Optimisation of Tamper Localisation and Recovery Watermarking Techniques

MAHMOUD ALI ALNAANAH

A thesis submitted in partial fulfilment of the requirement of Staffordshire University for the degree of Doctor of Philosophy

August 2018

Abstract

Digital watermarking has found many applications in many fields, such as: copyright tracking, media authentication, tamper localisation and recovery, hardware control, and data hiding. The idea of digital watermarking is to embed arbitrary data inside a multimedia cover without affecting the perceptibility of the multimedia cover itself. The main advantage of using digital watermarking over other techniques, such as signature based techniques, is that the watermark is embedded into the multimedia cover itself and will not be removed even with the format change.

Image watermarking techniques are categorised according to their robustness against modification into: fragile, semi-fragile, and robust watermarking. In fragile watermarking any change to the image will affect the watermark, this makes fragile watermarking very useful in image authentication applications, as in medical and forensic fields, where any tampering of the image is: detected, localised, and possibly recovered. Fragile watermarking techniques are also characterised by a higher capacity when compared to semi-fragile and robust watermarking. Semi-fragile watermarking techniques resist some modifications, such as lossy compression and low pass filtering. Semi-fragile watermarking can be used in authentication and copyright validation applications whenever the amount of embedded information is small and the expected modifications are not severe. Robust watermarking techniques are supposed to withstand more severe modifications, such as rotation and geometrical bending. Robust watermarking is used in copyright validation applications, where copyright information in the image must remains accessible even after severe modification.

This research focuses on the application of image watermarking in tamper localisation and recovery and it aims to provide optimisation for some of its aspects. The optimisation aims to produce watermarking techniques that enhance one or more of the following aspects: consuming less payload, having better recovery quality, recovering larger tampered area, requiring less calculations, and being robust against the different counterfeiting attacks.

Through the survey of the main existing techniques, it was found that most of them are using two separate sets of data for the localisation and the recovery of the tampered area, which is considered as a redundancy. The main focus in this research is to investigate employing image filtering techniques in order to use only one set of data for both purposes, leading to a reduced redundancy in the watermark embedding and enhanced capacity. Four tamper localisation and recovery techniques were proposed, three of them use one set of data for localisation and recovery while the fourth one is designed to be optimised and gives a better performance even though it uses separate sets of data for localisation and recovery.

The four techniques were analysed and compared to two recent techniques in the literature. The performance of the proposed techniques vary from one technique to another. The fourth technique shows the best results regarding recovery quality and Probability of False Acceptance (PFA) when compared to the other proposed techniques and the two techniques in the literature, also, all proposed techniques show better recovery quality when compared to the two techniques in the literature.

Acknowledgement

I would like to express my sincere gratitude to my supervisors, Dr. Alison Griffiths and Dr. Abdel-Hamid Soliman, for their patience and support during the course of this research work.

My faithful thanks goes to my wife and my family for their patience, love, and support through my study.

I also want to thank all of my friends for their help and good times we spent together.

This research had a deep impact on my academic knowledge and experience, and it gave me a good insight on research, which will be very helpful in my future career.

Table of Contents

List of Figures

List of Tables

Abbreviations

2D	Two dimensional
AES	Advanced Encryption Standard.
BCH	Bose-Chaudhuri-Hochquenghem code.
CCITT	Consultative Committee for International Telephony and Telegraphy.
CRC	Cyclic Redundancy Check.
DCT	Discrete Cosine Transform.
DFC	Digital Fountain Codes.
DFT	Discrete Fourier Transform.
DPCM	Differential Pulse-Code Modulation.
DWT	Discrete Wavelet Transform.
HFCs	High Frequency Contents.
IWT	Integer Wavelet Transform.
JPEG	Joint Photographic Experts Group.
LSB	Least Significant Bit.
LSBs	Least Significant Bits.
MLP	Multilayer Perception neural network.
MSB	Most Significant Bit.
MSBs	Most Significant Bits.
MSE	Mean Squared Error.
MT	Mellin Transform
NCC	Normalised Correlation Coefficient.

PFA	Probability of False Acceptance.
PFD	Probability of False Detection.
PFR	Probability of False Rejection.
PSNR	Peak Signal to Noise Ratio.
px	Pixel.
RLE	Run-Length Encoding.
RLF	Random Linear Fountain codes.
ROE	Region of Embedding.
ROI	Region of Interest.
RONI	Region of Non-Interest.
SARI	Self-Authentication-and-Recovery Image Watermarking System.
SSIM	Structural SIMilarity Index.
SVD	Singular Value Decomposition.
SV	Singular Value.
VQ	Vector Quantisation.
XOR	Exclusive OR (logical operation).

List of Symbols

$A(i,j)$	The authentication data stored in a block $B(i,j)$.
$\hat{A}(i,j)$	The authentication data calculated for a block $B(i,j)$.
A_{S1}, A_{S1}, ...	The authentication data stored in sub-blocks B_{S1}, B_{S2}, etc.
$\hat{A}_{S1}$, $\hat{A}_{S1}$, ...	The authentication data calculated for sub-blocks B_{S1}, B_{S2}, etc.
$B(i,j)$	A block with indices i, j in the horizontal and the vertical directions respectively.
B_S	A sub-block.
B_{S1}, B_{S2}, ...	Sub-blocks with numbers 1, 2, ... etc.
b	Number of bits in general.
C	A constant.
d	Number of deleted (i.e. set to zero) LSBs in an image.
$F(u,v)$	Representation of an image in the frequency domain as a function of the horizontal (u) and vertical (v) indices
$f(m,n)$	Representation of an image in the spatial domain as a function of the horizontal (m) and vertical (n) spatial indices
G	Generator polynomial.
HH1, HH2, ...	Level 1, 2, ... etc. of High-High portion in wavelet transform.
HL1, HL2, ...	Level 1, 2, ... etc. of High-Low portion in wavelet transform.
I	Identity matrix.
I	A symbol for an image in general.
$\overline{I}$	The mean value of an image (I).
I_1, I_2, ...	Images with numbers 1, 2, ... etc.

i , j	The indices of a block in general (as in $B(i, j)$), where i and j refer to the horizontal and the vertical directions respectively.
$\hat{i}$, $\hat{j}$	The indices of a block mapping in general (as in $B(\hat{i}, \hat{j})$), where $\hat{i}$ and $\hat{j}$ refer to the horizontal and the vertical directions respectively.
K	A secret key.
K1, ***K2***, ...	Secret keys 1, 2, ... etc.
L	Number of authentication, i.e. localisation, bits in a block or bit sequence.
LH1, LH2, ...	Level 1, 2, ... etc. of Low-High portion in wavelet transform.
LL1, LL2, ...	Level 1, 2, ... etc. of Low-Low portion in wavelet transform.
M , N	The width (M) and the height (N) of an image in general.
m , n	The spatial indices of the pixels in an image in the horizontal (m) and the vertical (n) directions.
mod	Modulo operation.
P	The power in a pixel.
Q	Quality factor.
p , q	Initial conditions in general.
$A(i, j)$	The recovery data stored in a block $B(i, j)$.
r , c	Number of rows (r) and columns (c) in a matrix.
sv_1 , sv_2 , ...	The singular values of a matrix.
S	A binary sequence.
T	Total number of blocks.
U , Σ , V	The components of a matrix that are generated using Singular Value Decomposition.

u , v	The indices in the frequency domain of an image, in the horizontal (u) and the vertical (v) directions.
W	A watermark in general.
$W(i,j)$	The watermark in a block $B(i,j)$.
$w(x,y)$	A spatial filtering window in the position (x,y) .
x , y	The location of a pixel in an image in the horizontal (x) and the vertical (y) directions.
Z	A matrix in general.
α , β	Control parameters.
Γ	A threshold in general.
ϵ	Number of ones in a binary sequence.
b_r	Number of recovery bits in a block.
λ	Fidelity level.
$\sigma(I)$	The standard deviations of an image I .
$\sigma(I_1,I_2)$	The cross covariance of two images I_1 and I_2 .

Chapter 1: Introduction

This chapter presents a general introduction about watermarking and its different types and applications. The motivation, aim, objectives, methodology, and contributions of this research are also presented in this chapter.

1.1- General Background

The idea behind digital watermarking [1-3] is to embed arbitrary information inside a multimedia cover work in a way that is not perceptible. The cover work could be an image, a video, or an audio file. The nature of the embedded information depends on the application and the goal of watermarking.

A watermarked cover work might get modified before being received by the designated recipient. The modifications are called attacks, especially of they are intentionally applied by an intruder to change the information inside the watermark. Based on the the ability to extract the watermark after attacks; watermarking techniques are categorised into: robust, semi-fragile, and fragile techniques. These categories are described as follows:

- Robust watermarking techniques [4, 5] are supposed to resist severe attacks, such as: cropping, rotation, scaling, bending, spatial distortion, high lossy compression, and filtering. Using frequency domain transforms, such as Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT), makes the watermarking technique capable of resisting lossy compression and filtering attacks; however, geometrical attacks, such as rotation and bending, are the most difficult attacks to deal with because of the loss of synchronisation with the watermark data [6, 7]. Geometrical attacks could be global, i.e. applied to whole image, or local, i.e. applied to some parts of the image. Local geometrical attacks are the most difficult ones to deal with.

  Many techniques have been proposed to deal with geometrical attacks, some of them are:

- Transformation invariant domains techniques [8, 9], which rely on the fact that any translation in the spatial domain will not affect the amplitude of 2D Discrete Fourier Transform (DFT), and any rotation in the the spatial domain will affect DFT by the same angle [10], also, any scaling in the spatial domain will not affect the amplitude of 2D Mellin Transform (MT) [11].
- Image normalisation techniques [12-14], which rely on embedding the watermark on a normalised version of the image, the normalisation is done using image moments [7]
- Template based synchronisation techniques [15, 16], which rely on inserting a synchronisation pattern in the spatial or the frequency domain, this pattern is used for resynchronising the watermark. This technique compromises the capacity of the watermark because it uses part of the original capacity for synchronisation purposes, also, it is susceptible to template removal attacks [4].
- Synchronisation techniques based on image features [17-19], which rely on detecting intrinsic image features, such as edges, texture, corners, etc., before and after the attack and using these features for synchronising the watermark. These techniques are hard to analyse because of the lack of predefined mathematical models that characterise the transformation used in the attacks, also, there is no guarantee on the results or guarantee on finding appropriate features for synchronisation.

Robust watermarking techniques are mainly used in copyright validation applications, where even after modifying the cover work it is possible to extract the watermark, or a part of it, which will be used to identify the copyright information of the cover work.

- Semi-fragile watermarking techniques resist some attacks, such as lossy compression and low-pass filtering [20]. Semi-fragile techniques are used sometimes in image authentication and are less frequently used in image recovery due to their low capacity [21-23].
- Fragile watermarking techniques are characterised by their high sensitivity

to modification and their high capacity, which makes them frequently used in tamper localisation and recovery applications [24-27].

More theoretical information about watermarking is presented in Chapter 2.

1.2- Motivation and Problem Statement

Image authentication plays an important role in many fields, such as medical and forensic applications [24, 27]. One way of authentication is to calculate a signature information, such as a hash function, and attach it to the image as meta-data; however, this method does not have the ability to locate or recover the tampering. In addition, the attached meta-data might be removed due to format change [28]. Fragile watermarking provides an excellent choice for authentication applications because it provides the ability to locate and even recover the tampered area; also, the watermark will not be removed due to format change.

This research is focused on tamper localisation and recovery due to its importance in the field of authentication and because it is still an active field of research [25-27].

In tamper localisation and recovery, the image is usually partitioned into non overlapping blocks, and two sets of data are generated for each block. The first set is used to localise the tampered blocks, and it is usually composed of the parity or a hash of the block. The localisation data is usually stored in the block from which it is generated. The second set of data is used to recover the tampered block, and it is composed of an approximation of the block it is supposed to recover. The recovery data is mapped into a different block from the one from which it is generated.

A literature survey was conducted, and the different techniques used in tamper localisation and recovery were investigated. Through the survey, some shortcomings of the technique in the literature were found, these shortcomings resulted in reducing the capacity of the watermark or reducing the quality of recovered image. The main shortcoming that was found is the use of two different sets of data, one for localisation and another for recovery. Since both sets are related to the data inside the block and the data in each set can be useful for both purposes of localisation and recovery, it is more suitable to have one set of data

that is used for both purposes instead of using one set of data for each purpose. Having one set of data for each purpose results in redundancy and wast of the watermark data, and it could be avoided by having one set of data for both purposes.

Other shortcomings were also found, such as using multiple copies of the recovery data and the overcomplicated partitioning and mapping schemes that are not effective in enhancing the performance of the watermarking techniques.

1.3- Aim and Objectives

The aim of this research is to develop some optimised tamper localisation and recovery watermarking techniques that enhance the recovered image quality while maintaining high recoverable area ratio and being resistive to the different types of attacks. The optimisation is mainly based on using one set of data for both purposes of localisation and recovery instead of using a separate set of data for each purpose. To realise the aim of this research, the following objectives need to be achieved:

- Conduct a general literature survey in the field of image watermarking. This survey helps in determining the active research topics in the field of watermarking, and to focus the research in one topic. Based on this survey, tamper localisation and recovery was selected as a topic for this research.
- Conduct a literature survey in the existing watermarking techniques that are used for tamper localisation and recovery. The goal of this survey is to know the different methods, ideas, and challenges that exist in the research topic.
- Analyse the existing tamper recovery techniques in order to determine the shortcomings and the problems to which the research will contribute in finding new solutions or enhance existing ones.
- Design some algorithms for watermarking techniques that optimise tamper localisation and recovery methods in the surveyed literature and provide some solutions for the existing shortcomings.
- Validate the performance of the proposed techniques by comparing them to some of the existing ones using the validation parameters used in the

literature.

- Submit the findings to a journal in order to publish them.

1.4- Research Methodology

Research methodologies can be categorised into quantitative and qualitative based on the characteristics of the collected data [29]. In quantitative research, the data can be reduced into numbers, such as measuring temperature, pressure, number of bits, etc. In qualitative research, the data is usually expressed in words instead of numbers, such as describing people's feelings, judgments, beliefs, etc.

Quantitative methodology is used in this research, where the different aspects of the performance of the proposed watermarking techniques are measured and expressed as numbers, then they are compared to the numbers from other techniques. The details about the evaluation parameters used in this research are presented in Chapter 2.

1.5- Research Contributions and Statement of Originality

The contributions of this research could be summarised as follows:

- Proposing three techniques that use one set of data for both purposes of localisation and recovery. These techniques rely on direct comparison between the recovery data, i.e. the watermark image, and the watermarked image. Image filtering is applied on the resulting image after comparison, which is called the difference image, in order to localise the tampered area. A threshold is applied to the difference image so that any pixel greater than 0 will become 255, i.e. white. Image filtering is applied to the difference image to extract the tampered area. The resulting image after applying filtering to the difference image is called the mask image. Any white pixel in the mask image corresponds to a tampered pixel in the watermarked image. The differences between the three techniques are as follows:
    - The first technique uses random mapping for the recovery data. The spatial filter used in mask image generation returns 0 if the ratio of white

pixels inside its window is above certain threshold, which is selected to be 70%. Median filtering is also used to enhance the quality of the watermark image which is used to recover the tampered area. The technique is not sensitive to the tampering pattern; however, the quality of the recovered image degrades as the tampering ratio increases.

- The second technique divides the watermark image into 3 groups and maps these groups in the vertical, the horizontal, and the diagonal directions based on maximum-distance mapping. The spatial filter used for localisation uses 3x3 px window and returns white if its window has more than 3 white pixels inside it. The watermark is XORed with a random sequence in order to secure it. The bits inside the watermark pixels are randomly permuted to prevent knowing the secret sequence. The technique guarantees the recovery of any tampered area as long as its width and height does not exceed 50% of the width and the height of the watermarked image respectively.
- The third technique relies on measuring the high frequency contents (HFCs) in the watermark image to localise the tampered area. The HFCs in the tampered area appear because of XORing the watermark image with the same random sequence in the encoding and decoding stages, which cancel each others except for the tampered area, which is XORed only once and that results in the random contents in it. Maximum distance mapping is used in the third technique, which results in a large recoverable area of 50% of the image size. The technique guarantees the recovery of any tampered area as long as both of its width and height do not together exceed 50% of the width and the height of the watermarked image respectively.

- Proposing a fourth technique that has optimisation aspects such as using DCT to generate the recovery data and using maximum-distance mapping to ensure larger recoverable area. The technique guarantees the recovery of any tampered area as long as both of its width and height do not together exceed 50% of the width and the height of the watermark image respectively.

- Proposing random bit permutation inside each pixel in the second and the third techniques, and inside each block in the fourth technique. In these techniques maximum-distance mapping is used, which makes it possible to know the random sequence that is used to secure the watermark unless random bit permutation is used.

Some aspects of the proposed techniques exist in other techniques in the literature; however, the exact implementation of the techniques is novel, up to the knowledge of the author, and it was not proposed by other researchers, especially proposing image filtering to use recovery data in tamper localisation process.

1.6- Thesis Organisation

A general introduction is presented in Chapter 1, then Chapter 2 presents a general theoretical overview of some topics related to the thesis. A literature survey and analysis are presented in Chapter 3. The descriptions of the proposed techniques and the two techniques from the literature are presented in Chapter 4, followed by the experimental evaluation in Chapter 5. The final discussion, conclusion, and future work are presented in Chapter 6, followed by the references and the appendices at the end of the thesis.

1.7- Notes About Terminology

- The terms: localisation, authentication, and detection will be used interchangeably depending on the context; however, they give the same meaning, which is determining which part of the image is tampered with.
- For simplicity, the term "random" will be used in this thesis instead of the term "pseudorandom"; however, the term "pseudorandom" is scientifically more accurate.
- The suffix "px" will be used in the description of block size and it means "pixels", such as "2x2 px" which means "2x2 pixels".

1.8- The Image Database Used in this Research

The Kodak image database [30] is used in this research for the following reasons:

- Direct download availability of the database images.
- The number of images in the database is sufficient for acceptable statistical results.
- The database contains images with variety of detail levels and colour ranges, which makes it adequate for statistical calculations.
- The images in the database are coloured, which makes it useful if the research is extended to coloured images. The database images can be converted into grey-level whenever it is needed.

Kodak image database contains 24 true-colour images (i.e. each image has red, green, and blue channels with 8 bits assigned for each channel). The dimensions of the images are 768x512 px or 512x768 px. A preview of the images is shown in Figure 1.1 and Figure 1.2, and the image number is shown below each one.

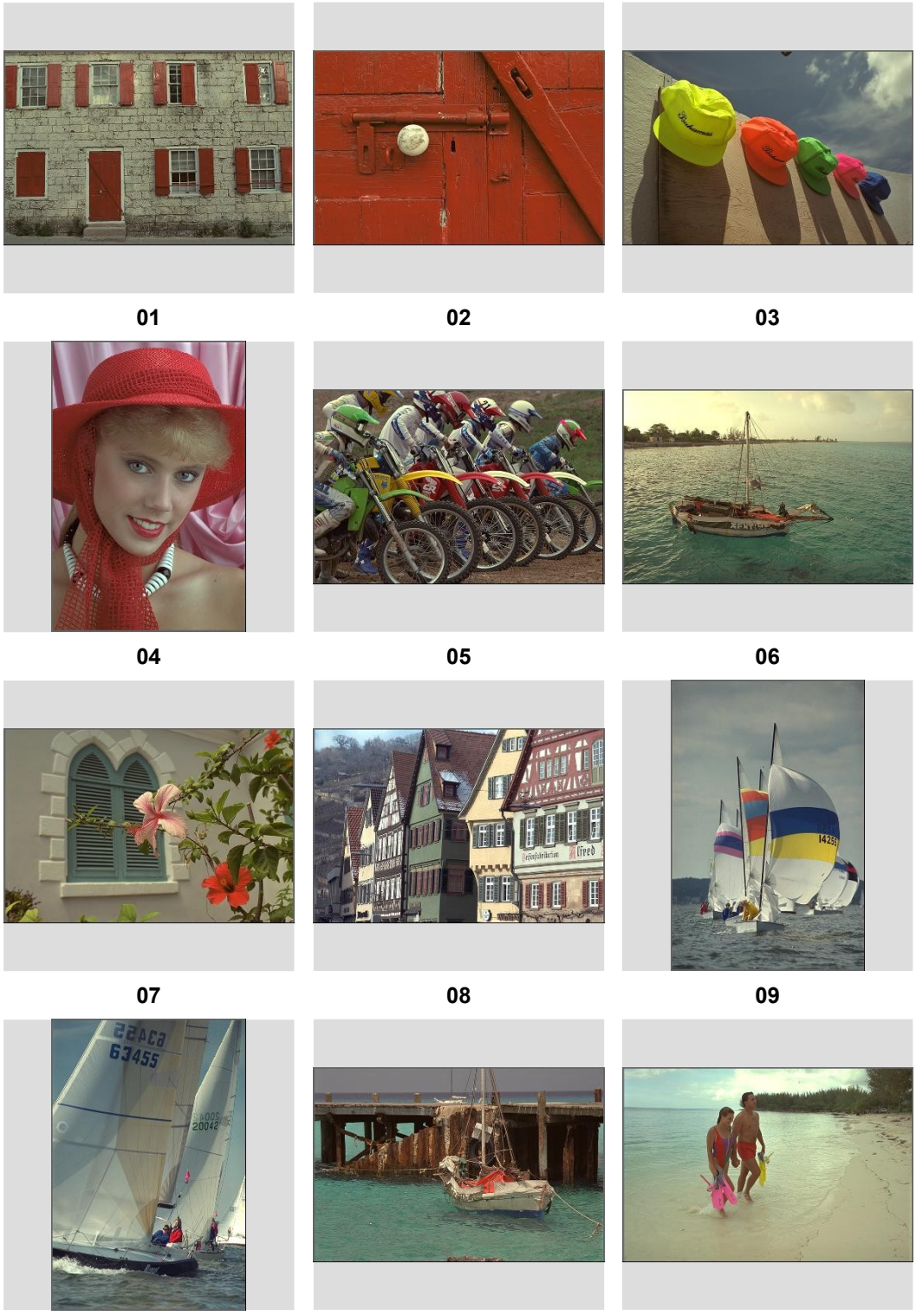


Figure 1.1: A preview for images 1 to 12 in the Kodak image database.

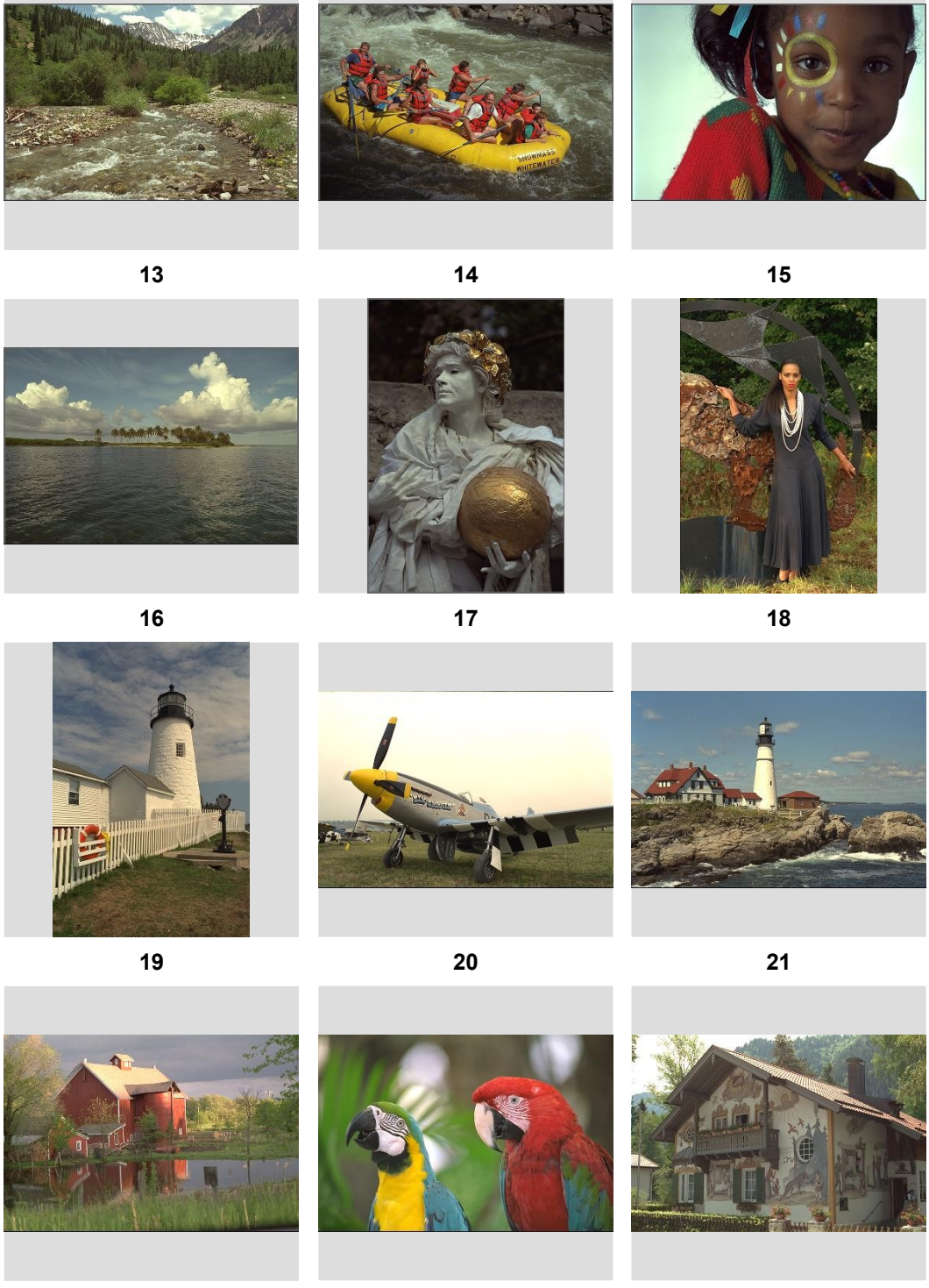


Figure 1.2: A preview for images 13 to 24 in the Kodak image database.

Chapter 2: General Theoretical Overview

2.1- Introduction

This chapter presents a general overview about some aspects of digital watermarking and an overview of some techniques that were used or mentioned in this research, such as: DCT, JPEG compression standard, Spatial Image Filtering, Singular Value Decomposition (SVD), and Cyclic Redundancy Check (CRC). The theoretical overview in this chapter is intended to give the reader general information about the intended topics without going into details.

2.2- Digital Watermarking Definition

Digital watermarking [1-3] is the embedding of arbitrary information, i.e. the watermark, in noise tolerant data, i.e. the cover work, such as: images, video and audio, in a way that is not perceptible by the observer.

Digital watermarking is different from meta-data [28], where in watermarking the cover work itself is used to embed the watermark, which means that the cover work is changed, while in meta-data the additional information is stored separately and the cover work remains intact. Therefore, watermarking can not be used with error-intolerant data such as computer programs or text files.

The cover work could either be an image, audio, or video file; however, this research is only concerned with image watermarking since it is widely used and audio/video watermarking is beyond the scope of this research.

The cover work might suffer certain modifications before reception, these modifications are known as ***attacks*** and they might be intentional or unintentional. In intentional attacks, the attacker usually tries to counterfeit the cover work without being detected, and that could be done by forging a new watermark for the counterfeited cover work. In forensics, for example, a fake image is counterfeited to appear as an authentic one. Therefore, the watermark is usually encrypted and a secret key is used to protect it from being forged by unauthorised intruders. The diagram shown in Figure 2.1 represents the general watermark embedding and

extraction process.

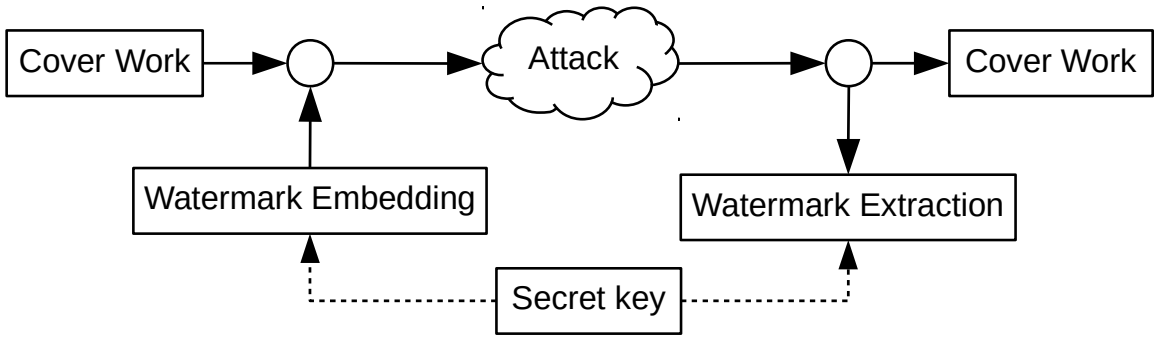


Figure 2.1: Watermark embedding and extraction process.

2.3- Applications of Digital Watermarking

There are many applications for digital watermarking, such as:

2.3.1- Copyright Protection

In this case, the embedded data represents a copyright information about the cover work, such as its: owner, date of production, etc.

2.3.2- Hardware Control

For example, a digital video player could be designed to play only the video stream that has an authentic watermark embedded in it.

2.3.3- Tamper Localisation and Recovery

The aim of tamper localisation is to locate any tamper in the cover work. The watermark is usually related to the cover work, such as using hash functions or parity check.

In the case of recovery, the watermark is usually an approximation of the cover work and its position is remapped in a way that the tampering is unlikely to affect the cover work and watermark in same position. This will enable the recovery of the damaged parts in the cover work from the undamaged parts in the watermark. Since this research is concerned with tamper localisation and recovery, more details about it will be provided later in section 2.7.

2.3.4- Steganography

Steganography [31] is a way of data hiding, in which the data is securely

exchanged by embedding it inside a cover work and extracting it at the receiving end by the authorised recipient. The advantage of steganography over encryption is that the cover work is not a potential target for an intruder to suspect that some secret data is hidden inside it.

2.4- Classifications of Digital Watermarking

There are many ways in which watermarking can be classified, the main ones are listed in this section.

2.4.1- Spatial and Frequency Domain Watermarking

Depending into which domain the watermark is embedded, watermarking is categorised as spatial or frequency domain watermarking. In spatial domain watermarking, the watermark is embedded directly into the pixels of the cover work, such as embedding it in the Least Significant Bits (LSBs) of image pixels. In frequency domain watermarking, the cover work is transformed into the frequency domain where the embedding is done. Different transforms could be used such as: DFT, DCT, and DWT.

2.4.2- Fragile, Semi-fragile, and Robust Watermarking

Watermarking techniques are categorised according to their robustness against modifications into: fragile, semi-Fragile, and robust watermarking techniques.

Fragile watermarking is usually used in authentication applications where any change to the cover work will alter the watermark. When this alteration is detected at the receiving end, this means that the cover work was tampered with and it is not authentic.

Semi-fragile watermarking techniques resist some attacks, such as lossy compression [20]. Semi-fragile watermarking techniques are used sometimes in image authentication and are used less frequently in image recovery due to their low capacity [21-23].

Robust watermarking is used in copyright applications, where even after modifying the cover work it is possible to extract the watermark, or a part of it, which will be used to identify some information about the cover work.

2.4.3- Reversible Watermarking

In reversible watermarking, also known as lossless watermarking, [32, 33] it is possible to fully recover the original cover work along with the watermark. This is usually achieved by applying lossless compression to the cover work and store the watermark data in the remaining space after compression.

2.4.4- Blind vs Informed (Non-Blind) Watermarking

In blind (or oblivious) watermarking, the original cover work is not required to extract the watermark; on the other hand, in informed watermarking the original cover work is required to extract the watermark correctly.

2.5- Types of Attacks on Watermarked Images

Watermarked images might suffer different types of attacks [34] and these attacks might be intentional or unintentional. Some examples of possible attacks are listed below:

2.5.1- Lossy Compression

In lossy compression, the pixel level is altered and the high frequency components are reduced, which could result in destroying the watermark. Lossless compression is not used for attacking watermarked images because it does not alter the pixel-values in the image.

2.5.2- Geometrical Attacks

Such as: scaling, rotation, cropping, and flipping. Geometrical attacks result in losing the spatial synchronisation needed for the watermarking algorithms to work.

2.5.3- Filtering

Such as applying low-pass filter which will remove the high frequency components in the image and alter the values of its pixels.

2.5.4- Collusion Attack

In a collusion attack [25], the watermark is detected and removed by using different watermarked versions of the cover work. This method, however, requires different watermarked versions of the cover work, which might not be available.

2.5.5- Vector Quantisation Counterfeiting Attack

In order to understand Vector Quantisation (VQ) counterfeiting attack [35], assume there is a watermarked image with an embedded watermark W. The watermarking is applied in an ***independent block-wise*** way where each block in the watermarked image is treated independently and it generates a corresponding block in the watermark. Assume there are some blocks in the same watermarked image, or from other watermarked images, that generate the same corresponding blocks in the watermark, then these blocks could be used as a codebook. From this codebook an approximation of the counterfeiting block is generated. By using the codebook, it is possible to generate a watermarked counterfeited image which will generate the same watermark and will be considered as an authentic one. This attack works without knowing the secret key that is used during watermarking, and sometimes even without knowing the watermark itself, especially if there are different images that use the same watermark.

The counterfeiting becomes harder as the block size in the watermarked image increases, also the watermarking process becomes more resistive to this type of attacks by introducing dependency between the blocks of the image. Collage attack [36] is a variation of VQ counterfeiting attack and it does not require the knowledge of the watermark, it only requires a number of images that use the same secret key.

2.6- Chaotic Maps and Watermark Shuffling

In order to increase the security and invisibility of the watermark, the watermark is shuffled using a chaotic map. One example of chaotic maps is Arnold's cat map [37, 38], which transforms the location (x_{old} , y_{old}) in an $M \times N$ image into new locations (x_{new} , y_{new}) according to the following equations:

$$\begin{aligned} x_{new} &= (x_{old} + y_{old}) \bmod M \\ y_{new} &= (x_{old} + 2\,y_{old}) \bmod N \end{aligned} \tag{2.1}$$

Where mod is modulo operation. For example, if the size of the image is 256x256 px, then the pixel at position (100,200) will be relocated to (100+200, 100+2*200) mod 256 = (300, 500) mod 256 = (44, 244).

Arnold's cat map could be expanded [39] to use the initial condition p and q as

follows:

$$x_{new} = (x_{old} + p\, y_{old}) \bmod M$$
$$y_{new} = (q\, x_{old} + (p\, q + 1)\, y_{old}) \bmod N \quad (2.2)$$

After some iterations, the image will look random and with more iterations it returns back to its original appearance, as seen in Figure 2.2, where the map is applied to 150x150 px bird image taken from Kodak database image number 23. The number of iterations is shown under each image.

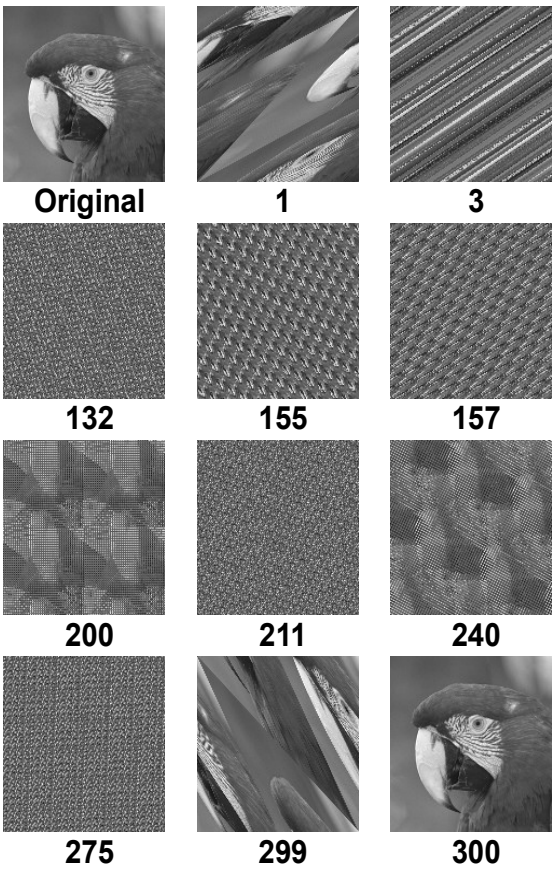


Figure 2.2: Arnold chaotic map applied to a 150x150 px image, the number of iterations is shown under each image.

2.7- The General Steps of Tamper Localisation and Recovery

This section provides a general introduction about the steps used in tamper localisation and recovery watermarking techniques, in order to have a good idea about the process before presenting the literature survey. This introduction is

concerned with the steps that are used in the techniques similar to the proposed ones, such as using the spatial domain for watermark embedding.

The encoding stage starts by dividing the image into blocks that have a size of one pixel or more, such as 2x2 px, 4x4 px, etc. Some LSBs in each block are reserved to store the watermark information, usually not more than 3 LSBs or otherwise the quality of the image will degrade to a noticeable level.

Two sets of data are usually extracted for each block. The first set is the localisation data which is used to locate any changes in the pixels inside the block, usually parity check or hash function is used for that purpose. The other set is the recovery data, which is usually a low resolution version of the block that is generated using its average or DCT. Localisation data is usually stored in the block it is generated from, and recovery data is stored in another block located as far away as possible from it. Encryption is applied to the watermark in order to secure it, one common method of encryption is to XOR the watermark with a random sequence that is generated based on a secret key.

The decoding stage involves the localisation of any tampering by calculating the localisation data and comparing it with the stored one. If any block is found to be tampered with, it will be recovered using the recovery data. When the block that has the recovery data is also tampered with, this is known as ***tamper coincidence*** and it results in a failure of the recovery of the tampered block.

When some parts of the image are not tampered with, the watermark data related to these parts is ignored and does not contribute to the quality of the untampered parts, which is considered as a waste of the watermark data, this problem is known as ***watermark-data waste problem***.

Tamper localisation and recovery techniques are categorised according to their recovery quality into: (a) ***Flexible*** quality techniques, where the quality of the recovered image increases as the tampered area decreases, and (b) ***Fixed*** quality techniques, where the quality of the recovered image is the same for different tampered area sizes.

2.8- Evaluation Parameters Used in Tamper Localisation and Recovery

This section introduces the main evaluation parameters that are used in tamper localisation and recovery techniques.

2.8.1- Peak Signal to Noise Ratio (PSNR)

Peak signal to noise ratio (PSNR) [40, 41] is used to measure the similarity between two images, the PSNR value increases as the similarity between them increases and it reaches infinity for identical images. In the case of tamper recovery, it is used to measure the similarity between the original unwatermarked image and the watermarked image, or between the original and the recovered image.

The PSNR between two images is defined as follows:

$$PSNR = 10\log_{10}\left(\frac{P_{MAX}}{MSE}\right) \tag{2.3}$$

where P_{MAX} is the maximum possible power in a pixel and is defined as follows:

$$P_{MAX} = (2^b - 1)^2 \tag{2.4}$$

where b is the number of bits assigned for each pixel. For example, if grey-level is stored in 8 bits then:

$$P_{MAX} = (2^8 - 1)^2 = 255^2 = 65025$$

MSE is the Mean Squared Error and is defined as follows:

$$MSE = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left(I_1(m,n) - I_2(m,n)\right)^2 \tag{2.5}$$

where I_1 and I_2 are the two images to be compared and they must have the same width (M) and height (N).

The PSNR gives a good indication about the similarity between two images and any distortion that results from the watermarking process; however, it might be deceiving sometimes [42] and does not reflect the measured perceptibility by the human eye. For example, adding a constant value to the whole image might give a PSNR value that is less than the one resulting from adding some noise to the image, but the noise will be more perceptible to the human eye since it is

concentrated in a few pixels.

In some cases, the value of the PSNR could be enhanced by introducing irrelevant information, as in the following test which shows the PSNR values resulting from setting some LSBs in the image to zero and how these values could be enhanced by storing irrelevant information instead of the deleted LSBs. The 24 images in Kodak image database were used in the following test, they were converted to grey-scale and 1 to 8 of their LSBs are set to zeros. The average PSNR for the 24 images was measured when the deleted LSBs are replaced by: zeros (Case 1), or random values (Case 2), or a constant value that equals $2^{(d-1)}$ (Case 3), where d is the number of deleted LSBs, and that is about half the maximum value that could be stored in the deleted LSBs. The results are shown in Table 2.1

Table 2.1: Average PSNR values when some LSBs are removed from the 24 images in Kodak image database

Number of deleted LSBs, (d)	Average PSNR (dB)		
	LSBs are replaced by 0	LSBs are replaced by random numbers	LSBs are replaced by a constant value ($2^{(d-1)}$)
1	51.077	51.140	51.211
2	42.617	44.141	46.406
3	35.656	37.912	40.742
4	29.111	31.781	34.704
5	22.843	25.759	28.742
6	16.758	19.695	22.634
7	10.369	13.814	17.026
8	6.669	8.780	13.413

It can be seen from Table 2.1 how the values of the PSNR were significantly improved by storing irrelevant information instead of the deleted one. The second case of storing random numbers is similar to the case of storing the watermark data in the LSBs of the image.

