register addressing mode, the clock cycles required for multiplication are 42 clock cycles for 80486, which is much longer than that of addition and logic operations, e.g., both are 1 clock cycles [92].

4.6 Chapter Summary

In this chapter, we propose a novel data hiding scheme for binary images authentication, in which the “flippability” of a pixel is assessed by the “Connectivity-Preserving” criterion. The uneven “embeddability” of the image is handled by embedding the watermark adaptively in those “embeddable” blocks. Different partition schemes such as fixed 3×3 block, non-interlaced and interlaced blocks are studied and the capacity varying from moderate to high has achieved. The problem of how to locate the “embeddable” pixel in a block is addressed such that the cryptographic signature can be incorporated, which is effective in detecting any tampering occurred to the watermarked image. Comparisons with prior methods show the superiorities of the proposed method in visual quality, capacity, computational complexity and security.

Chapter 5

Morphological Transform-Based Data Hiding

In Chapter 4, we present a pattern-based data hiding scheme in spatial domain. In this Chapter, we investigate the problem of data hiding on binary images in transform domain. To this end, we propose a morphological transform-based data hiding (*MTDH*) scheme. The reason that “*morphological*” is used to name the transform is due to the facts that the nonlinear wavelets are based on morphological operators. The basic idea of mathematical morphology (e.g., erosion, dilation, opening and closing) is to use a structuring element of varying sizes to examine the geometrical structure of an image by matching it in various locations of the image. The non-linear image operators resulted from this matching process can thus be used to analyze the geometrical and topological structure of an image [85, 86, 93–96]. One simple example of the morphological operators such as erosion can be obtained by taking the minimum over two samples, while dilation can be obtained by taking the maximum of the two samples. Detail definitions of the morphological operators can be found in [93, 94].

One intuitive idea for data hiding in transform domain is to use the detail coefficients as a location map to determine the data hiding locations. However, blind watermark extraction is difficult to be achieved since flipping a pixel has changed the corresponding

detail coefficients. Therefore, we further extend the “Connectivity-Preserving” idea presented in Chapter 4 and view flipping an edge pixel in binary images as shifting the edge location horizontally one pixel to the left or right and vertically one pixel up or down, namely, “Edge Shifting” phenomenon. We propose four morphological binary wavelet transforms interlaced in 3×3 blocks to track the shifted edges. The coefficients obtained from the interlaced transform are directly employed to determine the “embeddability” of a coefficient. The use of morphological binary wavelet transform (*MBWT*) addresses the “quantization error” issue arisen from the discrete cosine transform (*DCT*) *DC*-based approach [19]. The two processing cases in which the “flippability” condition of one is not affected by flipping the candidates of another can be combined, namely “Orthogonal Embedding”. The use of “Orthogonal Embedding” can achieve significant gain in capacity by slightly compromising the visual quality. Processing an image in 2×2 blocks and flipping at most two pixels in each 2×2 block allow us to solve the problem of “un-embeddability” of the block boundary and sharing rows or columns in the block-based approach. As a result, the efficiency of utilizing the “flippable” pixels has significantly improved. Security of the system is ensured by employing the hard authenticator watermark.

5.1 Background and Motivation

Data hiding for binary images in transform domain is known to be difficult due to the binary nature of the data. An early investigation of embedding the watermark in spatial domain to achieve unobtrusiveness and detecting the watermark in frequency domain to achieve high robustness can be found in [18]. A further investigation is made in which the watermark is embedded in the *DC* components of *DCT* [19]. A pre-blurring and a post-biased binarization are required in order to embed the watermark successfully. A discrete wavelet transform domain authentication technique [87] quantizes the coefficients

to even or odd multiples of a step size to embed the information. The difficulty in applying real-valued transforms [19,87,97] to binary images lies in two facts. One is due to the real valued nature of the coefficients. Embedding the watermark directly in the coefficients may cause the loss of the watermark in the post-binarization process after the inverse transform is computed. Another one is that the real-valued coefficients require extra memory space in the embedding process.

Swanson and Tewfik [98] proposed a theory of binary wavelet decomposition of binary images. Sweldens [99–101] introduced the lifting schemes, in which wavelets are not necessarily translating and dilating of one fixed function. A general and flexible approach for the construction of morphological wavelets by the lifting scheme is proposed by Goutsias and Heijmans [94,95], and Heijmans and Goutsias [96]. The morphological wavelet transform can be adapted easily to binary images for document image processing, analysis, compression and data hiding applications.

Most data hiding techniques for binary images are based on spatial domain. Typical methods are the pattern or rule based methods, which are more intuitive however more efforts are needed in matching different patterns or defining rules. Generally speaking, data hiding based on real transforms does not work well for binary images due to the distortion introduced in the pre/post-processing. We observe that the morphological binary wavelet transform [96] can be used to track the transitions in binary images efficiently. However, computing a single transform loses part of the transition information, which is undesirable for binary images data hiding. The “unembeddability” of the block boundary or sharing rows or columns in the block-based approach has decreased the efficiency of utilizing the “flippable” pixels, which can be addressed by employing “Orthogonal Embedding” in the proposed approach. In addition, directly using the detail coefficients as a location map to determine the data hiding locations creates difficulty in finding the same locations in the watermarked images.

Motivated by the above situations, we propose

1. To track the transitions between any two or three pixels in binary images efficiently using the interlaced morphological binary wavelet transform (*IMBWT*). The redundancy of *IMBWT* is eliminated by the lifting morphological binary wavelet transform (*LMBWT*).
2. To process the image based on 2×2 blocks, which is not only flexible in tracking transitions within a binary image, but also efficient in utilization of the “flippable” pixels.
3. To achieve low computational complexity by implementing the proposed transforms by means of some simple operations such as “exclusive OR”.
4. To track the shifted edges by employing the coefficients obtained from *IMBWT* to facilitate blind watermark extraction and cryptographic signature incorporation.
5. To achieve a large capacity by combining different processing cases which makes it possible to flip at most two pixels in some 2×2 blocks. To increase the system security by randomly choosing the processing cases and locations through a random key.

This Chapter is organized as follows. A brief review on 1-D *MBWT* and its extension to 2-D *MBWT* is presented in Section 5.2, followed by the proposed *IMBWT*. The binary images authentication by employing *IMBWT* is discussed in Section 5.3. Elimination of the redundancy in *IMBWT* by the proposed *LMBWT* is discussed in Section 5.4. Experimental results and discussions are presented in Section 5.5. Extending the flippability block to a larger neighborhood is discussed in Section 5.6. Applications of the proposed authentication system are presented in Section 5.7. Relationships of the proposed *CPDH* and *MTDH* schemes are discussed in Section 5.8 and Section 5.9 summarizes the Chapter.

5.2 Interlaced Morphological Binary Wavelet Transform

One intuitive idea in utilizing *MBWT* for data hiding would be to use the detail coefficients, since these coefficients contain the edges in horizontal, vertical and diagonal directions. However, flipping a pixel has changed the coefficients, as illustrated in Figure 5.1(a) and (b). It is not difficult to see that the same edges used to determine the data hiding locations cannot be found in the watermarked image. We observe that flipping an edge pixel in binary images is equivalent to shifting the edge location horizontally one pixel to the left or right and vertically one pixel up or down, as illustrated in Figure 5.1(c) and (d). In the figure, “1” and “0” represent the black and white pixels, respectively. To this end, we design an *IMBWT* to keep track of the shifted edges so that blind watermark extraction can be achieved.

Figure 5.1: Illustration of Changes in Coefficients ((a) and (b)) and “Edge Shifting” Phenomenon ((c) and (d)).

5.2.1 Signal Analysis and Synthesis

In this Section, we start the discussion with signal analysis and synthesis which is similar to [95, 96]. A brief review on the 1-D signal decomposition given in [96] is also included. Based on this, we further extend the decomposition scheme to 2-D signal and subsequently propose an interlaced transform for the data hiding application.

Consider a family of signal space V_l and detail space W_l at level l . The 1-D wavelet decomposition scheme comprises of one signal analysis operator ψ_l^+ and one detail analysis operator ϖ_l^+ . In addition, it consists of one signal synthesis operator ψ_l^- and one detail synthesis operator ϖ_l^- . Here “+” indicates that the corresponding operator maps a signal to a higher level, towards the direction of reducing the information. Whereas “-” indicates that the operator maps a signal to a lower level, towards the direction of restoring the information. The signal analysis operator $\psi_l^+ : V_l \rightarrow V_{l+1}$ maps a signal, $s_l \in V_l$, from level l to $l+1$ for the scaled signal $s_{l+1} = \psi_l^+(s_l)$, $s_{l+1} \in V_{l+1}$. The detail analysis operator $\varpi_l^+ : V_l \rightarrow W_{l+1}$ maps a signal from level l to $l+1$ for the detail signal $d_{l+1} = \varpi_l^+(s_l)$, $d_{l+1} \in W_{l+1}$. On the other hand, the signal synthesis operator $\psi_l^- : V_{l+1} \rightarrow V_l$ maps a signal from level $l+1$ back to level l to obtain an approximation of s_l , i.e., $\hat{s}_l = \psi_l^-(s_{l+1})$. The detail synthesis operator $\varpi_l^- : W_{l+1} \rightarrow V_l$ maps a detail signal back to level l so as to obtain the detail signal $e_l = \varpi_l^-(d_{l+1})$. The signal at level l is reconstructed by

$$s_l = \psi_l^-(s_{l+1}) + \varpi_l^-(d_{l+1}) \quad (5.1)$$

As discussed in [95,96,101], reconstructing the original signal is possible if the operators satisfy

$$\psi_l^- \psi_l^+ = \varpi_l^- \varpi_l^+ = I \quad (5.2)$$

$$\psi_l^- \varpi_l^+ = \varpi_l^- \psi_l^+ = \underline{0} \quad (5.3)$$

$$\psi_l^- \psi_l^+ + \varpi_l^- \varpi_l^+ = I \quad (5.4)$$

where “I” and “ $\underline{0}$ ” represent the identity and zero operators, respectively. Note that $\{\psi_l^-, \psi_l^+, \varpi_l^-, \varpi_l^+\}$ is a set of biorthogonal filter operators if the conditions in Equations (5.2)-(5.4) are satisfied. For the morphological analysis and synthesis scheme discussed in this Chapter, the bi-orthogonality is formulated in the operator terms only. For

simplicity, we delete the subindex l in the corresponding analysis and synthesis operators, since we consider only one step in the decomposition, i.e., the decomposition between V_0 and V_1 , W_1 . Let “ $\oplus$ ” represent the “exclusive Or (*Xor*)” operation. The signals at level $l + 1$ obtained by applying the analysis operators for a 1-D signal are given by

$$s(i) = \psi^+(s)(i) = s(2i + 1) \quad (5.5)$$

$$d(i) = \varpi^+(s)(i) = s(2i) \oplus s(2i + 1) \quad (5.6)$$

where i denotes the index of the signal at level $l + 1$ and $i = 0, 1, 2, \dots, N - 1$ for a 1-D signal of size N , $N \in \mathbb{Z}$ and $\mathbb{Z}$ denotes the nonnegative integer set. $s(i)$ and $d(i)$ are the coarse and detail signals obtained at level $l + 1$, respectively. The detail signal $d(i)$ contains 1s at each transition from 0 to 1 or vice versa in signal s . The transform in Equations (5.5) and (5.6) are named as even transform since it is calculated starting from an even sample. In Equation (5.5), every odd point is sampled as the coarse signal. The detail signal indicates whether there is a transition starting from the coarse signal. The synthesis operators are given by

$$\psi^-(s)(2i + 1) = \psi^-(s)(2i) = s(i) \quad (5.7)$$

$$\varpi^-(s)(2i + 1) = 0 \text{ and } \varpi^-(s)(2i) = d(i) \quad (5.8)$$

The signal at level l is reconstructed by

$$s(2i) = \psi^-(s)(2i) \oplus \varpi^-(s)(2i) = s(i) \oplus d(i) \quad (5.9)$$

$$s(2i + 1) = \psi^-(s)(2i + 1) \oplus \varpi^-(s)(2i + 1) = s(i) \quad (5.10)$$

The 1-D wavelet decomposition scheme can be extended to two dimensions by sequentially applying the 1-D analysis process to the rows and columns of a 2-D image. We now define a 2-D non-separable transform. Let i and j represent the indices of the signal at level

$l + 1$, where $i = 0, 1, 2, \dots, M - 1$ and $j = 0, 1, 2, \dots, N - 1$ for a 2-D coarse signal of size $M \times N$, $M, N \in \mathbb{Z}$. Designation of the samples in a 2×2 block is shown in Figure 5.2, where $s(2i, 2j)$ denotes the signal located at row $2i$ and column $2j$ at level l . To

$s(2i, 2j)$	$s(2i, 2j+1)$
$s(2i+1, 2j)$	$s(2i+1, 2j+1)$

Figure 5.2: Designations of Samples in a 2×2 Block.

define a 2-D morphological binary wavelet transform, one of the samples in a 2×2 block can be subsampled as the coarse signal. The differences between the sub-sampled sample and its vertical, horizontal, and diagonal (includes diagonal and anti-diagonal directions) neighbors determine the horizontal, vertical and diagonal detail signals. The resulting transformed signals remain binary. Notably, the coarse signal and detail signals will each be $1/4$ the size of the original signal. Further, the transform is lossless. As such, the transform is often reapplied to the coarse signal to produce higher level wavelet transforms. Let the operators for the coarse, horizontal, vertical and diagonal detail signals be denoted by ψ^{ee} , ϖ_h^{ee} , ϖ_v^{ee} and ϖ_d^{ee} , respectively. The superscript “ee” denotes “even-even” transform. The thus obtained transform is named as even-even transform since it is operated on a 2×2 block starting from the even-even coordinates. Other transforms starting from even-odd, odd-even and odd-odd coordinates will be defined in Section 5.2.2. The coarse signal, vertical, horizontal and diagonal detail signals at level $l + 1$ are obtained by applying the analysis operators

$$s^{ee}(i, j) = \psi^{ee+}(s)(i, j) = s(2i + 1, 2j + 1) \quad (5.11)$$

$$v^{ee}(i, j) = \varpi_v^{ee+}(s)(i, j) = s(2i + 1, 2j) \oplus s(2i + 1, 2j + 1) \quad (5.12)$$

$$h^{ee}(i, j) = \varpi_h^{ee+}(s)(i, j) = s(2i, 2j + 1) \oplus s(2i + 1, 2j + 1) \quad (5.13)$$

$$d^{ee}(i, j) = \varpi_d^{ee+}(s)(i, j) = s(2i, 2j) \oplus s(2i + 1, 2j) \oplus s(2i, 2j + 1) \oplus s(2i + 1, 2j + 1) \quad (5.14)$$

where $s^{ee}(i, j) \in V_{l+1}$ and $\{v^{ee}(i, j), h^{ee}(i, j), d^{ee}(i, j)\} \in W_{l+1}$. The synthesis operators of

a 2-D wavelet transform are given by

$$\psi^{ee-}(s)(2i, 2j) = \psi^{ee-}(s)(2i, 2j+1) = \psi^{ee-}(s)(2i+1, 2j) \quad (5.15)$$

$$= \psi^{ee-}(s)(2i+1, 2j+1) = s^{ee}(i, j)$$

$$\varpi_h^{ee-}(s)(2i, 2j+1) = h^{ee}(i, j) \quad (5.16)$$

$$\varpi_v^{ee-}(s)(2i+1, 2j) = v^{ee}(i, j) \quad (5.17)$$

$$\varpi_d^{ee-}(s)(2i, 2j) = v^{ee}(i, j) \oplus h^{ee}(i, j) \oplus d^{ee}(i, j) \quad (5.18)$$

$$\varpi^{ee-}(s)(2i+1, 2j+1) = 0 \quad (5.19)$$

Finally, the signal at level l can be reconstructed by

$$s(2i, 2j) = \psi^{ee-}(s)(2i, 2j) \oplus \varpi_d^{ee-}(s)(2i, 2j) \quad (5.20)$$

$$= s^{ee}(i, j) \oplus v^{ee}(i, j) \oplus h^{ee}(i, j) \oplus d^{ee}(i, j)$$

$$s(2i, 2j+1) = \psi^{ee-}(s)(2i, 2j+1) \oplus \varpi_h^{ee-}(s)(2i, 2j+1) \quad (5.21)$$

$$= s^{ee}(i, j) \oplus h^{ee}(i, j)$$

$$s(2i+1, 2j) = \psi^{ee-}(s)(2i+1, 2j) \oplus \varpi_v^{ee-}(s)(2i+1, 2j) \quad (5.22)$$

$$= s^{ee}(i, j) \oplus v^{ee}(i, j)$$

$$s(2i+1, 2j+1) = \psi^{ee-}(s)(2i+1, 2j+1) \oplus \varpi^{ee-}(s)(2i+1, 2j+1) = s^{ee}(i, j) \quad (5.23)$$

5.2.2 Interlaced Morphological Binary Wavelet Transform

The even-even transform defined in Equations (5.11)-(5.14) for a 2-D signal starts from the even-even coordinates of the signal. As the coarse signals are in the odd-odd locations, the transitions starting from odd-odd to other coordinates in the 2×2 block may easily be assessed from the detail signals obtained by Equations (5.11)-(5.14). However, the transitions starting from even-odd, odd-even and even-even coordinates may not be so easily assessed. This motivates us to design the complementary wavelet transforms operating on 2×2 blocks starting from the odd-even, even-odd and odd-odd coordinates of the signal. These transforms, together with the even-even transform, are collectively

named as “interlaced morphological binary wavelet transform (*IMBWT*)”. For a 1-D signal, the computation of a complementary transform starts with odd numbered samples, namely, “odd transform”. Let ψ^o and ϖ^o denote the operators for the coarse and detail signals, where the superscript “o” denotes the “odd” transform. The resulting transform yielding the coarse $s^o(i)$ and detail $d^o(i)$ signals is given by

$$s^o(i) = \psi^{o+}(s)(i) = s(2i + 2) \quad (5.24)$$

$$d^o(i) = \varpi^{o+}(s)(i) = s(2i + 1) \oplus s(2i + 2) \quad (5.25)$$

An illustration of decomposition of 1-D *IMBWT* is shown in Figure 5.3. Based on the

Figure 5.3: Illustration of 1-D *IMBWT*.

Figure 5.4: Different 2×2 Blocks in a Binary Image. (a) even-even blocks, (b) even-odd blocks, (c) odd-even blocks and (d) odd-odd blocks starting with (even, even), (even, odd), (odd, even) and (odd, odd) coordinates, respectively.

starting absolute coordinates (x, y) in the top left position shown in Figure 5.4. Each 2

$\times 2$ block $B_p(i, j)$ in a 2-D image can be classified as even-even block (*EEB*), even-odd block (*EOB*), odd-even block (*OEB*) and odd-odd block (*OOB*), which is given by

$$\begin{cases} B_p(i, j) \in EEBs & \text{when } (\text{mod}(x, 2) = 0) \wedge (\text{mod}(y, 2) = 0) \\ B_p(i, j) \in EOBs & \text{when } (\text{mod}(x, 2) = 0) \wedge (\text{mod}(y, 2) = 1) \\ B_p(i, j) \in OEBs & \text{when } (\text{mod}(x, 2) = 1) \wedge (\text{mod}(y, 2) = 0) \\ B_p(i, j) \in OOBs & \text{when } (\text{mod}(x, 2) = 1) \wedge (\text{mod}(y, 2) = 1) \end{cases} \quad (5.26)$$

where (x, y) denotes x th row and y th column pixel location of an image, (i, j) denotes the index of the current 2×2 block, $\text{mod}()$ denotes a modulo operation and “ $\wedge$ ” represents logical “And” operation. By sub-dividing an image into even-odd, odd-even and odd-odd 2×2 blocks, three additional transforms can be defined. These four 2×2 blocks within a 3×3 block is shown in Figure 5.5.

Figure 5.5: Four Different 2×2 Blocks in a 3×3 Block of a Binary Image.

Let the operators for the even-odd, odd-even and odd-odd transforms be denoted as:

$\{\psi^{eo}, \varpi_h^{eo}, \varpi_v^{eo}, \varpi_d^{eo}\}$, $\{\psi^{oe}, \varpi_h^{oe}, \varpi_v^{oe}, \varpi_d^{oe}\}$ and $\{\psi^{oo}, \varpi_h^{oo}, \varpi_v^{oo}, \varpi_d^{oo}\}$, respectively. Signals

obtained by applying the analysis operators for even-odd transform are given by

$$s^{eo}(i, j) = \psi^{eo+}(s)(i, j) = s(2i + 1, 2j + 2) \quad (5.27)$$

$$v^{eo}(i, j) = \varpi_v^{eo+}(s)(i, j) = s(2i + 1, 2j + 1) \oplus s(2i + 1, 2j + 2) \quad (5.28)$$

$$h^{eo}(i, j) = \varpi_h^{eo+}(s)(i, j) = s(2i, 2j + 2) \oplus s(2i + 1, 2j + 2) \quad (5.29)$$

$$d^{eo}(i, j) = \varpi_d^{eo+}(s)(i, j) = s(2i, 2j + 1) \oplus s(2i + 1, 2j + 1) \oplus s(2i, 2j + 2) \oplus s(2i + 1, 2j + 2) \quad (5.30)$$

Signals obtained by applying the analysis operators for odd-even transform are given by

$$s^{oe}(i, j) = \psi^{oe+}(s)(i, j) = s(2i + 2, 2j + 1) \quad (5.31)$$

$$v^{oe}(i, j) = \varpi_v^{oe+}(s)(i, j) = s(2i + 2, 2j) \oplus s(2i + 2, 2j + 1) \quad (5.32)$$

$$h^{oe}(i, j) = \varpi_h^{oe+}(s)(i, j) = s(2i + 1, 2j + 1) \oplus s(2i + 2, 2j + 1) \quad (5.33)$$

$$d^{oe}(i, j) = \varpi_d^{oe+}(s)(i, j) = s(2i + 1, 2j) \oplus s(2i + 1, 2j + 1) \oplus s(2i + 2, 2j) \oplus s(2i + 2, 2j + 1) \quad (5.34)$$

Signals obtained by applying the analysis operators for odd-odd transform are given by

$$s^{oo}(i, j) = \psi^{oo+}(s)(i, j) = s(2i + 2, 2j + 2) \quad (5.35)$$

$$v^{oo}(i, j) = \varpi_v^{oo+}(s)(i, j) = s(2i + 2, 2j + 1) \oplus s(2i + 2, 2j + 2) \quad (5.36)$$

$$h^{oo}(i, j) = \varpi_h^{oo+}(s)(i, j) = s(2i + 1, 2j + 2) \oplus s(2i + 2, 2j + 2) \quad (5.37)$$

$$d^{oo}(i, j) = \varpi_d^{oo+}(s)(i, j) = s(2i + 1, 2j + 1) \oplus s(2i + 1, 2j + 2) \oplus s(2i + 2, 2j + 1) \oplus s(2i + 2, 2j + 2) \quad (5.38)$$

where $s^k(i, j)$, $v^k(i, j)$, $h^k(i, j)$ and $d^k(i, j)$ represent the coarse signal, vertical, horizontal and diagonal detail signals respectively obtained by computing transform k at level $l + 1$, $k \in \{ee, eo, oe, oo\}$. For simplicity, we use s_k , v_k , h_k and d_k . The synthesis operators can be similarly defined for even-odd, odd-even and odd-odd transforms.

To process an image in 2×2 blocks, there are four single processing cases (*SPCs*), i.e., even-even, even-odd, odd-even and odd-odd, can be defined depending on the main processing blocks to be *EEBs*, *EOBs*, *OEBs* and *OOBs*. Consider the even-even processing case, where the main processing blocks are *EEBs*, the three 2×2 blocks that are interlaced with the main processing blocks to the horizontal, vertical and diagonal directions are the *EOBs*, *OEBs* and *OOBs*, which are shown as the subsidiary blocks in Figure 5.6. In this figure, the main processing block starts from (even, even) coordinate and the

Figure 5.6: Main and Subsidiary Blocks for Even-Even Processing Case.

subsidiary blocks start from (even, odd), (odd, even) and (odd, odd) coordinates as shown in Figure 5.4. Noticeably, in applying the *IMBWT* for data hiding in binary images, we always process the images based on 2×2 blocks. However, the “flippability” of a coarse signal is determined by considering 3×3 blocks, which consist of both the main processing blocks and the subsidiary blocks.

5.3 Data Hiding Based on *IMBWT*

As discussed in Section 5.2, the suitable data hiding location is the location that flipping an edge pixel is equivalent to shifting the edges horizontally and vertically one position. A *Horizontal Edge* exists if there is a transition between two neighboring pixels along the vertical direction and a *Vertical Edge* exists if there is a transition between two neighboring pixels along the horizontal direction. Flipping a pixel causes changes to its

four surrounding 2×2 blocks as illustrated in Figure 5.7. Observed from the figure,

Figure 5.7: Illustration of Flipping a Pixel Changes the Four Surrounding 2×2 Blocks.

flipping the pixel in the lower right corner of the *EEBs* (main processing blocks) shifts the changes to the three subsidiary blocks, i.e., *EOBs*, *OEBs* and *OOBs*. The coefficients obtained from *IMBWT* can be employed to identify such locations for data embedding so that the shifted edges can be tracked for the convenience of blind watermark extraction. In the following Sections, we shall discuss the flippability conditions for different cases.

5.3.1 Single Processing

As mentioned above, there are four Single Processing Cases (*SPCs*). We define the **Flippability Condition** (“**Cross**” condition) for a *SPC* as follows. A coarse signal is “flippable” if both horizontal and vertical edges exist. Specifically, a horizontal edge exists if either h_{ee} or h_{oe} equals to 1 and a vertical edge exists if either v_{ee} or v_{eo} equals to 1 for the even-even processing case. The flippability condition $f_{ee}(i, j)$ is given by

$$f_{ee}(i, j) = (h_{ee}(i, j) \oplus h_{oe}(i, j)) \wedge (v_{ee}(i, j) \oplus v_{eo}(i, j)) \quad (5.39)$$

Equation (5.39) thus identifies a coarse signal $s_{ee}(i, j)$ (e.g., pixel $s(2i + 1, 2j + 1)$ at the lower right location in Figure 5.5(b)) to be “flippable” if $f_{ee}(i, j) = 1$. Since v_{ee} , v_{eo} , h_{ee} , $h_{oe} \in \{0, 1\}$ and $(v_{eo}(i, j) \oplus v_{ee}(i, j) = 1) \wedge (h_{oe}(i, j) \oplus h_{ee}(i, j) = 1)$, we have $\bar{v}_{eo}(i, j) = v_{ee}(i, j)$, $\bar{v}_{ee}(i, j) = v_{eo}(i, j)$, $\bar{h}_{ee}(i, j) = h_{oe}(i, j)$ and $\bar{h}_{oe}(i, j) = h_{ee}(i, j)$, where $\bar{v}_{ee}$ is the comple-

ment of v_{ee} . Flipping $s_{ee}(i, j)$ leads each coefficient to become its complement, as a result, the flippability condition $f'_{ee}(i, j)$ becomes

$$\begin{aligned} f'_{ee}(i, j) &= (\bar{h}_{ee}(i, j) \oplus \bar{h}_{oe}(i, j)) \wedge (\bar{v}_{ee}(i, j) \oplus \bar{v}_{eo}(i, j)) \\ &= (h_{oe}(i, j) \oplus h_{ee}(i, j)) \wedge (v_{eo}(i, j) \oplus v_{ee}(i, j)) \\ &= f_{ee}(i, j) \end{aligned} \quad (5.40)$$

Hence, the shifted edges have been tracked. The same set of operations can be employed to find the “flippable” locations for the extraction process. Consider an input image Y . The 4-connectivity in a 3×3 block centered at a pixel is calculated as the transitions from “1” to “0” or vice versa from the center pixel to its 4-neighbors. Specifically, the numbers of transitions in the horizontal and vertical directions, represented by $\mathcal{N}_h$ and $\mathcal{N}_v$, are calculated from $Y(2i + 1, 2j + 1)$ to its 4-neighbors (see Figure 5.5(a)) and given by

$$\mathcal{N}_h = \sum_{k=0}^1 Y(2i + 1, 2j + k) \oplus Y(2i + 1, 2j + 1 + k) \quad (5.41)$$

$$\mathcal{N}_v = \sum_{k=0}^1 Y(2i + k, 2j + 1) \oplus Y(2i + 1 + k, 2j + 1) \quad (5.42)$$

By satisfying the “Cross” condition in Equation (5.39), $\mathcal{N}_h$ and $\mathcal{N}_v$ do not change by flipping the center pixel, hence, the 4-connectivity is preserved. Further, the center pixel has one white 4-neighbor horizontally and vertically, therefore it is an edge pixel both horizontally and vertically, which can be easily verified from the list of patterns satisfying the “Cross” condition shown in Figure 5.8. Similarly, the flippability conditions for the

Figure 5.8: List of Patterns that Satisfy the “Cross” Condition, excluding patterns that differ only by rotation, complement and mirroring. Pixels in grid represent “don’t care” pixels.

odd-odd, even-odd and odd-even processing cases are given by

$$f_{oo}(i, j) = (v_{oo}(i-1, j-1) \oplus v_{oe}(i-1, j)) \wedge (h_{oo}(i-1, j-1) \oplus h_{eo}(i, j-1)) \quad (5.43)$$

$$f_{oe}(i, j) = (v_{oe}(i-1, j) \oplus v_{oo}(i-1, j)) \wedge (h_{oe}(i-1, j) \oplus h_{ee}(i, j)) \quad (5.44)$$

$$f_{eo}(i, j) = (v_{eo}(i, j-1) \oplus v_{ee}(i, j)) \wedge (h_{eo}(i, j-1) \oplus h_{oo}(i, j-1)) \quad (5.45)$$

where the candidate pixels are $Y(2i, 2j)$, $Y(2i, 2j+1)$ and $Y(2i+1, 2j)$, respectively; $i \geq 1$ and $j \geq 1$ for $f_{oo}(i, j)$; $i \geq 1$ for $f_{oe}(i, j)$ and $j \geq 1$ for $f_{eo}(i, j)$. In terms of pixel values, Equations (5.39) and (5.43)-(5.45) can be further expressed as

$$f_{ee}(i, j) = (Y(2i+1, 2j) \oplus Y(2i+1, 2j+2)) \wedge (Y(2i, 2j+1) \oplus Y(2i+2, 2j+1)) \quad (5.46)$$

$$f_{oo}(i, j) = (Y(2i, 2j-1) \oplus Y(2i, 2j+1)) \wedge (Y(2i-1, 2j) \oplus Y(2i+1, 2j)) \quad (5.47)$$

$$f_{oe}(i, j) = (Y(2i, 2j) \oplus Y(2i, 2j+2)) \wedge (Y(2i-1, 2j+1) \oplus Y(2i+1, 2j+1)) \quad (5.48)$$

$$f_{eo}(i, j) = (Y(2i+1, 2j-1) \oplus Y(2i+1, 2j+1)) \wedge (Y(2i, 2j) \oplus Y(2i+2, 2j)) \quad (5.49)$$

In summary, the **Flippability Conditions** do help in identifying all the candidate pixels for which the pixels directly above and below the candidate pixel have different colors and the pixels immediately to the left and right of the candidate pixel have different colors. In addition, the two candidate pixels cannot be 4-adjacent, i.e., horizontally or vertically adjacent to each other by considering the visual quality of the resultant image. It can be concluded that the flippability of the center pixel in each 3×3 block is independent of the center pixel value.

5.3.2 Control Sharp Corner (Isolated Corner) Flipping

Controlling sharp corner or isolated corner flipping (*CSCF*) is necessary to avoid the obtrusiveness to human eyes, which is especially applicable for text images. Assume the total number of black pixels in a 3×3 block centered at the candidate pixel is $\mathcal{N}_b$. Let

Figure 5.9: List of Sharp Corner Patterns (a)-(h) and Isolated Corner Patterns (i)-(p) in a 3×3 Block. Pixels in grid represent “don’t care” pixels.

us consider the processing case to be even-even and the flippability condition $f_{ee}(i,j)$ in Equation (5.39) is true. In order to identify the “sharp corner” patterns shown in Figure 5.9(a)-(d) for the $(i,j)th$ location, we firstly check $s_{ee}(i,j)$ and $\mathcal{N}_b$ in each 3×3 block to ensure that $S_{s1} = (s_{ee}(i,j) = 0) \wedge (\mathcal{N}_b = 5)$ is true. Secondly, determine the dominant edge block ($\mathcal{D}_g(i,j)$) and pair block ($\mathcal{P}_b(i,j)$) in a 3×3 block. The $\mathcal{D}_g(i,j)$ is the 2×2 block in which both horizontal and vertical edges exist, e.g., $h_{ee}(i,j) \wedge v_{ee}(i,j) = 1$ in EEB of Figure 5.9(a). The $\mathcal{P}_b(i,j)$ is the 2×2 block that lies to the diagonal direction of $\mathcal{D}_g(i,j)$, e.g., OOB in Figure 5.9(a). Thirdly, ensure that no more black pixels exist in $\mathcal{P}_b(i,j)$. The condition is given by

$$\Pi = \Pi_1 \vee \Pi_2 \vee \Pi_3 \vee \Pi_4 \quad (5.50)$$

where

$$\Pi_1 = v_{ee}(i, j) \wedge h_{ee}(i, j) \wedge \bar{d}_{oo}(i, j) \quad (5.51)$$

$$\Pi_2 = v_{eo}(i, j) \wedge h_{ee}(i, j) \wedge \bar{d}_{oe}(i, j) \quad (5.52)$$

$$\Pi_3 = v_{ee}(i, j) \wedge h_{oe}(i, j) \wedge \bar{d}_{eo}(i, j) \quad (5.53)$$

$$\Pi_4 = v_{eo}(i, j) \wedge h_{oe}(i, j) \wedge \bar{d}_{ee}(i, j) \quad (5.54)$$

The candidate pixel is excluded from flipping when $\Pi = 1$, where “ $\vee$ ” denotes logical “Or” operation. Similarly, the condition for identifying the “isolated corner” patterns shown in Figure 5.9(i)-(l) for $s_{ee}(i, j) = 0$ is given by

$$\Lambda = \Lambda_1 \vee \Lambda_2 \vee \Lambda_3 \vee \Lambda_4 \quad (5.55)$$

where

$$\Lambda_1 = v_{ee}(i, j) \wedge h_{ee}(i, j) \wedge d_{oo}(i, j) \wedge v_{oo}(i, j) \wedge h_{oo}(i, j) \quad (5.56)$$

$$\Lambda_2 = v_{eo}(i, j) \wedge h_{ee}(i, j) \wedge d_{oe}(i, j) \wedge v_{oe}(i, j) \wedge \bar{h}_{oe}(i, j) \quad (5.57)$$

$$\Lambda_3 = v_{ee}(i, j) \wedge h_{oe}(i, j) \wedge d_{eo}(i, j) \wedge \bar{v}_{eo}(i, j) \wedge h_{eo}(i, j) \quad (5.58)$$

$$\Lambda_4 = v_{eo}(i, j) \wedge h_{oe}(i, j) \wedge d_{ee}(i, j) \wedge \bar{v}_{ee}(i, j) \wedge \bar{h}_{ee}(i, j) \quad (5.59)$$

The candidate pixel is excluded from flipping if $\Lambda = 1$. Similarly, the conditions to identify other patterns shown in Figure 5.9 can be formulated.

5.3.3 Double Processing

The capacity of the proposed scheme is determined by the number of pixels that satisfy the flippability condition, which can be increased significantly by combining two single processing cases, namely, Double Processing Case (*DPC*). Observed from Equations (5.46) and (5.47), the flippability condition of the even-even case, i.e., $f_{ee}(i, j)$, is not affected

by flipping the candidates in the odd-odd case, and vice versa. This “nonintersection” property makes the even-even and odd-odd processing cases can be combined. The same applies to the even-odd and odd-even processing cases. For the convenience of discussion, the two processing cases that can be combined is named as a “Pair Case”. The flippability conditions checking for a “Pair Case” is shown in Figure 5.10. The capacity can be roughly

Figure 5.10: Flippability Conditions Checking for a “Pair Case” Using IMBWT.

doubled by combining two processing cases, which can be intuitively seen from the increase of the number of candidate pixels, i.e., from $\lfloor \frac{1}{4} \times M \times N \rfloor$ to $\lfloor \frac{1}{2} \times M \times N \rfloor$ for an image of size $M \times N$. This idea is motivated by Quantization Index Modulation based data hiding method [48, 102]. The even-even and odd-odd processing cases can be viewed as two orthogonal sets of embedders indexed at the 2×2 blocks starting from even-even and odd-odd co-ordinates, denoted as $\mathcal{Q}_{ee}$ and $\mathcal{Q}_{oo}$, respectively. Here the “embedders” refer to an ensemble of the embedding functions of a “Pair Case”, e.g., even-even and odd-odd processing cases. We define the designated 2×2 block $\mathcal{D}_b(i, j)$ (shown in Figure 5.11) as the 2×2 block that contains the two candidate pixels for a “Pair Case”, which can be chosen from one of the processing blocks of the two processing cases to be combined. Only one candidate pixel is chosen to hide the data in each $\mathcal{D}_b(i, j)$, hence, the maximum

Figure 5.11: Designated 2×2 Blocks.

capacity $\mathcal{C}_\zeta \leq \lfloor \frac{1}{4} \times M \times N \rfloor$. By using *DPC*, the choice of the embedder in each $\mathcal{D}_b(i, j)$ depends on the output from a random key $\mathcal{R}_s$, where $\mathcal{R}_s \in \{0, 1\}$. For the m th $\mathcal{D}_b(i, j)$ block, we choose

$$\mathcal{Q}_k = \begin{cases} \mathcal{Q}_{ee} & \text{when } \mathcal{R}_s(m) = 1 \\ \mathcal{Q}_{oo} & \text{when } \mathcal{R}_s(m) = 0 \end{cases} \quad (5.60)$$

By choosing $\mathcal{Q}_{ee}$, we check $f_{ee}(i, j)$ first, mark the current pixel as the “embeddable” location if $f_{ee}(i, j) = 1$ and proceed to the next $\mathcal{D}_b(i, j)$. Otherwise, $f_{oo}(i, j)$ will be checked if $f_{ee}(i, j) \neq 1$. On the other hand, by choosing $\mathcal{Q}_{oo}$, we check $f_{oo}(i, j)$ first, if $f_{oo}(i, j) \neq 1$, $f_{ee}(i, j)$ will be checked. From the above discussion, it is not difficult to see that there may be two pixels being flipped in some 2×2 blocks, e.g., the *EEBs* in Figure 5.11, even if at most one candidate will be flipped in each $\mathcal{D}_b(i, j)$.

5.3.4 Double Processing with Distortion Control

The “nonintersection” property of the even-even and odd-odd, or even-odd and odd-even processing cases renders it possible to minimize the distortion for double processing cases, namely, Double Processing with Distortion Control (*DPDC*). Embedding data using either *DPC* or *DPDC* is named as “Orthogonal Embedding”. Using *DPDC*, each time when a

$\mathcal{D}_b(i, j)$ is processed, an index $\hat{k}$ (which is subsequently used to choose the corresponding embedder $\mathcal{Q}_{\hat{k}}$) is chosen such that the distortion between the original pattern $\mathcal{P}_{ro}$ and watermarked pattern $\mathcal{P}_{rw}$ with respect to embedder $\mathcal{Q}_{\hat{k}}$ is minimized, which is given by

$$\hat{k} = \arg \min_k \| \mathcal{P}_{rw}(\mathcal{Q}_k, k) - \mathcal{P}_{ro}(\mathcal{Q}_k, k) \| \quad (5.61)$$

where $\| \cdot \|$ represents distortion measures such as the visual distortion table [15] and $k \in \{ee, oo\} \vee k \in \{eo, oe\}$. To compare the visual distortions caused using different embedders, the list of patterns that satisfy the flippability conditions should be ranked.

The capacity increase by employing “Orthogonal Embedding”, i.e., using *DPC* or *DPDC*, over that of the interlaced 3×3 block in *CPDH* approach can be illustrated in Figure 5.12. It is not difficult to see from the figure that the missed good candidate, e.g., “L” pattern in the interlaced 3×3 block scheme, has been successfully identified by employing our proposed “Orthogonal Embedding”. Hence, significant increase in capacity can be achieved.

Figure 5.12: Illustration of Capacity Increase by Employing “Orthogonal Embedding” in *MTDH* in (b) versus the Interlaced 3×3 of *CPDH* in (a). Here gray represents black pixel.