Sometimes, the PSNR value of the recovered image is intentionally improved by storing a constant value in the deleted LSBs, as in the third case, which imposes a problem when comparing the published PSNR values of different techniques. Therefore, there should be enough awareness about this problem when dealing with the PSNR.

2.8.2- Normalised Correlation Coefficient (NCC)

NCC measures the similarity between two images and is calculated using the following expression:

$$NCC = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N} I_1(m,n)\, I_2(m,n)}{\sqrt{\sum_{m=1}^{M}\sum_{n=1}^{N} I_1(m,n)^2}\sqrt{\sum_{m=1}^{M}\sum_{n=1}^{N} I_2(m,n)^2}} \tag{2.6}$$

where I_1 and I_2 are two images with the same width (M) and height (N).

In general, NCC varies from 1 when the two compared signals are identical, to -1 when one signal is the negative of the other, i.e. The first signal = -1 x The second signal.

2.8.3- Structural SIMilarity Index (SSIM)

SSIM was introduced by Wang and Bovik [43, 44] and it can be expressed as follows:

$$SSIM = \frac{\sigma(I_{1,} I_2)}{\sigma(I_1)\sigma(I_2)} \quad \frac{2\sigma(I_1)\sigma(I_2)}{(\sigma(I_1))^2+(\sigma(I_2))^2} \quad \frac{2\overline{I_1}\,\overline{I_2}}{(\overline{I_1})^2+(\overline{I_2})^2} \tag{2.7}$$

where I_1 and I_2 are two images, $\overline{I_1}$ and $\overline{I_2}$ are their means, $\sigma(I_1)$ and $\sigma(I_2)$ are their standard deviations, and $\sigma(I_1, I_2)$ is their cross covariance. The three terms in equation 2.7 measure the distortion in correlation, contrast, and luminance, respectively.

2.8.4- Tamper Localisation Failure Rate Metrics

The main metrics that are used to measure the failure rate of tamper localisation are: Probability of False Acceptance (PFA), Probability of False Rejection (PFR), and Probability of False Detection (PFD).

PFA, PFR, and PFD will be defined based on the following definitions:

$T_{tampered}$: The total number of the tampered blocks in the image.

$T_{authentic}$: The total number of the authentic blocks in the image.

$T = T_{tampered} + T_{authentic}$: The total number of blocks in the image.

$T_{false\,rejection}$: The number of authentic blocks incorrectly detected as tampered ones.

$T_{false\,acceptance}$: The number of tampered blocks incorrectly detected as authentic ones.

PFA is the probability of classifying a tampered block as an authentic one, and it is defined as follows:

$$PFA = \frac{T_{false\,acceptance}}{T_{tampered}} \tag{2.8}$$

PFR is the probability of classifying an authentic block as a tampered one, and it is defined as follows:

$$PFR = \frac{T_{false\ rejection}}{T_{authentic}} \tag{2.9}$$

PFD is the probability of incorrect detection of a block in the image, and it is defined as follows:

$$PFD = \frac{T_{false\ rejection} + T_{false\ acceptance}}{T} \tag{2.10}$$

The most accurate measurement of failure metrics is when single pixels are considered in the calculations. Therefore, in this research the calculation of failure metrics will be conducted based on single pixels.

To estimate the relationship between the number of authentication bits and PFA, assume that a block has a total of b bits where L bits of them are assigned as authentication bits, as shown in Figure 2.3.

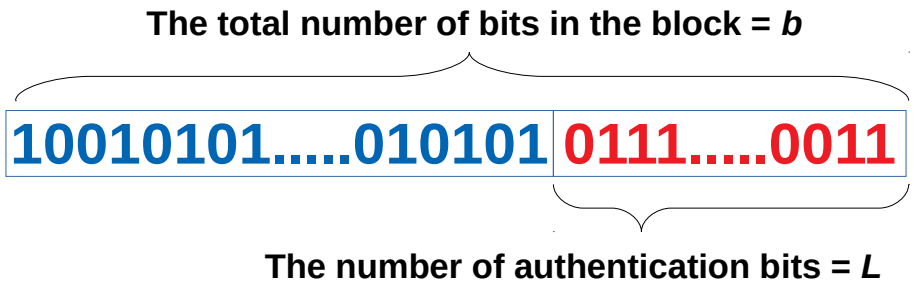


Figure 2.3: An illustration of the authentication bits in a block.

The contents of the block are considered authentic only when the authentication bits are equal to the number generated at the encoding stage, and assuming a uniform random distribution for the possible contents of the block, then the probability of considering the block as an authentic one is $1/(\text{The number of possiblities for } L \text{ bits})$ which is equal to $1/2^L$. However, only one

of the possible contents of the block is authentic, which means that the probability of having an authentic block is $1/(\text{The number of possiblities for } b \text{ bits})$, which is equal to $1/2^b$, therefore:

$$PFA = P(\text{Considering a block as an authentic one}) - P(\text{Having an authentic block}) = \frac{1}{2^L} - \frac{1}{2^b}$$

where $P()$ refers to the probability, or:

$$PFA = 2^{-L} - 2^{-b} \tag{2.11}$$

Since $b > L$, then $2^{-b} \ll 2^{-L}$, which leads to:

$$PFA \approx 2^{-L} \tag{2.12}$$

2.9- Discrete Cosine Transform (DCT) and JPEG Compression Standard

This section is included because DCT and some of the techniques used in JPEG compression standard were used in some of the proposed techniques. For further information, the reader can refer to references [45-47].

2.9.1- Discrete Cosine Transform (DCT)

The DCT is a reversible linear transform that is widely used in lossy compression formats, such as ***Joint Photographic Experts Group*** (JPEG) format. DCT converts a signal from the spatial or time domain into the frequency domain, where the signal is expressed as a sum of cosine waveforms.

If $f(m,n)$ is an MxN matrix that represents the spatial information of an image, then it is transformed using DCT into $F(u,v)$ which is an MxN DCT coefficient matrix that contains the components of $f(m,n)$ along cosine waves that are expressed as $\cos\frac{(2m+1)u\pi}{2M}.\cos\frac{(2n+1)v\pi}{2N}$. The formulas [47] for calculating DCT and inverse DCT coefficients are:

$$F[u,v] = C[u]C[v]\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\cos\frac{(2m+1)u\pi}{2M}\cos\frac{(2n+1)v\pi}{2N}, \quad \begin{cases} 0 \le u \le M-1 \\ 0 \le v \le N-1 \end{cases} \tag{2.13}$$

$$f(m,n)=\sum_{u=1}^{M-1}\sum_{v=0}^{N-1} C[u]C[v]F(u,v)\cos\frac{(2m+1)u\pi}{2M}\cos\frac{(2n+1)v\pi}{2N},\ \begin{cases}0\le m\le M-1\\ 0\le n\le N-1\end{cases} \quad (2.14)$$

where

$$C[u]=\begin{cases}\frac{1}{\sqrt{M}}, & u=0\\ \frac{2}{\sqrt{M}}, & 1\le u\le M-1\end{cases} \quad (2.15)$$

$$C[v]=\begin{cases}\frac{1}{\sqrt{N}}, & v=0\\ \frac{2}{\sqrt{N}}, & 1\le v\le N-1\end{cases} \quad (2.16)$$

2.9.2- JPEG Compression Standard

This section presents a general overview of the JPEG compression standard. Only the main steps will be highlighted without going into detail. Figure 2.4 shows the main steps that are carried out in JPEG, these steps can be summarised as follows:

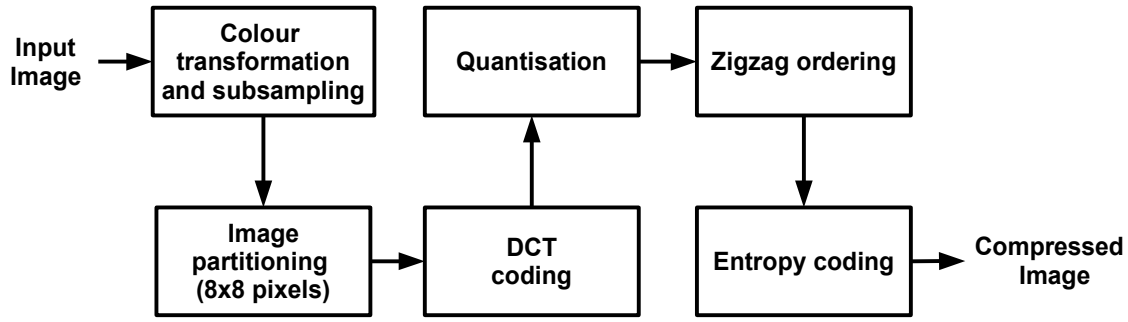


Figure 2.4: Block diagram for JPEG compression

2.9.2.1- Colour Transformation and Subsampling

The colour space of the input image is converted from RGB (Red, Green, Blue) into YCrCb colour space, where: Y is the luminance component, Cr and Cb are the chrominance components.

The human eye [45] has between 75 to 150 million rod photoreceptors, which are sensitive to the luminance component, compared to 6 to 7 million cone

photoreceptors, which are sensitive to the chrominance components. The human eye is less sensitive to the details in chrominance components and that gives the possibility of using subsampling to reduce the amount of information in the chrominance component by grouping adjacent pixels into one pixel (i.e. Cr and Cb components have less resolution than Y component). For example a 4:2:0 subsampling, which is used for high compression in JPEG, groups every 4 pixels in chroma components into one pixel which reduces their data size to one fourth.

2.9.2.2- Image Partitioning and DCT Coding

Each colour component is divided into 8x8 px blocks, then the DCT coefficients of each block are found.

2.9.2.3- Quantisation

Most of the lossy compression is done in the quantisation step, where the number of bits required by each DCT coefficient is reduced. Higher frequency components get fewer bits than lower frequency components because they are less noticeable by the human eye. The quantisation is carried out by dividing the DCT coefficient matrix by a quantisation matrix and rounding the result, a higher number in the quantisation matrix results in a fewer number of bits for a DCT coefficient. The standard quantisation matrices for luminance and chrominance components [47] are shown in Figure 2.5, these matrices correspond to a quality factor of 50.

Luminance quantisation matrix							
16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Chrominance quantisation matrix							
17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Figure 2.5: Standard JPEG quantisation matrices for luminance and chrominance components [47].

2.9.2.4- Zigzag Ordering

The quantised DCT coefficients are transformed into a sequence by reading them

in a zigzag manner [47] as shown in Figure 2.6.

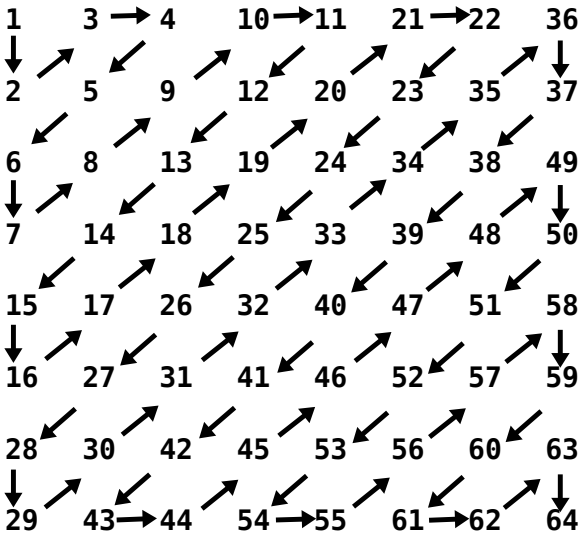


Figure 2.6: Zigzag ordering

2.9.2.5- Entropy Encoding

Lossless entropy encoding is applied to the quantised DCT components. The DC coefficients are encoded using Differential Pulse-Code Modulation (DPCM) (DC coefficient is in the top-left of the DCT matrix and it corresponds to zero frequency), while the AC coefficients (all coefficients except the DC one) are encoded using Run-Length Encoding (RLE). The details of the entropy encoding stage will be skipped because they are beyond the scope of this research.

2.10- Image Filtering in the Spatial Domain

Spatial domain filters [45-47] (also known as spatial masks, kernels, templates, and windows) are divided into linear and non-linear filters. Linear filters use linear operations, such as finding the mean value, and they have correspondence in the frequency domain, while non-linear filters use non-linear operations, such as finding the median value, and they do not have correspondence in the frequency domain. Spatial domain filtering is considered to have more versatility than frequency domain filtering because non-linear filtering can be done in it.

In spatial filtering, a sliding window $w(x,y)$ (also called a mask or a kernel) moves along the image and a linear or a non-linear operation is performed on the image pixels encompassed by this window. The spatial filter produces a new pixel

with coordinates equal to the coordinates of the centre of the window as shown in Figure 2.7.

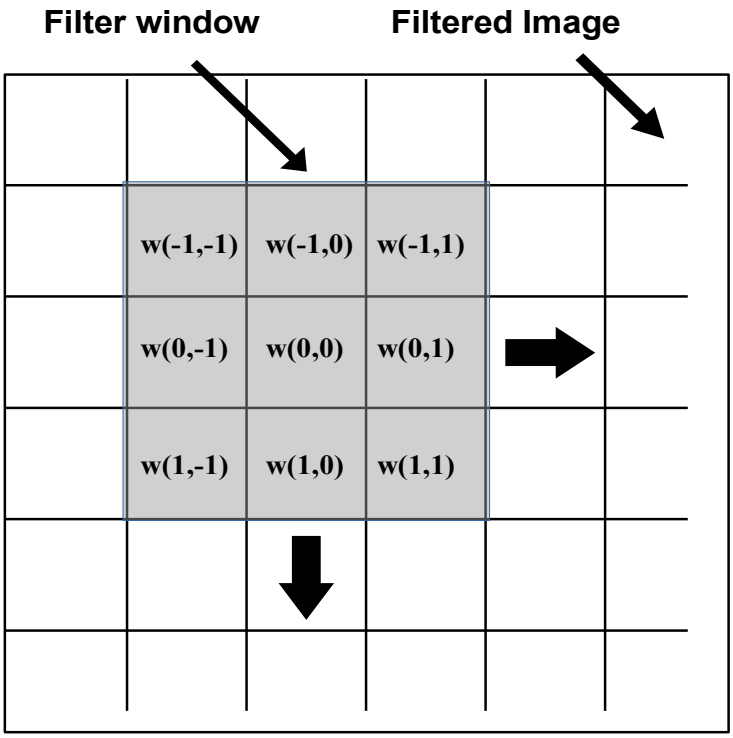


Figure 2.7: Spatial filter sliding window

One example of linear filters is averaging filter, where the outcome of the filter operation is the summation of the pixels encompassed by the window divided by the number of pixels inside the window. A median filter is an example of non-linear filters, where the outcome of the filter operation is the median of the pixels encompassed by the filter window. Any operation could be defined for the filter operation, such as calculating the maximum or the minimum of the encompassed pixels.

2.11- Singular Value Decomposition (SVD)

In Singular Value Decomposition (SVD) [48], an $r \times c$ matrix Z is decomposed into the product of three matrices in the form:

$$Z = U \Sigma V^{*} \tag{2.17}$$

Where:

r and c are the number of rows and columns in the matrix. U is an $r \times r$ unitary

matrix (i.e. its conjugate transpose is also its inverse, or $U U^{*}=U^{*} U=\mathbf{I}$ where $\mathbf{I}$ is the identity matrix). If Z is a real valued matrix, then U is an orthogonal matrix (i.e. its transpose is also its inverse, or $U U^{T}=U^{T} U=\mathbf{I}$). The conjugate transpose of a complex valued matrix is the transpose of the conjugates of its elements.

Σ is a diagonal $r \times c$ matrix with non-negative real numbers on its diagonal, the values in the diagonal of Σ (i.e. $sv_{1,} sv_{2,}$...) are called the Singular Values (SVs).

V is a $c \times c$ unitary matrix. If Z is a real valued matrix, then V is an orthogonal matrix, V^{*} is the conjugate transpose of V .

2.11.1- Geometrical Interpretation of SVD

In the special case when Z is an $r \times r$ real square matrix with a positive determinant: U , V^{*} , and Σ are real $r \times r$ matrices as well. U and V^{*} can be regraded as rotation matrices, while Σ can be regarded as a scaling matrix. Thus the expression $U \Sigma V^{*}$ can be interpreted as a composition of three transformations: a rotation or reflection (U), a scaling (Σ) , and another rotation or reflection (V^{*})

2.11.2- Invariance of Singular Values to Rotation, Transposition, and Flipping

Since singular values in Σ correspond to scaling along the different dimensions, the singular values do not change when rotation, transposition, or flipping are applied to the matrix [49].

2.12- Cyclic Redundancy Check (CRC)

A Cyclic Redundancy Check (CRC) code [50] is commonly used for error detection in digital networks and storage devices, where a fixed-length checksum is generated from a stream of data and this checksum will most probably change if any change occurs to the data stream. The CRC code is attached to the end of the data stream before transmission so that it can be compared to the one generated at the receiving end, or simply the CRC is calculated for the data stream with the CRC attached to it and if the result is 0, then the CRC of the data stream is the same as the attached one and the data stream is assumed to be uncorrupted.

CRC codes are commonly used because they are simple to implement in hardware and easy to analyse mathematically and efficient in detecting errors that are caused by noise. CRC generation functions are considered as hash functions since they produce a fixed-length checksum value.

The basic idea behind CRC is in treating the input data as a long polynomial that is divided using long division by what is called a ***generator polynomial,*** where the remainder of this division is the value of the CRC code. To generate an ***n-bit*** CRC code, a generator polynomial of power ***n*** is needed. This polynomial has ***n+1*** terms, i.e. bits, and the highest and the lowest terms in it must be 1. The polynomial coefficients are calculated according to the arithmetic of ***finite field***, where addition operation can always be performed without carry between digits, i.e. the long division in CRC calculation is carried out using XOR operation. In CRC calculation, the quotient is discarded because it is not needed.

2.12.1- An illustration for Manual Calculation of CRC Code

An illustrative hypothetical example is presented to understand the basic operations in CRC calculation and how to perform it manually. Assume that a 4-bit CRC code is to be generated for the following binary sequence:

S = 10101011

which could be expressed as the following polynomial:

$$\mathbf{S = 1*x^7+0*x^6+1*x^5+0*x^4+1*x^3+0*x^2+1*x^1+1*x^0 = x^7+x^5+x^3+x+1}$$

The binary representation of the generator polynomial is:

G = 11001

which is expressed as the following polynomial:

$$\mathbf{G = x^4+x^3+1}$$

The calculation of an **b-bit** CRC code starts by appending **b** zeros (4 zeros in this example) to the input sequence **S**, therefore **S** becomes **101010110000**. The resulting binary sequence is XORed with **G** as follows:

- **G** is aligned below **S** so that the leftmost 1 in **G** is aligned to the leftmost 1 in **S**, then they are XORed and the remaining digits in **S** are appended to

the result of the XOR operation.

- **G** is XORed again to the result as in the previous step and the process is repeated until the remainder (which is the CRC value) is less than **G**.

The manual calculation for the previous example is illustrated as follows:

```
    101010110000
XOR 11001
---------↓↓↓↓↓↓↓
    011000110000
XOR  11001
----------↓↓↓↓↓↓
     00001110000
XOR      11001
--------------↓↓
         0010100
XOR        11001
---------------
           01101  -> CRC = remainder = 1101
```

2.12.2- Implementation of CRC Using Shift Register

CRC can be implemented using a shift-left register. For n-bit CRC code an n-bit shift register is required. The CRC calculation proceeds as follows:

- The calculation starts by initialising the shift-left register with zeros.
- The leftmost bits of the input binary sequence, including the appended zeros, are fed into the register until a one pops from the left side of the register.
- The contents of the register are then XORed with the rightmost **n** bits of the generator polynomial, the leftmost bit is excluded because its XOR result is always 0.
- The remaining bits of the input binary sequence continue to be fed into the register until a one pops out from the left side of the register and the XOR operation in the previous step is repeated until all the bits in the input sequence are fed into the register, then the contents of the register will be

the value of the CRC code.

Figure 2.8 illustrates the calculation of the CRC for the previous example.

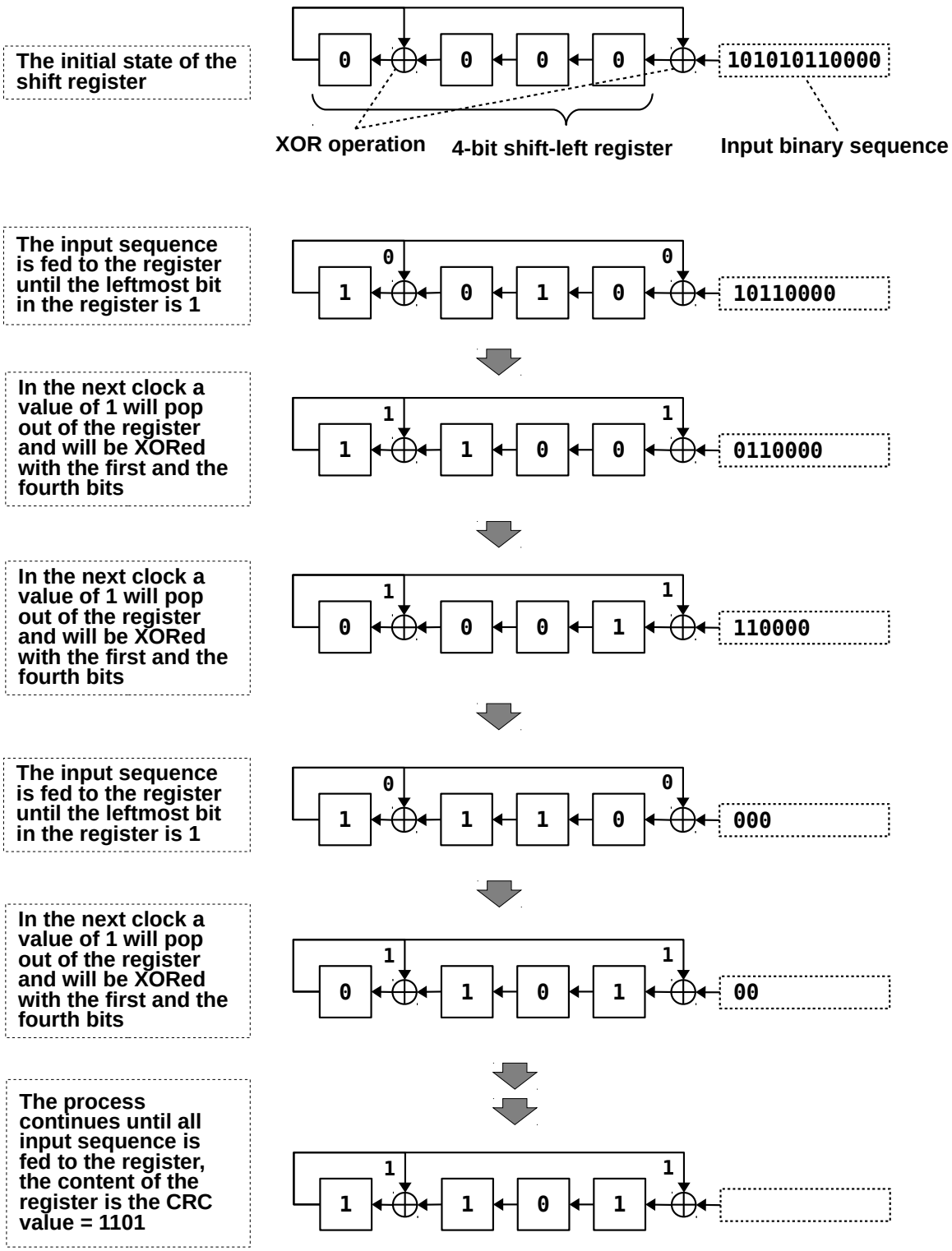


Figure 2.8: An illustration of using a shift register for CRC calculation.

2.12.3- Using a Look-Up-Table for CRC Calculation

CRC function is linear and therefore CRC calculation can be performed by

applying it to the separate parts of the input sequence. To illustrate, let the input number **S** be broken into a sequence of half bytes or nibbles **S1, S2, ..., Sk** (1 nibble = 4 bits), then:

CRC(S1,S2) = CRC(S2 XOR CRC(S1))

CRC(S1,S2,S3) = CRC(S3 XOR CRC(S1,S2))

....

CRC(S1,S2, ..., Sk) = CRC(Sk XOR CRC(S1, S2, ..., S(k-1))

In another word, the CRC of **S** is found by finding the CRC of the first nibble then XORing the result with the second nibble and find the CRC of the result and so on until the CRC of the last nibble in **S** is found.

The following example shows how to do it manually for the previous example:

```
input sequence = 10101011 -> S1 = 1010, S2 = 1011

To find CRC for S1 = 1010

    10100000
XOR 11001
---------↓↓↓
    01101000
XOR  11001
----------↓↓
     0001100 -> CRC = 1100

XOR CRC(S1) with S2 -> 1100 XOR 1011 = 0111

to find CRC for 0111

    01110000
XOR  11001
----------↓↓
    00010100
XOR    11001
------------
     0001101 -> CRC = 1101
```

The calculation of CRC can be sped up by storing the CRC of each nibble in a

look-up-table instead of calculating it each time, this will consume more memory but will increase the speed of calculation dramatically.

The principles that were presented in the previous examples are applied for CRC with larger block size. For example for CRC16-CCITT (from French: *Comité Consultatif International Téléphonique et Télégraphique* [51]) standard, the CRC is 16 bit wide, with a polynomial $\mathbf{G = x^{16}+x^{12}+x^5+1}$ [50], a look-up-table of 256 16-bit long elements could be used to find the CRC value for blocks of 1 byte long.

2.12.4- Initial Value, XOR Output, Reflected Input, and Reflected Output

The actual implementation of CRC includes some pre and post processing stages. The first one is the initial value of the CRC that is XORed with the start of the input sequence, this initial value is important to solve the problem of leading zeros in the input sequence, which have no effect on the value of the CRC regardless of their length, unless the initial value of the CRC is different from 0, usually it is initialised to all ones. Regarding the XOR output stage, sometimes the final value of the CRC is XORed with a number, usually all ones. The other two stages are reflect input and reflect output stages which determine whether the input bytes to the encoder and the final CRC output will be reflected. In reflection, the bit positions are flipped so that the higher bits become lower and vice versa.

A CRC calculator for different CRC standards along with their specification can be found in [52], and a good tutorial for CRC calculation can be found in [50]. Appendix D.4 has a GNU Octave implementation for CRC16 where the parameters can be changed to find the CRC for different standards.

2.13- Summary

Some aspects of digital watermarking were presented in this chapter, such as digital watermarking definition, applications, classifications, types of attacks, chaotic maps, general steps for tamper localisation and recovery, as well as the evaluation parameters used in tamper localisation and recovery techniques. Some topics that are mentioned in this research were also introduced, such as: Discrete Cosine Transform (DCT), JPEG compression standard, Image filtering in spatial domain, Singular Value Decomposition (SVD), and Cyclic Redundancy Check

(CRC). The overview in this chapter was intended to be general and simplified without going into details, in order to give the reader some introductory information to make the understanding of the following chapters easier.

Chapter 3: Literature Survey and Analysis for Tamper Localisation and Recovery Techniques

3.1- Introduction

This chapter presents a literature survey for fragile and semi-fragile watermarking techniques that perform not only localisation but both localisation and recovery. The survey will highlight the main properties of these techniques, such as number of bits used for localisation, number of bits used for recovery, methods of localisation-data generation, methods of recovery-data generation, localisation-data mapping, and recovery-data mapping. The papers in this survey are presented according to the year of publication and in an ascending order. The last section in this chapter presents some analysis of the techniques in the literature.

3.2- Fragile Tamper Localisation and Recovery Techniques

Due to their higher capacity and sensitivity to tampering, fragile watermarking techniques are the most used techniques in the literature for tamper localisation and recovery, this section presents a literature survey about the most important and recent fragile tamper localisation and recovery techniques.

The first work in tamper recovery was proposed by Fridrich and Goljan [53] in 1999. In their technique the image is divided into 8x8 px blocks and the recovery data consists of the DCT transform of each block. The DCT coefficients were quantised using JPEG quantisation matrix that corresponds to a quality factor of 50%. After quantisation, 1 or 2 LSBs could be used to store 64 or 128 bits of the resulting DCT coefficients. The quality factor is determined by the matrix used in the quantisation of the DCT coefficients, as was discussed in Section 2.9.2.3. No localisation bits were used in this method and the recovery bits were mapped into another block that is at a distance approximately equals 1/3 of image size and the direction of mapping is randomly chosen, the detection is done by back tracing any

tampered block. Even though the technique is simple and gives good recovery quality with only 1 or 2 LSBs being used, but the recovery area is small, not exceeding 1/3 of image dimensions, and the direction of embedding could be detected, which makes the method vulnerable to attacks.

A hierarchical localisation and recovery method is presented by Lin et al. [54] in which three levels of tamper localisation are applied. The image is divided into 4x4 px blocks and each block is further divided into 2x2 px sub-blocks, each sub-block has 2 bits assigned for tamper localisation and 6 bits assigned for recovery, which means that 2 LSBs are assigned for the watermark. The first localisation bit is set to one if the average of the 4x4 px block is larger or equal to average of the 2x2 px sub-block, the second localisation bit is the odd parity of the average of the 2x2 px block, the third detection level depends on the inspection of 3x3 block-neighbourhood. Recovery bits represents the average of 6 MSBs of the 2x2 px block. Localisation bits are stored in the same corresponding block while recovery bits are stored in another block determined by a mapping method that depends on a secret key. The mapping method used does not ensure enough distance between the tampered block and its recovery data; therefore, some blocks will not be recovered even when the tampered area is small, e.g. 25%.

Haouzia and Noumeir [24] presented a survey about image authentication techniques such as: cryptography, fragile and semi-fragile watermarking, and digital signatures. One major disadvantage of digital signature based techniques is their inability of tamper localisation. Authentication techniques were divided into strict and selective authentication, in strict authentication techniques no image processing is tolerated, while in selective techniques some image processing, such as lossy compression, is permitted.

Noriega et al. [55] proposed a method that uses two different watermarking techniques, one semi-fragile watermark is used for authentication and the other is a fragile one and consists of an approximation of the original image (referred to as the digest image). This approximation is generated using DCT and re-compressed using arithmetic codes, then redundancy is added using Bose-Chaudhuri-Hochquenghem (BCH) code to detect errors. Both watermarks are embedded in the Integer Wavelet Transform (IWT) domain. Secret keys, ***K1*** and ***K2***, are used to generate the first watermark and to permute, i.e. shuffle, the second one.

Successful recovery is shown for tampering of 10% of the image and noise insertion of about 5%; however, using compression codes and error correction codes adds more complexity to the algorithm. The recovery capability in this method is limited to the correction ratio of the BCH code.

Zhang et al. [56] used a hybrid block-wise and pixel-wise approach. The localization is done on 8x8 px blocks and the recovery is carried out in a pixel-by-pixel manner within these blocks. Three LSBs are reserved for the watermark, the recovery data is 160 bits per block and it is calculated by performing XOR operation between five MSBs in two different blocks which gives 320 bits that are mapped into two different blocks throughout the image. The remaining 32 bits of the watermark in each block are used for localisation. The quality of the recovered image degrades dramatically as the tampered area increases because the recovery data is distributed throughout the image in a random manner.

In their paper, Zhang et al. [57] proposed two techniques, the first one relies on a ***reference sharing mechanism*** where the image is divided into 8x8 px blocks and 3 LSBs are reserved for the watermark, 160 bits are reserved for recovery data, and 32 bits are reserved for localisation data which consists of the hash of 5 MSBs and the recovery data in that block. The recovery bits are generated by permuting 5 MSBs from the whole image according to a secret key and dividing them into groups with *E* elements in each group, then *E/2* recovery bits are generated by multiplying the group with a random matrix of size (*E/2)x(E),* and by using modulo-2 summation this will produce linear system of *E/2* equations. The tampering will be reflected as unknown values in this system and Gaussian elimination could be used to find these unknowns as long as the system solvable. The recovery in the proposed technique is probabilistic and works for small areas, up to 1/3 of image size in the best case [57], and it works better for large *E,* such as *E* >1024, which makes the technique require more time for computation.

Zhang et al. [58] proposed a flexible-quality recovery technique that claims avoiding the tampering coincidence and the watermark-data waste problems. In their technique the image is divided into 8x8 px blocks and 3 LSBs are set to 0 in order to store the watermark in them, which gives a total of 192 bits. 161 bits are used to store the recovery data, which is generated by quantising 23 DCT coefficients into 7 bits for each one. The recovery data is randomly distributed

throughout the image. The localisation bits consist of 31 bits hash function that is generated for the 5 MSBs and the recovery data from each 8x8 px block. The localisation data is stored in the same block from which it is generated. Compositive reconstruction and compressive sensing are used to estimate the contents of the tampered blocks if their recovery data is missing due to tampering. The proposed method suffers from high complexity and low recovery quality at high tamper ratios.

Dadkhah et al. [59] modified the LSB hierarchical method proposed by Chaluvadi and Prasad [60] in which the image is divided into 2x2 px blocks and a 12 bit watermark is stored in 3 LSBs of each block. The watermark consists of 5 bits for the average of a particular 2x2 px block and 5 bits for the average of another block, which means that there are two copies of the average value of each block distributed throughout the image, and two bits are left for parity check. The parity check was considered as the first level for integrity check, which might lead to a false decision since different pixel values could have the same parity.

Hisham et al. [61] presented a fragile watermarking technique for tamper localisation and recovery for the Holy Quran text images, the presented technique is based on the method proposed by Zain and Fauzi [62]. The mapping between the original block and the watermark block is carried out using the spiral numbering proposed by Afifah and Jasni [63]. Each 8x8 px block B is divided into 4x4 px blocks B_s and the recovery data consists of the average of each B_s block. The authentication of each block is done by calculating the parity of the B_s blocks and by comparing the average value of the B_s blocks to the original block. Some disadvantages of this method are the low resolution of the recovered image and the complicated method used for authentication, also the spiral mapping does not work for non-square images.

Tong et al. [64] divided the image into 2x2 px blocks and 3 LSBs are reserved for the watermark, that gives 12 bits for the watermark in each block. Two bits are used as authentication bits and 10 bits contain the recovery data which consists of two copies of the average of 5 MSBs of each block. The authentication bits are stored in the same block while the copies of the recovery bits are mapped in two different blocks using a chaotic map. The method suffers from high PFA because of using only two bits for authentication and that results in a low recovery quality.

In [65] Korus and Dziech present a model for the recovery problem that is inspired by the work in [57]. In their work, the tamper recovery problem is modelled as an erasure communication channel, and they used the Random Linear Fountain (RLF) codes [66] to recover the tampered part of the image. The model does not impose any criteria for the generation of localisation and recovery data; however, in their paper the image was partitioned into 8x8 px blocks and 3 LSBs were reserved for the watermark. 32 bits of the watermark are reserved for authentication using hashing function and 160 bits are reserved for recovery using the DCT transform of 5 MSB of the each block. Their technique gives a quality of 37 dB with a tamper ratio that reaches 50% of the image. The main disadvantage of this technique is its high computational complexity due to the use of fountain coding that is applied to the whole image. Another disadvantage of the proposed technique is the probabilistic behaviour of fountain codes. This behaviour is due to the random selection of the blocks that will be included in the linear system that represents the watermark. That means there is a probability that the generated linear system might not be solvable for some variables, i.e. some tampered blocks might not be recovered.

Nyeem [67] proposed a standard model for image watermarking and he defined a set of watermarking properties according to that model, also a set of expected attack models were developed. Self-authentication schemes have been analysed in order to detect and recover any possible alteration or tampering of medical images, also a watermarking embedding scheme and watermarking capacity control models were developed.

In another thesis, Jassim [68] implemented blind watermarking using one-level and two-level DWT, mobile number with international code was used as watermarking data. Fragile watermarking and a combined fragile and robust watermarking were implemented too. The main disadvantage of the proposed robust technique is its vulnerability against geometrical attacks such as rotation and cropping, due to the lost of synchronisation of the watermark data. The copyrighted data is restricted to mobile number, where more general types of data should be investigated. The main disadvantage of the proposed fragile authentication technique is the lack of ability to recover the tampered area.

Dadkhah et al. [69] divided the image into 4x4 px blocks and each block is further

divided into 2x2 px sub-blocks, two LSBs in each block were reserved for the watermark which gives 8 bits per each sub-block. Singular Value Decomposition (SVD) is used to generate the authentication data, which is stored into 3 bits, and 5 bits are assigned for the recovery data which are the average value of 5 MSBs the sub-block. The recovery bits are also used in the authentication process which improves the localisation rate and robustness against attacks. A random mapping is used for recovery bits, but the upper part of the image is mapped to lower part of the image and vice versa. The use of SVD and hierarchical block division increases the complexity of the algorithm without introducing noticeable efficiency and the mapping of recovery blocks is not efficient and suffers from tamper coincidence problem.

Tareef et al. [70] presented a tamper localisation and recovery technique for medical images. The electronic patient record and the Region Of Interest (ROI) are sparsely coded and embedded in the Region Of Non-Interest (RONI). If the image is tampered, then the ROI could be extracted from the embedded watermark in the RONI. Sparse coding (SC) and SVD are used in the proposed technique to increase robustness and reduce perceptibility of the proposed technique.

Eswaraiah and Sreenivasa [71] proposed a technique for medical images where the embedding of the recovery information of the ROI is done in the RONI. The ROI is divided into non overlapping 3x3 px blocks and the recovery data is generated from the grey level of the central pixel (8 bits) and the difference between it and each surrounding pixel with 6 bits for each pixel. The RONI is divided into 8x8 px blocks and each block is decomposed using wavelet transform and the recovery data is stored in the 2 LSB of LH1, HL1, LH2 and HL2 levels. One problem with this method is that it assumes the difference between adjacent pixels will not exceed 6 bits, which might not be the case. Another problem is when the middle pixel is changed, this will affect all surrounding pixels which makes the method more sensitive to errors and noise. The value of the grey level after performing inverse wavelet transform (IWT) might exceed 255 or become less than 0, this was solved by performing a check on the possibly affected blocks in the RONI and change their grey level prior to embedding, this will add more complexity and encoding time to the method and leads to a degradation in the image quality.

Singh et al. [72] proposed a method that uses 2x2 px blocks with 3 LSBs reserved for the watermark, DCT was used to generate recovery data which takes 10 bits in each block and 2 bits were reserved for authentication. The authentication bits are stored in the same block and recovery bits are mapped randomly using a secret key. The method shows good recovery quality; however, using only two bits for authentication and storing the authentication bits in the same block make this method vulnerable to attacks such as VQ and collage attacks and increase the PFA.

Sreenivas and Prasad [26] presented a survey about fragile watermarking techniques where they highlighted the main schemes used in fragile watermarking especially self-embedding schemes. The survey presented the main issues and problems faced when using self-embedding schemes and some of the work that has been done to solve these issues.

A comprehensive survey paper in the field of image authentication and integrity was presented by Korus [27]. Image authentication approaches were categorised into four main categories, which are: (a) Digital signatures (b) Authentication watermarks (c) Forensic analysis (d) Phylogeny reconstruction. The paper highlighted that the definition of image authenticity is more general and encompasses image integrity by ensuring the truthfulness of the presented image and taking into account using an unaltered image in an incorrect context, such as taking an image in a different time than the original one. The paper addressed active and passive approaches used in image authentication from different aspects such as (a) Analysis capabilities (b) Fundamental limitations (c) Documented vulnerabilities (d) Maturity and availability of software tools. The paper also described some recently proposed alternative approaches to image authentication that do not directly fall into any of the discussed classes. It also reviewed resources available in the research community such as publicly available datasets and software tools and it concluded with a discussion in the open problems and future research perspectives.

In [73] Sreenivas and Kamakshiprasad proposed a localisation technique that enhances the technique proposed by Chang et al. in [74]. The proposed technique utilises logistic chaotic map [75] and Arnold's cat map [37, 38] in the generation of the localisation data. The technique divides the image into 2x2 px blocks and uses

3 LSBs for watermark embedding. Based on their localisation technique, Sreenivas and Kamakshiprasad also proposed a tamper recovery technique, their technique divides the image into 2x2 px blocks and uses 3 LSBs for watermark embedding. In their technique, 4 bits are used for localisation and 8 bits are used for recovery. Two recovery sets are used, one has 5 bits and represents the mean of 5 MSBs of the block, the other is an approximation of the mean, which was previously calculated, and it is stored in 3 bits. The localisation bits are stored in the same block and the two recovery sets are stored in different blocks. Using two recovery sets enables the recovery of a larger tampered area; however, the quality of recovery is low considering that 3 LSBs are reserved for the watermark.

Qin et al. [76] used a different approach for image partitioning where the image is divided into 3x3 px overlapping blocks that are overlapped in 1 pixel in each direction. Recovery data is generated using the average of 6 MSBs of each block. The number of LSBs used for embedding the recovery data is 2 LSB in 4 pixels at the corners of each block and 1 LSBs in the pixels at the sides of each block. The number of LSBs used for localisation data embedding is 2, 3, or 4 LSBs that are stored in the centre pixel of each block and is generated depending on the complexity of the block. Inspection of the neighbouring blocks is also used for localisation where a block is considered as a tampered one if it is surrounded by 5 or more tampered blocks. Depending on the position of each pixel in each block, a set of equations describes the value of that pixel depending on its neighbouring blocks which enables a pixel-wise recovery of the tampered blocks. The proposed method has high level of complexity and since the blocks are overlapped this might lead to a reduced recovery capability especially when the tampered area is spread all over the image as in the case of adding noise to the image.

Qin et al. [77] proposed a technique that relies on VQ and index sharing, where a VQ codebook is constructed for the image and each non-overlapping block is represented by VQ index bits. The encoding process is carried out by dividing the image into non-overlapping 8x8 px blocks and reserving 1, 2, or 3 LSBs in each block for the watermark. 32 bits of the first LSB are reserved for the authentication bits, which are generated by finding a hash for the contents of the block along with the recovery bits. Each 8x8 px block is further divided into 4x4 px sub-blocks and the recovery bits are found for each sub-block using a compressed VQ

representation of the image. The recovery bits are permuted randomly in the image and a level of redundancy is applied for the recovery data in order to ensure high recoverable area ratio. One advantage of the proposed technique is the ability of recovering large tampered area, up to 80% of the image size. The proposed technique has flexible recovery quality, which depends on the number of LSBs reserved for the watermark and the size of the VQ codebook. The recovery quality ranges from 37 dB to 28 dB for tamper ratios that range from 10% to 80% respectively.

In the technique proposed by Shehab et al. [78], the watermark is stored in 2 LSBs, and the image is divided into 4x4 px blocks. The authentication bits are generated using SVD and they are stored in 12 bits, which leaves 20 bits for the recovery bits. The 4x4 px block is further divided into 2x2 px sub-blocks and 5 recovery bits are found for each one, the recovery bits represent 5 bits of the average value of each sub-block. Arnold chaotic mapping is used to distribute the recovery bits randomly throughout the image. The parameters controlling Arnold chaotic mapping are chosen based on a secret key. The proposed technique was tested against some attacks, such as: copy and paste attack, text addition, content removal, and VQ attacks. The proposed method suffers from low recovery quality because of the random mapping and the low number of recovery bits.

Haghighi et al. [79] proposed a technique that generates 2 recovery data sets based on lifting wavelet and halftoning techniques. The recovery data is mapped randomly based on Arnold Cat Map. The image is partitioned into 2x2 px blocks and 2 LSBs are reserved for the watermark. LSB rounding mechanism is proposed to enhance the recovery quality, and shift-aside operation is proposed to enhance the recovery rate. The technique suffer from low recovery quality and high complexity; however, having 2 recovery data sets increases the recoverable tampered area size.

Hemida et al. [80] proposed a tamper localisation and recovery technique that is targeting natural and text images. The authentication is done using 4 bits for 4x4 px blocks, while the recovery is based on 2x2 px blocks. The DCT of the 2x2 px block is used to generate the recovery data, which has variable length from 6 to 10 bits based on the nature of the bock contents. More recovery bits are assigned to the blocks containing text, and less are assigned to the ones containing natural

contents. The recovery data is XORed with a secret key to enhance the security of the technique. The recovery data is mapped randomly based on a secret key. A multi-stage neighbour detection strategy is used to improve the performance of tamper detection. In general, the proposed technique has low computational time, but it suffers from low PFA value and the variable length encoding of the recovery data requires extra bits to determine the length of the recovery data, which increases the watermark-data waste problem.

Niu et al. [81] investigated reducing the computational complexity of referencing sharing mechanism used in some tamper localisation and recovery techniques. In their paper, they reviewed the main literature where reference sharing mechanism was used. The main advantage of reference sharing mechanism is solving the tamper-coincidence problem and the vulnerability of recovery to tampering pattern. Reference sharing relies on converting the recovery information into a linear system and solving this system for any loss in recovery information. The tampering appears as unknown variables in the linear system. The main problem with this scheme is the high computational time required to solve the linear system, and this time increases dramatically as the size of the image increases. Niu et al. build their work based on the idea that if the encoded recovery matrix is more sparse, then the required computational time will be reduced. The experimental results presented in their paper shows an improvement in the computational time while maintaining the same restoration capability.

3.3- Semi-Fragile Tamper Localisation and Recovery Techniques

A few semi-fragile tamper recovery techniques have been proposed in the literature [82], this is due to their limited capacity and low sensitivity to tampering, this section reviews the most important techniques presented in the literature.

In [83] Lin and Chang proposed a design for Self-Authentication-and-Recovery Image (SARI) watermarking system based on their semi-fragile watermarking technique that was proposed in [84] which is considered as the first work using semi-fragile techniques for tamper localisation and recovery. Their system is compatible with JPEG image format and can detect malicious tampering while accepting the changes introduced in the image due to JPEG lossy compression.

Lin and Chang mentioned that SARI is based on two invariant properties of quantization-based lossy compression. The first property shows that if a transform-domain (such as DCT in JPEG) coefficient is modified to an integral multiple of a quantization step, which is larger than the steps used in later JPEG compressions, then this coefficient can be exactly reconstructed after later JPEG compression. The second one is the invariant relationships between two coefficients in a block pair before and after JPEG compression. In SARI, the second property is used to generate authentication signature, and the first one is used for watermark embedding. The proposed technique suffers from small-size and low quality of the recovered area.

Yafei et al. [85] proposed a semi-fragile technique for recovering missed blocks in block coded image formats, such as JPEG, by embedding the DCT coefficients of a low-quality version of the image into the LSBs of the DCT coefficients of the original image. The number of the generated DCT coefficients for each block depends on how much detail it contains where 7 coefficients are used for smooth blocks and 15 coefficients are used for high detail blocks.

Zhu et al. [86] proposed a semi-fragile tamper localisation and recovery watermarking technique in which the recovery problem is formulated as an irregular sampling problem, the recovery is performed through iterative projections onto convex sets. The proposed technique suffers from small recoverable area and the quality of recovery decreases as the tampered area increases.

Hasan and Hassan in [87] and Cruz et al. in [88] proposed some techniques that generate recovery information for Region of Interest (ROI) and the embedding is carried out in the remaining region, i.e. Region of Embedding (ROE). However, these techniques suffer from limited recovery area and can not be used for general purposes where all parts of the image are equally important.

Cheddad et al. [89] proposed a technique based on a steganography approach to protect scanned documents from forgery. In their technique the original image is converted into a halftone image and embedded into the first-level 2D Haar DWT. The technique suffers from low quality of the recovered area due to using half-tone mechanism. Another halftone-based technique was also proposed by Mendoza-Noriega et al. [90] where the halftone image is embedded into middle-frequency

DCT coefficients of the original image and the inverse halftoning process is carried out using Multi-Layer Perception neural network (MLP). The technique suffers from high computational complexity and low quality of the recovered image.

In [82] Korus et al. proposed a technique that aims at providing a practical recovery capability for lossy compressed JPEG images, their technique is based on the technique proposed in [65, 91] but with far less computation complexity, the reduced encoding and decoding time enables using their technique for high resolution images and on mobile devices. The reduction of computation time comes from dividing the image into small sub-images and applying Digital Fountain Codes (DFC) on them instead of the whole image. An improved model for calculating the probability of successful restoration is introduced also in this paper. In their proposed technique the damaged parts of the watermark are discarded and do not contribute to the recovery process which results in a constant recovery quality, 4 quality levels were defined from low fidelity with average PSNR of 28 dB to a high fidelity with average PSNR of 33 dB, the claimed achievable tampering rate vary between 67% and 20% depending on the selected quality level. In the encoding process, the image is compressed using JPEG compression with a quality factor Q_1 and the resulting JPEG image is used to generate the recovery bits which are grouped as units of macro-blocks of 16x16 px where each block has 4 sub-blocks of 8x8 px. The number of watermark bits corresponding to each macro block is $4\lambda\Lambda+2L$, where Λ is the number of recovery bits for each 8x8 block, L is the number of localisation, or hash, bits for the macro-block, and λ is the fidelity level. The generated recovery bits are encoded using RLF codes, then localisation bits appended to them, the resulting stream is scrambled and embedded in the coefficients of the JPEG image. In order for the technique to work, any further compression after encoding should have a compression quality that is higher than Q_1.

In [92] Chen et al. proposed a technique in which the recovery data consists of 5 bits for DC coefficient and 6 bits for 2 AC coefficients from the DCT coefficients of 8x8 px block after downsampling it to 4x4 px block. Recovery bits are used for both localisation and recovery purposes, and the DC and AC values in the watermark are embedded in 7 middle frequency DCT coefficients of two different blocks in order to increase robustness against attacks, the embedding blocks are

selected according to a secret key. Multi-neighbour characteristic and multi-threshold optimization are used in the localisation stage to improve the performance. The proposed technique suffers from severe distortion and low quality of the recovered area.

Semi-fragile techniques for tamper localisation and recovery are also proposed in the papers [93-95]

3.4- Analysis and Shortcomings of the Techniques Proposed in the Literature

In this section, analysis is carried out for the different stages of tamper localisation and recovery process and some shortcomings are highlighted. The analysis will not go deep into the different methods proposed in the literature, instead it will try to highlight the main approaches that could be used and the advantages and disadvantages of each one.

3.4.1- Using Separate Sets of Bits for Localisation and Recovery

The main shortcoming that has been found in the survey is the use of two separate sets of bits, one for the localisation and another for the recovery of the tampered area. The disadvantages of this can be summarised as follows:

- Reducing the capacity of the watermarking method, because both sets are related to the values of the pixels in each block and both sets can contribute to localisation and recovery processes, but since each set is used for one purpose only, that results in a redundancy which decreases the watermarking capacity.
- Increasing the complexity of watermarking: Because the localisation and the recovery are done as separate processes instead of doing them as a single process with a single set of bits.
- Reducing the localisation efficiency: Because a limited number of bits is used for localisation instead of using all of the available bits in the watermark and that increases the probability that some tampered pixels might go undetected, which increases the PFA. The PFA can be estimated

by equation 3.1, where L is the number of localisation, i.e. authentication, bits. For example, if 2 bits are used for localisation then 25% of the tampered blocks will be detected as valid ones, which leads to reduced recovery quality because these blocks will not be recovered.

$$PFA \approx 2^{-L} \tag{3.1}$$

- Increasing the ability of counterfeiting: Because when a limited number of bits is used in localisation, that will result in a larger number of blocks that can generate the same localisation bits and hence more ease of counterfeiting.

3.4.2- Multiple Copies of the Recovery Data

It has been noticed that some techniques use multiple copies of the recovery data to increase the size of the recoverable area and to overcome the problem of tamper coincidence; however, this increases the watermark-data waste problem.

3.4.3- Localisation Bit Mapping

Another shortcoming that has been noticed is that the localisation bits are stored in the same block they are generated from. This is done to maximise the localisation rate by simply considering the missing localisation bits in each block as an indication of the tampering of that block. Storing the localisation bits in the same block will make each block independent from the others and it could easily be counterfeited using VQ and collage attacks.

3.4.4- Generation of Localisation and Recovery Bits

The generation of localisation bits could take one of two distinct approaches, the first one is when localisation is equally sensitive to the change in any bit in the watermarked block and the best choice for that approach is to use a form of parity check or hash function. The other approach is to use localisation bits that are related to the energy in the watermarked block and the best choice for this approach is to use average value or DCT.

The problem with the first approach is that it gives the intruder more ability to replace the original block with another one that is visually very different but yet it produces the same localisation bits, which means more ability to counterfeit the

original block, meanwhile in the second approach the intruder's choices are more limited. Since the number of localisation bits is less than the number of bits in each block, a perfect localisation scheme is impossible and deliberately counterfeited blocks could go undetected.

3.4.5- Image Partitioning

The partitioning of the input image could be done in a pixel-wise or a block-wise manner. Pixel-wise methods have a better localisation rate whilst block-wise methods give more flexibility, especially if the watermark generation depends on a block-wise operation such as DCT or SVD. Nested blocking is used in some techniques; however, this increases the complexity of the technique.

3.4.6- Recovery Bit Mapping

It is very important to store the recovery bits of a certain block in another one that is separated from it by enough distance, which is determined by a mapping criteria. If the recovery bits are stored close to the block from which they are generated, this will result in a tamper coincidence problem.

There are two distinct approaches to map the recovery bits, the first one is to use a random distance that depends on a secret key. This will make the watermarking process more secure and the counterfeiting more difficult; however, the probability of tampering coincidence is higher in this method because some recovery bits will be close to the original block they are supposed to recover.

The other approach to ensure a ***maximum-distance*** in each direction between the recovery bits and the block from which they are generated, and this distance is half the size of the image.

In maximum-distance mapping [69] for MxN image, block $B_{x,y}$ is mapped to the block $B_{x+\Delta x,\, y+\Delta y}$ where:

$$\Delta x = \begin{cases} M/2, & x \le M/2 \\ -M/2, & x > M/2 \end{cases} \quad , \quad \Delta y = \begin{cases} N/2, & y \le N/2 \\ -N/2, & y > N/2 \end{cases} \tag{3.2}$$

The main disadvantage of the maximum-distance approach is that the position of the recovery bits is known to the intruder, which gives more ability of counterfeiting. One solution to this problem is to permute the order of the bits in

each pixel using a secret key, which could be considered as a permutation in the z direction, this will enhance the security of the watermarking technique whilst maintaining maximum-distance condition.

Another disadvantage of maximum-distance approach is its sensitivity to tampering pattern where the most suitable tampering patterns are vertical tampering that does not exceed half the width of the image or horizontal tampering that does not exceed half the height of the image, but any tampering that exceeds half of both the width and the height of the image will result in tampering coincidence.

3.4.7- Watermark Security

It is very important to encrypt the embedded watermark to prevent any intruder from knowing it. Using common encryption algorithms such as Advanced Encryption Standard (AES) [96] will cause error spread in the watermark [97] and can not be used for small watermark data size. Therefore, most methods use simple XOR operation with a random sequence to encrypt the watermark.

Using XOR operation imposes a security problem if the exact position of the watermark content is known, which is the case when maximum distance mapping is used, because if the intruder knows how to generate the watermark and its exact position then the random sequence will easily be known by comparing the generated watermark with the embedded one.

Some level of randomisation of the position of recovery bits must be ensured when XOR operation is used for encryption, this randomisation could be done for the position of the bits in watermark pixels, this solution will enhance the security of the watermarking whist maintaining maximum distance condition.

3.4.8- How to Resist VQ and Collage Attacks

The watermarked image could be attacked in different ways; however, the most difficult attacks to deal with are VQ and collage attacks, since they rely on replacing parts of the watermarked image with others that are taken from images that use the same secret key.

If each block of the watermark is encoded separately, then it could be replaced with any block from other image that uses the same secret key. Therefore, inter-

block dependency is required to prevent VQ and collage attacks.

In the case of tamper recovery, the intruder should insert the counterfeited pixels in both the image and the watermark in order for the tampering to be successful. By randomising the position of the watermark bits, the watermarking technique will be more immune against VQ and collage attacks since the exact position of the recovery bits is not known by the intruder.

One solution for VQ and collage attacks is to make the encryption of the image dependent on the number of each block [65] and on a unique serial number for each image, which makes each block distinct and can not be replaced by another from the same image or another image.

3.5- Summary

A literature survey for tamper localisation and recovery watermarking techniques was conducted in this chapter, fragile and semi-fragile watermarking techniques were surveyed, even though this research is intended for fragile ones. The survey focused on the techniques that perform not only localisation but both localisation and recovery. The main technical aspects of the surveyed techniques were highlighted, namely: the partitioning scheme, the number of bits used for localisation and recovery, the methods used for localisation and recovery data generation, mapping methods, recovered area quality and size, and the techniques advantages and shortcomings.

Varies methods have been implemented for each aspect of tamper localisation and recovery. Image partitioning is usually done using non-overlapping blocks and the block size depends mainly on the method used for recovery data generation, such as using averaging with 2x2 px blocks or DCT with 8x8 px blocks. Other partitioning schemes have been proposed in the literature, such as hierarchical and overlapping partitioning. Some techniques rely on having more than one copy of the recovery data to increase the recoverable area ratio and reducing tamper coincidence probability; however, that increases the watermark-data waste problem. Most of the techniques rely on storing the recovery data directly into the destination blocks whilst a fewer techniques store the recovery data as a linear system of equations, where the lost blocks are represented as unknown variables

and the system is solved to find these variables. Storing recovery data as a linear system helps in overcoming tamper coincidence problem and sensitivity to tampering pattern; however, it increases the decoding time significantly. The localisation data is usually stored in the block from which it is generated. Random or maximum-distance mapping is used to map the recovery data. Semi-fragile tamper recovery techniques suffers mainly from small recoverable area and low recovery quality.

The surveyed techniques were analysed in this chapter and some of the shortcomings were highlighted, such as: using two different sets of data for localisation and recovery, using multiple copies of the recovery data, and vulnerability against attacks and counterfeiting. The disadvantage of mapping techniques were also highlighted, such as higher tamper coincidence in random mapping and vulnerability to counterfeiting in maximum-distance mapping. Some solutions have been proposed, such as using bit permutation inside each pixel for maximum-distance mapping.

Chapter 4: The Description of the Proposed and the Referenced Techniques

4.1- Introduction

In this chapter, the challenge of using one set of data for tamper localisation and recovery is highlighted and the encoding/decoding stages of the proposed techniques are presented. This chapter also presents the description and the shortcomings of the two techniques in literature to which the proposed techniques will be compared, these two techniques are referred to as the ***referenced techniques***.

4.2- The Challenge of Using One Set of Data for Tamper Localisation and Recovery

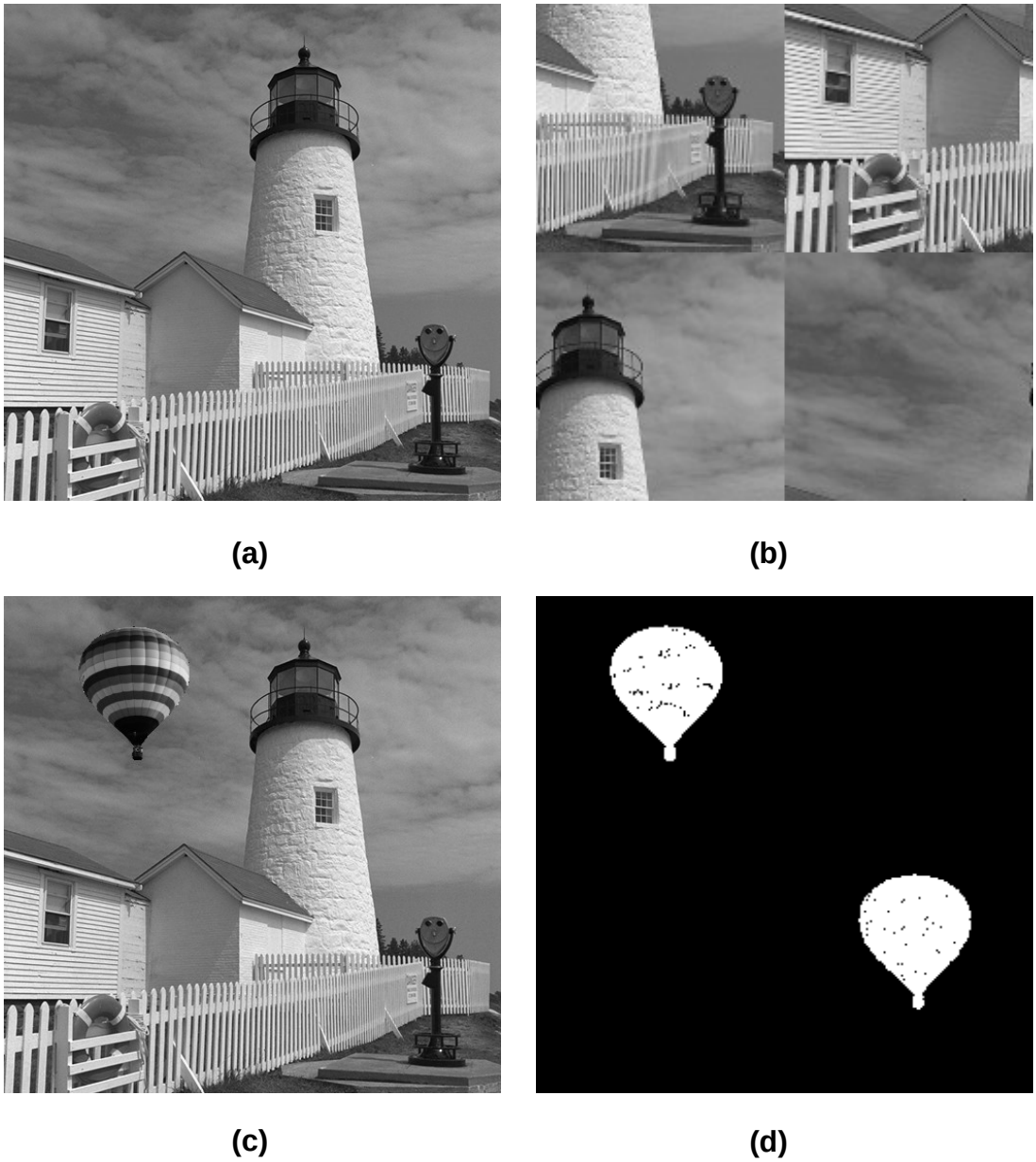
Two separate sets of data are used in most tamper localisation and recovery techniques, one set is used for localisation and it is usually embedded in the block it is related to, the other set is used for recovery and it is usually embedded in a block as far as possible from its related one. It has been noticed that the recovery data represents a low resolution version of the watermarked image and any tamper in the watermarked image could be localised with direct comparison between the watermark and the watermarked image. Therefore, the use of extra localisation data can be avoided and the recovery data only can be used for localisation and recovery.

The main problem that arises when using direct comparison between the watermark and the watermarked image is that the tampering will affect the watermark and watermarked image in two different places and when comparing them together there will be two regions that are different, one region coincides with the tampered area in the watermarked image while the other does not; therefore, it will not be possible to directly distinguish the tampered area.

The example shown in Figure 4.1 illustrates this problem. The original ***lighthouse*** image [30] is shown in Figure 4.1 (a) and the watermark is mapped using

maximum-distance mapping as shown in Figure 4.1 (b). The watermarked image is tampered by inserting an image of a hot air balloon as shown in Figure 4.1 (c). The inserted image will affect both the watermark and the watermarked image in the top-left corner but when the watermark is remapped to its original position, the tampering in it will be in the bottom-right corner. The watermark and the watermarked image are compared using XOR operation and a threshold is applied to the result of comparison so that any different pixels between the two images will have a white colour as shown in Figure 4.1 (d). From Figure 4.1 (d) it is noticed that there are two areas in the difference image, one coincides with the tampered area in the watermarked image while the other does not, this will impose a problem of distinguishing the tampered area.

The first three techniques proposed in this research overcome this problem and use the recovery data for the localisation and the recovery of the tampered area, which gives them more capacity than most of the proposed techniques in the literature, also they have the characteristic of employing image filtering in the localisation process. Each technique uses different criteria for the mapping of the localisation/recovery data, which requires appropriate filtering and localisation procedures for each technique.



(a) **(b)**

(c) **(d)**

Figure 4.1: An illustration of how the tampered area appears when the watermarked image is compared with the watermark. a) The original image. b) The watermark with maximum-distance mapping. c) The tampered image. d) The difference between the watermarked image and watermark. (Lighthouse image taken from [30])

4.3- Notes About the Selected Block Size of the Proposed Techniques

This section presents the reasons behind selecting 2x2 px blocks for the first, the second, and the third technique, and choosing 8x8 px for the fourth one.

The selected block size depends mainly on: the method used for generating the recovery data, the available number of bits provided by that block size, and the

number of deleted LSBs.

For the first, the second, and the third techniques, a block averaging is chosen to generate the recovery data for each block. As block size increases, the quality of the recovered area decreases and pixelation artefacts start to appear. For that reason, the block size needs to be as small as possible, preferably 2x2 px, as long as the available number of bits in the block is sufficient to store the generated watermark data.

The available number of bits inside a block depends on the number of deleted LSBs. The available number of bits when 2 or 3 LSBs are deleted is 8 or 12 bits respectively, which is sufficient to store the average value of the remaining MSBs in the block. Authentication data and/or extra recovery data could be stored in the remain space. Many researchers [54, 59, 64, 69, 72, 73] have chosen 2x2 px block size when averaging is used to generate the recovery data.

When 1 LSB is reserved for the watermark, only 4 bits will be available for recovery data, in this case 4 MSBs of the average value could be used to fit in the available space, as was done in the first proposed technique. Increasing the block size to 4x4 px will provide 16 bits for watermark; however, this number is more than needed and the quality of the recovered image will be worse than using 2x2 px block with only 4 MSBs.

To verify the previous statement, the 24 images in Kodak image database are converted to grey-scale and the average PSNR is calculated for two cases. The first one is when the images are converted into 2x2 px averaging blocks and 4 LSBs set to 0. The second case is when they are converted into 4x4 px averaging blocks with 1 LSB set to 0. The average PSNR is 23.6 dB for the first case and 21.1 dB for the second case, which confirm the that using 2x2 px blocks with 4 MSBs of the average is better than using 4x4 px block with 7 MSBs of the average.

For the fourth technique, the DCT is used to generate the recovery data. In this case a larger block size will be better for approximation, at the same time it will result in less precise localisation; therefore, a compromise is needed. A block size of 8x8 px is selected because it provides a good compromise and it has been used by many researchers [53, 56-58, 65]. A block size of 8x8 px is also used in JPEG

compression.

4.4- The Description of the Encoding and the Decoding Stages of the First Technique

The localisation/recovery bits in this technique are distributed in a random manner throughout the image. This makes the technique more resistive to VQ and collage attacks and less sensitive to the tampering pattern; however, the tampering coincidence in this technique is very high, which makes the quality of the recovered image very low. In this technique the image is divided into 2x2 px blocks and one LSBs is reserved for the watermark, which gives 4 bits in each block to store the watermark data that consists of 4 MSBs of the mean value of the block.

When comparing the watermark with the watermarked image, the tampered area will appear as a dense white area while the untampered area will appear as a dark noisy area. The nature of the noise is similar to salt and pepper noise. Median filtering is used in the localisation of the tampered area and in the enhancement of the quality of the recovered image.

This technique is suitable for applications where the number of sacrificed bits in the image should be as low as possible and the quality of the recovered area is not a priority.

The implemented code for the encoding and the decoding stages of the first proposed technique is provided in Appendix A.

4.4.1- The Encoding Stage of the First Technique

A general block diagram for the encoding stage is shown in Figure 4.2.

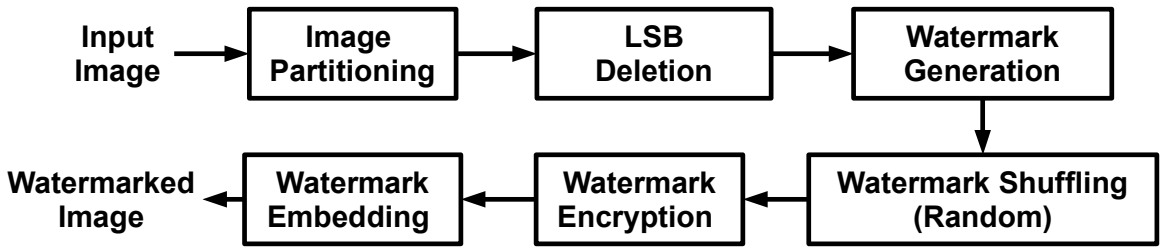


Figure 4.2: A block diagram for the encoding stage of the first proposed technique

The steps for the encoding stage are described as follows:

- **Image Partitioning:** The input image is divided into non-overlapping 2x2 px blocks. The dimensions of the input image must be multiples of 2.
- **LSB Deletion:** 1 LSB in the image is set to 0, i.e. deleted, in order to store the watermark in it.
- **Watermark Generation:** The watermark is generated by taking 4 bits of the average of each block. The watermark has half the width and half the height of the input image.
- **Watermark Shuffling:** Each row in the watermark image is rotated, i.e. circularly shifted, a random horizontal distance, then each column of the resulting image is rotated a random vertical distance. A secret key ***K*** is used as a seed for the generation of the random numbers used in the rotation process. The whole rotation process of the rows and the columns could be repeated to ensure a better shuffling for the watermark pixels. In the proposed technique the rotation is repeated once.
- **Watermark Encryption:** For securing the watermark, it is XORed with a random sequence. The secret key ***K*** is used as a seed for generating the random sequence. The whole watermark is XORed with a random sequence, which means that the random sequence for the individual blocks is not the same, which means the inability of using a block from the same image to replace another. A block from another image that is encrypted with same secret key can be used to replace a block from the current image as long as they have the same position. Therefore, it is necessary to assign a unique serial number for each encoded image and to make the generation of the random sequence dependant on the secret key and the serial number of the image.
- **Watermark Embedding:** The number of pixels in the watermark is the same as the number of 2x2 px blocks in the watermarked image. 4 MSBs of each pixel in the watermark are embedded in the LSB of the block that coincides with that pixel. The bits from the same pixel in the watermark are kept close to each other in order to give them a better chance of survival from tampering.

4.4.2- The Decoding Stage of the First Technique

A general block diagram for the decoding stage is shown in Figure 4.3.

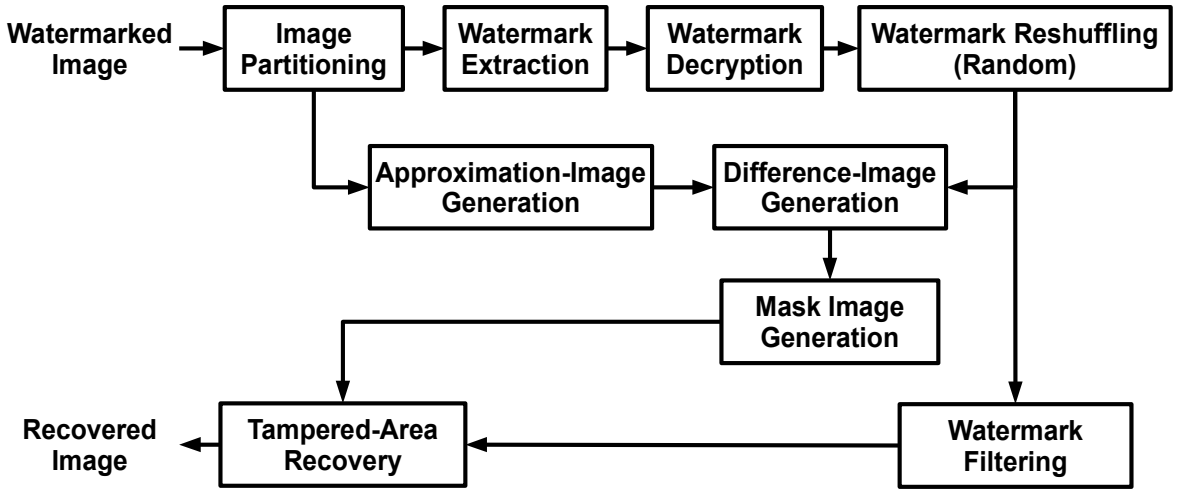


Figure 4.3: A block diagram for the decoding stage of the first proposed technique.

The lighthouse image in Figure 4.1 (a) will be used to illustrate the decoding stage. The image is tampered with by inserting an image of a hot-air balloon, as in Figure 4.1 (c). The steps for the decoding stage are described as follows:

- **Image Partitioning:** The input watermarked image is divided into non-overlapping 2x2 px blocks.
- **Watermark Extraction:** The pixels of the watermark are extracted from the LSB of each block.
- **Watermark Decryption:** The watermark is XORed with the random sequence generated in the encoding stage, this will return the pixels to their original values.
- **Watermark Reshuffling:** The columns and the rows of the watermark are rotated as in the encoding stage but with a reversed order and reversed direction, this will return the watermark pixels to their original positions, then 4 LSBs in the watermark are set to 0. The resulting image is referred to as the ***watermark image***. See Figure 4.4

Figure 4.4: An illustration of the watermark image in the decoding stage of the first technique

- **Approximation Image Generation:** An approximation image is generated from the watermarked image by setting 4 LSBs in each block to 0 and finding the block average, then 4 LSBs of the resulting image are set to 0 to match the watermark image. The resulting image is referred to as the ***approximation image***.

- **Difference Image Generation:** The watermark image is XORed with the approximation image, a threshold is then applied to the result so that each pixel with a value more than 0 will have a value of 255. The resulting image is referred to as the ***difference image***. Due to watermark shuffling, the tampering will be scattered throughout the watermark, and when it is compared with the approximation image, the tampered area will appear as a dense white area while the untampered area will appear as a dark area with noise that is similar to salt and pepper noise. See Figure 4.5.

Figure 4.5: An illustration of the difference image in the decoding stage of the first technique

- **Mask Image Generation:** The mask image determines the tampered area in the watermarked image and it is generated by applying a non-linear filter to the difference image. The non-linear filter returns 255 if the ratio of white pixels inside the window is above a threshold Γ, and it returns 0 otherwise. The lighthouse image is used to illustrate the best value of Γ, where it was tampered by erasing a central squared area. The ratio of the erased area varies from 10% to 50%. The ratio of white pixels was calculated for the tampered and the untampered areas. The results are shown in Table 4.1 which shows that the ratio of white pixels is always greater than 90% for the tampered area and less than 50% for the untampered area; therefore, 50%< Γ <90%. The value 70% was selected because it was found to give good results. The selection of the filter window size needs some optimisation because it must be large enough for the calculation of the ratio to be statistically accepted; however, a large block size will result in missing some pixels at the boundaries of the tampered area. A window size of 7x7 px was selected because it was found to give good results, as will be illustrated in Section 4.4.3.

Some tampered pixels at the boundaries of the tampered area are missed due to the use of the non-linear filter. To solve this problem, the localised area is enlarged by applying an averaging filter, then a threshold is applied so that any pixel with a grey level more than 0 will become 255. A window size of 5x5 px was selected because it was found to give good results. The resulting image from this step is referred to as the ***mask image***. See Figure 4.6.

Table 4.1: The ratio of white pixels in the difference image for the tampered and the untampered areas

Tamper ratio	10%	20%	30%	40%	50%
Ratio of white pixels in the tampered area	99.3%	98.5%	98.0%	97.4%	96.6%
Ratio of white pixels in the untampered area	9.3%	18.9%	28.5%	37.7%	46.7%

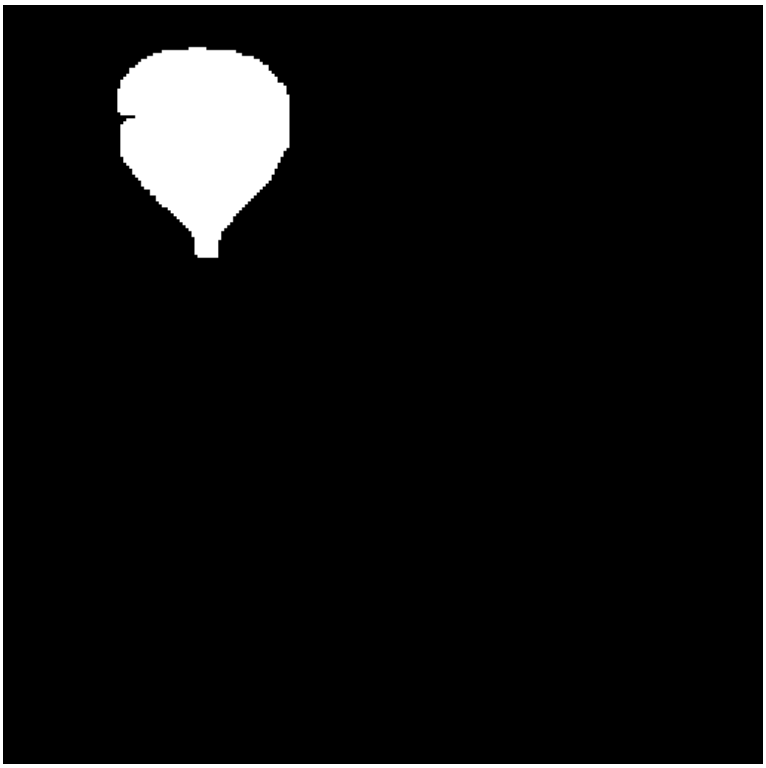


Figure 4.6: An illustration of the mask image in the decoding stage of the first technique

- **Watermark Filtering:** The watermark image is used to recover the tampered area; however, the noise in the watermark needs to be reduced. A median filter is used for this purpose because the noise is similar to salt

and pepper noise. A median filter with window size of 3x3 px is selected because it was found to give good results. The quality of the recovery depends on the percentage of the tampered area and the efficiency of the filter used for noise removal.

- **Tampered Area Recovery:** Finally, the filtered watermark image and the mask image are scaled up by a factor of 2 to match the watermarked image size. The tampered area is replaced by the corresponding area from the filtered watermark image. The location of the tampered area is determined by the mask image. Figure 4.7 shows the recovered image. The recovered area is recognisable in the image because of the use of median filtering and because only 4 bits are assigned for each pixel in the watermark.

Figure 4.7: An illustration of the recovered image in the decoding stage of first proposed technique.

4.4.3- Validation of the Selected Parameters in the Decoding Stage of the First Technique

This section presents a validation test for the selected values of the parameters in the decoding stage of the first technique, namely: the threshold value of the white pixels in the localisation window, the localisation filter window size, the averaging

filter window size, and the median filter window size.

The test starts by converting the 24 images in the Kodak database to grey-scale and cropping them to 512x512 px. Each of the 24 images is encoded using the first technique, then it is tampered with by inserting the content of the next image into it according to the white pixels in the pattern shown in Figure 4.8. The last image is tampered with using the first one. The percentage of white pixels in pattern is 10%, which represents the tampering ratio. Each image is then recovered and the average PSNR, PFA, and PFR are calculated, the PFA and the PFR are calculated according to equations 5.1 and 5.2. The recovery process is repeated 4 times, in each time one parameter is varied while the others are fixed. The variation of the parameters is shown in Tables 4.2, 4.3, 4.4, and 4.5. The fixed values for the parameters are as follows: threshold = 70%, localisation window size = 7x7 px, averaging filter window size = 5x5 px, and median filter window size = 3x3 px.

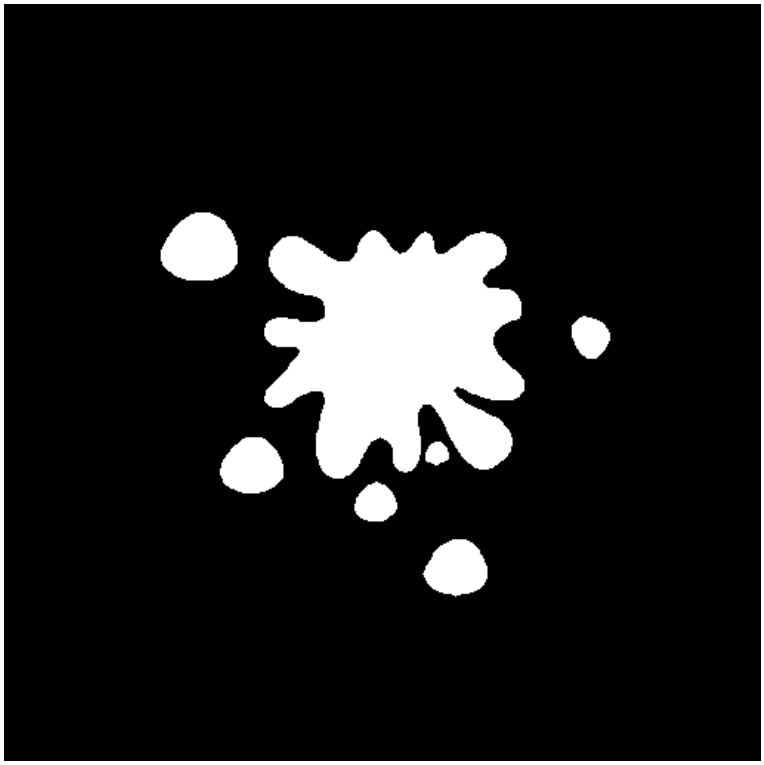


Figure 4.8: The pattern used in the tampering of Kodak database images which are used in parameter validation test of the first technique.

Table 4.2: The average value of PSNR, PFA, and PFR when varying the threshold value in the first technique.

Threshold %	average PSNR (dB)	average PFA %	average PFR %
50%	31.53	0.26%	4.49%
60%	31.81	0.74%	3.44%
70%	32.08	1.84%	2.39%
80%	32.12	4.80%	1.37%

Table 4.3: The average value of PSNR, PFA, and PFR when varying the localisation filter window size in the first technique.

Localisation filter window Size	average PSNR (dB)	average PFA %	average PFR %
3x3 px	31.75	0.45%	3.69%
5x5 px	31.90	1.07%	3.10%
7x5 px	32.08	1.84%	2.39%
9x9 px	32.03	3.02%	1.77%
11x11 px	31.39	5.20%	1.24%

Table 4.4: The average value of PSNR, PFA, and PFR when varying the averaging filter window size in the first technique.