5.3.5 Authentication Mechanism

Since the “embeddable” locations can be determined both in the watermark embedding and extraction processes, it is possible to incorporate the Cryptographic Signature (CS) as the hard authenticator watermark [89]. The inputs and some parameters to the watermark generation and embedding system is described in Table 5.1. The strategies for the hard authenticator watermark generation and embedding, and hard authenticator watermark extraction and verification are shown in Figures 5.13 and 5.14 and detailed in Tables 5.2 and 5.3, respectively. Noticeably, both “flippable” candidates should be cleared out to generate the intermediate image using $DPDC$ for a $\mathcal{D}_b(i, j)$ in which both candidates are “flippable”. This is due to the facts that minimizing distortion based on an asymmetric distortion table makes it difficult to locate the same suitable candidates for the watermarked image. A symmetric distortion table can thus be employed to solve this problem. In the verification, if $\mathcal{W}_p \neq \mathcal{W}'_p$, it implies that either Y_w is tampered or not from a legitimate user, e.g., a wrong $\mathcal{K}_p$ is used for verification. For exact authentication, we expect that $\mathcal{R}_d(\mathcal{W}_p, \mathcal{W}'_p) = 0$ (see Equation (2.3)).

5.4 Lifting Scheme

Let’s look at the flippability condition for the odd-odd processing case given in Equation (5.43), four coefficients, i.e., $v_{oo}(i-1, j-1)$, $v_{oe}(i-1, j)$, $h_{oo}(i-1, j-1)$ and $h_{eo}(i, j-1)$, need to be computed and 6 *Xor* and 1 logical “And” operations are required. The redundancy lies in that the center pixel has been calculated four times, which has been eliminated when the flippability condition is expressed in pixel values (see Equation (5.47)). Motivated by this observation, we design a transform to eliminate the redundancy in *IMBWT*.

Figure 5.13: Watermark Generation and Embedding Using *IMBWT*. Figure 5.14: Watermark Extraction and Verification Using *IMBWT*.

5.4.1 Forward and Inverse *LMBWT* for 1-D Signal

Let P and U represent the prediction and update operators, respectively. The lifting scheme for the even processing case consists of the following basic steps.

1. Split the signal into odd and even indexed sets, denoted as $s_o(i) = s(2i + 1)$ and

Table 5.1: Inputs to the Watermark Generation and Embedding Using *IMBWT*

-
1. the image Y .
 2. the payload watermark $\mathcal{W}_p$, whose length is $\mathcal{N}_w$.
 3. set $\mathcal{O}_p = 0, 1, 2$ and 3 for *CSCF*, *SPC*, *DPC* and *DPDC*, respectively.
 4. set the processing key $R_s \in \{0, 1\}$ to determine the sequence of choosing the two embedders in *DPC*.
 5. set $f_s(i, j) = 1$ or 0 to indicate that the watermark data being embedded in the $\mathcal{D}_b(i, j)$ or not.
 6. set $\mathcal{B}_f(i, j)$ to be the odd-even feature of $\mathcal{D}_b(i, j)$, where $\mathcal{B}_f(i, j) = \text{mod}(\mathcal{N}_b, 2)$, $\mathcal{N}_b$ is the number of black or white pixels in $\mathcal{D}_b(i, j)$ and $\text{mod}()$ is the modulo 2 operation.
 7. set $E_k()$ and $D_k()$ as the encryption and decryption algorithm, respectively. Set $\text{Hash}()$ as the hash function such as *MD5* and set $\mathcal{K}_s$ and $\mathcal{K}_p$ as a pair of private and public keys of a user.
 8. set $I_a(m) = 1$ or 0 to indicate whether s_k has been flipped or not, where the length of I_a , i.e., $\text{Len}(I_a) = \mathcal{N}_w$.
-

$s_e(i) = s(2i)$, respectively, where the index of the signal $i = 0, 1, 2, \dots, N - 1$.

2. Predict the odd sets from the even sets, using the prediction error, $e(i) = s(2i - 1) \oplus s(2i)$, to update the current prediction, i.e., $d(i) = s(2i) \oplus s(2i + 1) \oplus e(i) = s(2i - 1) \oplus s(2i + 1)$, where $d(0) = s(1)$.
3. The coarse approximation of the signal is the even sets of samples, $s_e(i)$, which is updated by the prediction error $d(i)$, i.e., $s(i) = s_e(i) \oplus U(d(i)) = s(2i) \oplus d(i)$.
4. Reverse the same prediction and update steps, the original signal can be perfectly reconstructed, i.e., $s(2i) = s(i) \oplus U(d(i)) = s(i) \oplus d(i)$ and $s(2i + 1) = P(s_e(i)) \oplus d(i) = s(2i - 1) \oplus d(i)$, where $s(1) = d(0)$.

The 1-D *LMBWT* is illustrated in Figure 5.15. Let ϖ^p and ψ^p be the operators, where, the superscript “ p ” denotes “prediction and update lifting” for 1-D signal. Using the

Table 5.2: Strategy of Watermark Generation and Embedding Using *IMBWT*

1. **1-level *IMBWT*.** Compute 1-level *IMBWT* of input image Y to obtain set of coefficients C_k , $\{C_k\} = \text{IMBWT}[Y]$, where $C \in \{s, v, h, d\}$ denotes the coarse signal, vertical, horizontal and diagonal detail signals, $k \in \{ee, eo, oe, oo\}$ represents different wavelet transforms.
2. **Embeddability determination.** Determine the embeddability of a coarse signal based on the flippability conditions discussed in Sections 5.3.1 to 5.3.4 for different $\mathcal{O}_p$.
3. **Intermediate image generation.** Generate the intermediate image Y_1 by setting the “embeddable” locations of Y to “0s”. For example, to generate Y_1 for those $\mathcal{D}_b(i, j)$ where $\mathcal{R}_s(m) \wedge \bar{f}_s(i, j) \wedge (\mathcal{O}_p = 2)$ is true (using *DPC*), we have

$$\begin{cases} \text{Set } Y_1(2i+1, 2j+1) = 0 \text{ and } f_s(i, j) = 1 & \text{if } f_{ee}(i, j) \\ \text{Set } Y_1(2i+2, 2j+2) = 0 \text{ and } f_s(i, j) = 1 & \text{elseif } f_{oo}(i, j) \\ \text{goto next } \mathcal{D}_b(i, j) & \text{else} \end{cases} \quad (5.62)$$

where $\bar{f}_s$ denotes complement of f_s , $i = 0, 1, \dots, M-1$ and $j = 0, 1, \dots, N-1$ are the indices of the signal in transform domain. $f_{ee}(i, j)$ and $f_{oo}(i, j)$ are determined by Equations (5.39) and (5.43), respectively. $m = 1, 2, \dots, \mathcal{N}_w$.

4. **Hard authenticator watermark $\mathcal{W}_r$ generation.** $\mathcal{W}_r = E_k(\mathcal{K}_s, \mathcal{W}_s)$, where $\mathcal{W}_s = H(Y_1) \oplus \mathcal{W}_p$ and $H(Y_1) = \text{Rpl}(\text{Hash}(Y_1))$. *Rpl()* is to replicate *Hash*(Y_1) to the length of $\mathcal{W}_p$.
5. **Hard authenticator watermark $\mathcal{W}_r$ embedding.** The m th $\mathcal{W}_r$ is embedded to enforce $\mathcal{W}_r(m) = \mathcal{B}_f(i, j)$ by flipping $s_k(i, j)$ if needed. For example, for those $\mathcal{D}_b(i, j)$ where $f_{ee}(i, j) \wedge \mathcal{R}_s(m) \wedge \bar{f}_s(i, j) \wedge (\mathcal{O}_p = 1)$ is true (using *SPC*), we have

$$\begin{cases} s_{ee}(i, j) = \bar{s}_{ee}(i, j) & \text{if } (\text{mod}(\mathcal{B}_f(i, j), 2) \neq \mathcal{W}_r(m)) \\ s_{ee}(i, j) & \text{else} \end{cases} \quad (5.63)$$

Set $f_s(i, j) = 1$ and set $I_a(m) = 1$ if $s_k(i, j)$ has been flipped.

6. **Post-updating.** When the candidate pixels from the subsidiary 2×2 blocks are flipped using *DPC* or *DPDC*, the diagonal coefficients of the main processing blocks being affected should be updated, e.g., update $d_{ee}(i+1, j+1) = \bar{d}_{ee}(i+1, j+1)$ by flipping $Y(2i+2, 2j+2)$ when the main processing blocks are *EEBs*.
7. **Watermarked image Y_w .** Y_w is obtained by computing inverse *IMBWT* (i.e., *IIMBWT*) for the main processing blocks. Compute *IIMBWT*[C_k] to reconstruct the candidate pixel, e.g., $Y(2i+1, 2j+1)$ for the even-even processing case. Undo flipping $s_k(i, j) = \bar{s}_k(i, j)$ for $I_a(m) = 1$, thereafter reconstruct other pixels in the 2×2 block, e.g., $Y(2i, 2j)$, $Y(2i, 2j+1)$ and $Y(2i+1, 2j)$ for the even-even processing case.

Table 5.3: Strategy of Watermark Extraction and Authentication Using *IMBWT*

1. **Hash value of Y_w .** Perform similar steps as that in the watermark embedding process to identify the “embeddable” coefficients. Clear the “embeddable” locations in Y_w to generate the intermediate image Y_{w1} , which is subsequently fed to a hash function such as *MD5* to generate the $Hash(Y_{w1})$ of Y_w .

2. **Watermark $\mathcal{W}'_p$ extraction.** Extract the m th authenticator watermark by $\mathcal{W}'_r(m) = \mathcal{B}_f(i, j)$ from $\mathcal{D}_b(i, j)$ whose coarse coefficient is “embeddable”. Decrypt $\mathcal{W}'_r$ to obtain

$$\begin{aligned}\mathcal{W}'_s &= D_k(\mathcal{W}'_r, \mathcal{K}_p) \\ &= H'(Y_1) \oplus \mathcal{W}'_p\end{aligned}\quad (5.64)$$

Finally obtain $\mathcal{W}'_p$ by

$$\mathcal{W}'_s \oplus H(Y_{w1}) = H'(Y_1) \oplus \mathcal{W}'_p \oplus H(Y_{w1}) = \mathcal{W}'_p \quad \text{when } H'(Y_1) = H(Y_{w1}) \quad (5.65)$$

where $H(Y_{w1})$ and $H'(Y_1)$ are the replicated $Hash(Y_{w1})$ and retrieved, replicated $Hash(Y_1)$, respectively.

3. **Watermark verification.** Determine the authenticity and integrity of Y_w by computing $\mathcal{R}_d(\mathcal{W}_p, \mathcal{W}'_p)$ using Equation (2.3).

Figure 5.15: Illustration of 1-D *LMBWT*.

index notation, the forward transform for 1-D signal is given by

$$s^p(i) = \psi^{p+}(s)(i) = s(2i) \oplus d^p(i) \quad (5.66)$$

$$d^p(i) = \varpi^{p+}(s)(i) = s(2i-1) \oplus s(2i+1) \quad (5.67)$$

where $d^p(0) = s(1)$. The prediction operation is designed to keep track of the transitions among every three neighboring pixels such as $s(2i-1)$, $s(2i)$ and $s(2i+1)$. The inverse transform for 1-D signal is given by

$$s(2i) = \psi^{p-}(s)(2i) + \varpi^{p-}(s)(2i) = s^p(i) \oplus d^p(i) \quad (5.68)$$

$$s(2i+1) = \psi^{p-}(s)(2i+1) + \varpi^{p-}(s)(2i+1) = s(2i-1) \oplus d^p(i) \quad (5.69)$$

where $s(1) = d^p(0)$. The 1-D *LMBWT* can be classified into even and odd processing cases depending on the co-ordinates of the coarse signal. The *LMBWT* even processing case for 1-D signal is defined such that the coarse signal takes the even indexed sample and is updated by the difference of its two neighboring odd indexed samples. The detail signal is the difference of the two odd indexed samples that surround the even indexed sample. For the odd processing case, left padding one sample before the first sample is required such that Equations (5.66) and (5.67) can be applied to the new data set. The appended sample should be discarded after the processing.

5.4.2 Forward and Inverse *LMBWT* for 2-D Signal

For the convenience of 2-D signal processing, the non-separable 2-D transform is defined. Note that each processing case of *LMBWT* corresponds to the corresponding transform, e.g., the even-even processing case corresponds to even-even transform. The 2-D *LMBWT* even-even processing case is defined such that the coarse signal takes even samples from both row and column directions and is updated by the difference of its horizontal and

vertical detail signals. The horizontal detail signal is the difference of the two odd indexed samples that surround the coarse signal vertically. The vertical detail signal is the difference of the two odd samples that surround the coarse signal horizontally. Whereas the diagonal detail signal is the difference of the four samples that are 8-connected to the coarse signal from its diagonal and anti-diagonal directions. The lifting strategy is based on a 3×3 block, which is shown in Figure 5.16 for the even-even processing case.

Let the operators for the coarse, horizontal, vertical and diagonal detail signals be ψ^{pu} ,

$s(2i-1, 2j-1)$	$s(2i-1, 2j)$	$s(2i-1, 2j+1)$
$s(2i, 2j-1)$	$s(2i, 2j)$	$s(2i, 2j+1)$
$s(2i+1, 2j-1)$	$s(2i+1, 2j)$	$s(2i+1, 2j+1)$

Figure 5.16: Designations of Samples in a 3×3 Block.

ϖ_h^{pu} , ϖ_v^{pu} and ϖ_d^{pu} , respectively. The superscript “ pu ” denotes “prediction and update lifting” and the subscript “ ee ” represents the even-even transform. The signals obtained by applying the analysis operators are given by

$$v_{ee}^{pu}(i, j) = \varpi_v^{pu+}(s)(i, j) = s(2i, 2j-1) \oplus s(2i, 2j+1) \quad (5.70)$$

$$h_{ee}^{pu}(i, j) = \varpi_h^{pu+}(s)(i, j) = s(2i-1, 2j) \oplus s(2i+1, 2j) \quad (5.71)$$

$$d_{ee}^{pu}(i, j) = \varpi_d^{pu+}(s)(i, j) = s(2i-1, 2j-1) \oplus s(2i-1, 2j+1) \oplus s(2i+1, 2j-1) \oplus s(2i+1, 2j+1) \quad (5.72)$$

$$s_{ee}^{pu}(i, j) = \psi^{pu+}(s)(i, j) = s(2i, 2j) \oplus u(i, j) \quad (5.73)$$

where $i = 0, 1, 2, \dots, M - 1$ and $j = 0, 1, 2, \dots, N - 1$, are the indices of the 2-D coarse signal of size $M \times N$ and $u(i, j) = v_{ee}^{pu}(i, j) \oplus h_{ee}^{pu}(i, j)$. The boundary conditions for Equations (5.70)-(5.73) are given by

$$\left\{ \begin{array}{ll} v_{ee}^{pu}(i, 0) = s(2i, 1) & \text{when } j = 0 \\ d_{ee}^{pu}(i, 0) = s(2i - 1, 1) \oplus s(2i + 1, 1) & \\ h_{ee}^{pu}(0, j) = s(1, 2j) & \text{when } i = 0 \\ d_{ee}^{pu}(0, j) = s(1, 2j - 1) \oplus s(1, 2j + 1) & \\ v_{ee}^{pu}(0, 0) = s(0, 1) & \text{when } i = 0 \text{ and } j = 0 \\ h_{ee}^{pu}(0, 0) = s(1, 0) & \\ d_{ee}^{pu}(0, 0) = s(1, 1) & \end{array} \right. \quad (5.74)$$

The signals at level l are reconstructed by applying the synthesis operators and given by

$$s(2i, 2j) = \psi^{pu-}(s)(2i, 2j) \oplus \varpi_v^{pu-}(s)(2i, 2j) = s_{ee}^{pu}(i, j) \oplus u(i, j) \quad (5.75)$$

$$s(2i, 2j + 1) = \psi^{pu-}(s)(2i, 2j + 1) \oplus \varpi_v^{pu-}(s)(2i, 2j + 1) = s(2i, 2j - 1) \oplus v_{ee}^{pu}(i, j) \quad (5.76)$$

$$s(2i + 1, 2j) = \psi^{pu-}(s)(2i + 1, 2j) \oplus \varpi_h^{pu-}(s)(2i + 1, 2j) = s(2i - 1, 2j) \oplus h_{ee}^{pu}(i, j) \quad (5.77)$$

$$\begin{aligned} s(2i + 1, 2j + 1) &= \psi^{pu-}(s)(2i + 1, 2j + 1) \oplus \varpi_d^{pu-}(s)(2i + 1, 2j + 1) \\ &= s(2i - 1, 2j - 1) \oplus s(2i - 1, 2j + 1) \oplus s(2i + 1, 2j - 1) \oplus d_{ee}^{pu}(i, j) \end{aligned} \quad (5.78)$$

where $u(i, j) = h_{ee}^{pu}(i, j) \oplus v_{ee}^{pu}(i, j)$. The boundary conditions for Equations (5.75)-(5.78) are given by

$$\left\{ \begin{array}{ll} s(2i, 1) = v_{ee}^{pu}(i, 0) & \text{when } j = 0 \\ s(2i + 1, 1) = s(2i - 1, 1) \oplus d_{ee}^{pu}(i, 0) & \\ s(1, 2j) = h_{ee}^{pu}(0, j) & \text{when } i = 0 \\ s(1, 2j + 1) = s(1, 2j - 1) \oplus d_{ee}^{pu}(0, j) & \\ s(0, 1) = v_{ee}^{pu}(0, 0) & \text{when } i = j = 0 \\ s(1, 0) = h_{ee}^{pu}(0, 0) & \\ s(1, 1) = d_{ee}^{pu}(0, 0) & \end{array} \right.$$

In the similar way, other processing cases of the *LMBWT* can be defined. Before applying the transform, one row, one column and both one row and one column data should be inserted before the first row, first column and both the first row and first column for odd-even, even-odd and odd-odd processing cases, respectively. The inserted data should be discarded after processing. s_k^{pu} , h_k^{pu} , v_k^{pu} and d_k^{pu} are used to represent the coefficients obtained for different processing cases, where $k \in \{ee, eo, oe, oo\}$. When *LMBWT* is applied to data hiding, updating the coarse signal is not necessary so that the watermark can be embedded in the coarse signal directly. The horizontal and vertical detail signals obtained from Equations (5.70) and (5.71) can be employed to determine the “flippability” of the coarse signal obtained in Equation (5.73) without being updated (i.e., no $u(i, j)$ item). For instance, determine $s_{ee}^{pu}(i, j)$ as “flippable” if $h_{ee}^{pu}(i, j) \wedge v_{ee}^{pu}(i, j) = 1$. It is worthwhile noticing that the images are still processed in 2×2 blocks despite the fact that *LMBWT* is defined based on 3×3 blocks. There are four processing cases, even-even, even-odd, odd-even and odd-odd using *LMBWT* depends on the coarse signal to be even-even, even-odd, odd-even and odd-odd, respectively.

The redundancy in *IMBWT* when being applied for data hiding for *SPC*, *DPC* and *DPDC* processing cases can be eliminated by using *LMBWT*, they differ and relate to each other in the following aspects.

1) **Number of transforms.** Only one transform needs to be computed for a single processing case using *LMBWT*, i.e., $\{s_k^{pu}, h_k^{pu}, v_k^{pu}, d_k^{pu}\} = \text{LMBWT}[Y]$ for an input image Y . Whereas at most four transforms are needed using *IMBWT*.

2) **Flippability condition.** For the (i, j) th coarse signal, the flippability conditions for the even-even and odd-odd processing cases of *LMBWT* are given by

$$P_{ee}(i, j) = v_{ee}^{pu}(i, j) \wedge h_{ee}^{pu}(i, j) \quad (5.79)$$

$$P_{oo}(i, j) = v_{oo}^{pu}(i, j) \wedge h_{oo}^{pu}(i, j) \quad (5.80)$$

In terms of pixel values, $P_{ee}(i, j)$ and $P_{oo}(i, j)$ can be expressed by

$$P_{ee}(i, j) = (Y(2i, 2j - 1) \oplus Y(2i, 2j + 1)) \wedge (Y(2i - 1, 2j) \oplus Y(2i + 1, 2j)) \quad (5.81)$$

$$P_{oo}(i, j) = (Y(2i, 2j + 1) \oplus Y(2i + 2, 2j + 1)) \wedge (Y(2i + 1, 2j) \oplus Y(2i + 1, 2j + 2)) \quad (5.82)$$

It is noticed that Equations (5.81) and (5.82) are equivalent to Equations (5.47) and (5.46) for the odd-odd and even-even processing cases of *IMBWT*, respectively.

3) **Watermark carrier coefficients.** In order to reconstruct the other three pixels in a main processing 2×2 block except the candidate pixel for *IMBWT*, the s_k that has been flipped needs to be flipped back, as shown in the dashed rectangle in Figure 5.13. While such a process is not needed using *LMBWT*. The m th watermark can be embedded by $s_k^{pu'}(i, j) = \bar{s}_k^{pu}(i, j)$ when $\mathcal{B}_f(i, j) \neq \mathcal{W}_r(m)$; Otherwise, $s_k^{pu'}(i, j) = s_k^{pu}(i, j)$, where $\bar{s}_k^{pu}(i, j)$ is the complement of $s_k^{pu}(i, j)$. The watermarked image Y_w is obtained by computing the inverse transform by $Y_w = ILMBWT[s_k^{pu'}, v_k^{pu}, h_k^{pu}, d_k^{pu}]$.