Averaging filter Window Size	average PSNR (dB)	average PFA %	average PFR %
3x3 px	32.26	4.02%	0.92%
5x5 px	32.08	1.84%	2.39%
7x7 px	31.64	1.03%	4.00%
9x9 px	31.23	0.65%	5.60%

Table 4.5: The average value of PSNR, PFA, and PFR when varying the median filter window size in the first technique.

Median filter Window Size	average PSNR (dB)	average PFA %	average PFR %
3x3 px	32.08	1.84%	2.39%
5x5 px	31.03	1.84%	2.39%
7x7 px	30.35	1.84%	2.39%
9x9 px	29.91	1.84%	2.39%

It can be seen in Table 4.2 that as the threshold value increases, the PSNR and the PFA values increase while the PFR values decrease. Since higher values of PSNR and lower values of PFA and PFR are required, a compromise is needed; therefore, a threshold value of 70% is selected. A compromise is also needed with the localisation filter windows size and the averaging filter window size; therefore, the selected window sizes are 7x7 px and 5x5 px for the localisation filter and averaging filter respectively. It can be seen from Table 4.5 that the average PFA and PFR are not affected by the median window size because they are already determined by the mask image; therefore, a window size of 3x3 px is selected because it provides the best value of PSNR.

4.5- The Description of the Encoding and the Decoding Stages of the Second Technique

The second technique divides the watermark pixels into three groups and maps each group based on maximum distance mapping. The mapping is carried out in three directions which are: the horizontal, the vertical, and the diagonal direction. This technique has higher recovery quality and better localisation accuracy when compared to the first one; however, it requires that the tampering is within a rectangular area that has half the width and half the height of the watermarked image, otherwise tampering coincidence occurs and recovery quality degrades dramatically. The technique divides the image into 2x2 px blocks and uses 2 LSBs to store the watermark, which gives 8 bits in each block for the watermark, 6 MSBs are used to store the average of 6 MSBs of the block and the remaining 2 LSBs in watermark are used to store the average of the deleted 2 LSBs in the block, which increases the recovery quality.

The implemented code for the encoding and the decoding stages of the second proposed technique is provided in Appendix B.

4.5.1- The Encoding Stage of the Second Technique

A general block diagram for the encoding stage is shown in Figure 4.9.

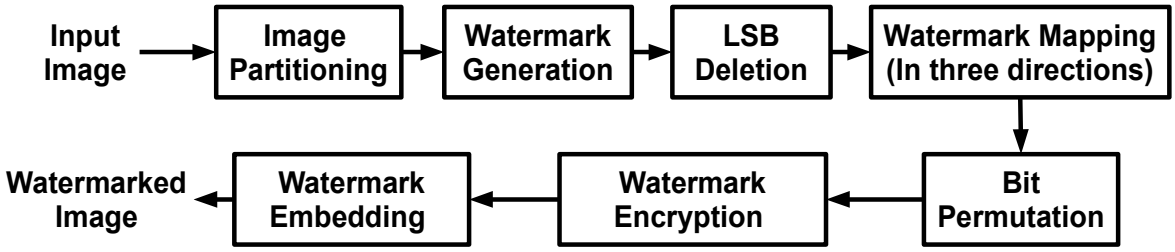


Figure 4.9: A block diagram for the encoding stage of the second proposed technique.

The steps for the encoding stage are described as follows:

- **Image Partitioning:** The input image is divided into non-overlapping 2x2 px blocks. The dimensions of the input image must be multiples of 12.
- **Watermark Generation:** The average of 6 MSBs of each block is calculated and stored in 6 MSBs in the watermark, this average will be used in tamper localisation/recovery process. The average of 2 LSBs of each block is calculated and is stored in 2 LSBs of the watermark, this average will be used to partially recover the deleted 2 LSBs in the input image. The watermark has half the width and half the height of the input image.
- **LSB Deletion:** 2 LSBs in the input image are set to 0 in order to store the watermark in them.
- **Watermark Mapping:** The watermark is divided into 3x3 px blocks, the pixels in each block are divided into 3 groups as shown in Figure 4.10 (a), the groups are mapped based on maximum-distance mapping. Group 1, 2, and 3 will be mapped horizontally, vertically, and diagonally, as shown shown in Figure 4.10 (b). The groups are selected so that no group has two pixels that are adjacent horizontally or vertically, this will enable the use of non-linear filter to localise the tampered area.

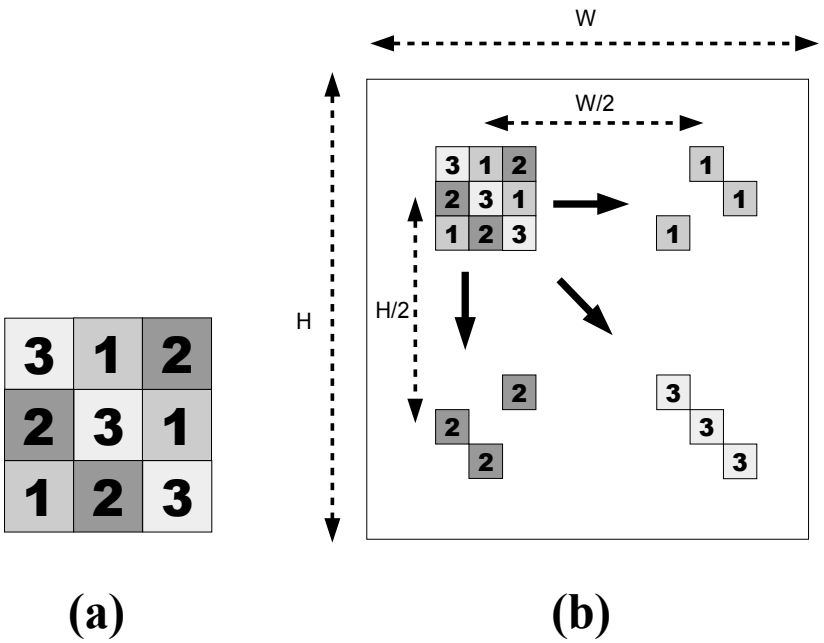


Figure 4.10: The groups of pixels in each 3x3 px block and their mapping. a) The distribution of each group. b) The mapping of each group.

- **Bit Permutation:** The bits inside each individual pixel in the watermark are permuted randomly using the secret key ***K*** as a seed, this will prevent knowing the random sequence used in securing the watermark even though maximum-distance mapping is used and the position of the watermark pixels is known, as discussed in section 3.4.6.
- **Watermark Encryption:** The watermark is encrypted as done in the first proposed technique by XORing it with a random sequence.
- **Watermark Embedding:** The number of pixels in the watermark is the same as the number of 2x2 px blocks in the watermarked image. The watermark is embedded in the watermarked image by storing each pixel in 2 LSBs of the block that coincides with that pixel.

4.5.2- The Decoding Stage of the Second Technique

A general block diagram for the decoding stage is shown in Figure 4.11.

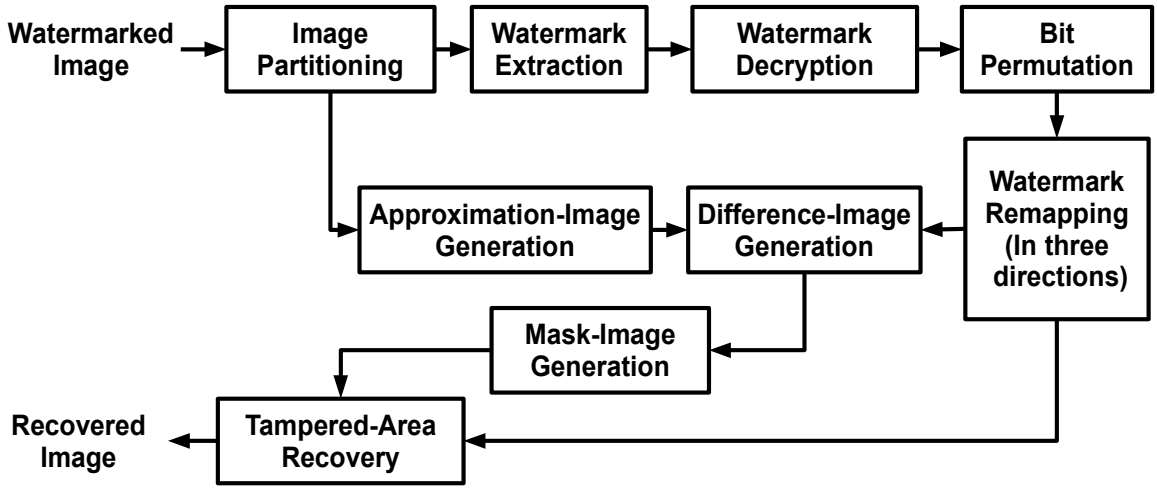


Figure 4.11: A block diagram for the decoding stage of the second proposed technique.

The lighthouse image in Figure 4.1 (a) will be used to illustrate the decoding stage. The image is tampered with by inserting an image of a hot-air balloon, as in Figure 4.1 (c). The steps for the decoding stage are described as follows:

- **Image Partitioning:** The input watermarked image is divided into non-overlapping 2x2 px blocks.
- **Watermark Extraction:** The pixels of the watermark are extracted from 2 LSBs of each block. The watermark has half the width and half the height of the watermarked image.
- **Watermark Decryption:** Similar to the first proposed technique, the watermark is XORed with the random sequence generated in the encoding stage.
- **Bit Permutation:** The bits of the pixels of the watermark are permuted as in the encoding stage, this will return the bits to their original positions.
- **Watermark Remapping:** The pixels of the watermark are mapped in three directions as in the encoding stage, this will return the pixels of the watermark to their original positions. At the end of this stage, the watermark represents a scaled-down version of the original image and the tampering will be scattered in three directions as shown in Figure 4.12. The resulting image is referred to as the ***watermark image***.

Figure 4.12: An illustration of the watermark image in the decoding stage of the second proposed technique. The tampering is scattered in three directions.

- **Approximation Image Generation:** An approximation image is generated from the watermarked image by setting 2 LSBs in each block to 0 and finding the block average. The resulting image is referred to as the ***approximation image***.
- **Difference Image Generation:** The watermark image is XORed with the approximation image, then 2 LSBs in the resulting image are set to 0 because the comparison depends only on the 6 MSBs of the two images. A threshold is then applied to the result so that any pixels value greater than 0 will become 255. The resulting image is referred to as the ***difference image,*** as shown in Figure 4.13. In the difference image, the tampered area is almost completely white and there are three copies of it, each one of these copies has at most 3 white pixels in any 3x3 px block, these areas are referred to as ***ghost areas***.

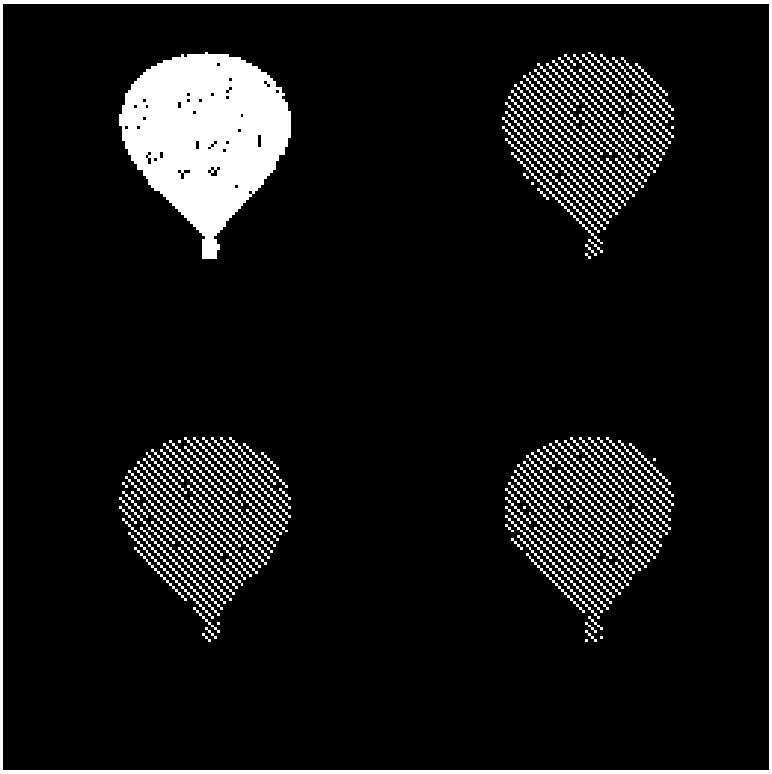


Figure 4.13: An illustration of the difference image in the decoding stage of the second proposed technique.

- **Mask Image Generation:** To generate the mask image, the ghost areas are removed by applying a non-linear filter with a window size of 3x3 px. If the window has more than 3 white pixels, then the central pixel is replaced with 255, otherwise, it is replaced with 0. The resulting image of this step is referred to as the ***mask image.*** See Figure 4.14.

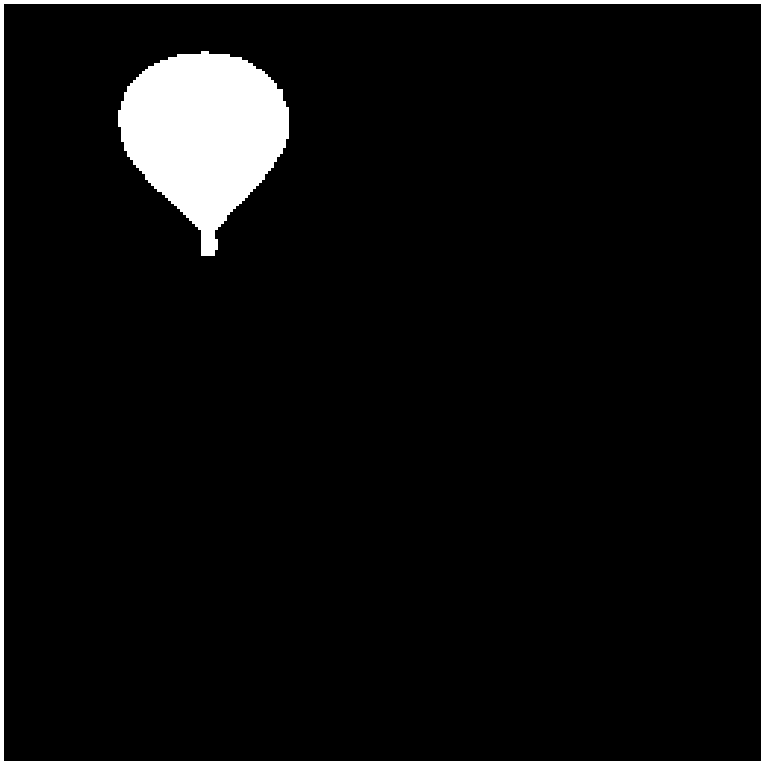


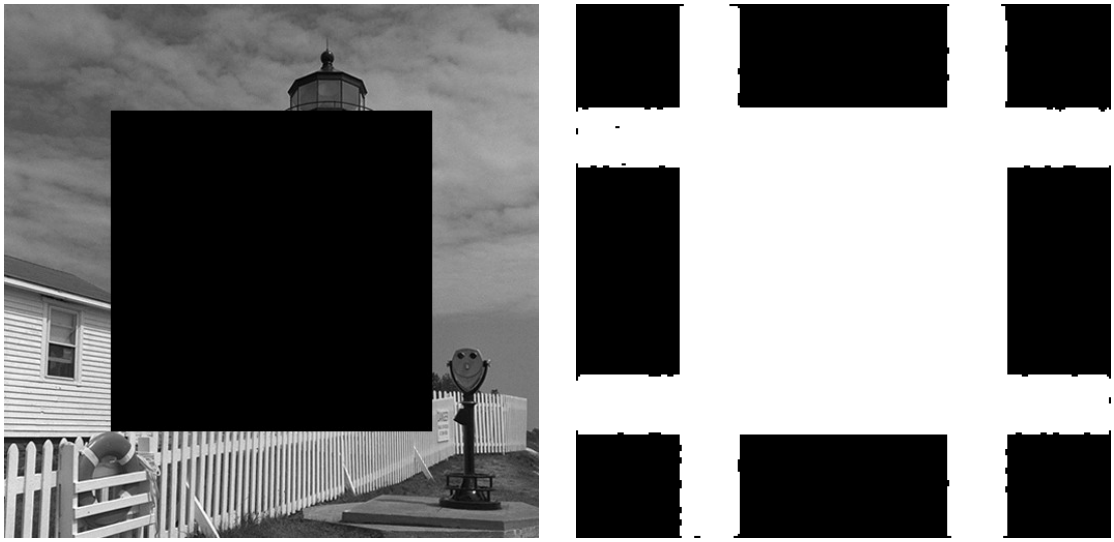
Figure 4.14: An illustration of the mask image in the decoding stage of second proposed technique.

- **Tampered Area Recovery:** Similar to the first proposed technique, the watermark image and the mask image are scaled up by a factor of 2 and the tampered area is replaced by the corresponding area from the watermark image. Also, the lost 2 LSBs in the untampered area in the watermarked image will be partially recovered by the 2 LSBs from the watermark image. Figure 4.15 shows the recovered image.

Figure 4.15: An illustration of the recovered image in the decoding stage of second proposed technique.

For the second proposed technique, the maximum recoverable area without tamper-coincidence is when the tampering is within a rectangular area that has half the width and half the height of the watermarked image, i.e. 25% tampering ratio, otherwise tamper-coincidence will occur. In Figure 4.16 (a), the lighthouse image is tampered with by deleting a central squared area that has 60% ratio of the image width and height. Figure 4.16 (b) shows the detected tampered area and Figure 4.16 (c) shows the recovered image.

The technique also suffers from high PFR when the tampering ratio is equal to or higher than 25%, as shown in Figure 4.16 (b) where large untampered area has been considered as a tampered area.



(a) **(b)**

(c)

Figure 4.16: An example of the second technique. a) The tampered area, where its width and height are 60% of the image dimensions. b) The mask image, which shows the detected tampered area. c) The recovered image.

4.6- The Description of the Encoding and the Decoding Stages of the Third Technique

The third technique employs the fact that the watermark is XORed with a random sequence at the encoding stage, and when the image is tampered with, the tampered area in the watermark will be the only area that is not XORed with the random sequence. When the watermark is XORed again with the random sequence at the decoding stage, all the pixels in the watermark will be XORed twice and hence they will return to their original values, except for the tampered

area which will be XORed only once and this will result in high-frequency contents (HFCs) in the tampered area of the watermark, these contents could be measured and used to localise the tampered area.

Maximum-distance mapping is used in the third technique, which will increase the recoverable area ratio to 50% of the watermarked image instead of 25% when compared to the second technique. Tampering coincidence will occur if the tampered area exceeds half the width and half the height of the watermarked image simultaneously, but it will not occur if the tampered area exceeds half of one dimension only.

As in the second technique, the third technique divides the image into 2x2 px blocks and uses 2 LSBs to store the watermark data; therefore, 8 bits will be available for the watermark in each block. The technique also uses 6 MSBs in the watermark to store the localisation/recovery data which is 6 MSBs of the average of the block. The remaining 2 LSBs in the watermark are used to store the average of the deleted 2 LSBs in the original image, this will partially recover them and will increase the recovery quality.

The implemented code for the encoding and the decoding stages of the third proposed technique is provided in Appendix C.

4.6.1- The Encoding Stage of the Third Technique

A general block diagram for the encoding stage is shown in Figure 4.17.

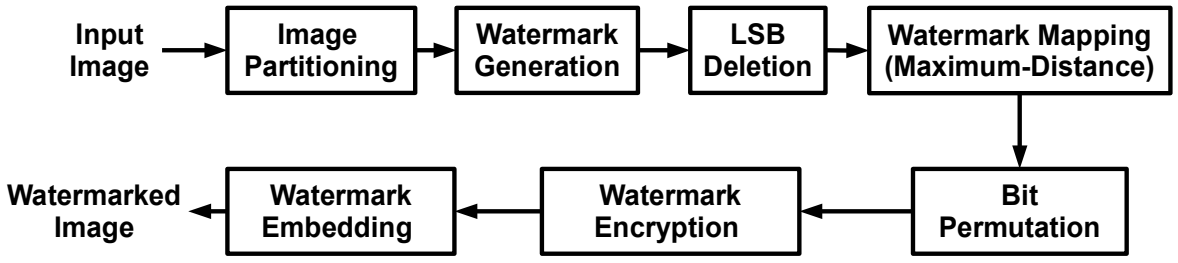


Figure 4.17: A block diagram for the encoding stage of the third proposed technique

The steps for the encoding stage are described as follows:

- **Image Partitioning:** The input image is divided into non-overlapping 2x2 px blocks. The dimensions of the input image must be multiples of 2.
- **Watermark Generation:** Similar to the second proposed technique, the

watermark is generated by finding the average of 6 MSBs and 2 LSBs of each block.

- **LSB Deletion:** 2 LSBs in the input image are set to 0 in order to store the watermark in them.
- **Watermark Mapping:** The watermark is mapped using maximum-distance mapping described in section 3.4.6, the mapping is done in both horizontal and vertical directions.
- **Bit Permutation:** The bits of the watermark pixels are permuted as in the second proposed technique.
- **Watermark Encryption:** Similar to the first and the second proposed techniques, the watermark is encrypted by XORing it with a random sequence.
- **Watermark Embedding:** Similar to the second proposed technique, each pixel in the watermark is stored in 2 LSBs of the block that coincides with that pixel.

4.6.2- The Decoding Stage of the Third Technique

A general block diagram for the decoding stage is shown in Figure 4.18.

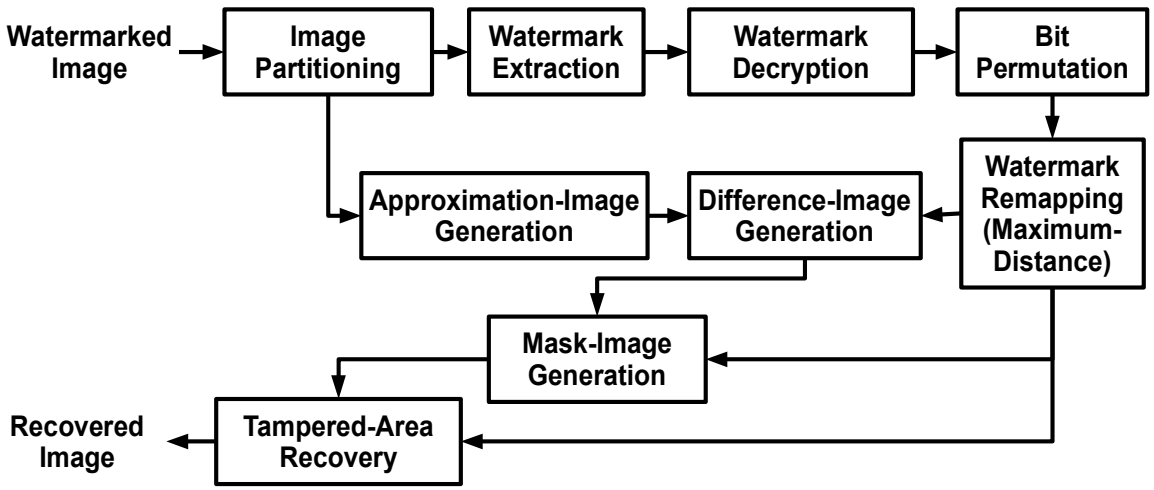


Figure 4.18: A block diagram for the decoding stage of the third proposed technique.

The lighthouse image in Figure 4.1 (a) will be used to illustrate the decoding stage. The image is tampered with by inserting an image of a hot-air balloon, as in Figure 4.1 (c). The steps for the decoding stage are described as follows:

- **Image Partitioning:** The input watermarked image is divided into non-overlapping 2x2 px blocks.
- **Watermark Extraction:** The pixels of the watermark are extracted from 2 LSBs of each block, the watermark has half the width and half the height of the watermarked image.
- **Watermark Decryption:** Similar to the first and second proposed techniques, the watermark is XORed with the random sequence generated in the encoding stage.
- **Bit Permutation:** The bits of the pixels of the watermark are permuted as in the encoding stage, this will return the bits to their original positions.
- **Watermark Remapping:** The watermark pixels are mapped using maximum-distance mapping, as in the encoding stage, this will return them to their original position. The resulting image is referred to as the ***watermark image***. See Figure 4.19.

Figure 4.19: An illustration of the watermark image in the decoding stage of the third proposed technique.

- **Approximation Image Generation:** Similar to the second proposed technique, the ***approximation image*** is generated from the watermarked

image.

- **Difference Image Generation:** Similar to the second proposed technique, the ***difference image*** is generated from the approximation and the watermark images. See Figure 4.20.

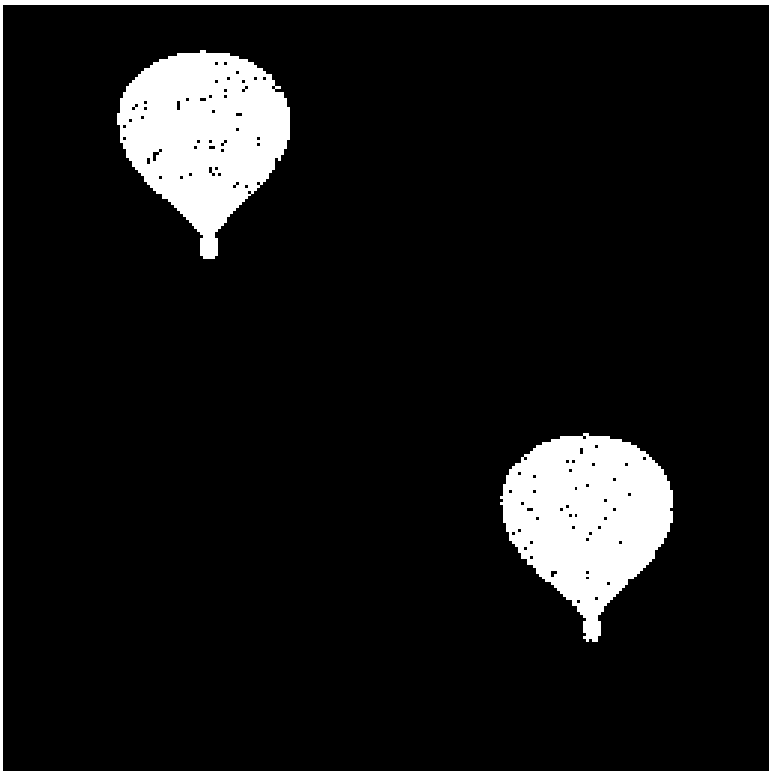


Figure 4.20: An illustration of the difference image in the decoding stage of the third proposed technique.

- **Mask Image Generation:** It can be noticed from the difference image that there are two copies of the tampered area, one coincides with the tampering in the watermark image and the other ***twin copy*** is shifted half the width and half the height of the image and it coincides with the actual tampered area in the watermarked image. The tampered area in the watermark image is distinguished by its HFCs. Therefore, the actual tampered area can be detected by measuring the HFCs in the watermark image and choosing the area that has less HFCs as follows:

  For each white pixel in the difference image, the HFCs in the watermark image are measured, this is done by using a window with an appropriate size and finding the DCT coefficients in this window, then the low-frequency coefficients are set to zero and the summation of the absolute values of the

remaining coefficients is found.

To determine the appropriate window size, the test in section 4.4.3 is carried out for the third technique and block sizes of 3x3 px, 5x5 px, 7x7 px, and 9x9 px were tested, the results are shown in Table 4.6. It can be seen that the average PSNR and PFA are significantly improved when selecting block sizes higher than 3x3 px; however, the improvement is not significant for sizes higher than 5x5 px. Since a larger block size requires more time in DCT calculation, a block size of 5x5 px is selected.

Table 4.6: The average value of PSNR, PFA, and PFR when varying the window size in the third technique.

Window size %	average PSNR (dB)	average PFA %	average PFR %
3x3 px	36.13	0.33%	0.46%
5x5 px	37.19	0.08%	0.43%
7x7 px	37.23	0.07%	0.43%
9x9 px	37.23	0.07%	0.43%

The first row and the first column in the DCT coefficients matrix are set to 0, because they correspond to the low frequency contents. The HFCs correspond to each white pixel in the difference image and its twin pixel are calculated. The tampered area is then detected by choosing the pixel with less corresponding HFCs. The resulting image is referred to as the ***mask image,*** as shown in Figure 4.21. Figure 4.22 shows the HFCs for watermark image, the tampered area in the watermark image is brighter than the rest of it.

Figure 4.21: An illustration of the mask image in the decoding stage of third proposed technique.

Figure 4.22: The high-frequency contents in the watermark image.

- **Tampered Area Recovery:** Recovering the tampered area is identical the one in the second proposed technique. The recovered image is similar to the one in Figure 4.15.

4.7- The Description of the Encoding and the Decoding Stages of the Fourth Technique

In this technique, the parameters used for the watermarking process are optimised to give the best recovery quality and resistance to counterfeiting. This is done by using blocks of size 8x8 px and generating the recovery data using the DCT of the block, which gives better quality than averaging 2x2 px blocks. Since it is not feasible to use the ideas that were used in the previous three techniques to avoid using separate sets of bits for localisation and recovery; 16 bits were assigned in each block to generate separate localisation data which is generated using CRC and specifically CRC16-CCITT that is used in communications protocols. Using CRC16 gives a very low PFA (about 2^{-16}) and makes the technique more resistive to counterfeiting than the previous proposed techniques (such as detecting pixel exchange in the same block which is not detectable by previous proposed techniques).

The generation of the recovery bits has some similarity with JPEG compression standard, the reader can refer to Section 2.9.2 for further information about JPEG compression standard.

The implemented code for the encoding and the decoding stages of the fourth proposed technique is provided in Appendix D.

4.7.1- The Encoding Stage of the Fourth Technique

A general block diagram for the encoding stage is shown in Figure 4.23.

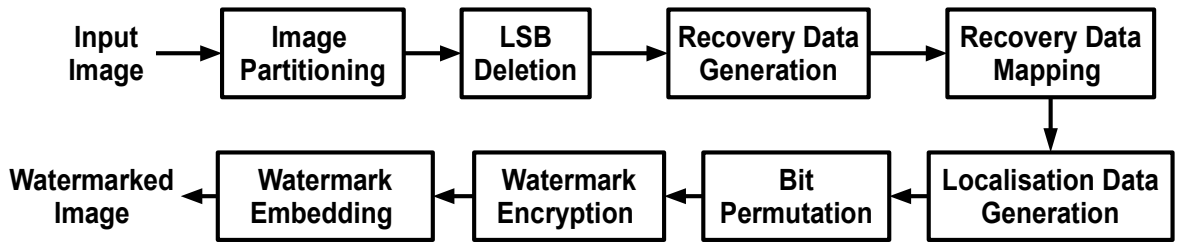


Figure 4.23: A block diagram for the encoding stage of the fourth proposed technique

The steps for the encoding stage are described as follows:

- **Image Partitioning:** The input image has a size of MxN where M and N are the width and the height of the image respectively. M and N must be multiples of 8. The image is divided into 8x8 px blocks $B(i,j)$, where

$1 \le i \le M/8$ and $1 \le j \le N/8$.

- **LSB Deletion:** 2 LSBs in $B(i, j)$ are set to 0 in order to store the watermark in them.

- **Recovery Data Generation:** The recovery data for $B(i, j)$ is generated by finding the DCT coefficients for $B(i, j)$, the DCT coefficients are quantized according to the standard JPEG luminance quantisation matrix which corresponds to 50% quality, the matrix is shown in Figure 2.5. The quantisation is carried out by dividing the elements in the DCT coefficient matrix by the elements in the quantisation matrix and rounding the results.

  Each DCT coefficient is stored in a fixed number of bits as shown in Figure 4.24. The numbers are chosen so that the total number of bits for all coefficients will be 112 bits. The DCT coefficients are then converted into a zigzag sequence as in Figure 2.6. The sequence is then converted into a bit stream which is stored into 14 bytes, and they represent the recovery data $R(i, j)$ of the block $B(i, j)$.

8	**7**	**6**	**5**	**4**	**3**	**2**	**0**
7	**6**	**5**	**4**	**3**	**2**	**0**	**0**
6	**5**	**4**	**3**	**2**	**0**	**0**	**0**
5	**4**	**3**	**2**	**0**	**0**	**0**	**0**
4	**3**	**2**	**0**	**0**	**0**	**0**	**0**
3	**2**	**0**	**0**	**0**	**0**	**0**	**0**
2	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**

Figure 4.24: The number of bits assigned for each DCT coefficient in the fourth proposed technique.

- **Recovery Data Mapping:** The recovery data generated for the block $B(i, j)$, which is referred to as $R(\hat{i}, \hat{j})$, is stored in another block $B(\hat{i}, \hat{j})$. The recovery data stored in $B(i, j)$, i.e. $R(i, j)$, is for the block $B(\hat{i}, \hat{j})$. The mapping of the recovery data is based on maximum-distance mapping, so that the block position (x, y) is mapped to $(x+\Delta x, y+\Delta y)$ where:

$$\Delta x=\begin{cases} M/2, & x\le M/2 \\ -M/2, & x>M/2 \end{cases} \quad , \quad \Delta y=\begin{cases} N/2, & y\le N/2 \\ -N/2, & y>N/2 \end{cases} \tag{4.1}$$

- **Localisation Data Generation:** The localisation data is generated by finding the CRC16 for the concatenation of the bytes in $B(i,j)$ and $R(i,j)$. The resulting 16 bits are converted into 2 bytes and they represent the authentication data $A(i,j)$ which will be stored in the block $B(i,j)$.

- **Bit Permutation:** $R(i,j)$ and $A(i,j)$ are stored in 2 LSBs of an 8x8 px block $W(i,j)$ and it represents the watermark to be stored in the block $B(i,j)$. The elements in $W(i,j)$ are permuted randomly according to a secret key ***K***.

- **Watermark Encryption:** $W(i,j)$ is XORed with a random sequence, the generation of the seed of the random sequence depends on the secret key ***K*** and the values of i and j, in this way no block in the image can be used to replace anther block in the same image. To make the technique resistant to VQ attack, the generation of the seed should also depend on a unique serial number for the image so that no block from another image could be used to replace a block from this image even if the same secret key is used in the encoding of both images.

- **Watermark Embedding:** Finally, 2 LSBs from $W(i,j)$ are stored in 2 LSBs in $B(i,j)$.

4.7.2- The Decoding Stage of the Fourth Technique

A general block diagram for the decoding stage is shown in Figure 4.25.

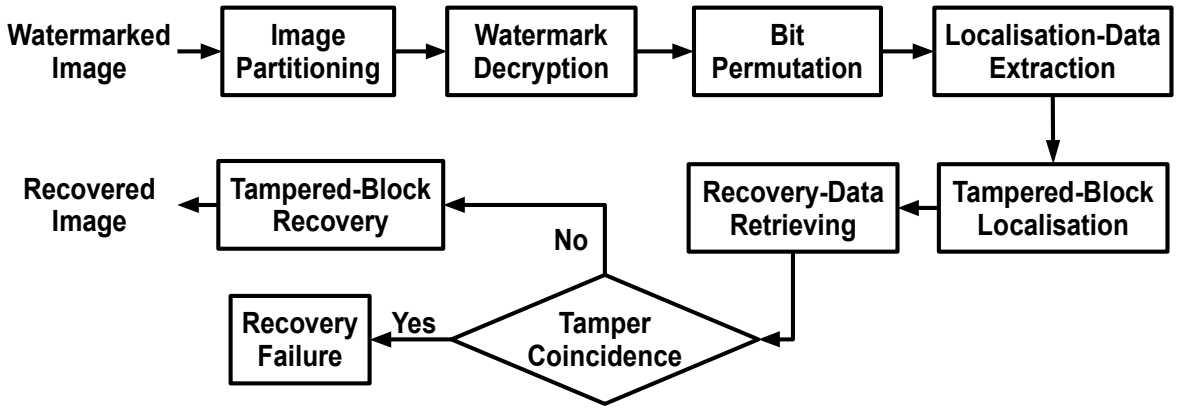


Figure 4.25: A block diagram for the decoding stage of the fourth proposed technique.

The steps for the decoding stage are described as follows:

- **Image Partitioning:** The input image has a size of MxN where M and N are the width and the height of the image respectively. M and N must be multiples of 8. The image is divided into 8x8 px blocks $B(i,j)$, where $1 \le i \le M/8$ and $1 \le j \le N/8$.
- **Watermark Extraction:** The watermark $W(i,j)$ is retrieved by taking 2 LSBs from $B(i,j)$, then 2 LSBs in $B(i,j)$ are set to 0. The size of $W(i,j)$ is 8x8 px.
- **Watermark Decryption:** $W(i,j)$ is decrypted by XORing it with the random sequence used in the encoding stage.
- **Bit Permutation:** The elements in $W(i,j)$ are permuted back to their original positions.
- **Localisation Data Extraction:** The authentication data $A(i,j)$ is retrieved from $W(i,j)$.
- **Tampered Block Localisation:** The CRC for the concatenation of the bytes in $B(i,j)$ and $R(i,j)$ is calculated and converted into 2 bytes which will be referred to as $\hat{A}(i,j)$. $\hat{A}(i,j)$ is then compared to $A(i,j)$, if they are equal then the block is authentic, otherwise it is tampered. The authentication process is carried out for all of the blocks in the image.
- **Recovery Data Retrieving:** If a block $B(i,j)$ is tampered, then its recovery data is retrieved from the block $B(\hat{i},\hat{j})$ which has maximum-distance from $B(i,j)$ as described before in the encoding stage, the recovery data will be referred to as $R(\hat{i},\hat{j})$.
- **Tampered Block Recovery:** If $B(\hat{i},\hat{j})$ is also tampered, then $B(i,j)$ can not be recovered, otherwise the recovery data $R(\hat{i},\hat{j})$ is used to recover the block $B(i,j)$. To recover $B(i,j)$, $R(\hat{i},\hat{j})$ is converted into a bit stream, then into a DCT sequence by taking the number of bits specified in Figure 4.24 and converting them into decimal values. The DCT sequence is then stored into 8x8 matrix which is de-quantised and converted back from DCT domain into spatial domain, then it is used to recover $B(i,j)$. The de-

quantisation is carried out by multiplying the elements in the DCT matrix with the elements in the quantisation matrix shown in Figure 2.5. The recovery process is carried out for all of the blocks in the image.

As was done for the previous techniques, the lighthouse image in Figure 4.1 (a) is tampered with by inserting an image of a hot-air balloon, as in Figure 4.1 (c). Figure 4.26 shows the detected tampered area by the fourth technique, it can be seen that at the edges of the tampered area some untampered pixels are included and that is due to using block size of 8x8 px, which means that the PFR is a little bit high for this technique; however, the PFA is extremely low due to using 16 bit CRC for block authentication. When it comes to practical use, a high PFR does not impose a problem as much as a high PFA; therefore, the high PFR of this method is accepted, especially with the advantages that it has, such as its resistance to counterfeiting and high recovery quality.

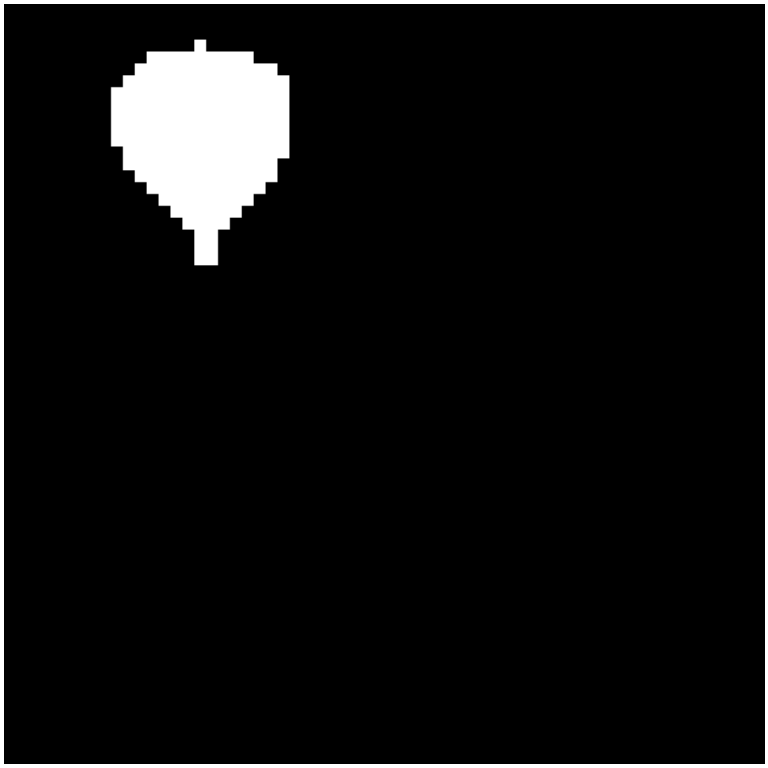


Figure 4.26: An illustration of the detected tampered area in the decoding stage of the fourth proposed technique.

4.8- The Description and the Shortcomings of the Referenced Techniques

The first referenced technique was proposed in 2013 by Tong et al. in [64] and the second one was proposed in 2014 by Dadkhah et al. in [69]. The techniques will be referred to as Tong's and Dadkhah's techniques. These two techniques were selected because they share some similarities with the proposed techniques, such as dividing the image into 2x2 px and generating the recovery data using the averages of these blocks. The referenced techniques have some advantages, such as low computational complexity for Tong's technique, and low PFA for Dadkhah's technique. The referenced techniques are also well-cited and used for comparison in many papers [79, 98-104]. In the following sections, the referenced techniques are described and their shortcomings are highlighted.