4) **Double processing cases.** When *DPC* or *DPDC* is chosen and the “embeddable” locations are in the subsidiary blocks, the diagonal detail coefficients of the main processing blocks being affected by the flipping need to be updated. For instance, if $s_{oo}^{pu}(i, j)$ (i.e., $Y(2i + 1, 2j + 1)$) is flipped, the diagonal detail signals of the four main processing blocks, i.e., the *OOBs* (whose candidate pixels are in light gray in Figure 5.10), need to be updated through $d_{ee}(a, b) = \bar{d}_{ee}(a, b)$, where $(a, b) \in \{(i, j), (i, j + 1), (i + 1, j), (i + 1, j + 1)\}$. However, only one diagonal detail signal of the *OOB* that includes the pixel being flipped needs to be updated using *IMBWT*.

5) **Computational complexity.** The major computations and buffer size required using *IMBWT* and *LMBWT* for an image of size $M \times N$ are shown in Table 5.4, which reveals that significant savings in buffer size can be achieved by employing *LMBWT*.

Table 5.4: Computational Complexity of Using *IMBWT* and *LMBWT*

	Processing Cases								Total Comp.	Buffer Size
	even-even		even-odd		odd-even		odd-odd			
	Co	Num	Co	Num	Co	Num	Co	Num		
<i>SPC</i> (<i>IMBWT</i>)	s,v, d,h	4 Agt 5 Xor	v	1 Agt 1 Xor	h	1 Agt 1 Xor			3/2 <i>S</i> Agt 7/4 <i>S</i> Xor	3/2 <i>S</i>
<i>CSCF</i> (<i>IMBWT</i>)	s,v, d,h	4 Agt 5 Xor	v,d	2 Agt 4 Xor	h,d	2 Agt 4 Xor	d	1 Agt 3 Xor	9/4 <i>S</i> Agt 4 <i>S</i> Xor	9/4 <i>S</i>
<i>DPC/DPDC</i> (<i>IMBWT</i>)	s,v, d,h	4 Agt 5 Xor	h,v	2 Agt 2 Xor	h,v	2 Agt 2 Xor	h,v	2 Agt 2 Xor	5/2 <i>S</i> Agt 11/4 <i>S</i> Xor	5/2 <i>S</i>
<i>SPC</i> (<i>LMBWT</i>)	s,v, d,h	4 Agt 5 Xor							<i>S</i> Agt 5/4 <i>S</i> Xor	<i>S</i>
<i>DPC/DPDC</i> (<i>LMBWT</i>)	s,v, d,h	4 Agt 5 Xor					h,v	2 Agt 2 Xor	3/2 <i>S</i> Agt 7/4 <i>S</i> Xor	3/2 <i>S</i>

Agt: Assignment; Co: Coefficients; Comp.: Computations; $S = M \times N$.

5.5 Experimental Results

5.5.1 Coefficients Comparison

Comparisons of the coefficients obtained by computing *LMBWT* and that of the two single transforms of *IMBWT* such as even-even and odd-odd transforms are made. In order to show the watermark carrier coefficients and to be comparable with other transforms, no updating of the coarse signal of *LMBWT* is performed. The results are shown in Figure 5.17. Obviously, the coefficients obtained from computing a single transform such as even-even or odd-odd transform of *IMBWT* lose some transition information. While most transitions in binary images can be tracked by the coefficients obtained from computing *LMBWT*, which is equivalent to *IMBWT*. The entropy-based cost function [98] is employed to measure the compactness of the signal representation, e.g., coefficients shown in Figure 5.17, which is given by

$$\mathcal{H}(r_o) = -r_o \log_2(r_o) - (1 - r_o) \log_2(1 - r_o) \quad (5.83)$$

Figure 5.17: Coefficients Comparison. (a) the original image. (b)-(m) the coarse signal and horizontal, vertical and diagonal detail signals obtained from: (b)-(e) *IMBWT* even-even (*EE*) transform, (f)-(i) *IMBWT* odd-odd (*OO*) transform and (j)-(m) *LMBWT* even-even processing case.

where r_o is the ratio of the number of foreground pixels, e.g., black pixels, versus the total number of pixels in the image. It can be easily seen from the results shown in Table 5.5,

Table 5.5: Entropy of Wavelet Coefficients

	Coarse signal	Vertical detail	Horizontal detail	Diagonal detail
<i>IMBWT(EE)</i>	0.4170	0.0264	0.0284	0.0066
<i>IMBWT(OO)</i>	0.4335	0.0260	0.0183	0.0098
<i>LMBWT</i>	0.4275	0.0612	0.0572	0.0246

compared with the entropy value 0.9238 of the original image, the small entropy values are obtained for the coefficients of a single transform of *IMBWT* and *LMBWT*. This indicates the large discrepancy between the number of foreground and background pixels. Hence a more compact signal representation can be achieved. Moreover, the entropy value of the detail coefficients obtained from *LMBWT* (i.e., *IMBWT*) is larger than that obtained from computing a single transform of *IMBWT*. This demonstrates that more transition

information can be tracked using the interlaced transform or *LMBWT*.

5.5.2 Visual Quality and Capacity

Extensive experiments are conducted to study the capacity and visual quality of the watermarked image obtained from the proposed scheme. The data hiding results for different processing cases shown in Figure 5.18 reveal the good visual qualities of the watermarked images. From *CSCF*, *SPC* to *DPC* or *DPDC*, the capacity has increased significantly at the cost of slightly compromising the visual quality of the images.

Comparisons of the capacity increase in terms of percentage for different processing cases are shown in Figure 5.19. Forty images of different types (e.g., cartoon, handwritten and text in different languages) and resolutions (e.g., 300dpi, 200dpi and 150dpi) are used in the experiments. The image sizes vary from 100×100 to 512×512 . The results agree with our observations that from *CSCF*, *SPC* to *DPC* or *DPDC*, the capacity has increased significantly. A capacity increase from 56% to 90% is achieved using *DPC* or *DPDC* versus *SPC*. The lowest increase (e.g., 56%) is resulted from a line drawing cartoon image which only contains thin lines. The capacity increase from 5% to 40% is achieved using *SPC* versus *CSCF*. Applying Equations (2.1) and (2.2) to evaluate the visual quality of the watermarked images obtained by employing *CSCF*, *SPC*, *DPC* and *DPDC* for the images whose capacities are shown in Figure 5.19, D_t and D_a versus different images are plotted in Figure 5.20 and Figure 5.21, respectively. The results conform to our visual observations that from *CSCF* to *SPC*, and *DPDC* or *DPC*, the total distortion increases in exchange for higher capacity. Employing *CSCF* results in the lowest D_a . By properly choosing the lower distortion pixel to flip for *DPDC*, D_a is lower than that achieved by employing *DPC*. While the D_a achieved using *SPC* is similar to that of using *DPDC*. It should be noted that D_a can be high for those images of low resolutions such as 150dpi, which can be seen from the last several images in Figure 5.21. This is due to the reason

Figure 5.18: Hiding Effects of Different Processing Cases. (a) the original image of size 268×268 and 300dpi, (b) hide 529 bits by employing *CSCF*, (c) hide 667 bits by employing *SPC*, (d) hide 1178 bits by employing *DPC* and (e) hide 1178 bits by employing *DPDC* processing cases. (f)-(i) the enlarged difference image.

that there are few good patterns exist in thin strokes of one or two pixels wide. Noticeably, by employing *DPC* or *DPDC*, there may be more than one pixel being flipped in each 3×3 block. In this case, the watermarked image with the center pixel taken from the original image is used for evaluation.

Figure 5.19: Comparisons of Capacity Increase for Different Processing Cases.

Figure 5.20: Comparisons of Total Distortion for Different Processing Cases.

5.5.3 Comparisons

Comparisons of the capacity and visual quality of the watermarked images obtained using our proposed method for different processing cases with that obtained by employing the

Figure 5.21: Comparisons of Average Per Pixel Distortion for Different Processing Cases.

methods proposed by Wu *et al.* [15], Tseng *et al.* [12], Lu *et al.* [63] and *CPDH* method proposed in Chapter 4 are made. The capacity achieved by employing different methods and the corresponding average per pixel distortion are shown in Tables 5.6 and 5.7, respectively. In the tables, the block sizes for other methods are chosen such that larger capacity is obtained under the constraints that the watermarked image is of acceptable visual quality. It can be seen from Table 5.6 that the capacity achieved by employing *DPDC* or *DPC* (i.e., the 4th column) of our proposed method can be about one (i.e., compared with the 5th column) to two (i.e., compared with the 6th column) times larger than that of Wu *et al.*'s method. It is about one (i.e., compared with the 9th column) to three (i.e., compared with the 10th column) times larger than that obtained by employing Lu *et al.*'s method. It is comparable and about one times larger than that of *CPDH* method (i.e., compared with the 11th and 12th columns) by employing *CSCF* (i.e., the 2nd column) or *SPC* (i.e., the 3rd column), and *DPC* or *DPDC* (i.e., the 4th column), respectively. The capacity increases at the cost of slightly degrading the visual quality as shown in Table 5.7. Whereas the capacity achieved by our proposed method using

Table 5.6: Comparisons of Capacity for Different Methods

File	Proposed Method			Wu <i>et al.</i>		Tseng <i>et al.</i>		Lu <i>et al.</i>		Yang & Kot	
	<i>CSCF</i>	<i>SPC</i>	<i>DPC</i> (<i>DPDC</i>)	B1	B2	B2	B3	B4	B1	B5	B6
HT	739	887	1627	1089	784	1446	1099	606	436	972	941
ET1	1006	1170	2040	1089	784	2028	1631	945	588	1237	1153
CT	1054	1341	2429	1089	784	2256	1694	968	588	1261	1159
ET2	529	667	1178	676	484	1404	917	593	360	711	677
ET3	591	698	1141	576	400	1278	938	535	351	683	667
ET4	1303	1684	2940	1225	841	2472	1883	1144	739	1722	1594
ET5	536	622	1094	625	441	1110	798	518	341	671	643
JT	1190	1514	2683	1089	784	2208	1764	983	621	1488	1364
ET6	1155	1419	2476	1056	715	2214	1736	1005	623	1449	1405
ET7	2515	3147	5565	2601	1764	4854	3787	2146	1401	3224	2974

HT: Handwritten Text; CT: Chinese Text; ET: English Text; JT: Japanese Text.

B1: 10×10; B2: 12×12; B3: 16×16; B4: 8×8; B5: IB 4×4; B6: IB 5×5.

Table 5.7: Comparisons of Average Per Pixel Distortion for Different Methods

File	Proposed Method				Wu <i>et al.</i>		Tseng <i>et al.</i>		Lu <i>et al.</i>		Yang&Kot	
	<i>CSCF</i>	<i>SPC</i>	<i>DPC</i>	<i>DPDC</i>	B1	B2	B2	B3	B4	B1	B5	B6
HT	0.45	0.57	0.56	0.55	0.53	0.47	0.82	0.85	0.64	0.64	0.47	0.47
ET1	0.46	0.56	0.59	0.58	0.50	0.44	0.87	0.86	0.67	0.65	0.48	0.47
CT	0.48	0.62	0.63	0.62	0.51	0.42	0.87	0.85	0.74	0.74	0.49	0.49
ET2	0.48	0.60	0.61	0.60	0.55	0.48	0.87	0.89	0.68	0.71	0.52	0.50
ET3	0.52	0.60	0.59	0.58	0.52	0.44	0.88	0.89	0.73	0.72	0.49	0.51
ET4	0.47	0.56	0.59	0.58	0.46	0.44	0.84	0.85	0.70	0.70	0.47	0.49
ET5	0.47	0.57	0.60	0.59	0.53	0.45	0.87	0.86	0.69	0.65	0.48	0.48
JT	0.53	0.62	0.63	0.62	0.48	0.44	0.86	0.85	0.76	0.76	0.51	0.50
ET6	0.51	0.60	0.62	0.60	0.45	0.42	0.86	0.88	0.73	0.73	0.49	0.51
ET7	0.49	0.58	0.60	0.59	0.49	0.43	0.86	0.85	0.70	0.69	0.49	0.49

DPC or *DPDC* (i.e., the 4th column) is comparable to that of Tseng *et al.*'s method (i.e., compared with the 7th and 8th columns) with quality control. However the average per pixel distortion achieved by our proposed method significantly outperforms that of Tseng

et al.'s method. Observed from Table 5.7, the good visual results comparable to that of Wu *et al.*'s method and *CPDH* scheme are achieved by employing *CSCF*. Employing *SPC* and *DPC* or *DPDC* slightly degrades the visual quality of the watermarked images, however, the capacity has increased significantly.

Comparisons of the data hiding results using different methods are made. In order to have a fair comparison, the same number of watermark bits that equals to the capacity obtained from Wu *et al.*'s approach with block size 10×10 is embedded in the image using different approaches. The block sizes chosen in the experiments are 16×16 , 7×7 and 5×5 for Tseng *et al.*, Lu *et al.* and *CPDH* scheme proposed in Chapter 4, respectively. The comparison results are shown in Figure 5.22. It is easily observed that the visual effects

Figure 5.22: Comparison of Data Hiding Results Using Different Methods. (a) the original image of size 245×245 and of 300dpi, (b)-(f) the watermarked image with 576 bits embedded by employing (b) the proposed method, (c) Wu *et al.*'s method, (d) Tseng *et al.*'s method, (e) Lu *et al.*'s method and (f) *CPDH* method.

of the watermarked image obtained using *CSCF* of our proposed method are comparable to that of Wu *et al.*'s method and *CPDH* scheme and are better compared with that of

Tseng *et al.*'s method and Lu *et al.*'s method. Apply Equation (2.2) to evaluate the visual distortion for the watermarked images, D_a is 0.51, 0.50, 0.74, 0.85 and 0.49 for Wu *et al.*'s method, our proposed method, Lu *et al.*'s method, Tseng *et al.*'s method and *CPDH* scheme, respectively, which agrees with our visual observation.

The major computations required for our proposed scheme are in order of $O(M \times N)$ and the basic computation is *Xor* as shown in Table 5.4. With reference to Table 4.5 in Chapter 4, the most time-consuming computation is the comparison required in order to find the best pattern in each block for Wu *et al.*'s method. The lower bound of the comparison is in order of $O(\mathcal{N}_k \times (\mathcal{N}_d \log_2 \mathcal{N}_d))$, where $\mathcal{N}_k$ is the total number of blocks and $\mathcal{N}_d$ is the number of determined pixels in each block. The computational load increases significantly with the increase of the block size. The most time-consuming computations are the calculation of the distance matrix for each pixel and comparisons of the distance matrix for any two pixels in each block for Tseng *et al.*'s method. The multiplication involved in the distance computation is considered time-consuming compared with the logical operations required by our proposed approach for the same addressing mode. The computational load also increases significantly for larger block size for *CPDH* approach. In addition, locating the “embeddable” pixels to incorporate the cryptographic signature in *CPDH* for $\mathcal{B}_z \geq 4$ not only leads to the decrease in capacity but also incurs the increase in computational load.

5.5.4 Watermark Effectiveness and Security Concerns

Experiments are conducted to verify effectiveness of the proposed hard authenticator watermark. A logo image is used as the payload watermark to be *Xored* with the cryptographic signature of the image so as to visually show the tamperings. The authentication results are shown in Figure 5.23. The logo image can be reconstructed successfully when

Figure 5.23: Authentication Results. (a) the original image of size 660×160 , (b) the watermarked image with 1155 bits embedded by employing *CSCF* and (c) the watermarked image that is tampered, in the 2nd line, “the” is changed to “a”. (d) the original logo image of size 50×20 , (e) the reconstructed logo image from the watermarked image, (f) the reconstructed logo image from the tampered image shown in (c).

no tampering occurs to the watermarked image. However, when tampering occurs or a wrong public key is used for verification, the reconstructed logo image is like a random noise.

Security of the proposed authentication scheme relies on the secrecy of the private key, breaking the system is equivalent to breaking the cryptographic algorithm such as RSA. In order to prevent the adversary from finding out the data hiding locations and subsequently embedding his own watermark by analyzing multiple watermarked images generated using the same private key, we suggest that one pair of key only be applied to a single image as discussed in Section 4.5.1 of Chapter 4. On the other hand, for the *CSCF*, *SPC* and *DPDC* processing cases, given a payload $\mathcal{W}_p$, where $\mathcal{W}_p \leq \mathcal{C}_{\zeta_i}$, where $\mathcal{C}_{\zeta_i}, i = 0, 1, 2$ denotes the maximum capacity for *CSCF*, *SPC* and *DPDC*, respectively. The “embeddable” locations that are actually chosen to hide the data depend on a random key. The probability that an adversary is able to figure out the data hiding locations is

$\mathcal{P}_f = \frac{1}{\binom{c_{\zeta_i}}{\mathcal{W}_p} \times \mathcal{W}_p!} = \frac{1}{c_{\zeta_i}(c_{\zeta_i}-1)\dots(c_{\zeta_i}-\mathcal{W}_p+1)}$. Generally we can assume that $\mathcal{C}_{\zeta_i}$ is a constant value for a given image. It is easily seen that the larger the $\mathcal{W}_p$, the lower the probability that an adversary can figure out the data hiding locations. By employing *DPC*, the choice and processing sequence of the two processing cases is determined by a random key. The probability that an attacker is able to figure out the data hiding locations is $\mathcal{P}_f = \frac{1}{\binom{c_{\zeta_1}}{\mathcal{W}_p} \times \mathcal{W}_p! \times \binom{2}{1}} = \frac{1}{2c_{\zeta_1}(c_{\zeta_1}-1)\dots(c_{\zeta_1}-\mathcal{W}_p+1)}$, which is about 1/2 that of *SPC*, assuming the two processing cases have similar maximum capacity. Hence, it is preferable to choose *DPC* from security point of view. In order to tackle the sensitivity of the proposed scheme to random noise, Error Correction Coding can be applied to each segment of the watermark bits, which sacrifices the capacity to gain the robustness.

5.6 Flippability Decision in Larger Block

Most data hiding methods determine the “flippability” of a pixel by considering 3×3 neighborhood. The denoise patterns in 5×5 neighborhood are employed to find the “flippable” locations in our earlier work [49, 53]. However, requiring the presence of a location map for the watermark retrieval definitely limits its application. Pairs of patterns that are of five pixel long contour segments are employed for the watermark embedding and retrieval in [16]. It is preferable to extend the flippability decision block from 3×3 to 4×4 or 5×5 to maintain the smoothness of the contour, which is explored in this Section. Specifically, the “flippability” condition is initially determined in 3×3 blocks by checking the relationship of the vertical and horizontal detail coefficients of *IMBWT*, e.g., using Equation (5.39). Patterns that satisfy this condition is collectively named as “Connectivity-Preserving (*CP*)” patterns due to the 4-connectivity among pixels is preserved. Extending the “Dominant Edge” block defined in Section 5.3.2, we define the “Dominant Edge” as the main edge in a 3×3 block by satisfying the “flippability” condition defined in Equations (5.39), (5.43), (5.44) and (5.45). Once the Dominant Edge

(Figure 5.24(a)) is determined, further checking direction is determined by following the edge counter clockwise (Figure 5.24(b)) or clockwise (Figure 5.24(c)), starting from the two edge pixels of the Dominant Edge.

Figure 5.24: Example of Dominant Edge and Follow Up Edges in 3×3 Blocks. Pixels in grid are “don’t care” pixels. The arrows show the Dominant Edge directions.

Generally, the information data can be embedded in binary images by flipping a single pixel [13–15, 49, 52, 53], namely “Single Flipping”. A pair of pixels, e.g., P_{mf} and P_{pf} , can be flipped to embed the information data by changing the starting point of the edge transition [16] for the “Transition Change (TC)” patterns, e.g., Figure 5.25(b), namely “Pair Flipping”. This is especially suitable for images of low resolutions such as images of 200dpi and below, in which the stroke is only one or two pixels wide. The CP patterns that can be paired with TC patterns, e.g., Figure 5.25(a), and the CP patterns that cannot be paired with TC patterns, e.g., Figure 5.25(c), (d) and (e), are determined based on the “flippability” condition in 4×4 or 5×5 blocks. To preserve the visual quality of the

Figure 5.25: Illustration of “Pair Flipping” (a)(b) and “Single Flipping” (c)(d)(e). The 3×3 blocks in dashed green square are the initial consideration blocks.

image, the “Triangle Noise” patterns shown in Figure 5.26 are to be excluded from being flipped for the “L” patterns in Figure 5.25(e).