4.8.1- The Description of Tong's Technique

The main feature of this technique is that it uses the two-dimensional chaotic map proposed in [105] to encrypt the watermark by shuffling its rows and columns using the outcomes of the map. The chaotic map is defined as follows:

$$\begin{aligned} x_{i+1} &= x_i - \alpha . y_i^2 \\ y_{i+1} &= \cos\left(\beta . \cos^{-1}(x_i)\right) \end{aligned} \quad , \; -1 < (x, y) < 1 \tag{4.2}$$

where α and β are control parameters. When $\alpha = 2$ and $\beta = 6$ the map exhibits a chaotic behaviour.

The implemented code for the encoding and the decoding stages of Tong's technique is provided in Appendix E.

A general block diagram for the encoding stage is shown in Figure 4.27.

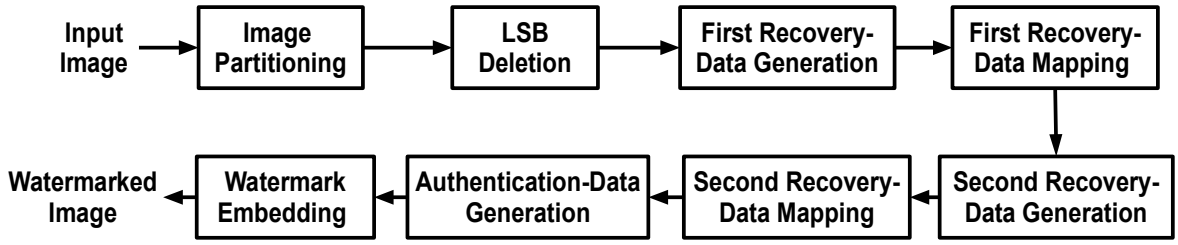


Figure 4.27: A block diagram for the encoding stage of Tong's technique.

The steps for the encoding stage are described as follows:

- **Image Partitioning:** The input image has a size of $M \times N$ where M and N are the width and the height of the image respectively. M and N must be multiple of 4. The image is divided into 2x2 px blocks $B(i,j)$, where $1 \le i \le M/2$ and $1 \le j \le N/2$.
- **LSB Deletion:** 3 LSBs in each $B(i,j)$ are set to 0 to store the watermark in them.
- **First Recovery Data Generation:** The first recovery data $W_1(i,j)$ is generated by taking 5 MSBs of the average of $B(i,j)$.
- **First Recovery Data mapping:** The image that is formed by all $W_1(i,j)$, which is referred to as W_1, is mapped, i.e. shuffled, by taking each row and rotate, i.e. circularly-shift, it for a random distance, then each column is rotated too. In this research, the whole rotation process is repeated once to make it more efficient. The random sequence used in the shuffling process is generated using the chaotic map in equation 4.2 with $\alpha = 2$ and $\beta = 6$. By using the initial values x_0, y_0 and after some iterations, the generated chaotic sequence is used as the random numbers in the shuffling process. The x values are used for the rows and the y values are used for the columns. The initial values x_0, y_0 and the number of iterations are considered as the secret key for the watermark.
- **Second Recovery Data Generation:** After shuffling W_1, a second recovery matrix W_2 is generated by taking a copy of W_1.
- **Second Recovery Data Mapping:** W_2 is circularly-shifted using maximum-distance mapping in both the horizontal and vertical directions, this will provide another chance for recovery when tamper coincidence occurs.
- **Authentication Data Generation:** Two authentication bits are generated for each block $B(i,j)$, the first bit is $A_1(i,j) = mod_2(\epsilon)$ where ϵ is total number of ones in $B(i,j)$, $W_1(i,j)$, and $W_2(i,j)$, $mod_2(\epsilon)$ is the remainder after dividing by 2. The second authentication bit $A_2(i,j) = \overline{A_1}(i,j)$ is the complement of A_1.

- **Watermark Embedding:** Finally, $W_1(i,j)$, $W_2(i,j)$, $A_1(i,j)$, and $A_2(i,j)$ are embedded in 3 LSBs of $B(i,j)$.

A general block diagram for the decoding stage is shown in Figure 4.28.

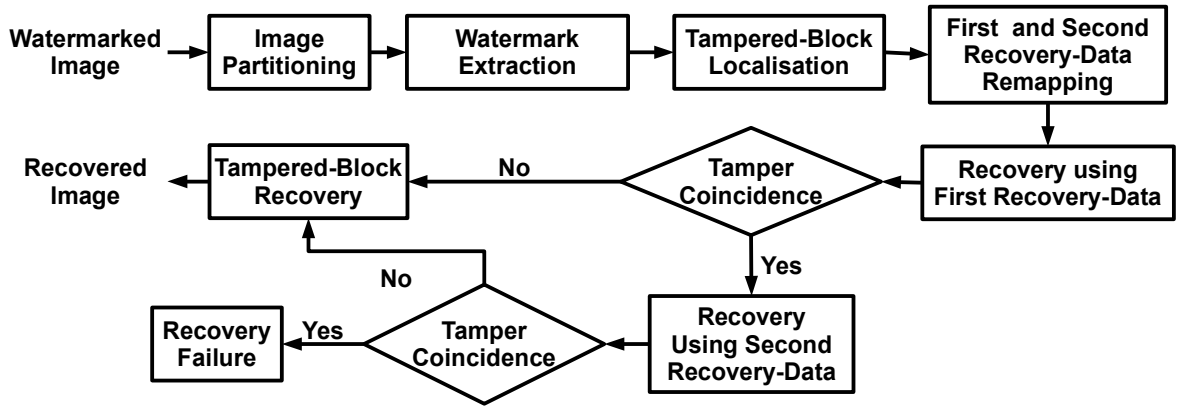


Figure 4.28: A block diagram for the decoding stage of Tong's technique.

The steps for the decoding stage are described as follows:

- **Image Partitioning:** The input watermarked image has a size of MxN where M and N are the width and the height of the image respectively. M and N must be multiple of 4. The input image is divided into 2x2 px blocks $B(i,j)$, where $1 \le i \le M/2$ and $1 \le j \le N/2$.

- **Watermark Extraction:** $A_1(i,j)$, $A_2(i,j)$, $W_1(i,j)$, and $W_2(i,j)$ are extracted from $B(i,j)$, then 3 LSBs in $B(i,j)$ are set to 0.

- **Tampered Block Localisation:** The number of ones ϵ in all of $B(i,j)$, $W_1(i,j)$, and $W_2(i,j)$ is calculated. $\hat{A}_1(i,j)$ and $\hat{A}_2(i,j)$ are generated, where $\hat{A}_1(i,j) = mod_2(\epsilon)$ and $\hat{A}_2(i,j) = \overline{\hat{A}_1}(i,j)$ is the complement of $\hat{A}_1(i,j)$, then $B(i,j)$ is authenticated, where $B(i,j)$ is authentic if $\hat{A}_1(i,j) = A_1(i,j)$ and $\hat{A}_2(i,j) = A_2(i,j)$. If $B(i,j)$ is tampered with, then $W_1(i,j)$, $W_2(i,j)$ are marked as invalid. The authentication process is carried out for all of $B(i,j)$.

- **First and Second Recovery Data Remapping:** W_1 is shuffled back by reversing the shuffling process that was carried out in the encoding stage. W_2 is remapped using maximum-distance mapping, then it is shuffled

back as was done for W_1.

- **Tampered Block Recovery:** If a block $B(i,j)$ is tampered with, $W_1(i,j)$ is used to recover it. If $W_1(i,j)$ is invalid, $W_2(i,j)$ is used for recovery, if $W_2(i,j)$ is also invalid, the block recovery fails. The recovery process is carried out for all tampered blocks in the image.

4.8.2- The Shortcomings of Tong's Technique

The main shortcomings of Tong's technique are summarised as follows:

- High PFA due to the limited number of authentication bits in each block, where 2 bits are used for authentication and that means $PFA \approx 2^{-2} \approx 25\%$ which means that about one fourth of the tampered blocks will remain unrecovered, which leads to low recovery quality. Figure 4.29 shows the recovered image after tampering a central squared area of the lighthouse image by inserting the contents of "Peppers" image [106] in it.

(a) (b)

Figure 4.29: An example of Tong's technique: a) The tampered image. b) The recovered image. (Peppers image taken from [106]).

- The position of the authentication bits is known and that makes the technique very vulnerable and the intruder can easily put any counterfeited content inside the block $B(i,j)$ as long as they generate the same $A_1(i,j)$ and $A_2(i,j)$.

- The watermark is not encrypted and that makes the technique vulnerable

against counterfeiting attacks.

- The authentication scheme is vulnerable against counterfeiting, so that replacing 5 MSBs in a block with any values that gives the same parity will not be detected.

4.8.3- The Description of Dadkhah's Technique

The main feature in this technique is the use of SVD for calculating the authentication bits for each block. The image is divided into 4x4 px blocks $B(i,j)$ and each block is further divided into four 2x2 px blocks B_{S1} to B_{S4}. 2 singular values (SVs) are found for each B_S block and 3 authentication bits are generated for each B_S based on these SVs. See section 2.11 for more information about SVD.

The authentication is done for $B(i,j)$ block even though the authentication bits are calculated for B_S blocks, i.e. $B(i,j)$ is considered authentic only if all its B_S blocks are authentic. Therefore, the total number of authentication bits used to authenticate $B(i,j)$ block is 12 bits, which makes the PFA about $2^{-12} \approx 0.0244\,\%$.

The calculation of the authentication bits for B_S block can be summarised as follows:

- Get the two SVs for the B_S block.
- If any SV has a fractional part, then the calculation of the authentication bits is carried out as follows:
  - Let $X1$ be 1 if the two leftmost digits in the fractional part of the first SV are greater than 50 and 0 else wise.
  - Let $X2$ be 1 if the two leftmost digits in the fractional part of the second SV are greater than 50 and 0 else wise.
  - Let $X3$ be 1 if the integer part of each SVs (as binary numbers) has even number of ones and 0 else wise.
  - The authentication bits are generated depending on the values of $X1$, $X2$, and $X3$ as follows:

$$Authentication\,bits = \begin{cases} 000 & if\ X1=1, X2=1, X3=1 \\ 100 & if\ X1=1, X2=0, X3=1 \\ 010 & if\ X1=1, X2=1, X3=0 \\ 001 & if\ X1=0, X2=1, X3=1 \\ 110 & if\ X1=0, X2=0, X3=1 \\ 101 & if\ X1=1, X2=0, X3=0 \\ 011 & if\ X1=0, X2=1, X3=0 \\ 111 & if\ X1=0, X2=0, X3=0 \end{cases}$$

- If both SVs are integer numbers, then the three authentication bits A_1, A_2, and A_3 are generated as follows:
  - A_1 is 1 if both SVs (as binary numbers) have even number of ones and 0 else wise.
  - A_2 is 1 if both SVs are less than or equal 50 and 0 else wise.
  - A_3 is the complement of A_1.

The code implemented for the encoding and the decoding stages of Dadkhah's technique is provided in Appendix F.

A general block diagram for the encoding stage is shown in Figure 4.30.

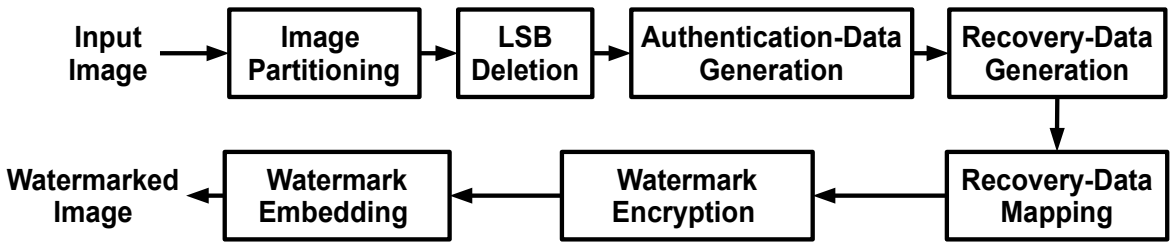


Figure 4.30: A block diagram for the encoding stage of Dadkhah's technique.

The steps for the encoding stage are described as follows:

- **Image Partitioning:** The input image has a size of $M \times N$ where M and N are the width and the height of the image respectively. M and N must be multiples of 4. The image is divided into 2x2 px blocks $B(i, j)$, where $1 \le i \le M/2$ and $1 \le j \le N/2$. $B(i, j)$ is divided into 4 of 2x2 px blocks B_{S1} to B_{S4}.
- **LSB Deletion:** 2 LSBs in each $B(i, j)$ are set to 0 to store the watermark in them.

- **Authentication Data Generation:** 3 authentication bits are found for each B_S block, let A_{S1} to A_{S4} refer to the authentication bits of the blocks B_{S1} to B_{S4} and $A(i,j)$ refer to the concatenation of all A_{S1} to A_{S4} in the block $B(i,j)$.
- **Recovery Data Generation:** The recovery information for $B(i,j)$ is 5 MSBs of the average of each sub-block B_S, this means there will be 20 bits of recovery information for $B(i,j)$.
- **Recovery Data Mapping:** For each block $B(i,j)$ in the upper half of the image, a block $B(\hat{i},\hat{j})$ is selected randomly from the lower half of the image, and for each block in the lower half of the image, a block is selected randomly from the upper half of the image. The mapping is done so that if the recovery information of $B(i,j)$ is stored in $B(\hat{i},\hat{j})$, then the recovery information of $B(\hat{i},\hat{j})$ should not be stored in $B(i,j)$. The recovery data stored in $B(i,j)$ is referred to as $R(i,j)$ and it consists of the concatenation of R_{S1} to R_{S4} which are the recovery data of sub-blocks B_{S1} to B_{S4}.
- **Watermark Encryption:** The authentication data $A(i,j)$ and the recovery data $R(i,j)$ are encrypted by XORing them with a random sequence. The generation of the random sequence is based on a secret key and is different for each $B(i,j)$ block, so that no block can be used as a replacement for another one from the same image.
- **Watermark Embedding:** The 12 bits in $A(i,j)$ and the 20 bits in $R(i,j)$, which are 32 bits in total, are stored in 2 LSBs in the block $B(i,j)$. The process is repeated for all $B(i,j)$ blocks in the image.

A general block diagram for the decoding stage is shown in Figure 4.31.

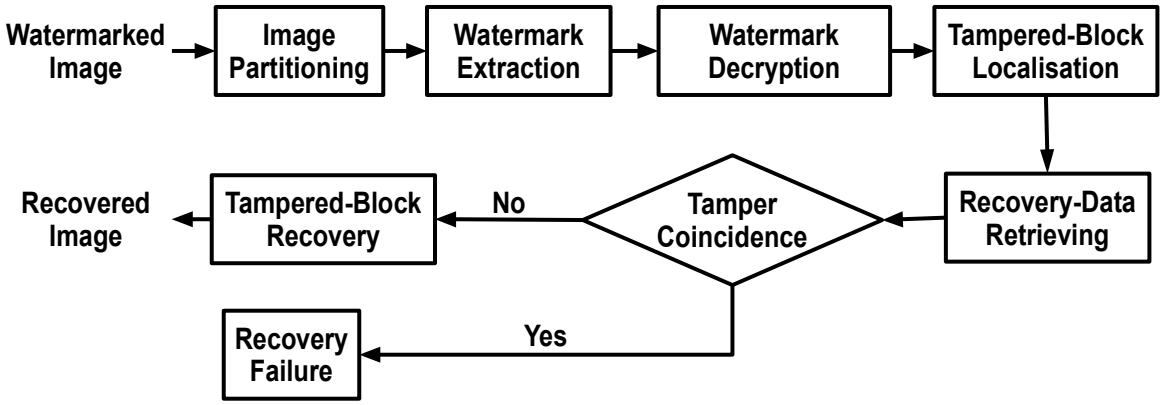


Figure 4.31: A block diagram for the decoding stage of Dadkhah's technique.

The steps for the decoding stage are described as follows:

- **Image Partitioning:** The input watermarked image has a size of MxN where M, N are the width and the height of the image respectively. M and N must be multiples of 4. The input image is divided into 4x4 px blocks $B(i,j)$, where $1 \le i \le M/2$ and $1 \le j \le N/2$.

- **Watermark Extraction:** The authentication bits $A(i,j)$ and the recovery bits $R(i,j)$ are retrieved from the block $B(i,j)$, then 2 LSBs in $B(i,j)$ are set to 0.

- **Watermark Decryption:** $A(i,j)$ and $R(i,j)$ are decrypted by XORing them with the random sequence as in the encoding stage

- **Tampered Block Localisation:** $B(i,j)$ is divided into four 2x2 px blocks B_{S1} to B_{S4} and the 3 authentication bits are calculated for each B_S. Let $\hat{A}_{S1}$ to $\hat{A}_{S4}$ refer to the calculated authentication bits for B_{S1} to B_{S4}, and $\hat{A}(i,j)$ refer to the concatenation of all $\hat{A}_{S1}$ to $\hat{A}_{S4}$ in the block $B(i,j)$.

  The block $B(i,j)$ is authenticated by comparing all $\hat{A}_{S1}$ to $\hat{A}_{S4}$ with all A_{S1} to A_{S4}. If anyone of them is not equal to the corresponding one, then $B(i,j)$ is considered as invalid (i.e. tampered block). The validation process is carried out for blocks $B(i,j)$.

  Another level of validation is carried out by retrieving the 20 recovery bits of $B(i,j)$ from its mapped block $B(\hat{i},\hat{j})$, which is mapped randomly in the opposite half of the image and must be valid too. The recovery information

is compared to 5 MSBs of the average of each B_{S1} to B_{S4} sub-block inside $B(i, j)$ and if it is found to be different, then $B(i, j)$ is considered invalid.

- **Recovery Data Retrieving:** After validating all of the blocks in the image, if any $B(i, j)$ is tampered, then it is recovered by retrieving its recovery bits from its mapped block $B(\hat{i}, \hat{j})$. If $B(\hat{i}, \hat{j})$ is invalid, then the recovery process fails. The recovery process is carried out for all tampered blocks in the image.

4.8.4- The Shortcomings of Dadkhah's Technique

The main shortcomings found in Dadkhah's et al. technique are summarised as follows:

- High tamper-coincidence: Due to the mapping between the blocks in the upper half and the lower half of the image. For example, if the tamper is only in the upper half of the image, then no tamper coincidence will occur; however, if the tamper extends across the upper and the lower halves of the the image, then tamper coincidence will occur. The probability of tamper coincidence for a block in the upper half of the image equals to the tampering ratio of the lower half of the image and vice versa. Therefore, if a central area is tampered then the tamper-coincidence probability equals the ratio of this area (i.e. tamper ratio). For example if 25% central area in the image is tampered then the tamper-coincidence probability is 25% which means that about 25% of the tampered blocks will not be recovered. Figure 4.32 shows an example of Dadkhah's technique, the tampering coincidence is obvious in part (b).

(a) (b)

Figure 4.32: An example of Dadkhah's technique: a) The tampered image. b) The recovered image.

- Any block tampering that is done by: rotation, flipping, or transposition of a block, will not be detected because the SVs do not change due to these operations (see section 2.11).

4.9- Summary

The beginning of this chapter highlighted the challenge of using one set of data for both purposes of localisation and recovery, where direct comparison between the pixels in the watermark and the watermarked image results in having two areas that are affected by the tampering. The challenge resides in distinguishing the area that corresponds to the actual tampering in the watermarked image.

The four proposed techniques were described. Three of them overcome the problem of using one set of data for localisation and recovery by employing image filtering to localise the actual tampered area. The first proposed technique is distinguished by using random mapping for the watermark pixels, which helps in localising the tampered area by using median filtering. The second technique maps the watermark pixels in three directions based on maximum-distance mapping, which enables using non-linear spatial filtering in localising the tampered area; however, this also results in a reduced recoverable area, about 25% of the image area. The third technique employs DCT to measure the high frequency contents that appear in the tampered area after XORing the watermark image with

the random sequence used in encryption, this enables recovering a tampering with ratio of 50% of the image area. The fourth technique is optimised to give the best results regarding recovery quality, PFA, and resistance of counterfeiting, which is achieved by using a larger block size of 8x8 px and employing DCT to approximate block contents. Some methods were proposed to optimise the security of the image when using maximum-distance mapping, such as permuting the bits in the individual watermark pixels or permuting the bits of the watermark in the individual block.

The last section in the chapter presented the description and the shortcomings of two referenced techniques. The first one is Tong's technique, which is characterised by introducing a novel chaotic map for watermark shuffling. The second one is Dadkhah's technique, which is characterised by using SVD for authentication data generation. The main shortcomings for Tong's technique are its low PFA, which results in low recovery quality, and its vulnerability against counterfeiting. Dadkhah's technique suffers mainly from high tamper coincidence probability, due to the used mapping method, which results in low recovery quality.

Chapter 5: Experimental Evaluation for the Proposed and the Referenced Techniques

5.1- Introduction

This chapter presents an experimental evaluation of the proposed techniques with respect to: PSNR, NCC, SSIM, , PFA, PFR, encoding time, and decoding time. The proposed techniques are compared to the two referenced techniques described before in Section 4.8.

All simulation codes in this research were programmed using GNU Octave version 4.2.2 [107] and all plots were made using Gnuplot version 5.2 [108]. Appendix G presents the code used in the experimental evaluation in this chapter.

All of the 24 images in Kodak image database are used in this evaluation. The evaluation proceeds as follows:

- All of the 24 images in Kodak database are converted to grey-scale and cropped to the central 512x512 px area.
- A central rectangular area in each image is tampered by replacing that area with one from the next image in the list. The first image is used to tamper the last one. The rectangular area has a height-to-width ratio of either 1:1 or 2:1 in order to adjust the shape of the area to match the recoverable area of each technique. The ratio for techniques 1, 2, and Dadkhah's is 1:1, while the ratio for techniques 3, 4, and Tong's is 2:1.
- The values of the pixels in the tampered area were adjusted if they have a close value to original pixels in the untampered image. The adjustment is done so that the difference between the old pixel and the new one is not less than 5, so that there will be a difference in pixel value even if 2 LSBs in the image are set to 0.
- The percentage of the rectangular area increased from 0 to 50% as in the following list: [0%, 5%, 10%, 15%, 20%, 24%, 25%, 30%, 35%, 40%, 45%,

49%, 50%]. The 24% and 49% values were used because some techniques have sudden change in evaluation results at 25% percent (The second technique) and at 50% (The third technique). The 0% ratio means that the image is not tampered.

- For each technique, each one of the 24 images is tampered with according to the list shown in the previous step, then the average values of the following parameters are measured for the 24 images: 1) PSNR. 2) NCC. 3) 4) SSIM. 5) PFA. 6) PFR. 7) Encoding time. 8) Decoding time. The values of PSNR, NCC, and SSIM are measured between the original unwatermarked image and the recovered image.

In the following sections, the experimental results for each parameter are presented and discussed in order to show how the proposed techniques perform when they are compared to each other and when they are compared to Tong's and Dadkhah techniques.

5.2- The Experimental Results for the Average PSNR for the Proposed and the Referenced Techniques

The average PSNR corresponding to each ratio is shown in Table 5.1 and Figure 5.1. The following notes should be considered about Table 5.1 and Figure 5.1:

- During recovery, the LSBs that are used to store the watermark are set to 0. This results in a reduced PSNR because leaving the watermark data in them will increase the PSNR (See Section 2.8.1); however, setting them to 0 gives more accurate representation of the performance of the technique (i.e. no irrelevant information is used to enhance the value of the PSNR).

- The experimental results include a hypothetical perfect recovery situation for Tong's and Dadkhah's techniques where: tamper coincidence, PFA, and PFR are all assumed to be 0, this should not be confused with the actual results of Tong's and Dadkhah's techniques. The assumed perfect recovery situation was included in the results to show how the proposed techniques perform better than Tong's and Dadkhah's techniques even at a hypothetical perfect recovery scenario.

Table 5.1: The experimental results of the average PSNR for the proposed and the referenced techniques.

Tamper Ratio	Average PSNR (dB)							
	Technique 1	Technique 2	Technique 3	Technique 4	Tong's Technique	Dadkah's Technique	Tong's (Perfect recovery)	Dadkah's (Perfect recovery)
0%	51.09	44.64	44.68	42.64	35.66	42.64	35.66	42.64
5%	35.42	38.55	38.38	39.66	28.64	34.38	33.82	36.84
10%	32.48	36.46	36.38	38.36	25.82	29.13	32.69	34.64
15%	30.83	35.12	34.9	37.46	23.91	26.18	31.83	33.31
20%	29.48	34.13	33.86	36.49	22.49	24.03	31.16	32.31
24%	28.59	33.53	33.3	36.14	21.56	22.55	30.75	31.7
25%	28.38	29.2	33.14	35.98	21.35	22.39	30.65	31.57
30%	27.22	20.81	32.49	35.47	20.42	20.85	30.19	30.93
35%	26.04	18.28	31.92	35.02	19.6	19.67	29.79	30.35
40%	24.71	16.75	31.41	34.57	18.86	18.44	29.42	29.86
45%	23.5	15.62	30.98	34.16	18.25	17.53	29.11	29.45
49%	22.34	14.91	30.52	34.02	17.77	16.81	28.86	29.15
50%	22.02	14.74	29.08	33.96	17.67	16.65	28.66	29.07

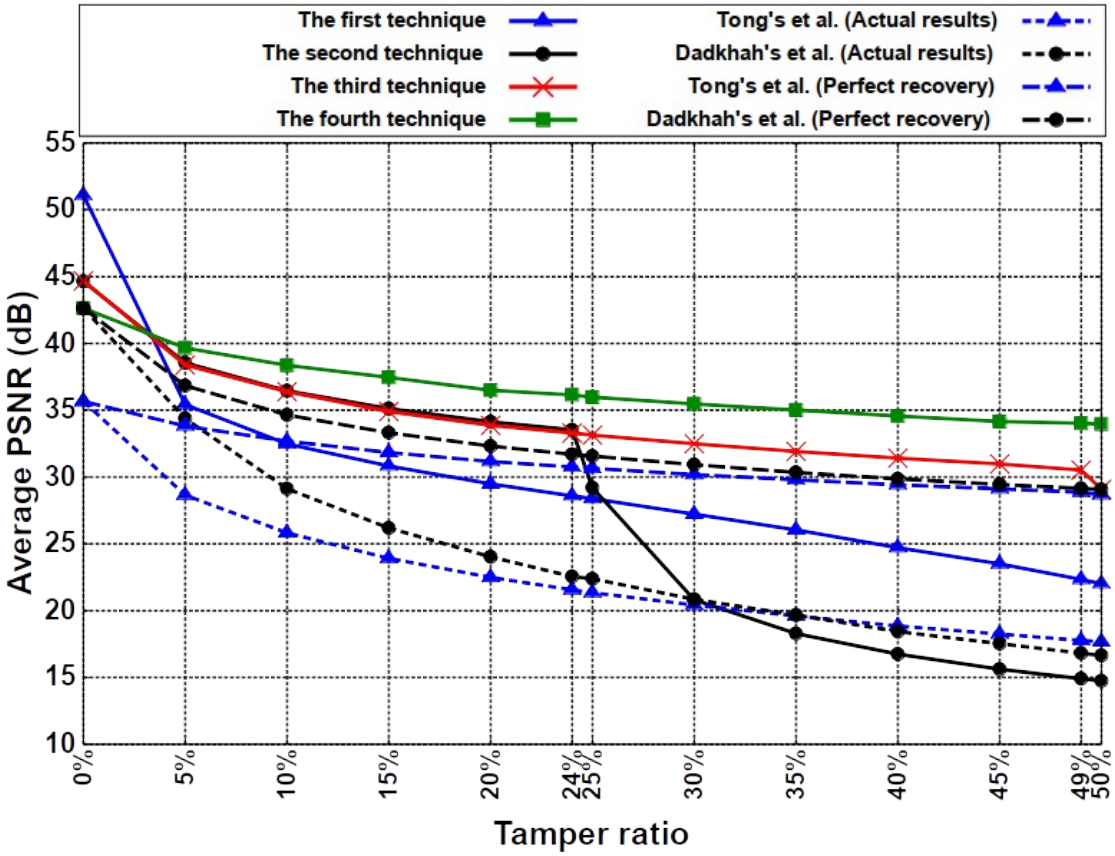


Figure 5.1: The experimental results of the average PSNR for the proposed and the referenced techniques.

It is possible to draw the following remarks from the experimental results of the average PSNR values:

- At no tampering: The first technique has the best results because only one LSB was used to store watermark information. Tong's technique has the worst results because it uses 3 LSBs. Techniques 2 and 3 perform better than technique 4 and Dadkhah's technique because of the partial recovery of the deleted LSBs in the original image.
- Technique 2 gives the same performance as technique 3 until 24% ratio, where its results drop dramatically after that due to tamper-coincidence and high PFR.
- Technique 4 gives the best overall performance, especially at high tamper ratio, which proves that using DCT with a large block size (8x8 px in this case) gives better results than using the average of a smaller block size (2x2 px in this case).

- All of the proposed techniques (except for technique 2 at high tamper ratio) outperform Tong's and Dadkah's technique and that is due to the high PFA (25%) in Tong's technique and high tamper-coincidence probability (which equals the tampering ratio) in Dadkhah's tachnique.
- All of the proposed techniques, except for technique 1 and technique 2 at high tamper ratio, outperform the perfect recovery scenario of Tong's and Dadkhah's techniques and that is because the proposed techniques use more recovery bits (6 bits for technique 2 and 3 when compared to 5 bits in Tong's and Dadkhah's techniques) and due to the partial recovery of the deleted LSBs in technique 2 and 3.

It can be concluded from the PSNR results that the fourth technique is the best one regarding the PSNR results, and the third technique comes in the second place.

5.3- The Experimental Results for the Average NCC and the Average SSIM for the Proposed and the Referenced Techniques

The experimental results of the average NCC and average SSIM are presented in this section, the average NCC values are presented in Table 5.2 and Figure 5.2 and the average SSIM values are presented in Table 5.3 and Figure 5.3. The SSIM is is calculated using the code *ssim_index.m* provided in [109]

It can be noticed that relative quality performance is generally the same regardless of the measuring method; however, the PSNR provides better assessments for high quality results, where there are small difference values, and that is because of the use of the logarithmic scale.

Table 5.2: The experimental results of the average NCC for the proposed and the referenced techniques.

Tamper Ratio	Average NCC							
	Technique 1	Technique 2	Technique 3	Technique 4	Tong's Technique	Dadkah's Technique	Tong's (Perfect recovery)	Dadkah's (Perfect recovery)
0%	0.999989	0.999916	0.999917	0.999939	0.999721	0.999939	0.999721	0.999939
5%	0.99914	0.999591	0.999518	0.999775	0.996843	0.999128	0.999383	0.999542
10%	0.998356	0.999305	0.999212	0.999655	0.993604	0.996898	0.999081	0.999177
15%	0.997646	0.999041	0.998923	0.999556	0.989961	0.993776	0.998804	0.998863
20%	0.996863	0.998797	0.998674	0.999427	0.98612	0.989761	0.998564	0.998577
24%	0.996233	0.998629	0.998512	0.999374	0.982936	0.985669	0.998393	0.998377
25%	0.996067	0.996946	0.998465	0.999348	0.982069	0.985115	0.998349	0.998332
30%	0.994979	0.980566	0.998253	0.999263	0.97794	0.979112	0.998138	0.998086
35%	0.993563	0.966594	0.998039	0.999174	0.973475	0.973036	0.99794	0.997835
40%	0.991438	0.954018	0.997824	0.999078	0.9687	0.964948	0.997745	0.997606
45%	0.988865	0.942173	0.997623	0.99898	0.964227	0.957534	0.997565	0.997394
49%	0.985622	0.933483	0.997392	0.998952	0.960277	0.950761	0.997421	0.997234
50%	0.984579	0.931228	0.996608	0.998952	0.959425	0.949011	0.997297	0.997188

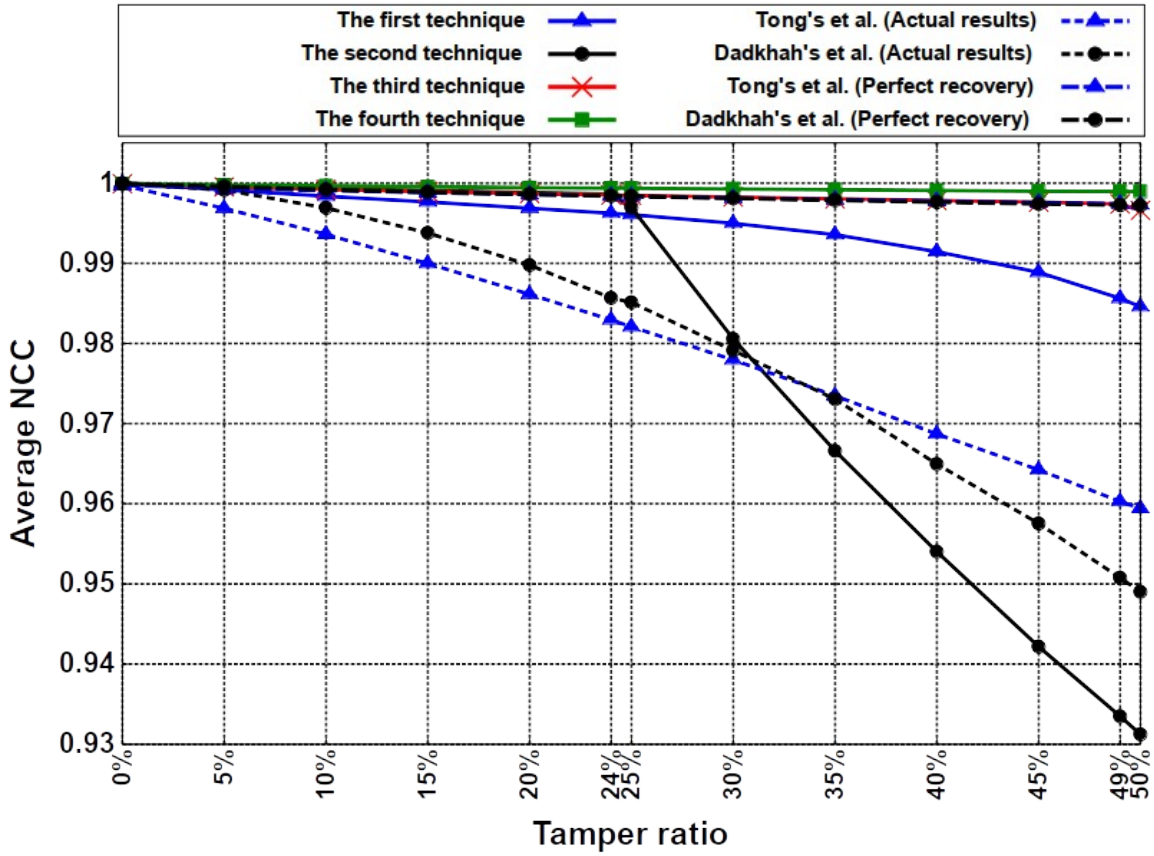


Figure 5.2: The experimental results of the average NCC for the proposed and the referenced techniques.

Table 5.3: The experimental results of the average SSIM for the proposed and the referenced techniques.

Tamper Ratio	Average SSIM							
	Technique 1	Technique 2	Technique 3	Technique 4	Tong's Technique	Dadkah's Technique	Tong's (Perfect recovery)	Dadkah's (Perfect recovery)
0%	0.998319	0.986236	0.986653	0.991929	0.968525	0.991929	0.968525	0.991929
5%	0.976466	0.976569	0.976538	0.986012	0.93276	0.976112	0.958462	0.98059
10%	0.954296	0.967603	0.967486	0.981108	0.896839	0.945905	0.948996	0.969397
15%	0.933491	0.958829	0.957676	0.976397	0.857231	0.913127	0.9395	0.958917
20%	0.90934	0.949737	0.948826	0.97009	0.817989	0.872494	0.930465	0.948448
24%	0.889084	0.942668	0.942312	0.967155	0.786029	0.836094	0.923819	0.940172
25%	0.883858	0.901602	0.940673	0.965996	0.777869	0.832135	0.922166	0.938223
30%	0.8528	0.75207	0.932489	0.961479	0.737273	0.781839	0.913926	0.927828
35%	0.817941	0.631456	0.924111	0.956592	0.695066	0.737202	0.905811	0.917582
40%	0.775032	0.530318	0.915008	0.951387	0.649282	0.682559	0.897346	0.907798
45%	0.728469	0.439616	0.906569	0.945942	0.605722	0.638337	0.889366	0.898494
49%	0.680519	0.374499	0.899211	0.945035	0.577022	0.598726	0.884016	0.891152
50%	0.666194	0.358903	0.886395	0.945035	0.572888	0.58769	0.882932	0.889104

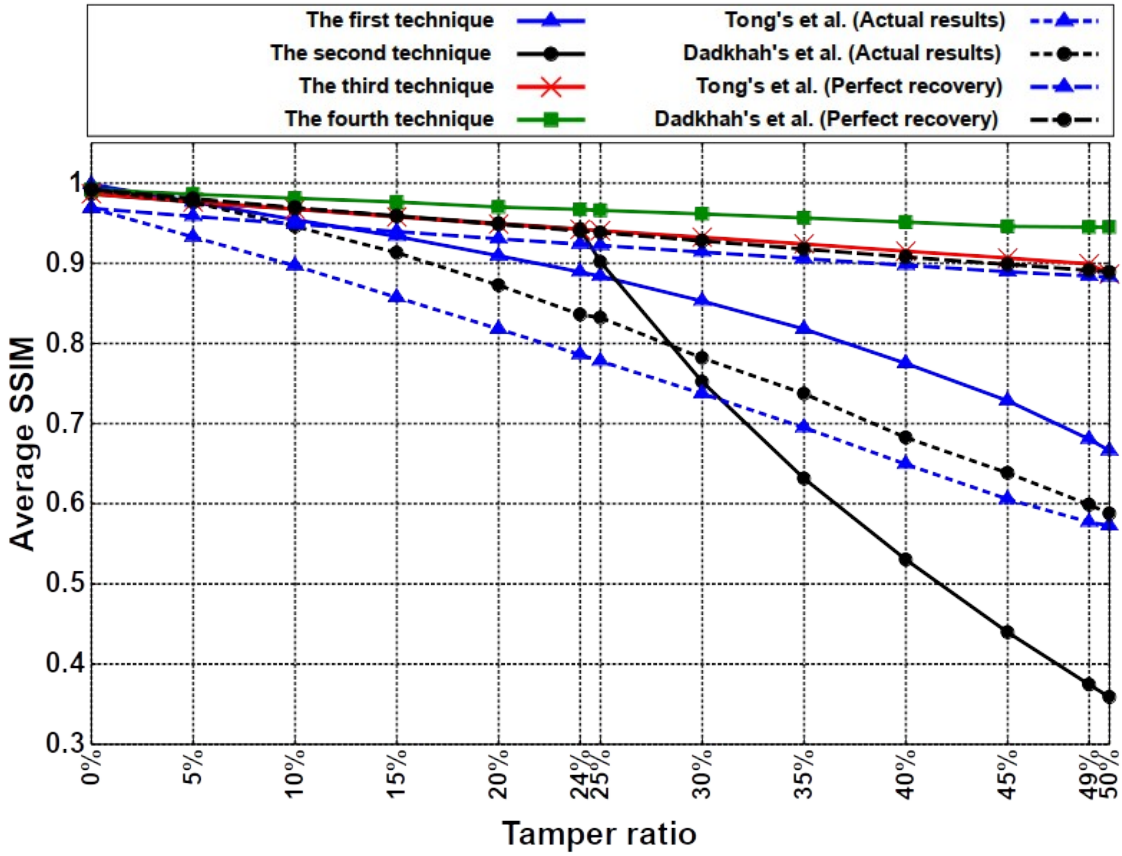


Figure 5.3: The experimental results of the average SSIM for the proposed and the referenced techniques.

5.4- The Experimental Results for the Average PFA for the Proposed and the Referenced Techniques

PFA is an important factor when it comes to tamper recovery and it has more critical importance when compared to PFR because accepting a tampered block as an authentic one is more dangerous than rejecting a valid one, especially that the rejected block will be recovered while the accepted one will not. Also, a high PFA will result in a lower recovery PSNR value.

The average PFA values corresponding to each ratio are shown in Table 5.4 and Figure 5.4. The following notes should be considered about Table 5.4 and Figure 5.4:

- The PFA was measured based on pixel level as in Equation 5.1, this gives more accurate results.

$$PFA = \frac{\text{Number of tampered pixels detected as authentic ones}}{\text{Total number of tampered pixels}} \quad (5.1)$$

- The scale of the y-axis in Figure 5.4 is a logarithmic scale for the values greater than 0, this scale was selected because some values are very small (like 0.002%) and some are large (like 24.9%).
- The values in Table 5.4 and Figure 5.4 are the percentage values, which means that they should be divided by 100 to get the actual values. For example the value 0.002 (i.e. 0.002%) represents the value of PFA = 0.002/100 = 0.00002.

Table 5.4: The experimental results for the average PFA for the proposed and the referenced techniques.

Tamper Ratio	Average PFA %					
	Technique 1	Technique 2	Technique 3	Technique 4	Tong's Technique	Dadkah's Technique
0%	0	0	0	0	0	0
5%	1.1548	0.0513	0.2	0	24.9994	0.0026
10%	1.1145	0.0558	0.1344	0	24.502	0.0129
15%	1.2063	0.0742	0.1089	0	24.7182	0.0128
20%	0.9629	0.0606	0.091	0	24.6133	0.0116
24%	0.8499	0.0566	0.0768	0	24.5842	0.0146
25%	0.8587	0.0438	0.0758	0	24.5861	0.0051
30%	0.7321	0.0335	0.0644	0	24.4428	0.0105
35%	0.6575	0.0263	0.0616	0	24.3961	0.0051
40%	0.5405	0.021	0.0558	0	24.4961	0.027
45%	0.3897	0.0089	0.0499	0	24.3321	0.0091
49%	0.3235	0.0043	0.0873	0	24.4192	0.0183
50%	0.2925	0.0043	0.442	0.002	24.3991	0.0173

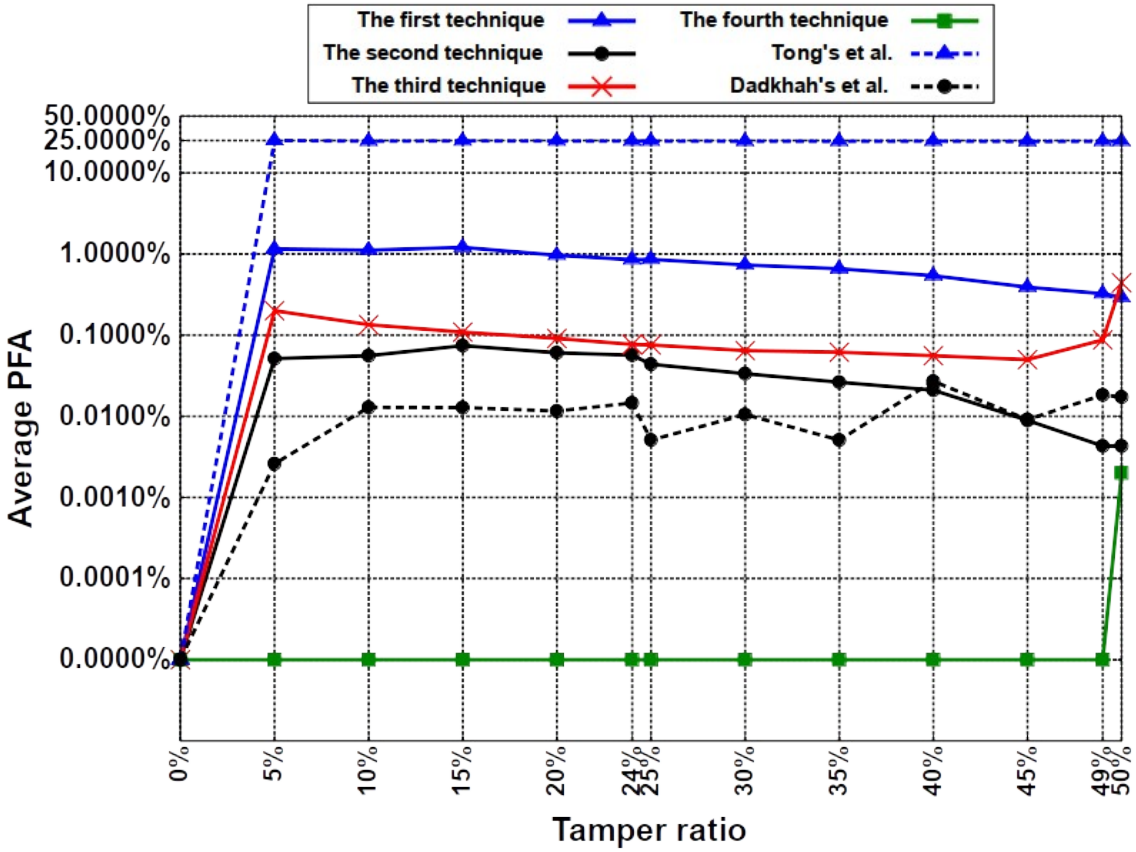


Figure 5.4: The experimental results for the average PFA for the proposed and the referenced techniques.

It is possible to draw the following remarks from the experimental results of the average PFA values:

- The best performance (i.e. lowest PFA) is for the fourth technique, and that is because of using 16 authentication bits, which means that $PFA \approx 2^{-16} \approx 0.0015\%$. In fact, no tampered blocks were missed except for one block in one image of the 24 images at 50% ratio and that results in a sudden increase in the value of PFA, nevertheless, the PFA values of the fourth technique are still the lowest.

- The performance of Dadkhah's technique comes in the second place because of using 12 authentication bits for each 4x4 px block, which should result in a PFA of about $2^{-12} \approx 0.024\%$; however, the experimental results are lower most of the time because of using the recovery information as a second authentication level, which reduces the PFA values.

- The second technique has a slightly better performance than the third technique because it uses a smaller filter window size of 3x3 px when

compared to 5x5 px in the third technique, and that gives more accurate results. Also the filtering method used in the second technique is more accurate than the one used in the third one. The PFA for the first technique is more than the second and the third one because of the lower accuracy of the filtering method used in it.

- Tong's technique has the worst values of the PFA which is about 25% and that agrees with the theoretical value of using 2 authentication bits for each block which gives $PFA \approx 2^{-2} = 25\%$.
- The fourth technique has an odd behaviour at tamper ratio of 50% because it happened that one tampered block in all of the images was detected as an authentic one. It means that 64 pixels out of (0.5 * 512 * 512 * 24) pixels, about 0.002%, are falsely accepted. This behaviour is expected when large number of images is tested even though the PFA is very low, which is about 0.0015%.
- The third technique has an odd behaviour at tamper ratio of 50%, where there is a sudden increase in the value of the PFA. This is because at that ratio and due to the use of filtering window, there will be tamper coincidence at the boundaries of the tampered area and that increases the PFA value.

It can be concluded from the PFA results that the fourth technique is the best technique regarding the PFA results, and Dadkhah's technique comes in the second place.

5.5- The Experimental Results for the Average PFR for the Proposed and the Referenced Techniques

The value of the PFR has less importance when compared to the PFA and less effect on the value of the PSNR. Therefore, a higher value of PFR is accepted when comparing to the PFA, especially when the reasons behind this high value are expected ones. As in the fourth technique, where some untampered pixels at the boundaries of the tampered area are included because of using a large block size (8x8 px in this case).

The average PFR values corresponding to each ratio are shown in Table 5.5 and

Figure 5.5. The following notes should be considered about Table 5.5 and Figure 5.5:

- The PFR was measured based on pixel level, as in Equation 5.2. This gives more accurate results.

$$PFR = \frac{\text{Number of authentic pixels detected as tampered ones}}{\text{Total number of authentic pixels}} \quad (5.2)$$

- The scale of the y-axis in Figure 5.5 is a logarithmic scale for the values greater than 0, this scale was selected because some values are very small and some are large.
- The values in Table 5.5 and Figure 5.5 are the percentage values, which means that they should be divided by 100 to get the actual values.

Table 5.5: The experimental results for the average PFR for the proposed and the referenced techniques.

Tamper Ratio	Average PFR %					
	Technique 1	Technique 2	Technique 3	Technique 4	Tong's Technique	Dadkah's Technique
0%	0	0	0	0	0	0
5%	0.4087	0.0902	0.1402	0.9501	0.0826	0.185
10%	0.8692	0.1345	0.0613	1.1413	0.0369	0.8501
15%	0.9135	0	0.3933	1.0191	0.253	0.3668
20%	1.459	0.2155	0.3277	3.1709	0.1936	0.6662
24%	1.7398	0.2482	0.2866	2.447	0.1574	1.2826
25%	1.7445	3.7208	0.2057	2.6111	0.0929	0.0034
30%	2.7099	15.9144	0.1307	2.503	0.0814	1.2348
35%	3.1464	26.1817	0.2765	3.0349	0.1921	0.3603
40%	3.9768	34.9109	0.6114	4.1192	0.3662	1.6812
45%	5.3081	43.3705	0.7067	6.1633	0.5149	0.481
49%	6.9284	49.996	0.6399	2.0923	0.3306	1.0859
50%	7.6475	51.5769	0.4325	0	0	1.1101

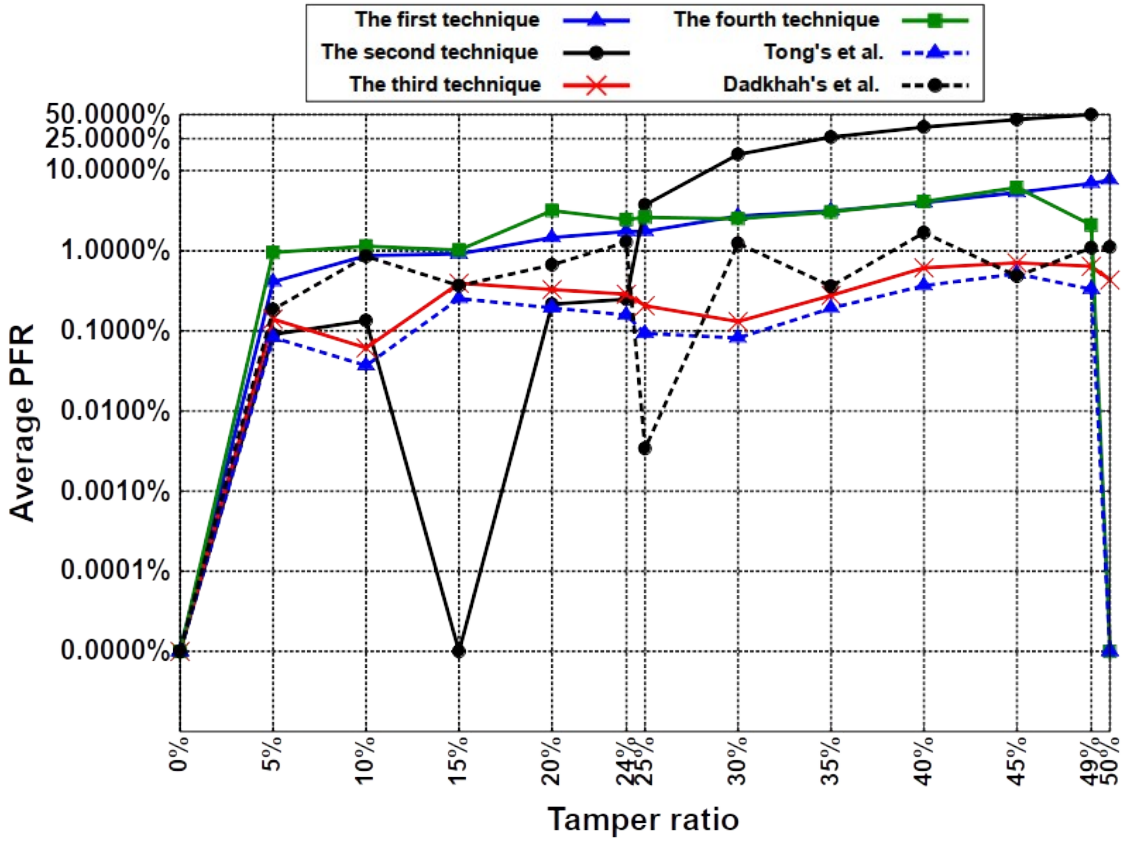


Figure 5.5: The experimental results for the average PFR for the proposed and the referenced techniques.

It is possible to draw the following remarks from the experimental results of the average PFR values:

- The main factor that affects the PFR values is the match between the boundaries of the tampered area and the boundaries of the image blocks. When a perfect match between the tampered area boundaries and the block boundaries occurs, this results in PFA = 0, this explains the odd behaviour of the second, and the fourth technique at 15% and 50% ratios respectively.
- Unlike the PFA, the number of the authentication bits has no effect on the PFR.
- The second technique has the worst results for ratios greater than 24% because of how the filter used for detection works.
- The third technique and Tong's technique seem to have best overall performance because of using 2x2 px block size, which provides better matching between the boundaries of the tampered area and the boundaries

of the blocks.

Tong's technique seems to have the best PFR performance, followed by the third technique, and that is because of using smaller block sizes. The values for other techniques (except the second one for tamper ratios greater than 24%) are acceptable, especially when the cause of the high PFR is understood.

5.6- The Experimental Results for the Encoding Time for the Proposed and the Referenced Techniques

For the sake of completeness, the average encoding time is included in the results. The time was measured on a ***2.10GHz Intel® Core™ i3-5010U CPU***. The average encoding time values corresponding to each ratio are shown in Table 5.6 and Figure 5.6. The following notes should be considered about Table 5.6 and Figure 5.6:

- The scale of the y-axis in Figure 5.6 is a piecewise linear scale, where the scale of the values greater than 1 is divided by 100. This scale was chosen to be able to show small and large values of encoding time.
- The following notes are also applied to the experimental results for the decoding time, which is presented in the following section.
    - It should be kept in mind that the measured encoding/decoding time values do not reflect the actual time of a real implementation of the techniques using C or C++ languages.
    - Best efforts were made to optimise the code and reduce the time, like using look-up-tables whenever it is possible; however, the code might not be the most optimised one and a lower encoding time is possible.
    - A variation in encoding/decoding time is expected, especially since the machine runs multiple tasks and other programs are using some of the machine processing power.

Table 5.6: The experimental results for the average encoding time for the proposed and the referenced techniques.

Tamper Ratio	Average encoding time (Seconds)					
	Technique 1	Technique 2	Technique 3	Technique 4	Tong's Technique	Dadkah's Technique
0%	0.3423	1.1563	1.1447	30.5612	0.364	21.085
5%	0.3418	1.1525	1.1437	30.7184	0.3644	21.259
10%	0.3434	1.1574	1.1442	30.6438	0.3667	21.1083
15%	0.3486	1.1701	1.1422	30.6307	0.3641	21.0224
20%	0.3408	1.1701	1.1438	30.6357	0.3654	21.2726
24%	0.3436	1.1791	1.1422	30.5662	0.3663	21.2725
25%	0.3428	1.1718	1.1426	30.6033	0.3643	21.1242
30%	0.3452	1.1731	1.1436	30.5604	0.3663	21.0909
35%	0.3419	1.1667	1.1431	30.5409	0.3647	21.0699
40%	0.3418	1.1561	1.1463	30.5153	0.363	21.1164
45%	0.3425	1.1566	1.1447	30.5821	0.3637	21.0282
49%	0.3434	1.1584	1.142	30.5837	0.3637	20.9926
50%	0.3458	1.1569	1.1464	30.6229	0.3647	20.9934

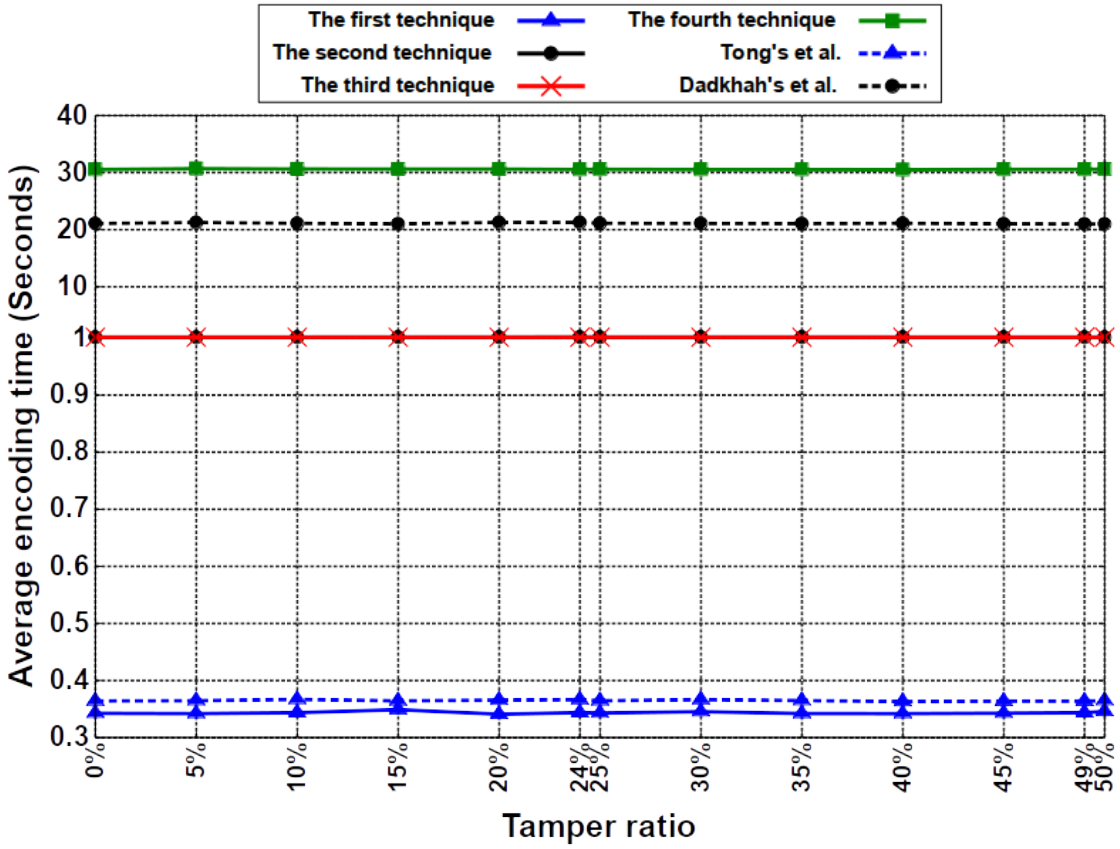


Figure 5.6: The experimental results for the average encoding time for the proposed and the referenced techniques.

It is possible to draw the following remarks from the experimental results of the average encoding time values:

- Because the images are not tampered yet, the encoding time is constant and does not depend on tamper ratio.
- The fourth and Dadkhah's techniques require significantly more encoding time because they perform the calculations on each block separately instead of applying them on the image as a whole, as other techniques do.

Both Tong's and the first techniques are considered to have the best performance, followed by the third technique. It should be remembered that the case might be different if a real implementation is carried out using C or C++ language, also, Tong's technique lacks many features, such as watermark encryption, which reduces it encoding time.

5.7- The Experimental Results for the Decoding Time for the Proposed and the Referenced Techniques

As was the case for the encoding time, the average decoding time is also included in the results. The time was measured on a ***2.10GHz Intel® Core™ i3-5010U CPU***. The average decoding time values corresponding to each ratio are shown in Table 5.7 and Figure 5.7. The following notes should be considered about Table 5.7 and Figure 5.7:

- The scale of the y-axis in Figure 5.6 is a piecewise linear scale, where the scale of the values greater than 10 is divided by 10. This scale was chosen to be able to show the small and large values of decoding time.
- The notes on the encoding time are also applicable here.

Table 5.7: The experimental results for the average decoding time for the proposed and the referenced techniques.

Tamper Ratio	Average decoding time (Seconds)					
	Technique 1	Technique 2	Technique 3	Technique 4	Tong's Technique	Dadkah's Technique
0%	3.7588	4.3592	1.5271	24.3584	1.0761	43.8284
5%	3.7652	4.3604	2.1044	24.7649	1.1384	44.2182
10%	3.7654	4.3758	2.6379	25.1408	1.2014	44.1729
15%	3.7565	4.3879	3.218	25.4676	1.2838	44.5587
20%	3.7625	4.3892	3.7854	25.97	1.3409	44.5534
24%	3.7637	4.4141	4.2148	26.2024	1.398	43.922
25%	3.763	4.4161	4.3183	26.2883	1.405	43.5972
30%	3.7603	4.3949	4.8715	26.7129	1.478	43.213
35%	3.752	4.366	5.4273	27.0353	1.5483	42.5625
40%	3.7593	4.3604	6.0311	27.4653	1.6251	41.4114
45%	3.7597	4.3628	6.5889	27.9208	1.7034	40.7209
49%	3.7808	4.3701	7.0108	28.1404	1.7657	39.6711
50%	3.8179	4.3601	7.098	28.1216	1.7707	39.4551

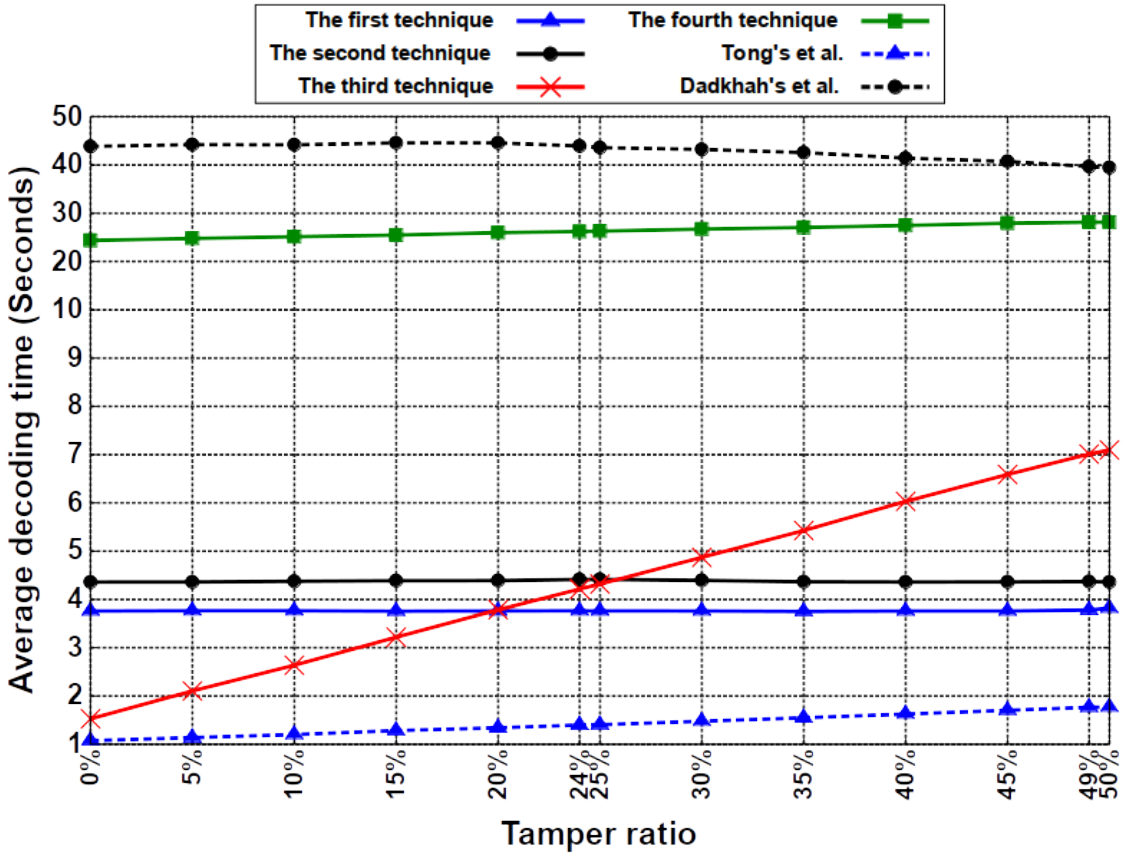


Figure 5.7: The experimental results for the average decoding time for the proposed and the referenced techniques.

It is possible to draw the following remarks from the experimental results of the average decoding time values:

- The fourth technique and Dadkhah's technique require significantly more encoding time because they perform the calculations on each block separately instead of applying them on the image as whole.
- The decoding time for the third technique increases linearly when the tamper ratio increases, this is because DCT calculations are required to distinguish each tampered block from its twin one. Therefore, more calculations are needed when the tampered area increases.
- There is a slight increase in decoding time for Tong's technique and the fourth technique because of the additional time needed to recover additional tampered area.
- The decoding time for the first and the second techniques seems to be constant for all tamper ratios and that is because the filtering used for localising the tampered area is applied for the whole image and the

recovery is done using a mask for the whole image and not for each block separately.

- There is a slight decrease in the decoding time for Dadkhah's technique as the tamper ratio increases, this could be due to the second authentication level that requires comparison with the recovery data for each block and which is skipped when the block is tampered and that results in a slightly less decoding time.

Tong's technique seems to be the best when it comes to decoding time. It should be remembered that the case might be different if a real implementation is carried out using C or C++ language, also, Tong's technique lacks many features, such as watermark encryption, which reduces it decoding time.

5.8- The Effect of Image Size on the Evaluated Parameters

Increasing the image size will not have any effect on the efficiency of the detection and the recovery for the proposed and referenced techniques because they are based on block-wise operations, also, the required encoding and decoding time has a linear relationship with image size (i.e. number of blocks in the image). Tables 5.8 and 5.9, and Figures 5.8 and 5.9 illustrate this linear relationship, they show the required encoding and decoding time for the lighthouse image when its size is increased from 120x120 px to 1200x1200 px in step of 120 px. 120x120 px was selected because it is accepted by all techniques. The tampering was done by replacing a squared area that has 45% of image width with the content from "peppers" image, both images were scaled before applying the tampering. The image size is measured in megapixels which is found by multiplying the image width by its height and dividing the result by 1 million. The time was measured on a ***2.10GHz Intel® Core™ i3-5010U CPU***.

Table 5.8: Encoding time vs image size for lighthouse image for the proposed and the referenced techniques.

Image size (megapixel)	Encoding time (Seconds)					
	Technique 1	Technique 2	Technique 3	Technique 4	Tong's Technique	Dadkah's Technique
0.0144	0.0797	0.0864	0.0706	1.7007	0.0971	1.3252
0.0576	0.1536	0.2659	0.2604	6.7652	0.1666	5.3916
0.1296	0.2329	0.5737	0.5698	15.2041	0.2534	12.0256
0.2304	0.3151	1.0198	1.0214	26.9472	0.346	21.4887
0.36	0.3991	1.5753	1.5883	42.2893	0.4359	33.4364
0.5184	0.4887	2.2849	2.2726	60.8669	0.5333	48.5468
0.7056	0.5763	3.0986	3.1133	83.1175	0.6367	65.4512
0.9216	0.6776	4.0488	4.0767	108.3992	0.7353	85.0598
1.1664	0.7701	5.102	5.1179	137.5072	0.8412	108.0352
1.44	0.8655	6.374	6.3696	169.5271	0.9471	132.193

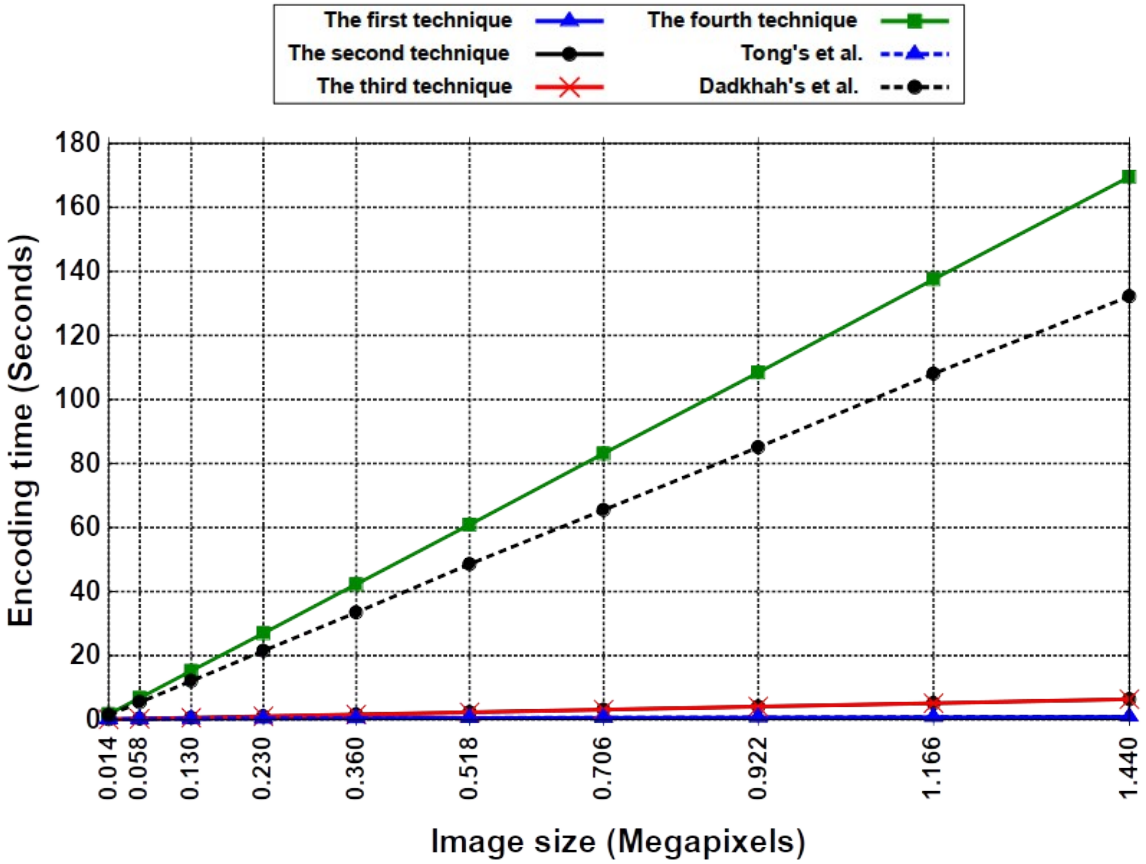


Figure 5.8: Encoding time vs image size for lighthouse image for the proposed and the referenced techniques.

Table 5.9: Decoding time vs image size for lighthouse image for the proposed and the referenced techniques.

Image size (megapixel)	Decoding time (Seconds)					
	Technique 1	Technique 2	Technique 3	Technique 4	Tong's Technique	Dadkah's Technique
0.0144	0.292	0.2716	0.2278	1.3459	0.2029	2.7718
0.0576	0.8971	0.9706	0.8325	5.6841	0.4624	11.101
0.1296	1.899	2.1135	1.854	11.9352	0.8281	25.2615
0.2304	3.2726	3.7342	3.277	22.728	1.233	44.8969
0.36	5.0424	5.785	5.1331	33.2008	1.7412	70.0066
0.5184	7.1213	8.3892	7.3264	51.2539	2.2723	99.8919
0.7056	9.6382	11.4152	10.022	65.0574	2.9213	136.2091
0.9216	12.5026	14.9404	12.9928	90.4702	3.6301	178.5067
1.1664	15.7116	18.7736	16.6517	108.16	4.416	225.2395
1.44	19.3861	23.3549	20.3933	141.586	5.2669	277.2823

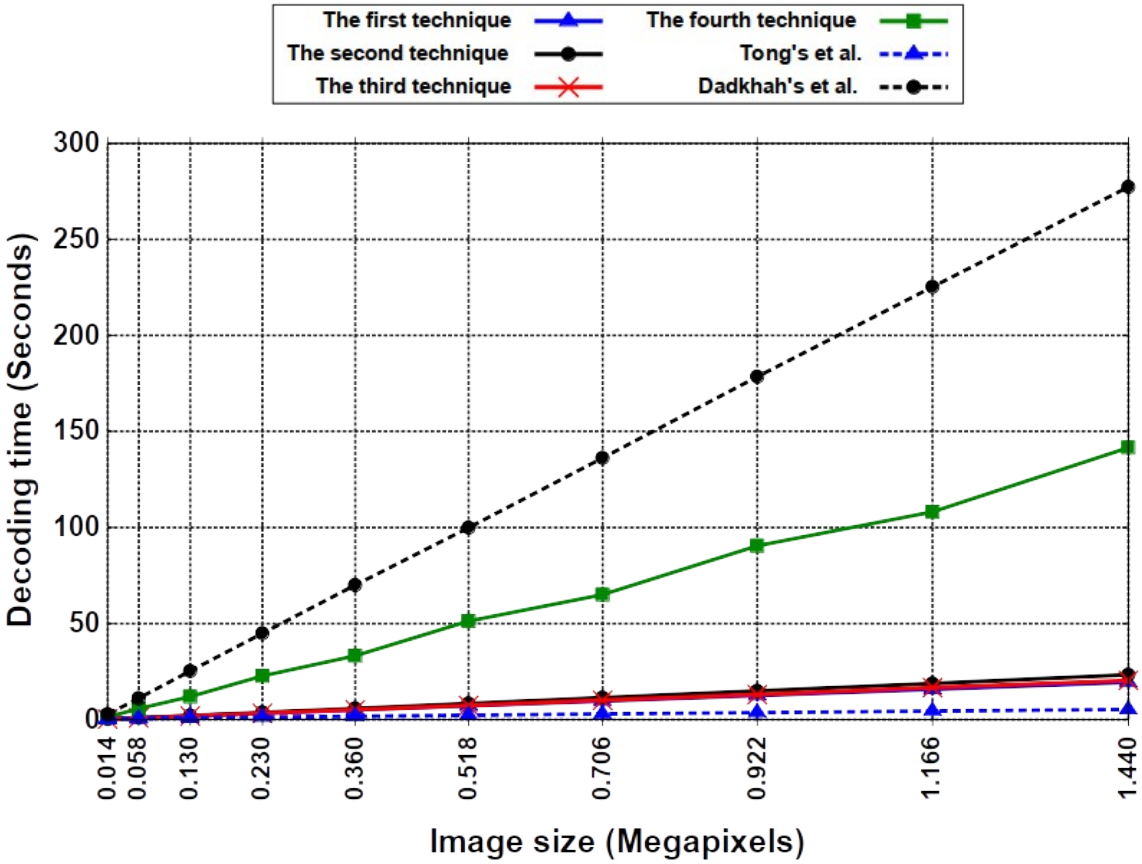


Figure 5.9: Decoding time vs image size for lighthouse image for the proposed and the referenced techniques.

5.9- Summary

This chapter presented an experimental evaluation of the proposed techniques and compared them to the referenced techniques. The evaluation was done for the following parameters: recovery quality (measured by PSNR, NCC, and SSIM), PFA, PFR, and encoding/decoding time. The effect of image resolution on encoding/decoding time was also measured. In general, the proposed techniques had better performance and the fourth proposed technique had the best general performance among the proposed and the referenced techniques.

The performance of the second technique decreases dramatically for tamper ratios equals to or greater than 25% and that is because of the nature of the used localisation filter. For tamper ratios below 25%, the second technique has approximately the same quality performance as the third one because of having the same recovery data generation strategy.

A hypothetical perfect recovery scenarios were tested for Tong's and Dadkhah's technique to show how the proposed techniques have better performance even when the referenced techniques have a perfect performance.

The PFA depends generally on the performance of the localisation filter in the case of the first, the second, and the third techniques, and it depends on the number of localisation bits for the other techniques. For this reason the fourth technique has the best PFA values followed by Dadkhah's techniques because they have 16 and 12 localisation bits respectively. The worst PFA values are for Tong's technique because it has only 2 localisation bits for each block.

The PFR depends mainly on the matching between the boundaries of the tampered area and the block partitioning, this is why the PFR drops to 0 whenever a perfect match occurs.

The required encoding/decoding time for the fourth and Dadkhah's techniques is significantly higher than the other techniques. This is because they require more calculations and these calculations are applied for each block separately instead of doing them to the whole image at once.

Because the proposed and reference techniques are based on block-wise operations, the increase in the image size has no effect on the recovery performance. Also, there is a linear relationship between the image size and encoding/decoding time of the proposed and the referenced techniques.

Chapter 6: Discussion, Future Work, and Conclusion

This chapter presents the final discussion about the advantages and shortcomings of the proposed techniques, and which technique has the most optimised performance. The possible future work is also presented in this chapter, followed by the final conclusion of this research.

6.1- Discussion

This section discusses the proposed techniques, their advantages and shortcomings and the technique that gives the best performance.

6.1.1- Limitations on Tampering Pattern

The use of maximum-distance mapping in the proposed techniques, except for the first one, gives them the ability to recover a large tampered area without tamper coincidence (up to 50% in the third and the fourth techniques); however, if any tamper exceeds half of both image dimensions, such as a diagonal line from top-left corner to bottom-right corner, it will not be fully recovered even if the tampered area is a small fraction of the image. Some techniques in the literature [57, 65] avoid this problem by making the recovery data consist of a linear combination of the information all over the image and solving this linear system for unknown variables due to tampering. However, the proposed techniques have less computational complexity, and maximum distance mapping gives acceptable results for most practical cases where the tampering usually affects a limited portion of the image.

6.1.2- Performance of the Proposed Techniques for Noise Tampering

The proposed techniques, except the fourth one, use image filtering for detecting the tampered area, and they rely on the assumption that the tampering is applied to a contiguous part of the image and that the tampering is not scattered as in the case of salt and pepper noise.

In the following example, the lighthouse image was tampered by adding salt and pepper noise with 0.1 density to the central part of the image as in Figure 6.1. The detected tampered area for the four proposed techniques is shown in Figure 6.2. A border line is drawn around the images to show their boundaries. The recovered image is shown in Figure 6.3. The fourth technique has the best performance in detecting and recovering all of the tampered blocks, while the performance of other techniques varies from detecting none of the tampered area, for the first technique, to some of it, for the second one, to most of it, for the third one.

Figure 6.1: The lighthouse image with salt and pepper noise tampering.

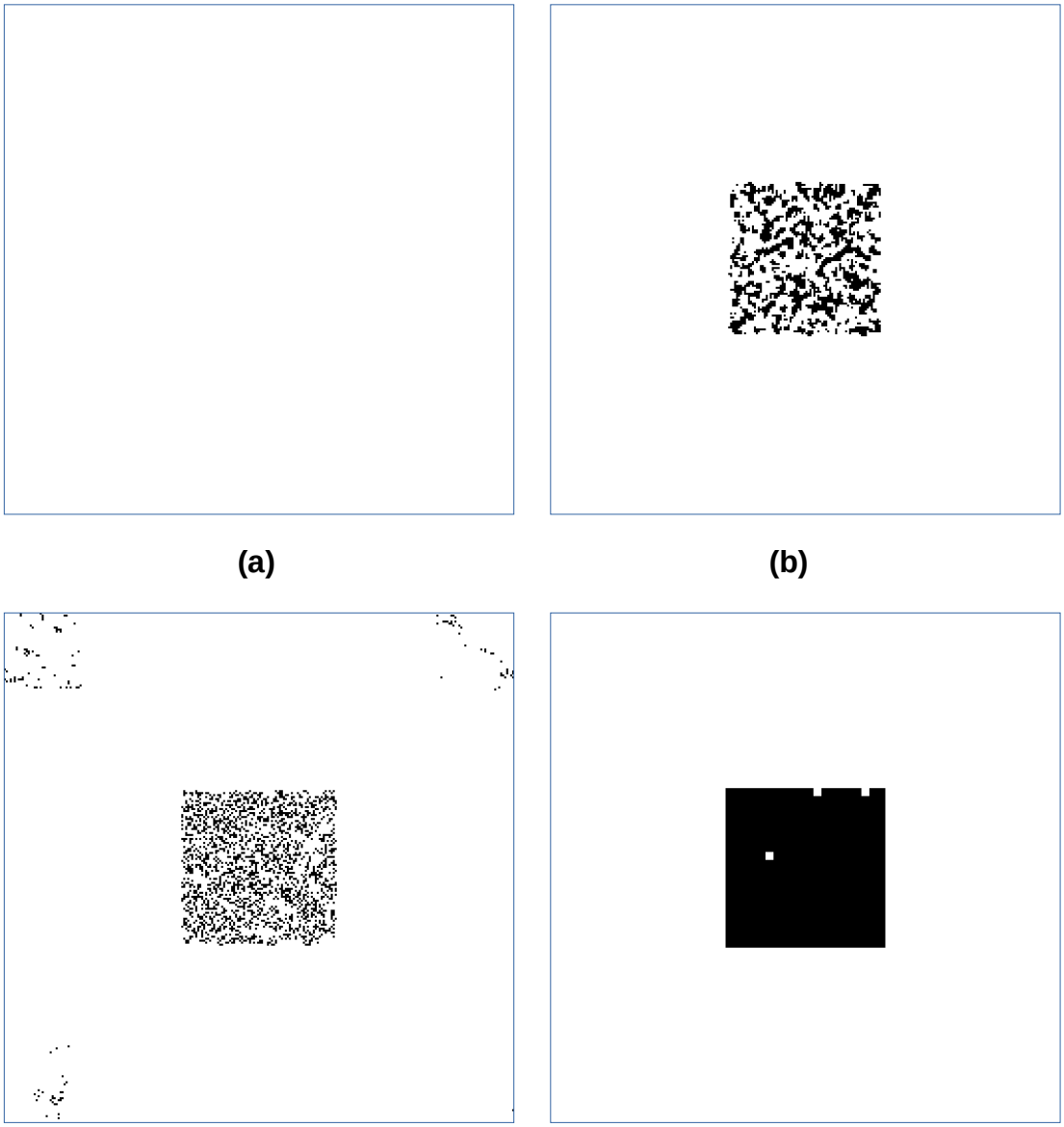


Figure 6.2: The detected tampered area (in black) for noise tampering in the four proposed techniques: a) The first technique. b) The second technique. c) The third technique. d) The fourth technique.