Figure 5.26: Example of “Triangle Noise” Patterns.

One orthogonal set of patterns to the CP and TC patterns are the “End Stroke (ES)” patterns shown in Figure 5.27, which can be identified by checking in the 4×4 or 5×5 block based on the directions of the dominant edges. The locations identified by employing the ES patterns are different from those identified by employing the CP or TC patterns. The orthogonality of the ES patterns to the CP and TC patterns can be employed to design a double data hiding scheme such that different information can be embedded one by one by employing the two types of patterns.

Figure 5.27: Example of “End Stroke” Patterns.

Proper post-checking constraints can be designed to ensure that the “embeddability” of a pixel is invariant in the data embedding process, thus facilitates blind watermark extraction and incorporation of cryptographic signature. This can be done by ensuring the invariance of the flippability conditions of those processed neighboring candidate pixels. The CP and TC patterns can be properly paired and classified in such a way that each category is isolated from other categories, hence no ambiguity or collusion exists in the data extraction.

5.7 Applications

The proposed authentication system can be applied to digital document authentication such as digital certificate (eCertificate) authentication. The parties involved in the eCertificate authentication process are: the Issuing Authority such as the universities, the Third Party such as the companies and the Certificate Holder such as the students. The Issuing Authority has an interface to communicate with the Certificate Holder, an interface to communicate with the Document Repository and an interface to communicate with the Third Party. All communications are carried out through secure transmission channel such as Secure Sockets Layer/Transport Layer Security protocols. The Certificate Holder or Third Party who intends to use the system must pre-register with the Issuing Authority by providing all necessary documents to identify themselves. The authentication processes shown in Figures 4.11 and 5.13 can be employed to perform the watermark embedding. A soft copy of the authenticated eCertificate will be issued to the Certificate Holder for verification and submission purposes upon the embedding process finishes.

Consider a scenario that a student is applying for a job or in the situations that require checking his educational background. He submits his digital watermarked eCertificates to the authorized expert by any valid way, e.g., through email, removable hard disk, CDs or floppy disk. If the expert, i.e., the Third Party wants to verify the authenticity of the eCertificate, he needs to log on the Issuing Authority's Website by providing the authentication parameters such as user name and password. The verification consists of two aspects. First, verify the originality of the eCertificate to ensure that it really comes from the claimed university with the named applicant. Second, verify the integrity of the eCertificate to ensure that it has not been tampered since originated. The authenticity of the eCertificate is checked by providing the corresponding public key which can be downloaded from the Issuing Authority's Website. Only if the eCertificate is really issued

by the claimed university and has not been tampered with, will the verification succeed. The watermark extraction and authentication processes shown in Figures 4.12 and 5.14 can be utilized for this purpose.

It is noticed that security of the proposed system relies on the secrecy of the private key. The key issue is how to deal with the situation when the private keys are compromised. Two solutions could be proposed. The first solution is to revoke the eCertificates periodically, as what is done in Public Key Infrastructure systems, e.g., the Certificate Authority revokes the certificate after certain periods and publishes it in the Certificate Revocation List. Once an eCertificate is revoked, the Certificate Holder must request the university to issue a new one. In this way, the key leakage problem can be deterred. The second solution is to decrease the number of eCertificates for which a key pair is applied. Instead of applying one key pair for a large amount of eCertificates, we could only apply it to each individual eCertificate. In this case, compromising certain key pairs will only affect the particular eCertificate. Of course, in doing so, the complexity of key distribution will be increased.

5.8 Relationships of *CPDH* and *MTDH*

The “Connectivity-Preserving” pattern-based data hiding (*CPDH*) presented in Chapter 4 and the morphological transform-based data hiding (*MTDH*) presented in this Chapter are closely related to each other in terms of basic ideas and flippability decision conditions. They differ in terms of processing units and efficiency of utilization of the “flippable” pixels.

5.7.1 Basic Ideas and Flippability Decision

The basic ideas of both schemes are based on the observations that if flipping a pixel in binary images does not destroy the connectivity between pixels, the visual quality of the resultant watermarked image is preserved. This idea in *CPDH* is implemented by defining a “Connectivity-Preserving” criterion. Whereas in *MTDH*, it is implemented by the “Cross” condition or the flippability condition for *CSCF*. To maintain a better visual quality of the resultant watermarked image, at least 4-connectivity and preferably 8-connectivity among pixels need to be preserved. In both schemes, a higher priority is given to the 4-connectivity among pixels. The “flippability” conditions for both schemes consist of three separable parts: 1) *4-Connectivity* condition, e.g., *VH* transition in *CPDH* or the “cross” condition to track the shifted edges in *MTDH*; 2) *8-Connectivity* condition, e.g., *C* transition in *CPDH* or the flippability condition of *CSCF* in *MTDH* and 3) *Corner* condition, e.g., *IR* transition in *CPDH*. The three conditions can be enforced separately in implementation. Possible combinations of the conditions can be: 1) *VH*; 2) *VH* and *C* and 3) *VH*, *C* and *IR*. From 1) to 3), a better visual quality of the resultant watermarked image can be expected but at the cost of decreasing the capacity. Typically, how to choose the “flippability” condition is application dependent. As a rule of thumb, for text images of high resolution such as English text images of 300dpi and above, imposing *4-connectivity* condition is sufficient to control the visual quality such that a larger capacity can be obtained. On the other hand, for text images that have ample sharp corners such as text images in Chinese, imposing *8-connectivity* condition is necessary which unavoidably decreases the total capacity. Similarly, sufficient thin strokes exist for text images in low resolution such as 200dpi and below. In this case, flipping a pixel may create isolated corner pixels in 3×3 blocks, therefore, the *corner* condition should be imposed.

5.7.2. Processing Units and Efficiency

The basic processing unit in *CPDH* is different from that of *MTDH*. In *CPDH*, we consider the whole block as a processing unit, where the block size $\mathcal{B}_z = 3, 4, 5, \dots, N$. In general, N does not go too large by taking the computational load into consideration, e.g., $N \leq 10$. Whereas in *MTDH*, we always consider the 2×2 blocks as the processing units, however, the “flippability” decision is based on 3×3 blocks. Processing an image in 2×2 blocks, not only prevents the computational load from getting too high as that in *CPDH* for large $\mathcal{B}_z$, but also increases the efficiency of utilization of the “flippable” pixels, denoted as η_e . Noticeably, the “un-embeddability” of the sharing rows or columns for the interlaced block scheme or the block boundaries for the non-interlaced block scheme in *CPDH* decreases η_e , namely “Block Boundary” effect. In addition, a large $\mathcal{B}_z$ also decreases η_e in that only one among many “flippable” pixels will be chosen. Generally, $\eta_e = 1/(n-1)^2$ and $\eta_e = 1/n^2$ for an interlaced or non-interlaced block of size $\mathcal{B}_z = n \times n$. As shown in Figure 5.28, η_e becomes lower with the increase of $\mathcal{B}_z$. Whereas $\eta_e = 1/4$ for each $\mathcal{D}_b(m, n)$ in *MTDH*,

Figure 5.28: Efficiency of Utilization of “Flippable” Pixels for *CPDH*. The numbers in x-axis represent the block sizes, n represents block size $(n+2) \times (n+2)$.

which is equivalent to that of the interlaced 3×3 block scheme in *CPDH*. However, two

candidate pixels can be chosen for both *DPC* and *DPDC* comparing only one can be chosen for the interlaced 3×3 block scheme in *CPDH*, which significantly increases the probability of a block to be “embeddable”. In addition, setting the criterion to locate the “embeddable” pixels to incorporate the cryptographic signature in *CPDH* approach not only cause the decrease in capacity but also leads to the increase in computational load for $B_z \geq 4$, especially for large B_z .

5.9 Chapter Summary

In this Chapter, we propose an interlaced morphological binary wavelet transform for the purpose of binary images data hiding and authentication. The relationship of the coefficients is utilized to identify the data hiding locations in morphological transform domain. The redundancy in the interlaced transform can be eliminated by the lifting scheme. We argue that flipping an edge pixel is equivalent to shifting the edge location one pixel position horizontally and vertically. The flippability is designed in such a way that the shifted edges can be tracked, thus facilitates blind watermark extraction and incorporation of cryptographic signature of the image. The two processing cases that flipping the candidate pixels in one does not affect the flippability of another are combined such that not only the capacity has significantly increased but also the visual distortion can be minimized. Comparisons with other methods proposed in literature show the superiority of the proposed scheme in being able to achieve large capacity with acceptable visual distortion while maintaining low computational cost. Relationship of the proposed *MTDH* scheme with *CPDH* scheme, and how to extend the flippability decision block to a larger block are discussed. Application of the proposed authentication system is presented. Experimental results enforce our claims.

Chapter 6

Authentication With Tampering Localization

In this Chapter, we propose a two-layer binary image authentication with tampering localization scheme, in which the first layer is targeted at the overall authentication and the second layer is targeted at identifying the tampering locations. Specifically, the “flippability” of a pixel is determined by the “Connectivity-Preserving” criterion discussed in Chapter 4 or the “flippability” conditions for different processing cases discussed in Chapter 5. The tampering localization scheme is built on top of the authentication schemes presented in Chapter 4 and Chapter 5. Conveniently, any block-based data hiding scheme whose “flippability” condition is invariant in the data embedding process can be employed. The image is partitioned into macro-blocks that are subsequently classified into eight categories. The block identifier is defined adaptively for each class and embedded in those “Qualified” and “Self-Detecting” macro-blocks in order to identify the tampered locations. The block identifier is defined in such a way that any changes occurred either to the “Qualified” or its neighboring “Un-Qualified” macro-blocks will render the retrieved block identifier different from the one embedded. The overall authenticity and integrity are ensured by embedding the cryptographic signature of the image.

6.1 General Overview

Authentication and tampering detection for grayscale and color images are discussed in [2, 89, 103–106], the image *ID* and block index are used as part of the input to the hash function to counter against the collage attack [89]. A hierarchical scheme for localizing the tampering is proposed in [103]. Discussions to counter against cutting and pasting, transplantation and advanced birthday attacks are described in [104]. These schemes are designed for grayscale and color images, in which the capacity of the image is rather high and most gray or color values can be utilized to hide the cryptographic signature. The method attempting to detect the traces of digital tampering in the absence of any form of digital watermark is discussed in [107], in which the statistical correlations of resampling are employed for tampering detection. Detecting the image photomontage using bi-coherence features is presented in [108] and method that differentiates photorealistic from photographic images using the first four order statistics of the subband coefficients in wavelet domain is proposed in [109].

To the best of our knowledge, very few papers reported in literature address the problem of tampering localization for binary images. In addition, no image forensics tools designed to detect forgeries in binary images in the absence of watermarks are available for now, due to the facts that the backgrounds of the binary images are in uniform white color, background inconsistency resulting from tampering is not valid for binary images. Tampering the contents of a text document can only be identified from semantic meaning, which poses challenges to the tampering detection and localization problem. The method that attempts to identify the tampered locations for binary images [17] performs watermarking on each sub-image independently by employing the secret key authentication scheme in [50]. Each sub-image is shuffled and divided into three regions, in which the first layer digital signature (*DS*) is computed on region *C* and inserted into

region B . The second layer DS is computed on regions B and C and inserted into region A . It is noticed that the uniform white or black area that does not participate in the watermarking process is easily tampered.

The uneven embedding capacity of binary images makes locating the tampering a difficult task, especially for those regions that have few or no “flippable” pixels. In this Chapter, we propose a two-layer authentication technique for binary images. The overall authentication is achieved in the first layer by embedding the cryptographic signature (CS) of the image. Localization of tampering is achieved in the second layer by embedding block identifier (BI) in the “Qualified” or “Self-Detecting” macro-blocks. Specifically, we group multiple overlapping 3×3 blocks to form a macro-block (MB) and classify the macro-blocks into “Qualified” macro-blocks ($QMBs$) and “Un-Qualified” macro-blocks ($UMBs$) based on the number of flippable pixels. The $QMBs$ are chained and the block identifier (BI) that is used to identify the tamperings occurred both to the QMB and its neighboring $UMBs$ is embedded. This Chapter is organized as follows. The “flippability” criterion and block signature generation are described in Section 6.2. Tampering detection and localization are discussed in Section 6.3. Experimental results and discussions are presented in Section 6.4 and Section 6.5 summarizes the Chapter.

6.2 Flippability Decision and Block Signature Generation

To design a tampering localization scheme for binary images, we require that the “flippability” of a pixel should be invariant in the data embedding process such that the “flippable” pixels in each macro-block can be determined both in the original and watermarked images. The “flippability” of a pixel can be determined by the “Connectivity-Preserving” criterion in $CPDH$ scheme discussed in Chapter 4 or the “flippability” condition for different processing cases in $MTDH$ scheme discussed in Chapter 5. In general, any block-based

data embedding algorithms that satisfy the invariance of the “flippability” of a pixel can be employed in the proposed tampering localization scheme.

Let us denote the block signature for a macro-block as $\mathcal{B}_s$. In generating the $\mathcal{B}_s$ for a macro-block, the macro-block is divided into 9 regions and each region consists of multiple finer-blocks (*FBs*) as shown in Figure 6.1. Each finer-block is of size $m \times n$ and centered

Figure 6.1: Macro-Blocks and Finer-Blocks.

at $f(x, y)$, where $m = 2a + 1$, $n = 2b + 1$, $a, b \in \mathbb{Z}$ and $\mathbb{Z}$ is nonnegative integer set. The feature code of a finer block, denoted as $f_c(x, y)$, is given by

$$f_c(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b g(s, t) \otimes f(x + s, y + t) \quad (6.1)$$

where the weights $g(s, t) = 1, 2, 4, \dots, 2^{r-1}$, $r \in \mathbb{Z}$, depend on the locations of the pixels in the finer-block and “ $\otimes$ ” represents element-wise multiply. An example of the weights assignment and the pixels of the image for a finer-block of size 3×3 is shown in Figure 6.2.

To generate the feature code for each finer-block, the “flippable” locations are to be set to fixed values, e.g., 0s, which is analogous to clearing the least significant bits in order to compute the hash value for grayscale images [89]. The mean feature code of the

Figure 6.2: Weights Assignment (a) and Pixels of the Image (b) for a Finer-Block of Size 3×3 .

(k, l) th region, which is denoted as $\bar{f}_c(k, l)$, is given by

$$\bar{f}_c(k, l) = \frac{1}{u \times v} \sum_{x=1}^u \sum_{y=1}^v f_c(x, y) \quad (6.2)$$

where u and v denote the number of the finer-blocks in each region of the macro-block in x and y directions, respectively. The $\mathcal{B}_s$ for region (k, l) of the macro-block is obtained below by mapping the $\bar{f}_c(k, l)$ to a binary bit via a look up table (LUT).

$$\mathcal{B}_s(k, l) = LUT(\bar{f}_c(k, l)) \quad (6.3)$$

In the proposed scheme, the finer-block size is chosen to be 3×3 , i.e., the overlapping 3×3 block is used, in which the patterns of the candidate pixels look like the chess board pattern.

6.3 Authentication and Tampering Localization

6.3.1 Macro-Block Classification

The “flippability” of the candidate pixel in each overlapping 3×3 block is determined by the “Connectivity-Preserving” criterion discussed in Chapter 4 or the “flippability” conditions for different processing cases discussed in Chapter 5. Thereafter, the macro-blocks are classified into eight categories based on the number of “flippable” pixels $\mathcal{N}_f$ in each macro-block as shown in Table 6.1.

Table 6.1: Macro-Blocks Classification

CLASS	CONDITIONS
C_0	the MB is all white
C_1	the MB is all black
C_2	$(\text{MB} \notin \{C_0, C_1\}) \wedge (\mathcal{N}_f = 0)$
C_3	$(\mathcal{N}_f \geq 1) \wedge (\mathcal{N}_f < T_0)$
C_4	$(\mathcal{N}_f \geq T_0) \wedge (\mathcal{N}_f < T_1)$
C_5	$(\mathcal{N}_f \geq T_1) \wedge (\mathcal{N}_f < T_2)$
C_6	$(\mathcal{N}_f \geq T_2) \wedge (\mathcal{N}_f < T_3)$
C_7	$\mathcal{N}_f \geq T_3$

The reason why $\mathcal{N}_f$ is chosen as the criterion to classify the macro-blocks lies in the fact that once a location is tampered, the “flippability” condition will most probably change and $\mathcal{N}_f$ will change. Hence, the class that the macro-block belongs to will change as well. The sensitivity of the “flippability” condition to changes is utilized to classify the macro-blocks. From the table, it is not difficult to see that the class type varies with the changes of the number of “flippable” pixels in each macro-block.

6.3.2 Distance Computation

The $MBs \in \{C_4, C_5, C_6, C_7\}$ are named as the “Qualified” macro-blocks ($QMBs$), which will participate in the block chaining process. In computing the distance between two consecutive $QMBs$, we define the horizontal distance and vertical distance as the distance from the previous QMB to the current QMB in the horizontal and vertical directions, which are denoted by $\mathcal{D}_x$ and $\mathcal{D}_y$, respectively. Let $\mathcal{I}_c$, $\mathcal{J}_c$ and $\mathcal{I}_p$, $\mathcal{J}_p$ represent the indices of the current and its previous $QMBs$ for the horizontal and vertical directions. The sign $\mathcal{S}_x$ for $\mathcal{D}_x$ is defined as the index difference of the current and its previous $QMBs$ in the horizontal direction, which is given by

$$\begin{cases} \mathcal{S}_x = 0 & \text{when } \mathcal{I}_c - \mathcal{I}_p < 0 \\ \mathcal{S}_x = 1 & \text{otherwise} \end{cases} \quad (6.4)$$

While $\mathcal{D}_y$ is calculated by $\mathcal{D}_y = \mathcal{J}_c - \mathcal{J}_p$ and $\mathcal{D}_x$ is calculated by

$$\begin{cases} \mathcal{D}_x = \mathcal{N}_x - 1 - (\mathcal{I}_p - \mathcal{I}_c) & \text{when } (\mathcal{S}_x = 0) \wedge (|\mathcal{I}_c - \mathcal{I}_p| > \lfloor \mathcal{N}_x/2 \rfloor) \\ \mathcal{D}_x = |\mathcal{I}_c - \mathcal{I}_p| & \text{otherwise} \end{cases} \quad (6.5)$$

where $\mathcal{N}_x$ is the number of the macro-blocks in the horizontal direction and $|\mathcal{I}_c - \mathcal{I}_p|$ is the absolute value of $\mathcal{I}_c - \mathcal{I}_p$. For the first QMB , we have $\mathcal{D}_x = \mathcal{I}_c$ and $\mathcal{D}_y = \mathcal{J}_c$. In processing the image in raster scan order, $\mathcal{D}_y$ is generally small, however, $\mathcal{D}_x$ is relatively large. When $(\mathcal{S}_x = 0) \wedge (|\mathcal{I}_c - \mathcal{I}_p| > \lfloor \mathcal{N}_x/2 \rfloor)$, $\mathcal{D}_x$ is the effective index difference of the current and its previous $QMBs$ in the horizontal direction.