(a) **(b)**

(c) **(d)**

Figure 6.3: The recovered image for noise tampering for the four proposed techniques: a) The first technique. b) The second technique. c) The third technique. d) The fourth technique.

6.1.3- Performance of the Proposed Techniques for Black and White (B&W) Text-Images

This section illustrates the performance of the proposed techniques for black and white (B&W) text-images. The top-left area of the 528x528 px image shown in Figure 6.4 is tampered as shown in Figure 6.5. The text and the rose shape are replaced to show how effective are the proposed techniques in recovering B&W text and shapes.

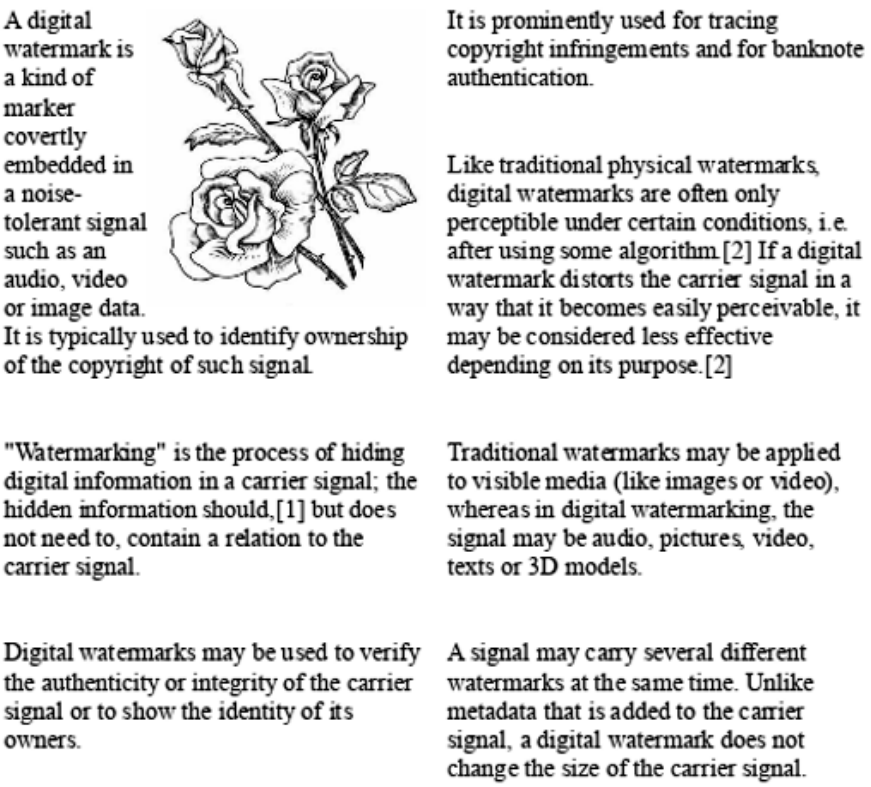


Figure 6.4: The used image for testing the performance of the proposed techniques for recovering B&W text images.

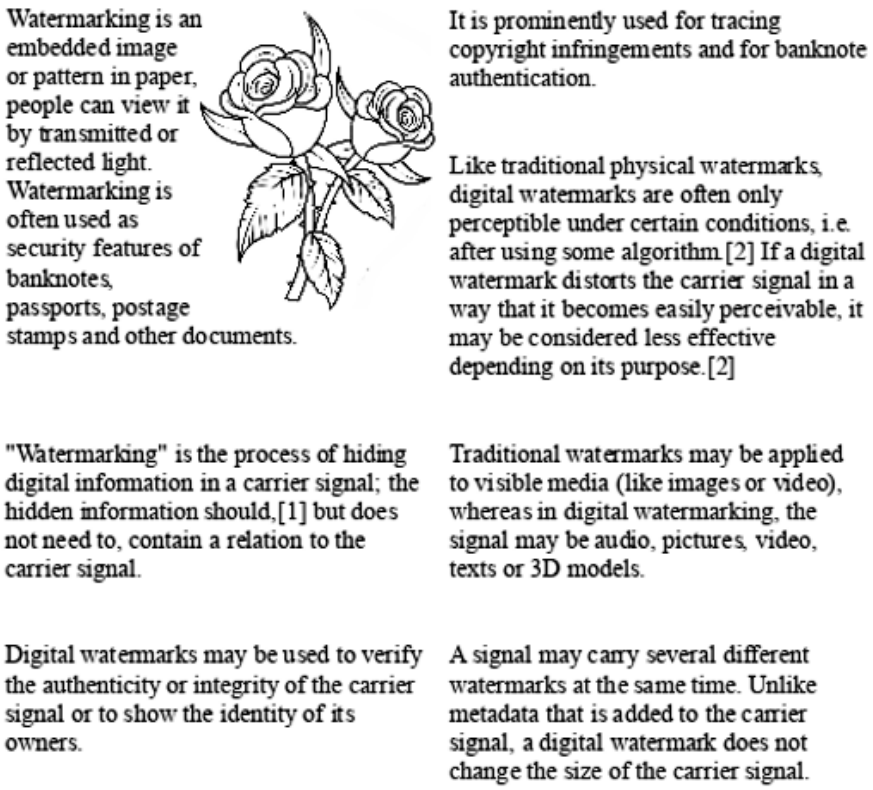


Figure 6.5: The tampered image used for testing the performance of the proposed techniques for recovering B&W text images. (The tampering text is taken from [110])

The detection results for the proposed techniques are shown in Figure 6.6, where the detected tampered area is shown in black. A border line is drawn around the images to show their boundaries. The recovered image is shown in Figure 6.7. The PSNR values are: 19.78 dB, 23.07 dB, 21.45 dB, and 26.54 dB for the techniques 1, 2, 3, and 4 respectively.

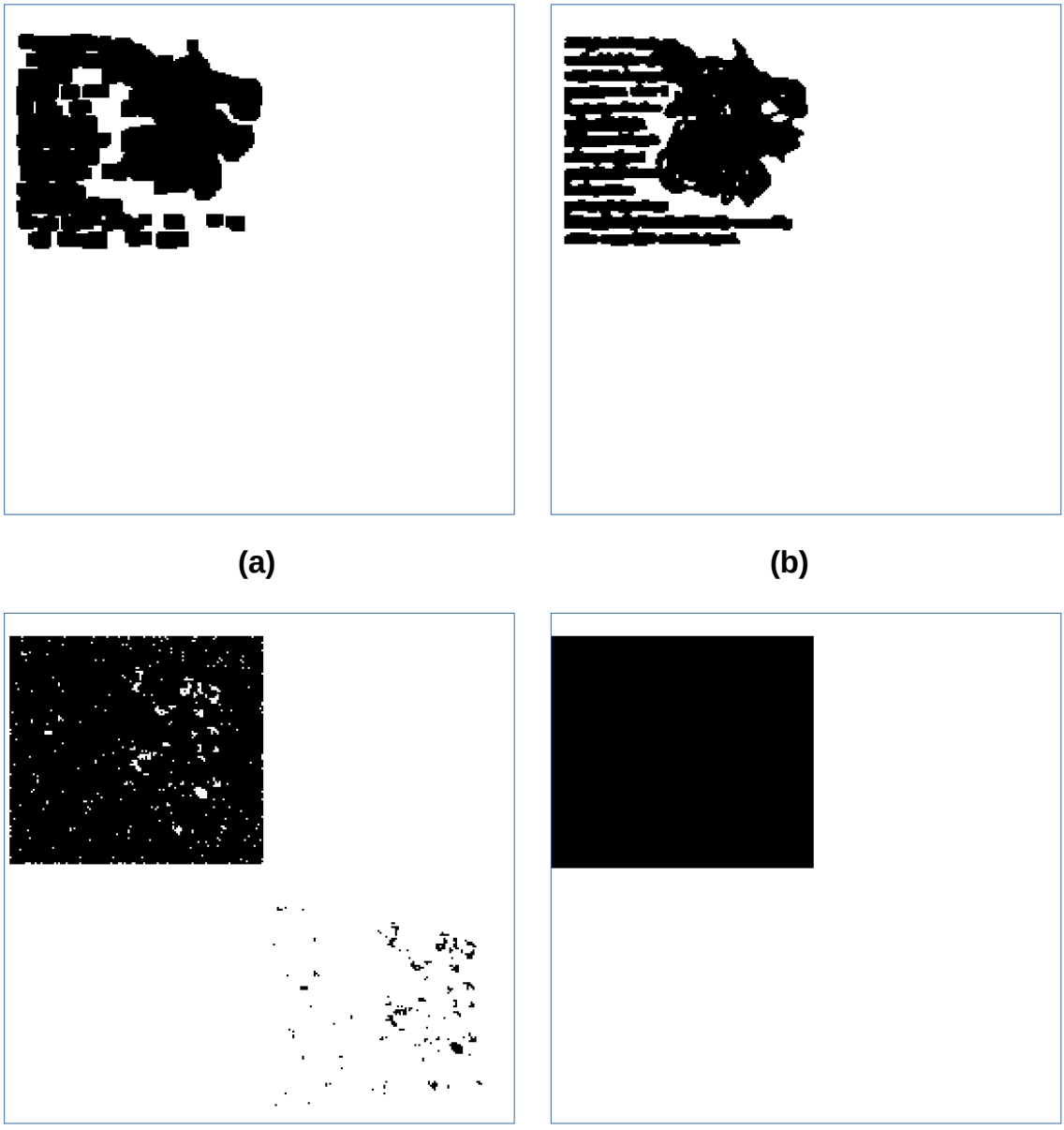


Figure 6.6: The detected tampered area for the text-image: a) The first technique. b) The second technique. c) The third technique. d) The fourth technique.

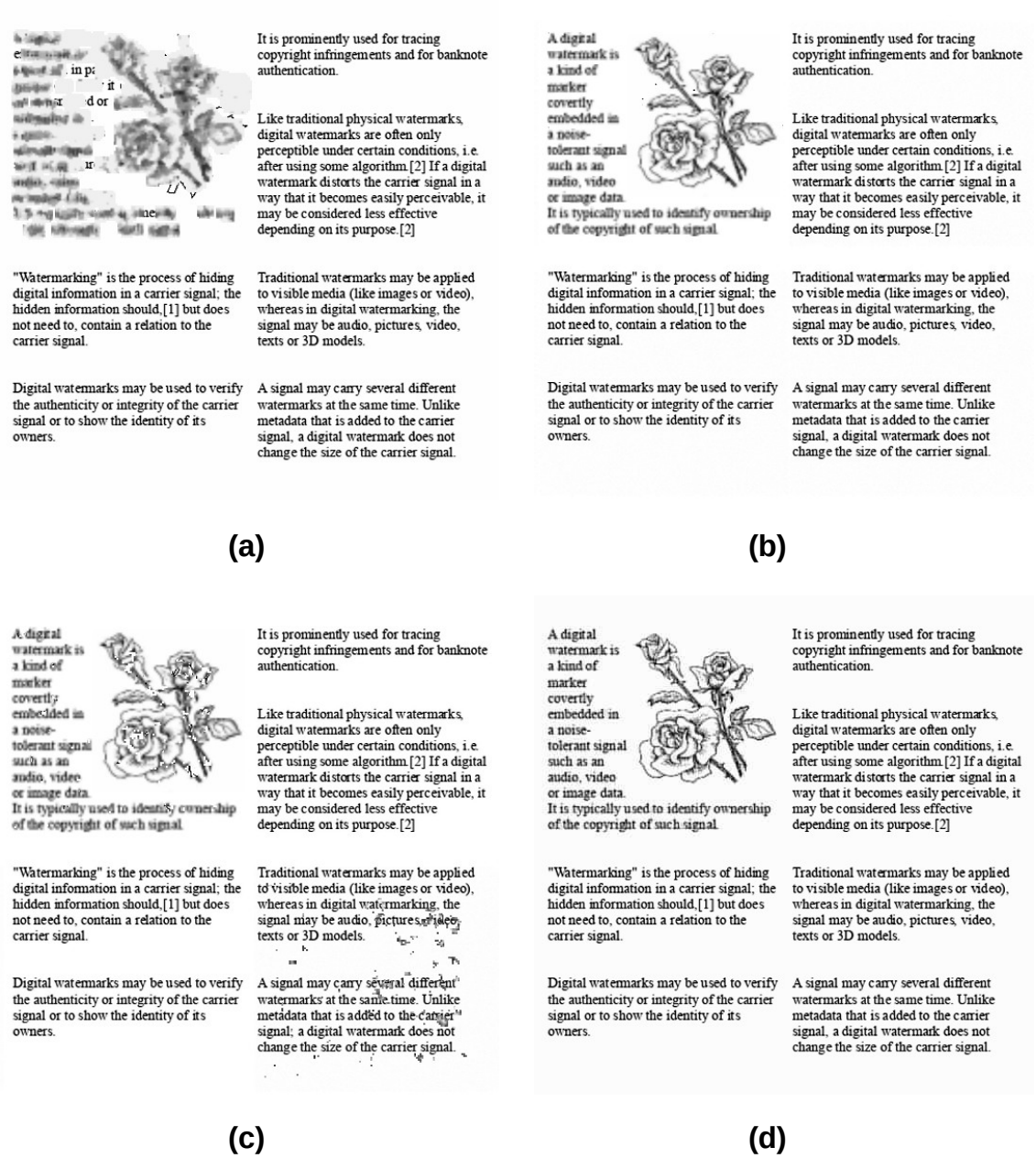


Figure 6.7: The recovered tampered area for the B&W text-image: a) The first technique. b) The second technique. c) The third technique. d) The fourth technique.

From previous figures and PSNR values, the fourth technique shows the best performance for recovering B&W text and shapes.

6.1.4- Performance of the proposed techniques against counterfeiting

Except for the fourth technique, the proposed techniques validate the average of 2x2 px blocks, which means that any block could be replaced by another that produces the same average, also, the exchange of the pixels inside the block will not be detected. On the other hand, the fourth technique is very robust against this

kind of tampering due to the use of CRC in the authentication process.

6.1.5- The Recommended Proposed Technique

From the analysis of the proposed techniques, it can be seen that the fourth technique has the best performance in terms of the recovered image PSNR and PFA, and it has an acceptable level of PFR. It also has the best performance under noise tampering and the best resiliency for counterfeiting; therefore, it is the recommended technique among the other proposed ones.

All of the four techniques have been discussed and analysed in this thesis, even though the fourth technique has the best performance and it is the recommended one to be used. The four techniques were investigated and presented in this research for the following reasons:

- To show the development of the ideas which led to a better technique. The first technique was the first developed one, then each technique came to provide more enhancements on the performance of the previous one.
- The main purpose of the first three techniques is to investigate the possible ways and the limitations on using one data set for both purposes of localisation and recovery. Presenting all of the investigated techniques will be very helpful for any further research to be taken in this direction.
- Proposing the fourth technique comes to aid the main goal of the research, which is to provide an optimised and high performance tamper localisation and recovery watermarking technique. One idea for achieving this goal was investigated by the first three techniques, which is having one set of data for localisation and recovery purposes. The other direction was investigated by the fourth technique, which is having a larger block size with the use of DCT as recovery data generation method.
- Despite the fact that the first and the second techniques have low performance when compared to fourth or even the third one; however, each technique has unique ideas that worth mentioning and investigating. For example, the use of three directional mapping in the second technique and the localisation filter used in the first one.

The following table shows the pros and cons of each technique and the kind of

applications that it could be used for:

Technique 1	**Pros**	• Insensitive to the tampering pattern. • More robust against VQ attack due to random mapping. • Consume only one LSB of the image. • Has low encoding/decoding time.
	Cons	• Recovery quality drops significantly as tampering ratio increases. • The recoverable tampered area with acceptable quality is smaller than other techniques. • Less resistive to counterfeiting than the fourth technique.
	Applications	• Good for application where the expected tamper ratio is small and the quality of the recovery is not a priority. This could be the case with personal images.
Technique 2	**Pros**	• Constant encoding/decoding time. • Low encoding/decoding time when compared to the fourth technique.
	Cons	• More sensitive the tampering pattern than other techniques. • The maximum recoverable tampering ratio is small, which is about 25% of image area. • Less resistive to counterfeiting than the fourth technique.
	Applications	• In general, the use of the third or the fourth technique is more preferred than the second one; however, it could be used for applications where the loss of information in the image is not a problematic issue and the expected tampering ratio is small and contiguous, such as personal images.
Technique 3	**Pros**	• Low encoding/decoding time when compared to the fourth technique. • Ability to recover large tampered area, up to 50% of the image area.
	Cons	• Sensitive the tampering pattern. • Less resistive to counterfeiting than the fourth technique.
	Applications	• Could be used for applications where the lost of information in the image is not a problem and the expected tampered area is contiguous, such as personal images.
Technique 4	**Pros**	• Higher recovery quality. • More resistive to counterfeiting when compared to the other techniques. • More secure than the other techniques. • Ability to recover large tampered area, up to 50% of the image area.
	Cons	• Sensitive the tampering pattern. • Has higher encoding/decoding time when compared to the other techniques.
	Applications	• Could be used for applications where higher quality recovery is required and the tampered area is expected to be contiguous, such as medical and forensic images.

6.2- Future Work

This section presents some future work that could be done based on the work in this research.

6.2.1- Solving the Problem of Sensitivity to Tampering Pattern

In this research, maximum-distance mapping was used to increase the recoverable area size while keeping low computational complexity; however, maximum-distance mapping introduces the problem of sensitivity to tampering pattern, where some tampering patterns are not recoverable even if they occupy a small fraction of the image area, such as a diagonal line from one corner of the image to the other. This problem could be solved by converting the information in the image blocks into a linear system and storing the results of this system into the watermark, in this way, the recovery will depend on the tampering ratio regardless of the pattern it takes. One major problem with this solution is the long time required for solving the large linear system it produces; therefore, research could be carried out to investigate computation time reduction.

6.2.2- Focusing Research Direction on Semi-Fragile Techniques that Work in Conjunction with Lossy Compression Techniques such as JPEG and JPEG2000

Fragile watermarking has some advantages, such as higher embedding capacity and more tamper detection sensitivity; however, it has one major disadvantage which is the inability to use lossy compression techniques with it and the necessity of using lossless compression techniques to store the watermarked images. Since the main advantage of using lossless techniques is to preserve the exact values of image pixels, which have been altered by the watermark, this makes using lossy compression more practical.

Semi-fragile watermarking is more practical for everyday applications, especially when it works in conjunction with lossy compression techniques such as JPEG and JPEG2000; however, the research taken for fragile watermarking is still helpful because it gives a good insight and platform for the operation of the semi-fragile techniques.

6.2.3- Applying Public-Key Encryption to Secure the Watermark

Using private key to encrypt the watermark imposes a security problem when it comes to exchanging this private key between the intended parties of communication. Therefore, public-key encryption is necessary to avoid knowing the private secret key by intruders when it being exchanged.

6.2.4- Applying Lossless Compression Techniques to the Watermark

Using lossless compression techniques, such as entropy coding, helps in reducing the watermark size and enabling using less LSBs to store it, or storing more watermark data in the same number of LSBs to get a better recovery quality. This could be applied particularly for the fourth technique, because of the flexibility of adding or removing some DCT coefficients to the watermark data without affecting the operation of technique.

6.2.5- Extension of the Proposed Techniques to Coloured Images

A direct application of the proposed techniques to coloured images is done by applying them separately to the RGB layers of the image; however, some steps might be taken to provide better performance for coloured images. For example, in the fourth technique the image could be converted to YCrCb colour space. Less LSBs could be taken from the Y channel and more LSBs could be taken from Cr and Cb channels because the Y channel has more information when compared to the Cr and the Cb channels. More LSBs could be assigned to the Y channel since it has more information and hence needs more space. Also the 16 bit detection data could be assigned for all of YCrCb channels instead of assigning 16 bit for each channel.

For the other three techniques, the image could be converted to YCrCb colour space and a larger block size, such as 4x4 px, could be assigned to Cr and Cb channels, this will reduce the size of the watermark. Also, since all of the colour layers are tampered in the same place, the detection could be improved by investigating using the detection results from all three channels to improve the detection rate for the tampered area.

6.3- Conclusion

Image watermarking plays an important role in many applications, such as: copyright tracking, hardware control, multimedia authentication and data hiding. Image watermarking techniques are divided into three categories based on their ability to resist attacks, i.e. modifications on the watermarked image. The first category is robust watermarking, which resist severe attacks, such as geometrical attacks, along with other less severe attacks, such as low pass filtering and lossy compression. Robust watermarking is used mainly in copyright tracking. The second category is semi-fragile watermarking, which resist less severe attacks, such as low pass filtering and lossy compression. The third category is fragile watermarking, which is characterised by its sensitivity to any modification to the image and its high capacity. Therefore, it is used mainly in authentication applications which include tamper localisation and recovery.

After conducting a general literature survey in the field of image watermarking; tamper localisation and recovery was selected to be the topic of this research. Tamper localisation and recovery attracted this research because it is an active and important topic in image watermarking and it is needed in many fields, such as medical and forensic applications. The images used in these applications contain critical information that attracts the intruders to tamper with and counterfeit them to gain some benefits or avoid some legal complications. It is very important to ensure the ability of localising and recovering any tamper that might be applied to these images; therefore, image watermarking is used to solve this issue.

Generally, the process of tamper localisation and recovery starts by dividing the image into non-overlapping blocks of nxn px and reserving some LSBs in each block to embed the watermark into them. Two separate sets of data are usually generated from each block, one set is used for localisation and it is stored in the same block. The other set is used for recovery and it is mapped in another block in the image, preferably as far as possible from the block from which the recovery data is generated. Tamper coincidence occurs if the block from which the recovery data is generated and the block in which it is stored are both tampered.

A literature survey was conducted in fragile watermarking techniques that are used in tamper localisation and recovery, and that is to determine the different methods,

shortcomings, trends, and issues in this field. Some of the shortcomings that found in the techniques proposed in the literature are:

- The use of two separate sets of data in the watermark, one set is used only for localisation while the other set is used only for recovery. Since both sets are related the data in their block, and they contain information that could be helpful for both purposes of localisation and recovery; it is a wast of data to use each one in only one purpose and it is more optimal to have one set of data that is used for both purposes of localisation and recovery. Another disadvantage of using two separate sets is that few number of bits are assigned for localisation data, which results in high PFA, which in turn results in low recovery quality. The PFA approximately equals to $2^{\text{Number of Localisatoin Bits}}$.
- Using multiple copies of recovery data, which is used to increase the recoverable area ratio; however, this increases the watermark-data waste problem.
- Complicating the encoding/decoding design without introducing significant improvement, such as using hierarchical authentication and hierarchical partitioning.
- Using few number of localisation bits, which results in large PFA values, which results in low recovery quality.

After reviewing the shortcomings of the techniques proposed in the literature, four techniques were proposed in this research. Each one of the proposed techniques gives some enhancements over the techniques proposed before. All of the proposed techniques are presented in this research in order to show their development and to present the ideas in each one.

Three of the four proposed techniques investigate the possibility of using one set of data for both purposes of localisation and recovery. In these techniques, the watermark image, which represents the recovery data, is compared directly with the watermarked images and the resulting difference image is treated with image filtering in order to extract a mask image that determines the location of the tampered area. The fourth technique uses two separate sets of data and it is optimised to give better performance in terms of recovered image quality and

resistance to counterfeiting.

A brief description of the four proposed techniques is presented as follows:

- In the first technique, the image is partitioned into 2x2 px blocks and one LSB is reserved for the watermark image, i.e. the recovery data, which is mapped randomly in the LSB. The watermark is secured by XORing it with a random sequence and using a secret key as the seed for randomisation. At the decoding stage, the difference image between the watermark image and watermarked image is found using XOR operation, and a threshold is applied to the resulting difference image so that any pixel that has a value greater than 0 will become 255. Spatial filtering is applied to the difference image to locate the tampered area. The spatial filter returns white, i.e. 255, if the ratio of white pixels inside the filter window is more than 70%. The resulting image of the filtering process is called the mask image and the white pixels in it correspond to the tampered area. Median filtering is used to enhance the watermark image before using it to recover the tampered area. The first technique is not sensitive to the tampering pattern; however, the recovered area quality degrades as the tampered area ratio increases.
- In the second technique, the image is partitioned into 2x2 px blocks and two LSBs are reserved for the watermark image, which is divided into 3 interleaved pixel groups. The three pixel groups are mapped based on maximum-distance mapping and they are mapped in the horizontal, the vertical, and the diagonal directions. The watermark is secured by XORing it with a random sequence and using a secret key as the seed for randomisation. The bits in each pixel are also permuted randomly based on the same secret key. The difference image between the watermark image and the watermarked image is found as in the first technique; however, the mask image is generated from the difference image using a filtering window of size of 3x3 px which returns white if the number of white pixels in it is more than 3. The technique guarantees the full recovery of any tampered area as long as it does not exceed half the width or half the height of the image, hence, the recoverable tampered area ratio reaches 25% of the image area.

- In the third technique, the image is partitioned into 2x2 px blocks and two LSBs are reserved for the watermark image, which is mapped diagonally using maximum-distance mapping. The watermark is secured and the bits in each pixels are permuted as in the second technique. Also, the difference image is generated as in the first and the second technique. The difference image will have two twin areas, one corresponds to the tampered area in the watermarked image and the other corresponds to the tampered area in the watermark image. In the watermark image, the tampered area will appear as a noisy area with high frequency contents, and that is because of XORing the watermark image with the random sequence to secure it. The mask image is generated from the difference image by selecting the area that corresponds the lower frequency contents in the watermark image. The frequency contents are measured by taking 5x5 px window and finding its DCT coefficients, then the first row and column of the DCT coefficients matrix are set to 0 and the sum of the absolute values of the remaining coefficients is found. The technique guarantees the full recovery of any tampered area as long as it does not simultaneously exceed half the width and half the height of the image, hence, the recoverable tampered area ratio reaches 50% of the image area.
- In the fourth technique, the image is partitioned into 2x2 px blocks and two LSBs are reserved for the watermark image, which is mapped diagonally and secured as in the third technique. Also, the bits in each block are permuted randomly based on the same secret key. In each block, 16 bits are dedicated for localisation and calculated using CRC16, while 112 are used to store the recovery data. The recovery data is generated by quantising the DCT coefficient matrix of the block. The fourth technique shows better recovery quality because of using larger block size and using DCT to generate the recovery data. As in the third technique, the fourth technique guarantees the full recovery of any tampered area as long as it does not simultaneously exceed half the width and half the height of the image.

The proposed techniques were compared to two techniques from the literature and showed better performance in term of recovery quality and PFA. The fourth

technique showed the best performance with average PSNR value of 42.64 dB for untampered image and from 36.84 dB to 29.07 dB for tamper ratios of 5% to 50%, and it has average PFA value of 0.00153%.

Some of the limitations of the proposed techniques are:

- Except for the first technique, the techniques are sensitive to the tampering pattern. This sensitivity is because of using maximum-distance mapping, which makes some patterns unrecoverable even if they occupy small percentage of the image, such as a diagonal line from top-left corner to bottom-right corner of the image.
- Except for the fourth technique, the techniques show less performance in detecting and recovering noise tampering. The fourth technique detects and recovers any noise tampering just like any other tempering; however, the other techniques miss some noise. The fourth technique shows the best performance, followed by the third, then the second. The first technique gives the worst performance regarding noise tampering.
- Except for the fourth technique, the techniques show less performance in detecting and recovering tampered text images. The fourth technique gives the best results regarding the detected area and the recovery quality, followed by the second, then the third. The first technique gives the worst results.

It is advisable to use the fourth technique for different applications, especially for sensitive ones that require higher recovery quality and more counterfeiting resistivity, such as medical and forensic applications. The other three techniques could be used for less sensitive applications, such as personal images.

The future work for this research includes:

- Solving the problem of sensitivity to tampering pattern by storing the recovery information as a linear system of equations. Recovery is curried out by solving this system for any missing variable that reflects a missing block due to tampering.
- Redirecting research toward semi-fragile techniques that work in conjunction with lossy compression techniques such as JPEG.

- Implementing public-key encryption to secure the watermark.
- Implementing lossless compression techniques to reduce the data size of the watermark.
- Extending the proposed techniques to work with coloured images.

References

[1] F. Y. Shih, *Digital watermarking and steganography: fundamentals and techniques*. Boca Raton, FL: Taylor & Francis, CRC Press, 2017.

[2] I. J. Cox, M. L. Miller, and J. A. Bloom, *Digital watermarking and steganography*. San Francisco: Morgan Kaufmann, 2008.

[3] A. Tefas, N. Nikolaidis, and I. Pitas, "Image Watermarking," *The Essential Guide to Image Processing*, pp. 597–648, 2009.

[4] H. Tao, L. Chongmin, J.M. Zain, and A.N. Abdalla, "Robust image watermarking theories and techniques: A review," *Journal of applied research and technology*, vol. 12, no. 1, pp.122-138, 2014.

[5] D. Kannan and M. Gobi, "An extensive research on robust digital image watermarking techniques: a review," *International Journal of Signal and Imaging Systems Engineering*, vol. 8, no. 1/2, pp.89-104, 2015.

[6] V. Licks and R. Jordan, "Geometric attacks on image watermarking systems, " *IEEE multimedia*, vol. 12, no. 3, pp. 68-78, 2005.

[7] D. Zheng, Y. Liu, J. Zhao, and A.E. Saddik, "A survey of RST invariant image watermarking algorithms," *ACM Computing Surveys (CSUR)*, vol. 39, no. 2, p. 5, 2007.

[8] 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 Transactions on image processing*, vol. 10, no. 5, pp.767-782, 2001.

[9] D. Zheng, J. Zhao, and A. El Saddik, "RST-invariant digital image watermarking based on log-polar mapping and phase correlation," *IEEE transactions on circuits and systems for video technology*, vol. 13, no. 8, pp.753-765, 2003.

[10] K. R. Rao, D. N. Kim, and J. J. Hwang, *Fast Fourier transform: algorithms and applications*. Dordrecht: Springer, 2010.

[11] A. D. Poularikas, *Transforms and applications handbook*. Boca Raton: CRC Press, 2011.

[12] M. Alghoniemy and A. Tewfik, "Geometric distortion correction through image normalization," *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532).*

[13] P. Dong, J.G. Brankov, N.P. Galatsanos, Y. Yang, and F. Davoine, "Digital watermarking robust to geometric distortions," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp.2140-2150, 2005.

[14] E. D. Tsougenis, G. A. Papakostas, and D. E. Koulouriotis, "Introducing the separable moments for image watermarking in a Tottaly moment-oriented framework," *2013 18th International Conference on Digital Signal Processing (DSP)*, 2013.

[15] D. Delannay and B.M. Macq, "Method for hiding synchronization marks in scale-and rotation-resilient watermarking schemes," *Security and Watermarking of Multimedia Contents IV, International Society for Optics and Photonics* ,Vol. 4675, pp. 548-555, 2002.

[16] W. Lu, H. Lu, and F.L. Chung, "Feature based watermarking using watermark template match," *Applied Mathematics and Computation*, vol. 177, no. 1, pp.377-386, 2006.

[17] C.W. Tang and H.M. Hang, "A feature-based robust digital image watermarking scheme," *IEEE transactions on signal processing*, vol. 51, no. 4, pp.950-959, 2003.

[18] M.-S. Wang and W.-C. Chen, "A majority-voting based watermarking scheme for color image tamper detection and recovery," *Computer Standards & Interfaces*, vol. 29, no. 5, pp. 561–570, 2007.

[19] Y. Yu, H. Ling, F. Zou, Z. Lu, and L. Wang, "Robust localized image watermarking based on *invariant regions," Digital Signal Processing*, vol. 22, no. 1, pp.170-180, 2012.

[20] C. K. Ho and C.-T. Li, "Semi-fragile watermarking scheme for authentication of JPEG images," *International Conference on Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004*., 2004.

[21] X. Zhu, A. T. Ho, and P. Marziliano, "A new semi-fragile image watermarking with robust tampering restoration using irregular sampling," *Signal Processing: Image Communication*, vol. 22, no. 5, pp. 515–528, 2007.

[22] M.-S. Wang and W.-C. Chen, "A majority-voting based watermarking scheme for color image tamper detection and recovery," *Computer Standards & Interfaces*, vol. 29, no. 5, pp. 561–570, 2007.

[23] G. D. Duan, X. Zhao, J. P. Li, and J. M. Liao, "A novel semi-fragile digital watermarking algorithm for image content authentication, localization and recovery," *Dianzi Xuebao(Acta Electronica Sinica)*, vol. 38, no. 4, pp. 842-847, 2010

[24] A. Haouzia, and R. Noumeir, "Methods for image authentication: a survey," *Multimedia tools and applications*, vol. 39, no. 1, pp.1-46, 2008.

[25] P. Singh and R. S. Chadha, "A survey of digital watermarking techniques, applications and attacks," *International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 2, no. 9, pp. 165-175, 2013.

[26] K. Sreenivas and V. K. Prasad, "Fragile watermarking schemes for image authentication: a survey," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 7, pp. 1193–1218, Feb. 2017.

[27] P. Korus, "Digital image integrity – a survey of protection and verification techniques," *Digital Signal Processing*, vol. 71, pp. 1–26, 2017.

[28] H.-C. Huang and W.-C. Fang, "Metadata-based image watermarking for copyright protection," *Simulation Modelling Practice and Theory*, vol. 18, no. 4, pp. 436–445, 2010.

[29] N. Walliman, *Research methods: the basics*. London,: Routledge, 2018.

[30] R. W. Franzen, *True Color Kodak Images*. [Online]. Available: http://r0k.us/graphics/kodak/. [Accessed: 03-Nov-2018].

[31] A. Cheddad, J. Condell, K. Curran, and P. M. Kevitt, "Digital image steganography: Survey and analysis of current methods," *Signal Processing*, vol. 90, no. 3, pp. 727–752, 2010.

[32] A. Khan, A. Siddiqa, S. Munib, and S. A. Malik, "A recent survey of reversible watermarking techniques," *Information Sciences*, vol. 279, pp. 251–272, 2014.

[33] L. Luo, Z. Chen, M. Chen, X. Zeng, and Z. Xiong, "Reversible image watermarking using interpolation technique," *IEEE Transactions on information forensics and security*, vol. 5, no. 1, pp. 187-193, 2010.

[34] C. Song, S. Sudirman, M. Merabti, and D. Llewellyn-Jones, "Analysis of Digital Image Watermark Attacks," *2010 7th IEEE Consumer Communications and Networking Conference*, 2010.

[35] M. Holliman and N. Memon, "Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes," *IEEE Transactions on Image Processing*, vol. 9, no. 3, pp. 432–441, 2000.

[36] M. Goljan, "Cryptanalysis of the Yeung–Mintzer fragile watermarking technique," *Journal of Electronic Imaging*, vol. 11, no. 2, p. 262, Jan. 2002.

[37] G. Peterson, "Arnold's cat map," *Math 45 - Linear algebra,* Fall 1997*. [Online]. Available:* https://mse.redwoods.edu/darnold/math45/laproj/Fall97/Gabe/catmap.pdf . [Accessed: 03-Nov-2018]

[38] Z.H. Guan, F. Huang, and W. Guan, "Chaos-based image encryption algorithm," *Physics Letters A*, vol. 346, no. 1, pp.153-157, 2005.

[39] M. Moniruzzaman, M.A.K Hawlader, and M.F. Hossain, "An image fragile watermarking scheme based on chaotic system for image tamper detection," *Informatics, Electronics & Vision (ICIEV), 2014 International Conference*, IEEE, pp. 1-6, 2014

[40] W. Lin and C.-C. J. Kuo, "Perceptual visual quality metrics: A survey," *Journal of Visual Communication and Image Representation*, vol. 22, no. 4, pp. 297–312, 2011.

[41] M. Pedersen, "Full-Reference Image Quality Metrics: Classification and Evaluation," *Foundations and Trends® in Computer Graphics and Vision*, vol. 7, no. 1, pp. 1–80, 2011.

[42] Z. Wang and A. Bovik, "Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009.

[43] Z. Wang and A. C. Bovik, "A universal image quality index," *IEEE signal processing letters*, vol. 9, no. 3, pp. 81-84, 2002.

[44] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[45] R. C. Gonzalez and R. E. Woods, *Digital image processing*. New York, NY: Pearson, 2008.

[46] R. C. Gonzales, R. E. Woods, and S. L. Eddins, *Digital image processing using MATLAB*. Berkshire: McGraw Hill, 2011.

[47] A. C. Bovik, *The essential guide to image processing*. London: Academic Press, 2009.

[48] S. Banerjee, "Linear Algebra and Matrix Analysis for Statistics," *CRC Press*, 2014.

[49] J. Gan and Y. Zhang, "A study of singular value decomposition of face image matrix," *Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference,* IEEE, vol. 1, pp. 197-199, 2003.

[50] F. M. Gieszczykiewicz, "A PAINLESS GUIDE TO CRC ERROR DETECTION ALGORITHMS INDEX V3.00 (9/24/96)," *Fil's FAQ-Link-In Corner: CRC algorithms.* [Online]. Available: http://www.repairfaq.org/filipg/LINK/F_crc_v3.html. [Accessed: 03-Nov-

2018].

[51] *CCITT (Consultative Committee for International Telephony and Telegraphy) (Linktionary term)*. [Online]. Available: https://www.linktionary.com/c/ccitt.html. [Accessed: 04-Nov-2018].

[52] "Online CRC Calculator," *Online CRC-8 CRC-16 CRC-32 Calculator*. [Online]. Available: http://crccalc.com/. [Accessed: 03-Nov-2018].

[53] J. Fridrich and M. Goljan, "Images with self-correcting capabilities," *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference,* IEEE, vol. 3, pp. 792-796, 1999.

[54] P.L. Lin, C.K. Hsieh, and P.W. Huang, "A hierarchical digital watermarking method for image tamper detection and recovery," *Pattern recognition*, vol. 38, no. 12, pp. 2519-2529, 2005.

[55] J.A.M. Noriega, B.M. Kurkoski, M.N. Miyatake, and H.P. Meana, "Image authentication and recovery using BCH error-correcting codes," *International Journal of Computers*, vol. 5, no. 1, pp. 26-33. 2011.

[56] X. Zhang, S. Wang, Z. Qian, and G. Feng, "Self-embedding watermark with flexible restoration quality," *Multimedia Tools and Applications*, vol. 54, no. 2, pp. 385–395, 2011.

[57] X. Zhang, S. Wang, Z. Qian, and G. Feng, "Reference Sharing Mechanism for Watermark Self-Embedding," *IEEE Transactions on Image Processing*, vol. 20, no. 2, pp. 485–495, 2011.

[58] X. Zhang, Z. Qian, Y. Ren, and G. Feng, "Watermarking With Flexible Self-Recovery Quality Based on Compressive Sensing and Compositive Reconstruction," IEEE *Transactions on Information Forensics and Security*, vol. 6, no. 4, pp. 1223–1232, 2011.

[59] S. Dadkhah, A. A. Manaf, and S. Sadeghi, "Efficient Two Level Image Tamper Detection Using Three LSB Watermarking," *2012 Fourth International Conference on Computational Intelligence and Communication Networks,* IEEE, pp. 719-723, 2012.

[60] S. B. Chaluvadi and M. V. N. K. Prasad, "Efficient image tamper

detection and recovery technique using dual watermark," *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC),* IEEE, pp. 993-998, 2009.

[61] S. I. Hisham, A. N. Muhammad, J. M. Zain, and G. Badshah, "Localization Watermarking for Authentication of Text Images in Quran with Spiral Manner Numbering," *2013 Taibah University International Conference on Advances in Information Technology for the Holy Quran and Its Sciences,* IEEE, pp. 24-29, 2013.

[62] J.M. Zain, and A.R. Fauzi, "Medical image watermarking with tamper detection and recovery," *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pp. 3270-3273, 2006.

[63] M. Afifah Nailah and M.Z. Jasni, "Using Spiral Scan Technique for Medical Image Watermarking with Tamper Detection and Recovery". 2007.

[64] X. Tong, Y. Liu, M. Zhang, and Y. Chen, "A novel chaos-based fragile watermarking for image tampering detection and self-recovery," *Signal Processing: Image Communication*, vol. 28, no. 3, pp.301-308, 2013.

[65] P. Korus and A. Dziech, "Efficient method for content reconstruction with self-embedding," *IEEE Transactions on Image Processing*, vol. 22, no. 3, pp.1134-1147, 2013.

[66] D.J. Mackay, "Fountain codes," *IEE Proceedings - Communications*, vol. 152, no. 6, pp. 1062-1068, 2005.

[67] H.M.A. Nyeem, "A digital watermarking framework with application to medical image security," Ph.D dissertation, Queensland University of Technology, 2014

[68] T.D. Jassim, "Combined robust and fragile watermarking algorithms for still images," Ph.D dissertation, University of Bradford, 2014.

[69] S. Dadkhah, A. A. Manaf, Y. Hori, A. E. Hassanien, and S. Sadeghi, "An effective SVD-based image tampering detection and self-recovery using

active watermarking," *Signal Processing: Image Communication*, vol. 29, no. 10, pp. 1197–1210, 2014.

[70] A. Tareef, A. Al-Ani, H. Nguyen, and Y. Y. Chung, "A novel tamper detection-recovery and watermarking system for medical image authentication and EPR hiding," *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, IEEE, pp. 5554-5557, 2014.