6.3.3 Block Identifier Formation

The block identifier consists of the block signature ($\mathcal{B}_s$), the horizontal and vertical distances, $\mathcal{D}_x$ and $\mathcal{D}_y$, the sign $\mathcal{S}_x$ and the features of the “Un-Qualified” macro-blocks

(*UMBs*) $\mathcal{F}_{ub}$. Given an image, it is firstly divided into multiple macro-blocks. The “flipability” of the candidate pixels in each finer-block is determined and $\mathcal{N}_f$ for each macro-block is calculated. Thereafter, the macro-blocks are classified based on $\mathcal{N}_f$ and $\mathcal{B}_s$ for each macro-block is generated. The distance between the two consecutive *QMBs* is computed. While for $MB \in \{C_3\}$, namely the “Self-Detecting” macro-block, it does not participate in the block chaining process, however, the $\mathcal{B}_s$ will be embedded for “Self-Detecting” changes. The $\mathcal{D}_x$ and $\mathcal{D}_y$ are represented by a fixed length binary sequence, e.g., $\mathcal{D}_x = 4$ bits and $\mathcal{D}_y = 3$ bits. To compute $\mathcal{F}_{ub}$, each class C_i of the *UMBs* is assigned a value, where $i = 0, 1, 2, 3$. The total number of macro-blocks for class C_i of the *UMBs* ($\mathcal{N}_{C_i}$) is calculated to form the set $\mathcal{N}_u$, i.e., $\mathcal{N}_u = \{\mathcal{N}_{C_0}, \mathcal{N}_{C_1}, \mathcal{N}_{C_2}, \mathcal{N}_{C_3}\}$. The minimum and maximum of $\mathcal{N}_u$ and their respective class values are mapped to binary bits to generate the binary features, e.g., 4 bits are used currently. A look up table can be set up for security concerns for the mapping. In order to prevent the adversary from swapping any two macro-blocks of the *UMBs*, especially for $MBs \in \{C_2, C_3\}$, the minimum, maximum, median and mean of the block indices of C_2 and C_3 are also mapped to binary bits. These binary bits are subsequently combined with the binary features by performing *Xor* to generate the $\mathcal{F}_{ub}$. The *BI* defined in Figure 6.3 is given by

$$\left\{ \begin{array}{ll} BI = \mathcal{B}_s & \text{if } MB \in C_3 \\ BI = \mathcal{B}_s \parallel \mathcal{D}_x & \text{if } MB \in C_4 \\ BI = \mathcal{B}_s \parallel \mathcal{S}_x \parallel \mathcal{D}_x \parallel \mathcal{D}_y & \text{if } MB \in C_5 \\ BI = \mathcal{B}_s \parallel \mathcal{S}_x \parallel \mathcal{D}_x \parallel \mathcal{D}_y \parallel \mathcal{F}_{ub} & \text{if } MB \in C_6 \\ BI = \mathcal{B}_s \parallel \mathcal{S}_x \parallel \mathcal{D}_x \parallel \mathcal{D}_y \parallel \mathcal{F}_{ub} & \text{if } MB \in C_7 \end{array} \right. \quad (6.6)$$

where “ $\parallel$ ” denotes the “concatenation” operation. It is not difficult to see that *BI* varies with the changes of the macro-block types. Obviously the larger the $\mathcal{N}_f$ is, the more accurate the localization results will be.

Figure 6.3: Block Identifier For Different Classes.

6.3.4 Authentication Data Generation and Embedding

The macro-blocks are employed as the data hiding units in order to provide tampering localization information. The cryptographic signature (CS) and block identifier generation and embedding process shown in Figure 6.4 is described as follows.

1. Divide the image into multiple macro-blocks.
2. Determine the “flippability” for each finer-block and compute $\mathcal{N}_f$ for each macro-block.
3. Classify the macro-blocks into eight categories and form the block identifier.
4. Embed the block identifier by flipping the “flippable” pixels (if needed) in each macro-block to enforce the odd-even feature of the 3×3 block.
5. Embed the overall authentication data, i.e., cryptographic signature, $\mathcal{W}_s$, in those macro-blocks containing more “flippable” pixels than that required to embed the BIs , e.g., $MBs \in \{C_7\}$. The $\mathcal{W}_s$ is generated by setting all the “flippable” locations to “0s” to generate image Y_1 . Feed Y_1 into $Hash()$ to generate the hash value $\mathcal{H}_o$, e.g., $MD5$ is

employed, which generates 128 bits hash value. Finally generate $\mathcal{W}_s = E_k(\mathcal{H}_o, \mathcal{K}_s)$, where $\mathcal{K}_s$ is the private key of the authorized user and $E_k()$ is the encryption algorithm such as *RSA*.

6. Obtain the watermarked image Y_w when all the macro-blocks have been processed. In Figure 6.4, the process shown in dashed lines is used to generate the cryptographic signature.

Figure 6.4: Authentication Data Generation and Embedding.

6.3.5 Authentication and Tampering Localization

Since the “flippability” of the pixels in each macro-block does not change in the data hiding process, the same process can be carried out to determine the number of “flippable” pixels for each macro-block of the watermarked image Y_w . Thereafter, the macro-blocks are classified, the distance between two consecutive *QMBs* and the features of the *UMBs* are computed. Hence, the block identifier can be re-created. The details of the data extraction, authentication and tampering localization processes in Figure 6.5 are described below.

1. Perform the same process as that in the data embedding process to form BI_w for the watermarked image.

2. Identify the current macro-block as “tampered” if the new computed block signature $\mathcal{B}_{sw}$ is different from the one extracted from the watermarked image $\mathcal{B}_{se}$.

3. Identify the *UMBs* between two consecutive *QMBs* as “tampered” if the new calculated $\mathcal{S}_{xw}$, $\mathcal{D}_{xw}$, $\mathcal{D}_{yw}$ or $\mathcal{F}_{ubw}$ are different from those extracted. The tampered area lies between the previous and the current *QMBs*.

4. Verify integrity and authenticity. The authentication data $\mathcal{W}'_s$ is extracted from those blocks containing more “flippable” pixels than that required to embed the block identifiers, e.g., $MBs \in \{C_7\}$, by computing the odd-even features of the corresponding 3×3 blocks. Thereafter, $\mathcal{W}'_s$ is decrypted by providing the public key $\mathcal{K}_p$ of the authorized user or owner to $D_k(\mathcal{W}'_s, \mathcal{K}_p)$ to obtain $\mathcal{H}'_o$, where $D_k()$ is the decryption algorithm. The same process is performed to generate the hash value $\mathcal{H}_w$ of Y_w , e.g., clear the “flippable” locations to generate the intermediate image Y_{w1} and feed Y_{w1} to a hash function $Hash(Y_{w1})$ such as *MD5* to obtain $\mathcal{H}_w$. Comparison of $\mathcal{H}'_o$ with $\mathcal{H}_w$ gives the authentication results.

Figure 6.5: Data Extraction, Authentication and Tampering Localization.

6.4 Experimental Results and Discussions

6.4.1 Tampering Localization Results

Extensive experiments are carried out to test the efficiency of tampering detection and localization mechanism, the results are shown in Figure 6.6. In these experiments, the

Figure 6.6: Results of Data Hiding and Tampering Localization. (a) the original image of size 280×280 , (b) the watermarked image with 1446 bits embedded with MB of size 31×31 , (c) the tampered image. Several regions in the image are tampered by erasing, cropping, cutting and pasting, and (d) the tampering localization results. The blocks painted in black are the blocks that are tampered.

parameters are chosen as $T_0 = 9$, $T_1 = 13$, $T_2 = 17$ and $T_3 = 21$ based on the definition of the block identifier in Equation (6.6). These parameters are chosen so that the full block identifier for each class can be embedded. A single processing case of *MTDH* is employed. Generally, the threshold for a class should be chosen such that the full or majority of the block identifier for the corresponding class can be embedded. However,

if the threshold is chosen to be lower than that required to embed a complete block identifier for the corresponding class, the actual length of block identifier that has been embedded is equal to the number of flippable pixels N_f in the macro-block. In this case, the localization of the tampering is determined by comparing the extracted BI_e with the re-computed BI_w of the watermarked image for the length of N_f . This is possible due to the fact that N_f in each macro-block can be determined in both the embedding and extraction processes. It is easy to see that good localization is achieved due to the small size of the macro-blocks. Generally, the accuracy of the localization relates to the size of tampered regions and macro-block size. Embedding the block identifier consumes some capacity, however, if the size of the macro-block is chosen to be sufficiently large, the decrease in capacity is small. Therefore, a good compromise should be made between the capacity required to embed other payload watermark, e.g., cryptographic signature, and the accuracy of identifying the tampered locations. The overall authentication results obtained are analogous to that described in Figure 4.17 in Chapter 4. Even a single pixel change renders the authentication fail.

Now we shall discuss how to adjust the parameters to manage the capacity in each macro-block such that the overall authentication bits, e.g., cryptographic signature, can be fully embedded. Let $N_c(k)$ represent the actual bits consumed in the k th macro-block for embedding the block identifier and let C_s represent the number of bits required to embed the cryptographic signature. For a total of n macro-blocks in an image, the available number of bits N_a to embed the cryptographic signature is given by

$$N_a = \sum_{k=1}^n (N_f(k) - N_c(k)) \quad (6.7)$$

where k is the index of the macro-block and $N_c(k)$ is given by

$$\begin{cases} N_c(k) = T_3 & \text{when } N_f(k) \geq T_3 \\ N_c(k) = N_f(k) & \text{when } N_f(k) < T_3 \end{cases} \quad (6.8)$$

To ensure the whole cryptographic signature can be embedded, $N_a \geq C_s$ should hold, which can be achieved by increasing the macro-block size; decreasing the threshold value or employing an embedding algorithm with larger capacity. Increasing the macro-block size can be easily implemented, however, the accuracy of tampering localization will be slightly decreased. Decreasing the threshold value, e.g., T_0, T_1, T_2, T_3 , will render the block identifier not to be fully embedded. Hence, the sensitivity to changes will decrease, which leads to miss detection of tamperings.

Miss detection of tamperings will most likely occur for those macro-blocks that do not have enough “flippable” pixels to embed a complete block identifier for the corresponding class, e.g., $MBs \in \{C_3, C_4, C_5\}$. In addition, if the chosen threshold for a class is lower than that required to embed the complete block identifier for the corresponding class, miss detection of tamperings may also occur. On the other hand, false alarm will most likely occur in those macro-blocks whose neighboring macro-blocks are tampered, which results in the changes either in the distance between two consecutive $QMBs$ or in the features of the $UMBs$. The probability of miss detection (P_{md}) and probability of false alarm (P_{fa}) defined in Equations (2.4) and (2.5) in Chapter 2 are used to evaluate the performance of the proposed tamperings detection and localization scheme. Forty text images of different sizes and resolutions are used in our experiments. The tamperings include erasing, cropping, cutting and pasting and tampering in the uniform black or white regions. Among a total of 731 tamperings, only 12 tamperings cannot be correctly localized and 11 tamperings are falsely identified, which results in $P_{md} = 1.64\%$ and $P_{fa} = 1.5\%$. All the miss-detected tamperings belong to small erasure, which can be

detected by the authentication layer.

6.4.2 Comparisons

Comparisons of our proposed tampering localization scheme with that proposed by Kim *et al.* [17] are made. The localization of tamperings in [17] is achieved by dividing the image into sub-images of size 128×128 . Each sub-image is independently shuffled, thereafter it is divided into two regions, A and B . Each shuffled sub-image is further divided into multiple small blocks of size 8×8 each. The uniform white and black sub-images and those sub-images with a total number of black or white pixels less than 6 are not used in the watermark embedding process according to the paper. A message authentication code of 96 bit long is generated in region B and subsequently embedded in region A , in which one bit is embedded in each small block. Similar tamperings are performed to the watermarked images generated by using our proposed and Kim *et al.*'s methods. The data hiding and tampering localization results are shown in Figures 6.7 and 6.8.

Observed from the results, the advantages of our proposed method lie in the following aspects. The localization accuracy achieved by our proposed method is about 16 times higher than that of Kim *et al.*'s method, due to a smaller macro-block size of 31×31 is employed in our method comparing the sub-image of size 128×128 employed in Kim *et al.*'s method. In addition, the visual quality of the watermarked image is not easy to control for Kim *et al.*'s method due to one pixel is forced to flip in each small block of size 8×8 in the chosen regions, especially for those smooth text images. Furthermore, as hinted in [17], the independent way of watermarking each sub-image for Kim *et al.*'s method makes it difficult to localize tamperings when a birthday attack is mounted, i.e., the sub-image is replaced with another faked sub-image [17], though the overall authentication layer can detect the changes.

of the author's watermark, *coefficient and quantization selection keys* (which we describe later) and the quantization parameter Δ are necessary to embed and to extract the mark. The watermark can be an encrypted version of the author identification which is used to establish sender authenticity. The only user-defined parameter is the positive integer L_{max} which is the maximum wavelet decomposition level. The host image and watermark are denoted $f(m, n)$ and $w(i)$,

(a)

of the author's watermark, *coefficient and quantization selection keys* (which we describe later) and the quantization parameter Δ are necessary to embed and to extract the mark. The watermark can be an encrypted version of the author identification which is used to establish sender authenticity. The only user-defined parameter is the positive integer L_{max} which is the maximum wavelet decomposition level. The host image and watermark are denoted $f(m, n)$ and $w(i)$,

(b)

of the author's watermark, *coefficient and quantization selection keys* (which we describe later) and the quantization parameter Δ are necessary to embed and to extract the mark. The watermark can be an encrypted version of the author identification which is used to establish sender authenticity. The only user-defined parameter is the positive integer L_{max} which is the maximum wavelet decomposition level. The host image and watermark are denoted $f(m, n)$ and $w(i)$,

(c)

Figure 6.7: Comparison of Data Hiding Results. (a) the original image of size 922×426 , (b) the watermarked image with 8,897 bits embedded by employing our proposed *MTDH* approach with block size 31×31 and double processing case, (c) the watermarked image by employing Kim *et al.*'s method with 2,016 bits embedded.

6.4.3 Discussions and Security Considerations

The proposed scheme for identifying the tampered locations is based on the observation that any tamperings occurred to the *QMB* will change its class type or block signature. On the other hand, any tamperings occurred between two consecutive *QMBs* may render

a new *QMB* be generated and hence the distances or features of the *UMBs* between them will change. The secrecy of the *LUT* that is used to map the features further makes it difficult for a tampering to go undetected. However, if the image contains large uniform or “non-flippable” regions between two consecutive *QMBs*, the localization results may not be so accurate. For images of small size, the capacity may not be large enough to embed both the block identifier and cryptographic signature. In this case, only cryptographic signature may be embedded. Generally, a message authentication code of length 128 is considered to be acceptable in terms of security.

Let us denote the average probability of error for the k th macro-block is $P_B(k)$, the probability for a bit in the block identifier or cryptographic signature to be wrong is P_e and assume j bits in the block are randomly flipped, $P_B(k)$ is given by

$$P_B(k) = \sum_{j=1}^{N_f(k)} \binom{N_f(k)}{j} P_e^j (1 - P_e)^{N_f(k)-j} \quad (6.9)$$

For severe tampering, the extracted block identifier or cryptographic signature are essentially independent of that embedded. It is reasonable to set $P_e = \frac{1}{2}$ under this assumption, thus, $P_B(k)$ becomes

$$P_B(k) = \left(\frac{1}{2}\right)^{N_f(k)} \sum_{j=1}^{N_f(k)} \binom{N_f(k)}{j} \quad (6.10)$$

Therefore, the total probability of error P_M for a total of n *MBs* is given by

$$P_M = \sum_{k=1}^n \left(\frac{1}{2}\right)^{N_f(k)} \sum_{j=1}^{N_f(k)} \binom{N_f(k)}{j} \quad (6.11)$$

It is worthwhile noting that $\mathcal{B}_s$ of the macro-block is generated by mapping the mean feature code in each region to a binary bit. In total, there are 2^9 options of assigning the weights in each 3×3 finer-block, which option is chosen can be kept secret and

only known to the receiver. Without imposing any constraint on the runs of “0” and “1” in the *LUT* [110], there are 2^{512} , i.e., 1.34×10^{154} options for setting up the *LUT*. Hence the probability for an adversary to figure out the *LUT* to compromise $\mathcal{B}_s$ is low. Even for the worst case, assume that the adversary tries to figure out the binary bits in each macro-block, with 0.5 probability, a total of $2^{9 \times M \times N}$ attempts is needed, where $M \times N$ is the number of the macro-blocks being affected by the tampering. Generally, $M \times N \geq 4$, the probability that the tampering go undetected is $p \leq \frac{1}{2^{36}} = 1.45 \times 10^{-11}$, which is considered small. It should be noted that currently 9 bits is used to represent $\mathcal{B}_s$ for each macro-block, which can be increased with the increase of the size of the macro-blocks, hence the security level will be improved as well. Analogous to the block chaining scheme proposed in [104], the *QMBs* in our proposed scheme are chained and the distances between two *QMBs* and the features of the “Un-Qualified” macro-blocks are coded. In order to tackle the sensitivity of the proposed scheme to random noise, Error Correction Coding (*ECC*), e.g., *BCH*(31,26,1) can be used to encode the watermark bits, which consists of the *BI* and *CS* for each macro-block. Of course, in using *ECC*, the total capacity will drop but the robustness has improved.

6.5 Chapter Summary

In this Chapter, we present a novel blind two-layer data hiding scheme for binary images authentication and tampering localization. Conveniently, any block-based data hiding scheme whose “flippability” of the pixels does not change in the embedding process can be incorporated. The proposed technique of dividing the images into macro-blocks and embedding the block identifier in each macro-block is effective in detecting tamperings occurred to the watermarked image, both in the “Qualified” and “Un-qualified” macro-blocks. The block signature, the distance between two “Qualified” macro-blocks and

the features of the “Un-qualified” macro-blocks can track the changes effectively. The proposed two-layer authentication, i.e., the overall image authentication layer and the tampering detection and localization layer, is effective in detecting any changes and in the meantime the locations being tampered can be identified. Experimental results validate the arguments made.

of the author's watermark, *coefficient and quantization selection keys* (which we describe later) and the quantization parameter Δ are necessary to embed and to extract the mark. The watermark can be an encrypted mark of the author identification which is used to establish author authenticity. The only user-defined parameter is the positive integer L_{max} which is the maximum wavelet decomposition level. The host image and watermark are denoted $f(m, n)$ and $w(i)$,

(a)

of the author's watermark, *coefficient and quantization selection keys* (which we describe later) and the quantization parameter Δ are necessary to embed and to extract the mark. The watermark can be an encrypted mark of the author identification which is used to establish author authenticity. The only user-defined parameter is the positive integer L_{max} which is the maximum wavelet decomposition level. The host image and watermark are denoted $f(m, n)$ and $w(i)$,

(b)

of the author's watermark, *coefficient and quantization selection keys* (which we describe later) and the quantization parameter Δ are necessary to embed and to extract the mark. The watermark can be an encrypted mark of the author identification which is used to establish author authenticity. The only user-defined parameter is the positive integer L_{max} which is the maximum wavelet decomposition level. The host image and watermark are denoted $f(m, n)$ and $w(i)$,

(c)

of the author's watermark, *coefficient and quantization selection keys* (which we describe later) and the quantization parameter Δ are necessary to embed and to extract the mark. The watermark can be an encrypted mark of the author identification which is used to establish author authenticity. The only user-defined parameter is the positive integer L_{max} which is the maximum wavelet decomposition level. The host image and watermark are denoted $f(m, n)$ and $w(i)$,

(d)

Figure 6.8: Comparison of Tampering Localization Results. (a)(b) the tampered watermarked images for the images shown in Figure 6.7(b)(c), several regions are tampered by erasing, cropping, cutting and pasting and (c)(d) the tampering localization results for (a)(b). The blocks that are highlighted in black are being tampered.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we study the problem of data hiding on binary images for authentication purposes. We investigate and propose algorithms that fall in spatial and transform domains. In the spatial domain techniques, methods in object and pixel level are proposed. The problem of localization of the tamperings for binary images is also addressed.

In the first category, we propose a novel object-based method for text document images authentication. The inter character and word spaces are integrated and the relationships between the left and right white spaces of a character are employed for the watermark embedding and retrieval. The fact that coding the watermark does not touch the characters is employed to generate the message authentication codes to ensure the integrity of the images. Compared with conventional line, word and character shifting methods, the superiority of our proposed method has been experimentally demonstrated. Our scheme achieves high capacity which is comparable to existing character shifting method [6], but with improved visual quality and better security. This method is especially suitable for the applications that do not allow the distortions introduced by flipping pixels in binary images.

In the second category, we propose a pattern-based data hiding method with an aim to preserve the connectivity among pixels in 3×3 blocks. To this end, a “Connectivity-Preserving” criterion is defined and used to assess the “flippability” of a pixel in binary images. Specifically, the transitions from the processing pixel to its eight neighbors in the horizontal and vertical directions, isolated and sharp corners are calculated. The invariant features in flipping a pixel in binary images are employed to find the “embeddable” locations for both the watermark embedding and extraction to achieve blind watermark extraction. The adaptive way of finding the “embeddable” blocks to carry the watermark data addresses the uneven embedding capacity problem in binary images. The proposed fixed 3×3 block, non-interlaced block and interlaced block partition schemes achieve from moderate to large capacity with good visual quality. The problem of how to locate an “embeddable” pixel in a block is addressed, which facilitates the incorporation of cryptographic signature to detect any tampering occurred to the watermarked image. Comparisons with other existing methods show the superiority of the proposed method in the following aspects. The system security of the proposed scheme is comparable to that of the digital signature scheme such as *RSA* and the capacity achieved is about one times larger than that of Wu *et al.*’s method [15] using the interlaced block scheme. The visual quality of the watermarked image obtained in our approach outperforms other methods by embedding high capacity payload watermark.

In the third category, we investigate the feasibility of data hiding for binary images in transform domain. Existing method that employs real discrete cosine transform does not work well for binary images due to the errors introduced in pre-blurring and post-biased binarization process. We further extend the “Connectivity-Preserving” idea and view flipping an edge pixel in binary images as shifting the edge locations horizontally and vertically one pixel. Based on this observation, we propose an interlaced morphological wavelet transform such that the transitions in binary images and the shifted edges due

to flipping pixels can be tracked, which facilitates blind watermark extraction and incorporation of the cryptographic signature. The redundancy in the interlaced transform can be eliminated by the prediction and update lifting scheme. Processing an image based on 2×2 blocks ensures efficient utilization of the “flippable” pixels that addresses the “un-embeddability” of the block boundary or sharing rows and columns in the block-based approach. The two processing cases that flipping the candidate pixels in one case does not affect the “flippability” condition of another case can be employed for “Orthogonal Embedding” such that not only the capacity has significantly increased and but also the distortion can be minimized. The proposed method achieves larger capacity compared with other existing schemes, which is comparable to Tseng *et al.*’s algorithm [12] but with better visual quality in terms of average per pixel distortion. The capacity increases at the cost of slightly compromising the visual quality, e.g., a decrease of 0.02 to 0.16 in average per pixel distortion compared with Wu *et al.*’s method [13–15] and our connectivity-preserving pattern-based approach. Lower computational complexity is another advantage of the proposed approach compared with other existing algorithms.

The “uneven” distribution of the “flippable” pixels in binary images makes it difficult to locate the tamperings. In our investigation, on top of the two data hiding schemes proposed, we introduce a new concept of macro-block which is formed by grouping multiple finer-blocks. The macro-blocks are classified based on the sensitivity of the “flippability” condition to changes. The block identifier is defined adaptively for each class and embedded in the “Qualified” and “Self-Detecting” macro-blocks. The block identifier is defined in such a way that the tampering can be localized no matter whether it occurs to the “Qualified” or “Un-Qualified” macro-blocks. The macro-block signature, the distance and features between two “Qualified” macro-blocks are used to track the changes. Experimental results show good performance in the accuracy of localization, which is about 16 times more accurate than that achieved in Kim *et al.*’s approach [17]. Our proposed

scheme also achieves low probability of miss detection, e.g., 1.64%, and low probability of false alarm, e.g., 1.5%.

7.2 Future Work

Avenues for further study arising from the present research are briefly summarized below.

1. The segment size is chosen to be three in the proposed object-based data hiding scheme. It is possible to extend the segment size to be larger than three by enforcing the relationship of a group of characters such that certain robustness of the watermark may be achieved. The statistical features of the character spaces such as mean, variance and higher order statistics can be utilized for watermark embedding. Further investigation on JBIG2 standard which is for lossy and lossless bi-level image compression [111, 112] may provide more insights in the object-based data hiding techniques.

2. The “Connectivity-Preserving” pattern-based and morphological binary wavelet transform-based approaches proposed in this thesis achieve good performance in terms of capacity, visual quality, system security and computational complexity. Estimation of the capacity bound, i.e., the maximum number of bits that can be embedded in the host image without degrading the image quality perceptually based on certain distortion measure, would be an interesting direction which is worth investigating. It would be helpful to formulate a theoretical model on it.

3. The tampering detection and localization approach proposed in this thesis achieves tampering localization by embedding features related to the contents of the macro-blocks, which unavoidably introduce distortions to the host data. In addition, inserting the watermark at the time of recording the image is needed which would limit its application. Methods that employ the statistics of the image data itself to detect the forgeries for

color and gray images have been investigated in [107–109]. Whereas no method has been proposed for binary images due to the reason that manipulations such as cutting and pasting in uniform white background is not easy to be detected. Can we employ the statistical features or context relations between words and sentences of binary images for tampering localization? This would be an interesting direction to explore.

4. Most algorithms designed for binary images authentication are fragile, it would be beneficial to design a semi-fragile data hiding method. The semi-fragile method should have the capability of differentiating malicious tamperings from those content-preserving modifications such as format conversion, JBIG2 compression, as well as random bit errors caused in the file transmission. This would be an interesting topics to explore.

5. We have investigated the interesting properties of morphological binary wavelet transform and applied it for binary image data hiding. It would be interesting to investigate features of the coefficients in multi-levels and study the relationships among siblings and parent-child pairs of the coefficients, which may provide more insights in tampering localization problem.

6. Matrix embedding for high payload has been investigated in [113,114]. Our current schemes (i.e., connectivity-preserving pattern-based and morphological transform-based) code two bits for each flipping of the pixels [113]. How to increase the embedding efficiency for per flipping such that not only the capacity can be significantly increased but also the distortion can be decreased and security against steganalysis can be enhanced, which is an interesting direction to investigate.

Appendix A

Capacity Analysis for *CPDH*

In this Appendix, we analyze the capacities for the interlaced block (*IB*) and non-interlaced block (*NIB*) schemes (without imposing *LEP* criterion) from a theoretical point of view. We assume that the “flippability” of a pixel to be “flippable”, denoted by p , is independent to other pixels in the block since at most one pixel is flipped in each block. We show that the maximum achievable capacity for the *IB* and *NIB* schemes are determined by p as well as the block size $\mathcal{B}_z$. In this discussion, we assume that the “flippability” of each pixel is the same for the convenience of discussion.

A.1 Capacities for Interlaced Block Scheme

We firstly show that the capacity is related to block size $\mathcal{B}_z$ when p is fixed. The total number of blocks $\mathcal{N}_k$ for an interlaced block of size $n \times n$ is $\mathcal{N}_k(n) = \lfloor M/(n-1) \rfloor \times \lfloor N/(n-1) \rfloor$ and the probability for an interlaced block to be “embeddable” is $1 - (1-p)^{(n-2)^2}$, as discussed in Section 4.2.3. Without loss of generality, we ignore the floor function in the discussion. The total capacity $\mathcal{C}_\zeta$ is given by

$$\mathcal{C}_\zeta(n) = [1 - (1-p)^{(n-2)^2}] [M/(n-1) \times N/(n-1)] \quad (\text{A.1})$$

Let's denote $P(n) = (1-p)^{(n-2)^2}$, the ratio of $P(n+1)$ and $P(n)$, denoted by $\Delta P(n)$, is given by

$$\begin{aligned}\Delta P(n) &= \frac{P(n+1)}{P(n)} \\ &= (1-p)^{2n-3}\end{aligned}\tag{A.2}$$

Since $n \geq 3$ and $0 < 1-p < 1$, we have $0 < \Delta P(n) < 1$, i.e., $P(n)$ monotonically decrease, which means $1 - P(n)$, i.e., the first item in Equation (A.1) monotonically increase. The amount of increase, $\Delta I_p(n)$, is given by

$$\begin{aligned}\Delta I_p(n) &= (1 - P(n+1)) - (1 - P(n)) \\ &= (1-p)^{(n-2)^2} - (1-p)^{(n-1)^2}\end{aligned}\tag{A.3}$$

Since $0 < 1-p < 1$ and $n-2 \geq 1$, hence $0 < \Delta I_p(n) < 1$. On the other hand, the ratio of $\mathcal{N}_k(n+1)$ and $\mathcal{N}_k(n)$, denoted by $\Delta \mathcal{N}_k(n)$, is given by

$$\begin{aligned}\Delta \mathcal{N}_k(n) &= \frac{\mathcal{N}_k(n+1)}{\mathcal{N}_k(n)} \\ &= \left(\frac{n-1}{n}\right)^2 < 1\end{aligned}\tag{A.4}$$

From $\Delta \mathcal{N}_k(n) < 1$, we know that $\mathcal{N}_k(n)$, i.e., the second item in Equation (A.1) monotonically decrease. The amount of decrease, $\Delta D_k(n)$, is given by

$$\begin{aligned}\Delta D_k(n) &= \mathcal{N}_k(n) - \mathcal{N}_k(n+1) \\ &= MN \frac{2n-1}{n^2(n-1)^2}\end{aligned}\tag{A.5}$$

Let $\frac{2MN}{n^2(n-1)^2} = S$, we have $(n-1)S \leq \Delta D_k(n) \leq nS$. Generally, the block size n is not too large and the image size $M \times N$ is not too small, hence, it is reasonably to assume that $\Delta D_k(n) \geq 1$. Knowing that $0 < \Delta I_p(n) < 1$, it can be concluded that $\Delta D_k(n)$ dominant the changes of $\mathfrak{C}_\zeta$ with the increase of block size $\mathcal{B}_z$. From the above discussion, it can be concluded that the smaller the block size, the larger the capacity can be achieved for a

given p . The same conclusion can be drawn for non-interlaced block scheme as well.

Secondly, using an example, we show that the achievable capacity is related to p . The ratio of the capacities for an interlaced block with size $n = 3$ and $n = 4$, denoted as $\mathfrak{C}_{\zeta_{i3}}$, is given by

$$\begin{aligned}\Delta\mathfrak{C}_{\zeta_{i3}} &= \frac{[1 - (1 - p)^{(3-2)^2}][\frac{MN}{(3-1)^2}]}{[1 - (1 - p)^{(4-2)^2}][\frac{MN}{(4-1)^2}]} \\ &= \frac{9}{16 - 24p + 16p^2 - 4p^3}\end{aligned}\tag{A.6}$$

To achieve $\Delta\mathfrak{C}_{\zeta_{i3}} > 1$, $4p^3 - 16p^2 + 24p - 7 > 0$ should hold. Let $X = 4p^3 - 16p^2 + 24p$, X versus different probabilities plotted in Figure A.1 indicates that roughly when $p > 0.375$, $X > 7$ holds, i.e., the capacity achieved for $n = 3$ is larger than that achieved for $n = 4$. Conversely, when $p \leq 0.375$, the capacity of $n = 4$ is larger. This argument

Figure A.1: X Versus Different Probabilities p .

can be verified by directly plotting the capacity versus different block sizes for different probabilities, which is shown in Figures A.2. In the figure, $M \times N = 1600$.

A.2 Capacities for Non-Interlaced Block Scheme

Knowing that the capacity is affected by the block size for a given p , we now show that the capacity is also affected by p for non-interlaced block. The capacity for a non-interlaced

Figure A.2: Capacity Versus Different Block Sizes for Interlaced Block Scheme for Different Probabilities, (a) 0.2, (b) 0.3, (c) 0.3875 and (d) 0.45.

block with block size $n \times n$ is given by

$$\mathcal{C}_{\zeta_n} = [1 - (1 - p)^{(n-2)^2}] [M/n \times N/n] \quad (\text{A.7})$$

Similarly, the ratio of the capacities for a non-interlaced block of size $n = 3$ and $n = 4$, denoted as $\Delta \mathcal{C}_{\zeta_{n3}}$, is given by

$$\begin{aligned} \Delta \mathcal{C}_{\zeta_{n3}} &= \frac{[1 - (1 - p)^{(3-2)^2}] [\frac{MN}{3^2}]}{[1 - (1 - p)^{(4-2)^2}] [\frac{MN}{4^2}]} \\ &= \frac{16}{9(4 - 6p + 4p^2 - p^3)} \end{aligned} \quad (\text{A.8})$$

To achieve $\Delta\mathcal{C}_{\zeta_{n3}} > 1$, $54p - 36p^2 + 9p^3 > 20$ should hold. Let $T = 54p - 36p^2 + 9p^3$, T versus different probabilities p plotted in Figure A.3 shows that to achieve $T > 20$, i.e., $\Delta\mathcal{C}_{\zeta_{n3}} > 1$, roughly $p > 0.5375$ should hold. This can be confirmed by directly plotting capacity versus different $\mathcal{B}_z$ for different probabilities as shown in Figure A.4, where $M \times N = 160000$.

Figure A.3: T Versus Different Probabilities p .

Figure A.2 and Figure A.4 agree with our observation that the capacity is determined by the block size as well as the probability of each pixel to be “flippable”. This argument is supported by the extreme case of $p = 1$, in which each pixel is determined “flippable”. In this case, the smaller the block size, the larger the capacity. Since the largest number of blocks is obtained by choosing the smallest block size. Hence, $n = 3$ gives the largest capacity. However, for natural images, $p = 1$ is difficult to be achieved, hence the maximum capacity is achieved with varying block sizes for different images, which can be seen from Figure 4.13. The main reason lies in that the probability of each pixel to be “flippable” varies for natural images.

(a)

(b)

(c)

(d)

Figure A.4: Capacity Versus Different Block Sizes for Non-interlaced Block Scheme for Different Probabilities, (a) 0.3, (b) 0.45, (c) 0.58 and (d) 0.69.

Author's Publications

Journal Papers

1. Huijuan Yang and Alex C. Kot, "Binary Image Authentication With Tampering Localization By Embedding Cryptographic Signature and Block Identifier", *IEEE Signal Processing Letters*, vol. 13, no. 12, pp. 741-744, Dec. 2006.
2. Huijuan Yang and Alex C. Kot, "Pattern-Based Data Hiding for Binary Image Authentication By Connectivity-Preserving", regular paper, *IEEE Transactions On Multimedia*, vol. 9, no. 3, pp. 475-486, April 2007.
3. Huijuan Yang, Alex C. Kot and Susanto Rahardja, "Orthogonal Data Embedding for Binary Images in Morphological Transform Domain-A High Capacity Approach", regular paper, *IEEE Transactions on Multimedia*, accepted.
4. Huijuan Yang and Alex C. Kot, "A General Data Hiding Framework and Multi-Level Signature for Binary Images", *Lecture Notes in Computer Science*, accepted.

Patents

1. Alex C. Kot, Huijuan Yang and Haiping Lu, "Method, Software and Device for Hiding Data in Binary Images While Preserving Image Quality," *US patent provisional filed on 21 May 2004, patent application filed on 4 October 2004*.
2. Alex C. Kot, Huijuan Yang and Jun Liu, "Software and Method for Embedding Data in Two Color Images," *US patent application filed on 22 October 2005*.

Conference Papers

1. Huijuan Yang and Alex C. Kot, "Text Document Authentication by Integrating Inter Character and Word Spaces Watermarking", *Proc. of the 2004 IEEE Int.*

- Conf. on Multimedia and Expo. (ICME'2004)*, vol. 2, pp. 955-958, June 27-30, 2004, Taipei, Tai Wan.
2. Huijuan Yang and Alex C. Kot, "Data Hiding for Bi-level Documents Using Smoothing Techniques", *Proc. of the IEEE Int. Symp. on Circuits and Systems (ISCAS'2004)*, vol. 5, pp. 692-695, May 23-26, 2004, Vancouver, Canada.
 3. Jun Liu, Huijuan Yang and Alex C. Kot, "Relationships and Unification of Binary Images Data-Hiding Methods", *Proc. of the IEEE Int. Conf. on Image Processing (ICIP'2005)*, vol. 1, pp. 981-984, September 2005, Geneva, Italy.
 4. Huijuan Yang, Alex C. Kot and Jun Liu, "Semi-fragile Watermarking for Text Document Images Authentication", *Proc. of the IEEE Int. Symp. on Circuits and Systems (ISCAS'2005)*, vol. 4, pp. 4002-4005, May 2005, Kobe, Japan.
 5. Huijuan Yang and Alex C. Kot, "Data Hiding for Text Document Image Authentication by Connectivity-Preserving", *Proc. of The IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'2005)*, vol. 2, pp. 505-508, March 18-23, 2005, Philadelphia, USA.
 6. Huijuan Yang and Alex C. Kot, "Two-Layer Binary Image Authentication With Tampering Localization," *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'2006)*, vol. 2, pp. 309-312, May 14-19, 2006, Toulouse, France.
 7. Huijuan Yang, J. C. Patra and C. W. Chan, "An Artificial Neural Network-based Scheme for Robust Watermarking of Audio Signals", *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'2002)*, vol. 1, pp. 1029-1032, May 13-17, 2002, Florida, USA.
 8. Huijuan Yang and Alex C. Kot, "Data Hiding For Binary Images Authentication By Considering A Larger Neighborhood", *The IEEE Int. Symp. on Circuits and Systems (ISCAS'2007)*, pp. 1269-1272, 27-30 May 2007.
 9. Huijuan Yang, Alex C. Kot and Susanto Rahardja and Xudong Jiang, "High Capacity Data Hiding for Binary Images in Morphological Wavelet Transform Domain", *Proc. of the IEEE Int. Conf. on Multimedia & Expo. (ICME'2007)*, pp. 1239-1242, 2-5 July 2007.

Bibliography

- [1] Digital Vault, Data Encryption Software by StompSoft, <http://www.stompsoft.com/digital-vault.html>.
- [2] J. Fridrich, "Security of Fragile Authentication Watermarks with Localization", *Proc. SPIE Photonic West, Vol. 4675, Electronic Imaging 2002, Security and Watermarking of Multimedia Contents*, pp. 691-700, San Jose, California, January 2002.
- [3] M. Chen, E. K. Wong, N. Memon and S. Adams, "Recent Development in Document Image Watermarking and Data Hiding", *Proc. SPIE Conf. 4518: Multimedia Systems and Applications IV*, pp. 166-176, August 2001.
- [4] J. K. Brassil, S. Low, N. F. Maxemchuk and L. O'Gorman, "Electronic Marking and Identification Techniques to Discourage Document Copying", *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1495-1504, October 1995.
- [5] J. K. Brassil, S. Low, and N. F. Maxemchuk, "Copyright Protection for the Electronic Distribution of Text Documents", *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1181-1196, July 1999.
- [6] N. Chotikakamthorn, "Electronic Document Data Hiding Technique using Inter-Character Space", *The 1998 IEEE Asia-Pacific Conf. on Circuits and Systems*, pp. 419-422, November 1998.
- [7] N. Chotikakamthorn, "Document Image Data Hiding Technique Using Character Spacing Width Sequence Coding", *Proc. of Int. Conf. on Image Processing*, vol. 2, pp. 250-254, October 1999.
- [8] D. Huang and H. Yan, "Interword Distance Changes Represented by Sine Waves for Watermarking Text Images", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 12, pp. 1237-1245, December 2001.
- [9] Y.-W. Kim, K.-A. Moon, and I.-S. Oh, "A Text Watermarking Algorithm based on Word Classification and Inter-Word Space Statistics", *Proc. of Seventh Int. Conf. on Document Analysis and Recognition*, vol. 4, pp. 775-779, August 2003.
- [10] H.-K. Pan, Y.-Y. Chen and Y.-C. Tseng, "A Secure Data Hiding Scheme for Two-Color Images", *Proc. of the Fifth Symp. on Computers and Communications*, pp. 750-755, July 2000.
- [11] Y. C. Tseng and H.-K. Pan, "Secure and Invisible Data Hiding in 2-Color Images", *Proc. of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, vol. 2, pp. 887-896, 2001.

- [12] Y. C. Tseng and H.-K. Pan, "Data Hiding in 2-Color Images", *IEEE Trans. on Computers*, vol. 51, no. 7, pp. 873-878, July 2002.
- [13] M. Wu, E. Tang and B. Liu, "Data Hiding in Digital Binary Image", *IEEE Int. Conf. on Multimedia and Expo.*, vol. 1, pp. 393-396, New York, 30 July-2 August 2000.
- [14] M. Wu, "Multimedia Data Hiding", *Phd Thesis Dissertation, Princeton University NJ, June 2001*. [Online] Available: <http://www.ece.umd.edu/~minwu/research/phd-thesis.html>
- [15] M. Wu, and B. Liu, "Data Hiding in Binary Images for Authentication and Annotation", *IEEE Trans. on Multimedia*, vol. 6, no. 4, pp. 528-538, August 2004.
- [16] Q. Mei, E. K. Wong and N. Memon, "Data Hiding in Binary Text Document", *Proc. of SPIE*, vol. 4314, pp. 369-375, 2001.
- [17] H. Y. Kim and R. L. de Queiroz, "Alteration-Locating Authentication Watermarking for Binary Images", *Proc. Int. Workshop on Digital Watermarking*, pp. 125-136, 2004.
- [18] Y. Liu, J. Mant, E. K. Wong and S. Low, "Marking and Detection of Text Documents Using Transform-domain Techniques", *Proc. of SPIE, Electronic Imaging (EI'99) Conference on Security and Watermarking of Multimedia Contents*, vol. 3657, pp. 317-328, San Jose, CA, 1999.
- [19] H. Lu, X. Shi, Y. Q. Shi, A. C. Kot and L. Chen, "Watermark Embedding in DC Components of DCT for Binary Images", *2002 IEEE Workshop on Multimedia Signal Processing*, pp. 300-303, December 2002.
- [20] R. J. Anderson and F. A. P. Petitcolas, "On the Limits of Steganography", *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 474-481, May 1998.
- [21] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information Hiding-a Survey", *Proc. of the IEEE-Special Issue on Identification and Protection of Multimedia Information*, vol. 87, no. 7, pp. 1062-1078, July 1999.
- [22] M. D. Swanson, M. Kobayashi and A. H. Tewfik, "Multimedia Data-Embedding and Watermarking Technologies", *Proceedings of the IEEE*, vol. 86, no. 6, pp. 1064-1087, June 1998.
- [23] F. Hartung and M. Kutter, "Multimedia Watermarking Techniques", *Proc. of the IEEE-Special Issue on Identification and Protection of Multimedia Information*, vol. 87, no. 7, pp. 1079-1107, July 1999.
- [24] S. Katzenbeisser, F. A. P. Petitcolas, "Information Hiding Techniques for Steganography and Digital Watermarking", *Artech House, December 1999*.
- [25] I. J. Cox, M. L. Miller and J. A. Bloom: *Digital Watermarking*, San Mateo, CA: Morgan Kaufmann, 2001.
- [26] I. J. Cox and M. L. Miller, "The First 50 Years of Electronic Watermarking", *EURASIP Journal on Applied Signal Processing*, vol. 2, pp. 126-132, 2002.
- [27] M. Wu and B. Liu: *Multimedia Data Hiding*, Springer-Verlag, 2002.

-
- [28] P. Moulin and R. Koetter, "Data-Hiding Codes", *Proceedings of the IEEE*, vol. 93, no. 12, pp. 2083-2126, December 2005.
 - [29] C.-Y. Lin, M. Wu, J. A. Bloom, I. J. Cox, M. L. Miller and Y. M. Lui, "Rotation, Scale and Translation Resilient Watermarking for Images", *IEEE Trans. on Image Processing*, vol. 10, no. 5, pp. 767-781, May 2001.
 - [30] X. Kang, J. Huang, Y. Q. Shi and Y. Lin, "A DWT-DFT Composite Watermarking Scheme Robust to Both Affine Transform and JPEG Compression", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 776-786, August 2003.
 - [31] B. B. Zhu, M. D. Swanson, and A. H. Tewfik, "When Seeing Isn't Believing", *IEEE Signal Processing Magazine*, vol. 21, no. 2, pp. 40-49, March 2004,
 - [32] W. Stallings: *Cryptography and Network Security: Principles and Practice*, Prentice Hall 1999.
 - [33] C.-Y. Lin and S. F. Chang, "A Robust Image Authentication Method Distinguishing JPEG Compression from Malicious Manipulation", *IEEE Trans. on Circuits and Systems of Video Technology*, vol. 11, no. 2, pp. 153-168, February 2001.
 - [34] C.-S. Lu, H.-Y. Mark Liao, "Structural Digital Signature for Image Authentication: An Incidental Distortion Resistant Scheme", *IEEE Trans. on Multimedia*, vol. 5, no. 2, pp. 161-173, June 2003.
 - [35] Q. Sun and S.-F. Chang, "A Secure and Robust Digital Signature Scheme for JPEG2000 Image Authentication", *IEEE Trans. on Multimedia*, vol. 7, no. 3, pp. 480-494, June 2003.
 - [36] C. W. Wu, "On the Design of Content-Based Multimedia Authentication Systems", *IEEE Trans. on Multimedia*, vol. 4, no. 3, pp. 385-393, September 2002.
 - [37] L. Xie and G. R. Arce, "A Class of Authentication Digital Watermarks for Secure Multimedia Communication", *IEEE Trans. on Image Processing*, vol. 10, no. 11, pp. 1754-1764, November 2001.
 - [38] L. Xie, G. R. Arce and R. F. Graveman, "Approximate Image Message Authentication Codes", *IEEE Trans. on Multimedia*, vol. 2, no. 2, pp. 242-252, June 2001.
 - [39] P. Moulin and M. K. Mihcak, "A Framework for Evaluating the Data-Hiding Capacity of Image Sources", *IEEE Trans. on Image Processing*, vol. 11, no. 9, pp. 1029-1042, Vancouver, Canada, September 2002.
 - [40] D. Kundur, "Implications for High Capacity Data Hiding in the Presence of Lossy Compression", *Proc. of Int. Conf. on Information Technology: Coding and Computing*, no. 16, pp. 2127-2129, March 2000.
 - [41] T. Kalker, F. M. J. Willems, "Capacity Bounds and Constructions for Reversible Data-hiding", *Security and Watermarking of Multimedia Contents V. Edited by Edward J. Delp, III; Ping W. Wong, Proc. of the SPIE*, vol. 5020, pp. 604-611, 2003.
 - [42] P. H. W. Wong and O. C. Au, "A Capacity Estimation Technique for JPEG-to-JPEG Image Watermarking", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 746-752, August 2003.

- [43] J. Fridrich, M. Goljan and R. Du, "Lossless Data Embedding for All Image Formats", *Proc. of SPIE Electronic Imaging 2002: Security and Watermarking of Multimedia Contents*, vol. 4675, pp. 572-583.
- [44] J. Tian, "Reversible Data Embedding Using a Difference Expansion", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 890-896, August 2003.
- [45] A. M. Alattar, "Reversible Watermark Using Difference Expansion of a Generalized Integer Transform", *IEEE Trans. on Image Processing*, vol. 13, no. 8, pp. 1147-1155, August 2004.
- [46] M. U. Celik, G. Sharma, A. M. Tekalp and E. Saber, "Lossless Generalized-LSB Data Embedding", *IEEE Trans. on Image Processing*, vol. 14, no. 2, pp. 253-266, February 2005.
- [47] L. Kamstra, and H.J.A.M. Heijmans, "Reversible Data Embedding into Images Using Wavelet Techniques and Sorting", *IEEE Trans. on Image Processing*, vol. 14, no. 12, pp. 2082-2090, December 2005.
- [48] B. Chen and G. W. Wornell, "Quantization Index Modulation: A Class of Provably Good Methods for Digital Watermarking and Information Embedding", *IEEE Trans. on Information Theory*, vol. 47, no. 4, pp. 1423-1443, May 2001.
- [49] H. Yang and A. C. Kot, "Data Hiding for Bi-level Documents Using Smoothing Techniques", *Proc. of the 2004 IEEE Int. Symp. on Circuits and Systems (ISCAS'2004)*, vol. 5, pp. 692-695, May 2004.
- [50] H. Y. Kim and R. L. de Queiroz, "A Public-Key Authentication Watermarking for Binary Images", *Proc. of the IEEE Int. Conf. on Image Processing (ICIP'2004)*, vol. 5, pp. 3459-3462, October 2004.
- [51] S. V. D. Pamboukian, H. Y. Kim and R. L. de Queiroz, "Watermarking JBIG2 Text Region for Image Authentication", *Proc. of the IEEE Int. Conf. on Image Processing (ICIP'2005)*, vol. 2, pp. 1078-1081, September 2005.
- [52] H. Y. Kim, "A New Public-Key Authentication Watermarking for Binary Document Images Resistant to Parity Attacks", *Proc. IEEE Int. Conf. on Image Processing (ICIP'2005)*, vol. 2, pp. 1074-1077, September 2005.
- [53] H. Yang, A. C. Kot and J. Liu, "Semi-fragile Watermarking for Text Document Images Authentication", *Proc. of the IEEE Int. Symp. on Circuits and Systems (ISCAS'2005)*, vol. 4, pp. 4002-4005, May 2005.
- [54] J. Liu, H. Yang and A. C. Kot, "Relationships And Unification of Binary Images Data-Hiding Methods", *Proc. of the IEEE Int. Conf. on Image Processing (ICIP'05)*, vol. 1, pp. 981-984, September 2005.
- [55] J. Zhao and E. Koch, "Embedding Robust Labels into Images for Copyright Protection", *Proc. of the Int. Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Technologies*, pp. 1-10, Vienna, August 1995.
- [56] S. Hu, "Document Image Watermarking Algorithm Based on Neighborhood Pixel Ratio", *Proc. of the Int. Conf. of Acoustics, Speech, and Signal Processing 2005 (ICASSP'05)*, vol. 2, pp. 841-844, March 2005.

- [57] D. Zou and Y. Q. Shi, "Formatted Text Document Data Hiding Robust to Printing, Copying and Scanning", *Proc. of IEEE Int. Symp. on Circuits and Systems, 2005 (ISCAS'2005)*, vol. 5, pp. 4971-4974, May 2005.
- [58] T. Amamo and D. Misaki, "A Feature Calibration Method for Watermarking of Document Images", *Proc. 5th Int. Conf. on Document Analysis and Recognition*, pp. 91-94, Bangalore, India, 1999.
- [59] G. Pan, Z. Wu and Y. Pan, "A Data Hiding Method for Few-Color Images", *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'02)*, vol. 4, pp. 3469-3472, May 2002.
- [60] H. Lu, A. C. Kot and Y. Q. Shi, "Distance-Reciprocal Distortion Measure for Binary Document Images", *IEEE Signal Processing Letters*, vol. 11, no. 2, pp. 228-231, February 2004.
- [61] J. Cheng and A. C. Kot, "Objective Distortion Measure for Binary Images", *IEEE Region 10 Conference TENCON 2004*, vol. 1, pp. 355-358, November 2004.
- [62] A. T. S. Ho, N. B. Puhon, P. Marziliano, A. Makur and Y. L. Guan, "Perception Based Binary Image Watermarking", *Proc. of the 2004 Int. Symp. on Circuits and Systems (ISCAS'04)*, vol. 2, pp. 37-40, May 2004.
- [63] H. Lu, A. C. Kot and J. Cheng, "Secure Data Hiding in Binary Images for Authentication", *Proc. of the 2003 Int. Symp. on Circuits and Systems (ISCAS'03)*, vol. 3, pp. 806-809, May 2003.
- [64] K.-F. Hwang and C.-C. Chang, "A Run-length Mechanism for Hiding Data into Binary Images", *Proc. of Pacific Rim Workshop on Digital Steganography*, pp. 71-74, Kitakyushu, Japan, July 2002.
- [65] V. Korzhik, G. M.-Luna, D. Marakov and I. Marakova, "Watermarking of Binary Messages in Conditions of an Additive Binary Noise Attack", *IEEE Signal Processing Letters*, vol. 10, no. 9, pp. 277-279, September 2003.
- [66] C.-H. Tzeng and W.-H. Tsai, "A New Approach to Authentication of Binary Images for Multimedia Communication with Distortion Reduction and Security Enhancement", *IEEE Communications Letters*, vol. 7, no. 9, pp. 443-445, September 2003.
- [67] T.-H. Liu and L.-W. Chang, "An Adaptive Data Hiding Technique for Binary Images", *Proc. of the 17th Int. Conf. on Pattern Recognition (ICPR'2004)*, vol. 4, pp. 831-834, August 2004.
- [68] H. Y. Kim and A. AFif, "Secure Authentication Watermarking for Binary Images", *Proc. of the XVI Brazilian Symp. on Computer Graphics and Image Processing (SIB-GRAPI'03)*, pp. 199-206, October 2003.
- [69] H. Y. Kim and A. AFif, "A Secure Authentication Watermarking for Halftone and Binary Images", *International Journal of Image Systems and Technology*, vol. 14, no. 4, pp. 147-152, December 2005.
- [70] A. Markur, "Self-Embedding and Restoration Algorithms for Document Watermark", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'05)*, vol. 2, pp. 1133-1136, March 2005.

- [71] J. Fridrich, M. Goljan, P. Lisonek and D. Soukal, "Writing on Wet Paper", *IEEE Trans. on Signal Processing*, vol. 53, no. 10, pp. 3923-3935, October 2005.
- [72] M. Wu, J. Fridrich, M. Goljan and H. Gou, "Handling Uneven Embedding Capacity in Binary Images: A Revisit", *SPIE Conference on Security, Watermarking and Stegonography, San Jose, CA, Proc. of SPIE*, vol. 5681, pp. 194-205, January 2005.
- [73] M. S. Fu and O. C. Au, "Data Hiding Watermarking for Halftone Images", *IEEE Trans. on Image Processing*, vol. 11, no. 4, pp. 477-484, April 2002.
- [74] S.-C. Pei and J.-M. Guo, "Hybrid Pixel-based Data Hiding and Block-based Watermarking for Error-diffused Halftone Images", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 867-884, August 2003.
- [75] M. Jiang, N. Memon, E. Wong and X. Wu, "Quantitative Steganalysis of Binary Images", *Proc. of the IEEE Int. Conf. on Image Processing (ICIP'04)*, vol. 1, pp. 29-32, October 2004.
- [76] J. Cheng, A. C. Kot, J. Liu and H. Cao, "Steganalysis of Binary Text Images", *Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'05)*, vol. 4, pp. 689-692, March 2005.
- [77] J. Cheng, A. C. Kot, J. Liu and H. Cao, "Steganalysis of data hiding in binary text images", *Proc. IEEE Int. Symp. on Circuits and Systems (ISCAS'2005)*, vol. 5, pp. 4405-4408, May 2005.
- [78] J. Cheng, A. C. Kot, J. Liu and H. Cao, "Detection of Data Hiding in Binary Text Images", *Proc. IEEE Int. Conf. on Image Processing (ICIP'2005)*, vol. 3, pp. 73-76, September 2005.
- [79] M. Jiang, E. Wong, N. Memon and X. Wu, "Steganalysis of Halftone Images", *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP'05)*, vol. 2, pp. 793-796, March 2005.
- [80] X. Yu, Y. Wang and T. Tan, "Steganalysis of Data Hiding in Binary Images", *Proc. of the 17th Int. Conf. on Pattern Recognition (ICPR'2004)*, vol. 4, pp. 877-880, August 2004.
- [81] I. J. Cox, J. Kilian, F. T. Leighton and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia", *Proc. First Int. Workshop on Information Hiding*, R. Anderson, Ed., Cambridge, U. K.: Springer-Verlag, pp. 183-206, May/June 1996.
- [82] I. J. Cox, J. Kilian, F. T. Leighton and T. Shamoon, "Secure Spread Spectrum Watermarking for Multimedia", *IEEE Trans. on Image Processing*, vol. 6, no. 12, pp. 1637-1687, December 1997.
- [83] A. Rosenfeld, "Connectivity in Digital Pictures", *Journal of the Association for Computing Machinery*, vol. 17, no. 1, pp. 146-160, January 1970.
- [84] F. Y. Shih and W.-T. Wong, "Fully Parallel Thinning With Tolerance To Boundary Noise", *Pattern Recognition*, vol. 27, no. 12, pp. 1677-1695, 1994
- [85] R. C. Gonzalez and R. E. Woods: *Digital Image Processing*, 2nd Ed., Prentice Hall Inc., 2002.

-
- [86] A. K. Jain: *Fundamentals of Digital Image Processing*, Prentice Hall Inc., 1989.
 - [87] D. Kundur and D. Hatzinakos, "Digital Watermarking for Telltale Tamper Proofing and Authentication", *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1167-1179, July 1999.
 - [88] P. Fan and M. Darnell: *Sequence Design for Communications Applications*, Research Studies Press Ltd., 1996.
 - [89] P. W. Wong and N. Memon, "Secret and Public Key Image Watermarking Schemes for Image Authentication and Ownership Verification", *IEEE Trans. on Image Processing*, vol. 10, no. 10, pp. 1593-1601, October 2001.
 - [90] M. Holliman and N. Memon, "Counterfeiting Attacks on Oblivious Block-Wise Independent Invisible Watermarking Schemes", *IEEE Trans. Image Processing*, vol. 9, no. 3, pp. 432-441, 2000.
 - [91] Lydia I. Kronsjo: *Algorithms: Their Complexity and Efficiency*, John Wiley & Sons Ltd., 1987.
 - [92] Barry B. Brey: *The Intel Microprocessors: 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, Pentium Pro Processor, Pentium II, Pentium III, and Pentium 4: Architecture, Programming, and Interfacing*, Six Edition, Prntice Hall, Upper Saddle River, New Jersey, 2003.
 - [93] Henk J. A. M. Heijmans, *Morphological Image Operators*, Boston, MA: Academic, 1994.
 - [94] J. Goutsias and Henk J. A. M. Heijmans, "Multiresolution Signal Decomposition Schemes-Part I: Linear and Morphological Pyramids", *CWI, Amsterdam, The Netherlands, Tech. Report. PNA-R9810*, October 1998.
 - [95] J. Goutsias and Henk J. A. M. Heijmans, "Nonlinear Multiresolution Signal Decomposition Schemes-Part I: Morphological Pyramids", *IEEE Trans. on Image Processing*, vol. 9, no. 11, pp. 1862-1876, November 2000.
 - [96] Henk J. A. M. Heijmans and J. Goutsias, "Nonlinear Multiresolution Signal Decomposition Schemes-Part II: Morphological Wavelets", *IEEE Trans. on Image Processing*, vol. 9, no. 11, pp. 1897-1913, November 2000.
 - [97] Y. Zhao, P. Campisi and D. Kundur, "Dual Domain Watermarking for Authentication and Compression of Cultural Heritage Images", *IEEE Trans. on Image Processing*, vol. 13, no. 3, pp. 430-448, March 2004.
 - [98] M. D. Swanson and A. H. Tewfik, "A Binary Wavelet Decomposition of Binary Images", *IEEE Trans. on Image Processing*, vol. 5, no. 12, pp. 1637-1650, December 1996.
 - [99] W. Sweldens, "The Lifting Scheme: A New Philosophy in Biorthogonal Wavelet Constructions", *Proc. SPIE: Wavelet Applications Signal Image Processing III*, vol. 2569, A. F. Lain and M. Unser, Eds., 1995, pp. 68-79.
 - [100] W. Sweldens, "The Lifting Scheme: A Custom-Design Construction of Biorthogonal Wavelets", *Appl. Comput. Harmon. Anal.*, vol. 3, pp. 186-200, 1996.

- [101] W. Sweldens, "The Lifting Scheme: A Construction of Second Generation Wavelets", *SIAM J. Math. Anal.*, vol. 29, pp. 511-546, 1998.
- [102] C. Fei, D. Kundur and Raymond H. Kwong, "Analysis and Design of Secure Watermark-Based Authentication Systems", *IEEE Trans. on Information Forensics and Security*, vol. 1, no. 1, pp. 43-55, March 2006.
- [103] M. U. Celik, G. Sharma, E. Saber and A. M. Tekalp, "Hierarchical Watermarking for Secure Image Authentication With Localization", *IEEE Trans. on Image Processing*, vol. 11, no. 6, pp. 585-595, June 2002.
- [104] P. S. L. M. Barreto, H. Y. Kim and V. Rijmen, "Toward Secure Public-Key Block-wise Fragile Authentication Watermarking", *IEE Proceedings on Vision, Image and Signal Processing*, vol. 149, no. 2, pp. 57-62, April 2002.
- [105] A. H. Ouda and M. R. El-Sakka, "A Practical Version of Wong's Watermarking Technique", *Proc. of the IEEE 2004 Int. Conf. on Image Processing*, vol. 4, pp. 2615-2618, October 2004.
- [106] Y. Yuan and C. T. Li, "Fragile Watermarking Scheme Exploiting Non-deterministic Block-wise Dependency", *Proc. of the 17th Int. Conf. on Pattern Recognition (ICPR'2004)*, vol. 4, pp. 849-852, August 2004.
- [107] A. C. Popescu and H. Farid, "Exposing Digital Forgeries by Detecting Traces of Resampling", *IEEE Trans. On Signal Processing*, vol. 53, no. 2, pp. 758-767, February 2005.
- [108] T.-T. Ng and S.-F. Chang, "Blind Detection of Digital Photomontage using Higher Order Statistics", *ADVENT Technical Report #201-2004-1, Columbia University, June 2004*.
- [109] S. Lyu and H. Farid, "How Realistic is Photorealistic?", *IEEE Trans. on Signal Processing*, vol. 53, no. 2, pp. 845-850, February 2005.
- [110] M. Wu, "Joint Security and Robustness Enhancement for Quantization Based Data Hiding", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 831-841, August 2003.
- [111] B. Martins, and S. Forchhammer, "Lossless, Near-lossless, and Refinement Coding of Bilevel Images", *IEEE Trans. on Image Processing*, vol. 8, no. 5, pp. 601-613, May 1999.
- [112] *Coding of Still Pictures*, ISO/IEC JTC1/SC29/WG1 (ITU-T SG8), 1999, [Online] Available at: <http://www.jpeg.org/jbig/jbigpt2.html>.
- [113] R. Crandall, "Some Notes on Steganography", posted on *Steganography Mail List*, [Online] Available at: <http://os.inf.tu-dresden.de/~westfeld/crandall.pdf>.
- [114] J. Fridrich and D. Soukal, "Matrix Embedding for Larger Payload", *IEEE Trans. on Information Forensics and Security*, vol. 1, no. 3, pp. 390-395, September 2006.