[71] R. Eswaraiah and E. S. Reddy, "Robust medical image watermarking technique for accurate detection of tampers inside region of interest and recovering original region of interest," *IET Image Processing*, vol. 9, no. 8, pp. 615–625, Jan. 2015.

[72] D. Singh and S. K. Singh, "Effective self-embedding watermarking scheme for image tampered detection and localization with recovery capability," *Journal of Visual Communication and Image Representation*, vol. 38, pp. 775–789, 2016.

[73] K. Sreenivas and V. Kamakshiprasad, "Improved image tamper localisation using chaotic maps and self-recovery," *Journal of Visual Communication and Image Representation*, vol. 49, pp. 164–176, 2017.

[74] C.-C. Chang, K.-N. Chen, C.-F. Lee, and L.-J. Liu, "A secure fragile watermarking scheme based on chaos-and-hamming code," *Journal of Systems and Software*, vol. 84, no. 9, pp. 1462–1470, 2011.

[75] A. Kanso and N. Smaoui, "Logistic chaotic maps for binary numbers generations," *Chaos, Solitons & Fractals*, vol. 40, no. 5, pp. 2557–2568, 2009.

[76] C. Qin, P. Ji, X. Zhang, J. Dong, and J. Wang, "Fragile image watermarking with pixel-wise recovery based on overlapping embedding strategy," *Signal Processing*, vol. 138, pp. 280–293, 2017.

[77] C. Qin, P. Ji, J. Wang, and C.C. Chang, "Fragile image watermarking scheme based on VQ index sharing and self-embedding," *Multimedia Tools and Applications*, vol. 76, no. 2, pp. 2267-2287, 2017.

[78] A. Shehab, M. Elhoseny, K. Muhammad, A.K. Sangaiah, P. Yang, H. Huang, and G. Hou, "Secure and robust fragile watermarking scheme for medical images," *IEEE Access*, vol. 6, pp.10269-10278, 2018.

[79] B.B. Haghighi, A.H. Taherinia, and A. Harati, "TRLH: Fragile and blind dual watermarking for image tamper detection and self-recovery based on lifting wavelet transform and halftoning technique," *Journal of Visual Communication and Image Representation*," vol. 50, pp.49-64, 2018.

[80] O. Hemida, Y. Huo, H. He, and F. Chen, "A restorable fragile watermarking scheme with superior localization for both natural and text images," *Multimedia Tools and Applications*, pp.1-31, 2018.

[81] D. Niu, H. Wang, and M. Cheng, "Reducing the Computational Complexity of the Reference-Sharing Based Self-embedding Watermarking Approach," *Cloud Computing and Security Lecture Notes in Computer Science*, pp. 635–643, 2018.

[82] P. Korus, J. Bialas, and A. Dziech, "Towards Practical Self-Embedding for JPEG-Compressed Digital Images," *IEEE Transactions on Multimedia*, vol. 17, no. 2, pp. 157–170, 2015.

[83] C.Y. Lin and S.F. Chang, "SARI: self-authentication-and-recovery image watermarking system" *Proceedings of the ninth ACM international conference on Multimedia*, ACM, pp. 628-629, Oct. 2001.

[84] C.Y. Lin and S.F. Chang, "Semifragile watermarking for authenticating JPEG visual content," *Security and Watermarking of Multimedia Contents II*, vol. 3971, pp. 140-152, May. 2000,

[85] S. Yafei, Z. Li, W. Guowei, and L. Xinggang, "Reconstruction of missing blocks in image transmission by using self-embedding," *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium*, IEEE, pp. 535-538, 2001.

[86] X. Zhu, A. T. Ho, and P. Marziliano, "A new semi-fragile image watermarking with robust tampering restoration using irregular sampling," *Signal Processing: Image Communication*, vol. 22, no. 5, pp. 515–528, 2007.

[87] Y.M. Hasan and A.M. Hassan, "Tamper detection with self-correction hybrid spatial-DCT domains image authentication technique," *Signal Processing and Information Technology, 2007 IEEE International Symposium*, IEEE, pp. 369-374, Dec. 2007.

[88] C. Cruz, J.A. Mendoza, M.N. Miyatake, H.P. Meana, and B. Kurkoski, "Semi-fragile watermarking based image authentication with recovery capability," *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference*, IEEE, pp. 1-4, Dec. 2009.

[89] A. Cheddad, J. Condell, K. Curran, and P. Mc Kevitt, "A secure and improved self-embedding algorithm to combat digital document forgery," *Signal Processing*, vol. 89, no. 12, pp.2324-2332, 2009.

[90] J.A. Mendoza-Noriega, B.M. Kurkoski, M. Nakano-Miyatake, and H. Perez-Meana, "Halftoning-based self-embedding watermarking for image authentication and recovery", *Circuits and Systems (MWSCAS), 2010 53rd IEEE International Midwest Symposium*, IEEE, pp. 612-615, Aug. 2010.

[91] P. Korus, J. Bialas, and A. Dziech, "High-quality self-embedding for jpeg-compressed digital images," *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*, IEEE, pp. 1-5, Sep 2013.

[92] F. Chen, H. He, and Y. Huo, "Self-embedding watermarking scheme against JPEG compression with superior imperceptibility," *Multimedia Tools and Applications*, vol. 76, no. 7, pp. 9681–9712, 2016.

[93] H. Wang, A. T. S. Ho, and X. Zhao, "A Novel Fast Self-restoration Semi-fragile Watermarking Algorithm for Image Content Authentication Resistant to JPEG Compression," *Digital Forensics and Watermarking Lecture Notes in Computer Science*, pp. 72–85, 2012.

[94] G. Li, S. Pei, G. Chen, W. Cao, and B. Wu, "A self-embedded watermarking scheme based on relationship function of corresponding inter-blocks DCT coefficient," *Computer Supported Cooperative Work in Design, 2009. CSCWD 2009. 13th International Conference*, IEEE, pp.

107-112, Apr. 2009.

[95] R. Chamlawi, A. Khan, and A. Idris, "Wavelet based image authentication and recovery," *Journal of Computer Science and Technology*, vol. 22, no. 6, pp. 795-804, 2007.

[96] S. Riaz and S.W. Lee, "Image authentication and restoration by multiple watermarking techniques with advance encryption standard in digital photography," *Advanced Communication Technology (ICACT), 2013 15th International Conference,* IEEE, pp. 24-28, Jan. 2013.

[97] Y. Zhang, J. Zheng, and M. Ma (Eds.), "Handbook of research on wireless security," *IGI Global,* 2008.

[98] I.A. Ansari, M. Pant, and C.W. Ahn, "SVD based fragile watermarking scheme for tamper localization and self-recovery," *International Journal of Machine Learning and Cybernetics*, vol.7, no. 6, pp.1225-1239, 2016.

[99] F. Arab, S.M. Abdullah, S.Z.M. Hashim, A.A. Manaf, and M. Zamani, "A robust video watermarking technique for the tamper detection of surveillance systems," *Multimedia Tools and Applications*, vol. 75, no. 18, pp.10855-10885, 2016.

[100] B. Wang, S. Zhou, X. Zheng, C. Zhou, J. Dong, and L. Zhao, "Image watermarking using chaotic map and DNA coding," *Optik-International Journal for Light and Electron Optics*, vol. 126, no. 24, pp.4846-4851, 2015.

[101] T. Luo, G. Jiang, M. Yu, and H. Xu, "Asymmetric self-recovery oriented stereo image watermarking method for three dimensional video system," *Multimedia Systems*, vol. 22, no. 5, pp.641-655, 2016.

[102] W.C. Wu and Z.W. Lin, "SVD-based self-embedding image authentication scheme using quick response code features," *Journal of Visual Communication and Image Representation*, vol. 38, pp.18-28, 2016.

[103] H. Shi, X. Wang, M. Li, J. Bai, and B. Feng, "Secure variable-capacity self-recovery watermarking scheme" *Multimedia Tools and Applications*,

vol. 76, no. 5, pp.6941-6972, 2017.

[104] S. Sarreshtedari, A. Abbasfar, and M.A. Akhaee, "A joint source–channel coding approach to digital image self-recovery", *Signal, Image and Video Processing*, vol. 11, no. 7, pp.1371-1378, 2017.

[105] L. Wang, Q. Ye, Y. Xiao, Y. Zou, and B. Zhang, "An image encryption scheme based on cross chaotic map," *Image and Signal Processing, 2008. CISP'08. Congress*, IEEE, vol. 3, pp. 22-26, May. 2008.

[106] *SIPI Image Database - Misc*. [Online]. Available: http://sipi.usc.edu/database/database.php?volume=misc. [Accessed: 04-Nov-2018].

[107] "GNU Octave," *The GNU Operating System and the Free Software Movement*. [Online]. Available: https://www.gnu.org/software/octave/. [Accessed: 03-Nov-2018].

[108] "gnuplot homepage," *gnuplot homepage*. [Online]. Available: http://www.gnuplot.info/. [Accessed: 03-Nov-2018].

[109] *Center for Neural Science, New York University*. [Online]. Available: http://www.cns.nyu.edu/~lcv/ssim/. [Accessed: 04-Nov-2018].

[110] B. Yu, *A Quick Glance at Digital Watermarking*. [Online]. Available: https://www.cse.wustl.edu/~jain/cse571-11/ftp/watermrk/index.html. [Accessed: 04-Nov-2018].

Appendix A: The Implemented Code for the First Proposed Technique

A.1: The Used Programming Language for Code Implementation

All of the code for all techniques was implemented using GNU Octave 4.2.2. Appendix A contains a list of the implemented code for the first proposed technique, the code consists of two files: ***encode.m, and decode.m.***

A.2: The Code for the Encoding Stage of the First Technique

```
% function I_enc=encode(I_in,key)
%
% The encoding function for the first proposed technique.
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function I_enc=encode(I_in,key)

[I_H,I_W] = size(I_in);
I_SZ = [I_H,I_W];

W_SZ =I_SZ/2;
W_W = W_SZ(2);
W_H = W_SZ(1);

% Generating the watermark
I_del = bitand(I_in,bin2dec('11111110'));
W = I_del(1:2:end,1:2:end)/4 +  ...
    I_del(1:2:end,2:2:end)/4 +  ...
    I_del(2:2:end,1:2:end)/4 +  ...
    I_del(2:2:end,2:2:end)/4;

% Shuffling the watermark according to the secret key.

rand('seed',key);

key_R = randi([1,W_W],[W_H,1]);
```

```
key_C = randi([1,W_H],[1,W_W]);

for count  = 1:2 % repeat Shuffling once
    for R = 1:W_H
        W(R,:)=circshift(W(R,:),[0, key_R(R)]);
    end
    for C = 1:W_W
        W(:,C)=circshift(W(:,C),[key_C(C),0]);
    end
end

% Encrypting the watermark by XORing it with a random sequence.

% Note: For simplicity, the randomisation depends only on the secret key.
%       However, to make the method resistive to VQ and collage attach
%       the randomisation should depend also on a unique serial
%       number for the watermarked image.

rand('seed',key);

seq  = uint8(randi([0,255],W_SZ));
W = bitxor(W,seq);

% Storing the watermark in the first LSB of the input image.

and_msk = bin2dec('00000001');

I_enc=I_del;

W = bitshift(W,-4);
I_enc(1:2:end,1:2:end) = I_enc(1:2:end,1:2:end) + bitand(W,and_msk);

W = bitshift(W,-1);
I_enc(1:2:end,2:2:end) = I_enc(1:2:end,2:2:end) + bitand(W,and_msk);

W = bitshift(W,-1);
I_enc(2:2:end,1:2:end) = I_enc(2:2:end,1:2:end) + bitand(W,and_msk);

W = bitshift(W,-1);;
I_enc(2:2:end,2:2:end) = I_enc(2:2:end,2:2:end) + bitand(W,and_msk);

return
```

A.3: The Code for the Decoding Stage of the First Technique

```
% function [I_fxd,vld_msk]=decode(I_in,key)
%
% The decoding function for the first proposed technique.
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [I_fxd,vld_msk]=decode(I_in,key)

[I_H,I_W] = size(I_in);
I_SZ = [I_H,I_W];

W_SZ = I_SZ/2;
W_H = W_SZ(1);
W_W = W_SZ(2);

% Getting the watermark from the first LSB of the input image
I_tmp = bitand(I_in, bin2dec('00000001'));

W = I_tmp(2:2:end,2:2:end);
W = bitshift(W,7);

W_tmp =  I_tmp(2:2:end,1:2:end);
W_tmp = bitshift(W_tmp,6);
W=W+W_tmp;

W_tmp =  I_tmp(1:2:end,2:2:end);
W_tmp = bitshift(W_tmp,5);
W=W+W_tmp;

W_tmp =  I_tmp(1:2:end,1:2:end);
W_tmp = bitshift(W_tmp,4);
W=W+W_tmp;

% Decrypting the watermark by XORing with the random sequence

% Note: For simplicity, the randomisation depends only on the secret key.
%       However, to make the method resistive to VQ and collage attach
%       the randomisation should depend also on a unique serial
%       number for the watermarked image.

rand('seed',key);
seq  = uint8(randi([0,255],W_SZ));
W = bitxor(W,seq);
```

```
% Set 4 MSBs in the watermark to zero
W = bitand(W, bin2dec('11110000'));


% Shuffling back the watermark
rand('seed',key);

key_R = randi([1,W_W],[W_H,1]);
key_C = randi([1,W_H],[1,W_W]);

for count  = 1:2 % repeat Shuffling once
    for C = 1:W_W
        W(:,C)=circshift(W(:,C),[-key_C(C),0]);
    end

    for R = 1:W_H
        W(R,:)=circshift(W(R,:),[0, -key_R(R)]);
    end
end


% Producing the difference image by comparing
% the watermarked image to the watermark

I_del = bitand(I_in, bin2dec('11111110'));

W_tmp = I_del(1:2:end,1:2:end)/4 + ...
        I_del(1:2:end,2:2:end)/4 + ...
        I_del(2:2:end,1:2:end)/4 + ...
        I_del(2:2:end,2:2:end)/4;

W_tmp = bitand(W_tmp, bin2dec('11110000'));
W_diff = bitxor(W,W_tmp);

W_diff(W_diff>0) = 255;

% Applying filter to the difference image
FLTR_SZ = 7;
THRESH = 0.7; % Threshold for the ratio of the white area

fun = @(x) ( sum(x(:)) > ((255*FLTR_SZ^2)* THRESH) );

W_diff = nlfilter(W_diff,[FLTR_SZ, FLTR_SZ],fun);
W_diff = uint8(W_diff)*255;

% Expanding the white area in the difference image
```

```
f = fspecial("average", 5);
W_diff = imfilter(W_diff, f,'same');
W_diff(W_diff>0) = 255;

% Finding the validity mask image.
vld_msk = imresize(W_diff,2,'linear');
vld_msk = ~(vld_msk > 0);

% Applying median filter to the watermark
W_fltr = medfilt2(W,[3 3]);

% Resizing the filtered watermark
I_fltr = imresize(W_fltr,2,'linear');

% Recovering the input image using the filtered watermark
I_fxd = I_del;
I_fxd(~vld_msk) = I_fltr(~vld_msk);

return
```

Appendix B: The Implemented Code for the Second Proposed Technique

Appendix B contains a list of the implemented code for the second proposed technique, the code consists of two files: ***encode.m, and decode.m.***

B.1: The Code for the Encoding Stage of the Second Proposed Technique

```
% function [I_enc]=encode(I_in,key)
%
% The encoding function of the second proposed technique
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [I_enc]=encode(I_in,key)
I_SZ = size(I_in);
I_W = I_SZ(2);
I_H = I_SZ(1);

W_SZ =I_SZ/2;
W_W = W_SZ(2);
W_H = W_SZ(1);

if sum(mod(I_SZ,12)) ~= 0
    disp('Dimensions of the image must be multiple of 12');
    return
end

% Store the average the 6 MSB of the input image in the watermark
and_msk = bin2dec('11111100');
I_del = bitand(I_in,and_msk);

W = I_del(1:2:end,1:2:end)/4 + ...
    I_del(1:2:end,2:2:end)/4 + ...
    I_del(2:2:end,1:2:end)/4 + ...
    I_del(2:2:end,2:2:end)/4;

W = bitand(W,and_msk);
```

```
% Store the average 2 LSB of the input image in the watermark

and_msk = bin2dec('00000011');
I_tmp = bitand(I_in,and_msk);

W_tmp = I_tmp(1:2:end,1:2:end) + ...
        I_tmp(1:2:end,2:2:end) + ...
        I_tmp(2:2:end,1:2:end) + ...
        I_tmp(2:2:end,2:2:end);
W_tmp=W_tmp/4;
W_tmp = bitand(W_tmp,and_msk);
W = W+W_tmp;

% Mapping each group of pixels in 3x3 px block using
% maximum-distance mapping
% The positions of the groups are as follows:
%  3  1  2
%  2  3  1
%  1  2  3

dst = 6;

%% Mapping the first group of pixels horizontally
W(1:3:end,2:3:end) = circshift(W(1:3:end,2:3:end),[0,W_W/dst]);
W(2:3:end,3:3:end) = circshift(W(2:3:end,3:3:end),[0,W_W/dst]);
W(3:3:end,1:3:end) = circshift(W(3:3:end,1:3:end),[0,W_W/dst]);

%% Mapping the second group of pixels vertically
W(2:3:end,1:3:end) = circshift(W(2:3:end,1:3:end),[W_H/dst,0]);
W(3:3:end,2:3:end) = circshift(W(3:3:end,2:3:end),[W_H/dst,0]);
W(1:3:end,3:3:end) = circshift(W(1:3:end,3:3:end),[W_H/dst,0]);

%% Mapping the third group of pixels diagonally
%% (horizontally then vertically)
W(1:3:end,1:3:end) = circshift(W(1:3:end,1:3:end),[0,W_W/dst]);
W(2:3:end,2:3:end) = circshift(W(2:3:end,2:3:end),[0,W_W/dst]);
W(3:3:end,3:3:end) = circshift(W(3:3:end,3:3:end),[0,W_W/dst]);

W(1:3:end,1:3:end) = circshift(W(1:3:end,1:3:end),[W_H/dst,0]);
W(2:3:end,2:3:end) = circshift(W(2:3:end,2:3:end),[W_H/dst,0]);
W(3:3:end,3:3:end) = circshift(W(3:3:end,3:3:end),[W_H/dst,0]);

% Shuffling the bits in each pixel in the watermark
% according to the secret key

%% Generating permutation matrix
rand('seed',key);
```

```
PRM = zeros([W_SZ,8]);
for c = 1:W_W
    for r = 1:W_H
        PRM(r,c,:) = randperm(8);
    end
end

%% Permuting the pixels of the watermark
PRM = PRM -1;
Wx = W * 0;

for l = 1:8
    M = uint8(ones(W_SZ));
    M = M  .* 2 .^ PRM(:,:,l);
    M = bitand(M,W);
    M = M ./ 2 .^ PRM(:,:,l);
    M = M  * 2 ^ (l-1);
    Wx = bitor(Wx,M);
end

W = Wx;

% Encrypting the watermark by XORing with the random sequence

% Note: For simplicity, the randomisation depends only on the secret key.
%       However, to make the method resistive to VQ and collage attach
%       the randomisation should depend also on a unique serial
%       number for the watermarked image.

rand('seed',key)
seq = randi([0,255],W_SZ);
W = bitxor(W,seq);

% storing the watermark in the 2 LSB of the image

I_enc = I_del;

and_msk = bin2dec('00000011');

I_enc(1:2:end,1:2:end) = I_enc(1:2:end,1:2:end) + bitand(W,and_msk);
W = bitshift(W,-2);
I_enc(1:2:end,2:2:end) = I_enc(1:2:end,2:2:end) + bitand(W,and_msk);
W = bitshift(W,-2);
I_enc(2:2:end,1:2:end) = I_enc(2:2:end,1:2:end) + bitand(W,and_msk);
W = bitshift(W,-2);
I_enc(2:2:end,2:2:end) = I_enc(2:2:end,2:2:end) + bitand(W,and_msk);
```

```
return
```

B.2: The Code for the Decoding Stage of the Second Proposed Technique

```
% function [I_fxd,vld_msk]=decode(I_in,key)
%
% The decoding function for the second proposed technique.
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [I_fxd,vld_msk]=decode(I_in,key)

I_SZ = size(I_in);
I_W = I_SZ(2);
I_H = I_SZ(1);

W_SZ = I_SZ/2;
W_W = W_SZ(2);
W_H = W_SZ(1);

if sum(mod(I_SZ,12)) ~= 0
    disp('Dimensions of the image must be multiple of 12');
    return
end

% Getting the watermark from the 2 LSB of the image
I_tmp = bitand(I_in,bin2dec('00000011'));
W = I_tmp(2:2:end,2:2:end);

W = bitshift(W,2);
W = W + I_tmp(2:2:end,1:2:end);

W = bitshift(W,2);
W = W + I_tmp(1:2:end,2:2:end);

W = bitshift(W,2);
W = W + I_tmp(1:2:end,1:2:end);

% Decrypting the watermark by XORing with the random sequence

% Note: For simplicity, the randomisation depends only on the secret key.
%       However, to make the method resistive to VQ and collage attach
%       the randomisation should depend also on a unique serial
%       number for the watermarked image.
```

```
rand('seed',key)
seq = randi([0,255],W_SZ);
W = bitxor(W,seq);


% Shuffling back the bits in each pixel in the watermark
% according the secret key

%% Generating permutation matrix
rand('seed',key);

PRM = zeros([W_SZ,8]);
for c = 1:W_W
    for r = 1:W_H
        PRM(r,c,:) = randperm(8);
    end
end


%% Permuting the pixels of the watermark
PRM = PRM -1;
Wx = W * 0;

for l = 1:8
    M = uint8(ones(W_SZ));
    M = M  * 2 ^ (l-1);
    M = bitand(M,W);
    M = M  / 2 ^ (l-1);
    M = M  .* 2 .^ PRM(:,:,l);
    Wx = bitor(Wx,M);
end

W = Wx;

% Re-mapping back each group of pixels in 3x3 px block
% using maximum-distance mapping
% The positions of the groups are as follows:
%  3  1  2
%  2  3  1
%  1  2  3

dst = 6;

%% Re-mapping the third group of pixels diagonally
%% (vertically then horizontally)
W(1:3:end,1:3:end) = circshift(W(1:3:end,1:3:end),[W_H/dst,0]);
```

```
W(2:3:end,2:3:end) = circshift(W(2:3:end,2:3:end),[W_H/dst,0]);
W(3:3:end,3:3:end) = circshift(W(3:3:end,3:3:end),[W_H/dst,0]);

W(1:3:end,1:3:end) = circshift(W(1:3:end,1:3:end),[0,W_W/dst]);
W(2:3:end,2:3:end) = circshift(W(2:3:end,2:3:end),[0,W_W/dst]);
W(3:3:end,3:3:end) = circshift(W(3:3:end,3:3:end),[0,W_W/dst]);


%% Re-mapping the second group of pixels vertically
W(2:3:end,1:3:end) = circshift(W(2:3:end,1:3:end),[W_H/dst,0]);
W(3:3:end,2:3:end) = circshift(W(3:3:end,2:3:end),[W_H/dst,0]);
W(1:3:end,3:3:end) = circshift(W(1:3:end,3:3:end),[W_H/dst,0]);

%% Re-mapping the first group of pixels horizontally
W(1:3:end,2:3:end) = circshift(W(1:3:end,2:3:end),[0,W_W/dst]);
W(2:3:end,3:3:end) = circshift(W(2:3:end,3:3:end),[0,W_W/dst]);
W(3:3:end,1:3:end) = circshift(W(3:3:end,1:3:end),[0,W_W/dst]);

% Taking average of 6 MSB of the image to compare it with W
and_msk = bin2dec('11111100');
I_del = bitand(I_in,and_msk);

W_tmp = I_del(1:2:end,1:2:end)/4 + ...
    I_del(1:2:end,2:2:end)/4 + ...
    I_del(2:2:end,1:2:end)/4 + ...
    I_del(2:2:end,2:2:end)/4;

W_tmp = bitand(W_tmp,and_msk);

% XOR the average with the watermark
W_diff = ((bitxor(W_tmp,bitand(W,and_msk)))>0)*255;

% Applying non-linear filter to remove non-tampered area

fun = @(x) (sum(x(:)) > 255*3); % more than 3 pixels are white

tmpr_W = nlfilter(W_diff,[3 3],fun);

tmpr_W = logical(tmpr_W);

% Generating validity matrix
vld_msk = logical(zeros(I_SZ));
vld_msk(1:2:end,1:2:end) = tmpr_W;
vld_msk(1:2:end,2:2:end) = tmpr_W;
vld_msk(2:2:end,1:2:end) = tmpr_W;
vld_msk(2:2:end,2:2:end) = tmpr_W;
```

```
vld_msk = ~vld_msk;

% Generating recovery image by scaling up the watermark
I_rec = imresize(W,2,'linear');

% recovering the tampered area
I_fxd  = I_del;
I_fxd(~vld_msk)  = I_rec(~vld_msk);

% Recover the lost 2 LSBs in the image
I_fxd(vld_msk) = I_fxd(vld_msk) +
bitand(I_rec(vld_msk),bin2dec('00000011'));

return
```

Appendix C: The Implemented Code for the Third Proposed Technique

Appendix C contains a list of the implemented code for the third proposed technique, the code consists of two files: ***encode.m, and decode.m.***

C.1: The Code for the Encoding Stage of the Third Proposed Technique

```
% function [I_enc]=encode(I_in,key)
%
% The encoding function of the third proposed technique
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [I_enc]=encode(I_in,key)

I_SZ = size(I_in);
I_W = I_SZ(2);
I_H = I_SZ(1);

W_SZ = I_SZ/2;
W_W = W_SZ(2);
W_H = W_SZ(1);

if sum(mod(I_SZ,2)) ~= 0
    disp('Dimensions of the image must be multiple of 2');
    return
end

% Store the average 6 MSB of the input image in the watermark
and_msk = bin2dec('11111100');
I_del = bitand(I_in,and_msk);

W = I_del(1:2:end,1:2:end)/4 + ...
    I_del(1:2:end,2:2:end)/4 + ...
    I_del(2:2:end,1:2:end)/4 + ...
    I_del(2:2:end,2:2:end)/4;

W = bitand(W,and_msk);
```

```
% Store the average 2 LSB of the input image in the watermark
and_msk = bin2dec('00000011');
I_tmp = bitand(I_in,and_msk);

W_tmp = I_tmp(1:2:end,1:2:end) + ...
        I_tmp(1:2:end,2:2:end) + ...
        I_tmp(2:2:end,1:2:end) + ...
        I_tmp(2:2:end,2:2:end);
W_tmp=W_tmp/4;
W_tmp = bitand(W_tmp,and_msk);
W = W+W_tmp;

% Mapping the watermark using maximum-distance
W = circshift(W,W_SZ/2);

% Randomly permuting the bits in each pixel in the watermark
% according to the secret key

%% Generating permutation matrix
rand('seed',key);

PRM = zeros([W_SZ,8]);
for c = 1:W_W
    for r = 1:W_H
        PRM(r,c,:) = randperm(8);
    end
end

%% permutating the pixels of the watermak
PRM = PRM -1;
Wx = W * 0;
for l = 1:8
    M = uint8(ones(W_SZ));
    M = M  .* 2 .^ PRM(:,:,l);
    M = bitand(M,W);
    M = M ./ 2 .^ PRM(:,:,l);
    M = M  * 2 ^ (l-1);
    Wx = bitor(Wx,M);
end

W = Wx;

% Encrypting the watermark by XORing it with a random sequence.

% Note: For simplicity, the randomisation depends only on the secret key.
%       However, to make the method resistive to VQ and collage attach
```

```
%       the randomisation should depend also on a unique serial
%       number for the watermarked image.

rand('seed',key)
seq = randi([0,255],W_SZ);
W = bitxor(W,seq);

% storing the watermark in the 2 LSB of the image
I_enc = I_del;

and_msk = bin2dec('00000011');

I_enc(1:2:end,1:2:end) = I_enc(1:2:end,1:2:end) + bitand(W,and_msk);
W = bitshift(W,-2);
I_enc(1:2:end,2:2:end) = I_enc(1:2:end,2:2:end) + bitand(W,and_msk);
W = bitshift(W,-2);
I_enc(2:2:end,1:2:end) = I_enc(2:2:end,1:2:end) + bitand(W,and_msk);
W = bitshift(W,-2);
I_enc(2:2:end,2:2:end) = I_enc(2:2:end,2:2:end) + bitand(W,and_msk);

return
```

C.2: The Code for the Decoding Stage of the Third Proposed Technique

```
% function [I_fxd,vld_msk,W_hfc]=decode(I_in,key)
%
% The decoding function of the third proposed technique
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [I_fxd,vld_msk,W_hfc]=decode(I_in,key)

get_W_hfc = (nargout == 3);

I_SZ = size(I_in);
I_W = I_SZ(2);
I_H = I_SZ(1);

W_SZ = I_SZ/2;
W_W = W_SZ(2);
W_H = W_SZ(1);

if sum(mod(I_SZ,2)) ~= 0
    disp('Dimensions of the image must be multiple of 2');
    return
```

```
end

% retreive the watermark from the 2 LSB of the image
I_tmp = bitand(I_in,bin2dec('00000011'));
W = I_tmp(2:2:end,2:2:end);

W = bitshift(W,2);
W = W + I_tmp(2:2:end,1:2:end);

W = bitshift(W,2);
W = W + I_tmp(1:2:end,2:2:end);

W = bitshift(W,2);
W = W + I_tmp(1:2:end,1:2:end);

% Decrypting the watermark by XORing it with a random sequence.

% Note: For simplicity, the randomisation depends only on the secret key.
%       However, to make the method resistive to VQ and collage attach
%       the randomisation should depend also on a unique serial
%       number for the watermarked image.

rand('seed',key)
seq = randi([0,255],W_SZ);
W = bitxor(W,seq);

%% Randomly permuting back the bits in each pixel in the watermark
%% according the secret key

%% Generating permutation matrix
rand('seed',key);

PRM = zeros([W_SZ,8]);
for c = 1:W_W
    for r = 1:W_H
        PRM(r,c,:) = randperm(8);
    end
end

%% permutating the pixels of the watermak
PRM = PRM -1;
Wx = W * 0;

for l = 1:8
    M = uint8(ones(W_SZ));
    M = M  * 2 ^ (l-1);
    M = bitand(M,W);
```

```
    M = M  / 2 ^ (l-1);
    M = M  .* 2 .^ PRM(:,:,l);
    Wx = bitor(Wx,M);
end

W = Wx;

% Mapping W using maximum-distance mapping
W = circshift(W,W_SZ/2);

% Taking average of 6 MSB of the image to compare it with W
and_msk = bin2dec('11111100');
I_del = bitand(I_in,and_msk);

W_tmp = I_del(1:2:end,1:2:end)/4 + ...
    I_del(1:2:end,2:2:end)/4 + ...
    I_del(2:2:end,1:2:end)/4 + ...
    I_del(2:2:end,2:2:end)/4;

W_tmp = bitand(W_tmp,and_msk);

% XOR the average with the watermark
tmpr_W = logical(bitxor(W_tmp,bitand(W,and_msk)));

% Defining the filter used for detecting high frequency contents

FLT_SZ = 5;
FLT_MAT = dctmtx(FLT_SZ);
MAT2=ones(FLT_SZ);
MAT2(1,:)=0;
MAT2(:,1)=0;
W_dbl = double(W);

% If W_hfc is requested, then it will be used in further calculations
if get_W_hfc
    fun = @(x) (sum(sum(abs( (FLT_MAT*x*FLT_MAT') .* MAT2))));
    W_hfc = nlfilter(W_dbl,[FLT_SZ FLT_SZ],fun);
    MAX = max(max(W_hfc));
    W_hfc=uint8(W_hfc/MAX*255);
else
    FLT_SZM = FLT_SZ-1;
    FLT_DF = FLT_SZM/2;
    W_dbl = padarray(W_dbl,[FLT_DF,FLT_DF]);
end

% Determining which part of tmpr_W refer to a tampered area based on
% high-frequency-contents
```

```
for r=1:W_H
    for c=1:W_W
        if tmpr_W(r,c)

            r2 = rem((r-1)+W_H/2,W_H)+1;
            c2 = rem((c-1)+W_W/2,W_W)+1;

            if get_W_hfc
                if W_hfc(r,c) < W_hfc(r2,c2)
                    tmpr_W (r,c) =true;
                    tmpr_W (r2,c2) = false;
                else
                    tmpr_W (r2,c2) = true;
                    tmpr_W (r,c) =false;
                end
            else
                tmp_mat = W_dbl(r:r+FLT_SZM,c:c+FLT_SZM);
                hfc1 = sum(sum(abs(FLT_MAT*tmp_mat*FLT_MAT') .* MAT2));
                tmp_mat = W_dbl(r2:r2+FLT_SZM,c2:c2+FLT_SZM);
                hfc2 = sum(sum(abs(FLT_MAT*tmp_mat*FLT_MAT') .* MAT2));

                if hfc1 < hfc2
                    tmpr_W (r,c) =true;
                    tmpr_W (r2,c2) = false;
                else
                    tmpr_W (r2,c2) = true;
                    tmpr_W (r,c) =false;
                end
            end
        end
    end
end

% Generating validity matrix
vld_msk = logical(zeros(I_SZ));
vld_msk(1:2:end,1:2:end) = tmpr_W;
vld_msk(1:2:end,2:2:end) = tmpr_W;
vld_msk(2:2:end,1:2:end) = tmpr_W;
vld_msk(2:2:end,2:2:end) = tmpr_W;

vld_msk = ~vld_msk;

% Generating recovery image by scaling up the watermark
I_rec = imresize(W,2,'linear');

% recovering the tampered area
I_fxd  = I_del;
```

```
I_fxd(~vld_msk)  = I_rec(~vld_msk);

% Recover the lost 2 LSBs in the image
I_fxd(vld_msk) = I_fxd(vld_msk) +
bitand(I_rec(vld_msk),bin2dec('00000011'));

return
```

Appendix D: The Implemented Code for the Fourth Proposed Technique

Appendix D contains a list of the implemented code for the fourth proposed technique. The main two functions in ***encode.m*** and ***decode.m*** call the functions in the files: ***blk2dct.m***, ***crc16.m***, ***dct2blk.m***, ***mat2zig.m***, ***set_global.m***, and ***zig2mat.m***. The file ***set_global.m*** must be called before calling the ***encode.m*** and ***decode.m*** files in order to set the global variables used by them.

D.1: The Code for the Encoding Stage of the Fourth Proposed Technique

```
% function I_enc=encode(I_in,key)
%
% The encoding function for the fourth proposed technique.
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function I_enc=encode(I_in,key)

[I_H,I_W] = size(I_in);
I_SZ = [I_H,I_W];

if sum(mod(I_SZ,8)) ~= 0
    disp('Dimensions of the image must be multiple of 8');
    return
end

global PRM64;

% Initialise the encoded image
I_enc = bitand(I_in,bin2dec('11111100'));

% Generate mapped image using maximum-distance mapping
I_map = circshift(I_enc,I_SZ/2);

for r=1:8:I_H
    for c=1:8:I_W
        W_dct = blk2dct(I_map(r:r+7,c:c+7));
```

```
        W_crc = crc16([I_enc(r:r+7,c:c+7)(:);W_dct(:)]);
        W_crc = uint8([bitshift(W_crc,-8),bitand(W_crc,255)]);

        % Concatinating DCT recovery bytes and CRC16 authentication bytes
        W_tmp = [W_dct(:);W_crc(:)];
        W_tmp = reshape(W_tmp,[4,4]);

        % Encrypting the watermark by XORing it with a random sequence.
        %
        % Note: The randomisation seed changes with the block number
        % in order to prevent using any block in the image to replace
        % another one. However, the randomisation should depend also
        % on a unique serial number for the image to prevent using
        % a bock from anther image to replace a block  that has the same
        % block number if the two images are using the same secret key.

        SD = key * 10^9 + r * 10^5 + c;

        rand('seed',SD);
        seq  = uint8(randi([0,255],[4,4]));
        W_tmp = bitxor(W_tmp,seq);

        % Storing the watermark into 2 LSB of 8x8 block
        W_blk = zeros(8,8);
        and_msk = bin2dec('00000011');

        W_blk(1:2:end,1:2:end) = bitand(W_tmp,and_msk);
        W_tmp = bitshift(W_tmp,-2);

        W_blk(1:2:end,2:2:end) = bitand(W_tmp,and_msk);
        W_tmp = bitshift(W_tmp,-2);

        W_blk(2:2:end,1:2:end) = bitand(W_tmp,and_msk);
        W_tmp = bitshift(W_tmp,-2);

        W_blk(2:2:end,2:2:end) = bitand(W_tmp,and_msk);

        % Randomly permuting W_blk
        W_blk = W_blk(PRM64);

        % Storing the watermark in the encoded image
        I_enc(r:r+7,c:c+7) = I_enc(r:r+7,c:c+7) + W_blk;
    end
end

return
```

D.2: The Code for the Decoding Stage of the Fourth Proposed Technique

```
% function [I_fxd,vld_msk]=decode(I_in,key)
%
% The decoding function for the fourth proposed technique.
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [I_fxd,vld_msk]=decode(I_in,key)

[I_H,I_W] = size(I_in);
I_SZ = [I_H,I_W];

if sum(mod(I_SZ,8)) ~= 0
    disp('Dimensions of the image must be multiple of 8');
    return
end

global XPRM64;

% Initialise the validity matrix
vld_msk = logical(zeros(I_SZ));

% Generating a matrix to store DCT recovery data
wdct_map = -1*ones(I_SZ);

% Initialise the decoded image
I_fxd = bitand(I_in,bin2dec('11111100'));

% Validating image blocks
for r=1:8:I_H
    for c=1:8:I_W
        % Getting the watermark from 2 LSB of I_blk
        W_blk = bitand(I_in(r:r+7,c:c+7),bin2dec('00000011'));

        % Randomly permute W_blk to the original position
        W_blk = W_blk(XPRM64);

        % Converting the watermark into bytes
        W_tmp = W_blk(2:2:end,2:2:end);

        W_tmp = bitshift(W_tmp,2);
        W_tmp = W_tmp + W_blk(2:2:end,1:2:end);

        W_tmp = bitshift(W_tmp,2);
```

```
            W_tmp = W_tmp + W_blk(1:2:end,2:2:end);

            W_tmp = bitshift(W_tmp,2);
            W_tmp = W_tmp + W_blk(1:2:end,1:2:end);

            % Decrypting the watermark by XORing it with a random sequence.
            %
            % Note: The randomisation seed changes with the block number
            % in order to prevent using any block in the image to replace
            % another one. However, the randomisation should depend also
            % on a unique serial number for the image to prevent using
            % a bock from anther image to replace a block  that has the same
            % block number if the two images are using the same secret key.

            SD = key * 10^9 + r * 10^5 + c;

            rand('seed',SD);
            seq  = uint8(randi([0,255],[4,4]));
            W_tmp = bitxor(W_tmp,seq);

            % Getting the DCT recovery bytes
            W_dct = W_tmp(1:14);

            % Getting the CRC16 bytes from the watermark
            W_crc = W_tmp(15:16);

            % Converting W_crc into 16 bit number
            W_crc=uint16(W_crc);
            W_crc = bitshift(W_crc(1),8)+W_crc(2);

            % Generating authentication bytes of the current
            % block and current DCT bytes
            I_crc = crc16([I_fxd(r:r+7,c:c+7)(:);W_dct(:)]);

            % Validating the current block

            if I_crc == W_crc
                vld_msk(r:r+7,c:c+7) = true;
                wdct_map(r:r+3,c:c+3) = W_tmp;
            end
        end
    end

    % Mapping the recovery matrix
    wdct_map=circshift(wdct_map,I_SZ/2);

    % Recovering the tampered blocks
```

```
for r=1:8:I_H
    for c=1:8:I_W
        if ~vld_msk(r,c)
            if wdct_map(r,c) >= 0
                W_dct = wdct_map(r:r+3,c:c+3);
                W_dct = W_dct(1:14);
                W_blk = dct2blk(W_dct);
                I_fxd(r:r+7,c:c+7) = W_blk;
            end
        end
    end
end

return
```

D.3: The Code in the File blk2dct.m

```
% function lsb_out = blk2dct(blk_in)
%
% Converts 8x8 px block to DCT coefficients and store them in 14 bytes.
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function lsb_out = blk2dct(blk_in)

global PRM64;
global Q
global BT_NUM
global LN
global DCT_MAT
global BT_LUT
global M2ZGZG64
global MAX_LEVEL

% Finding DCT coefficients
dct_blk = DCT_MAT*double(blk_in)*DCT_MAT';

% Quantising DCT coefficients
dct_blk = round(dct_blk ./ Q);

% Converting DCT block to a zigzag sequence
z_seq  = dct_blk(M2ZGZG64);

% Matching the size of z_seq to the size of BT_NUM
z_seq=z_seq(1:LN);
```

```
% Shifting negative values of DCT coefficients
z_seq(2:LN) = z_seq(2:LN) + 2 .^ (BT_NUM(2:LN) -1);

% Adjusting the levels of the the DCT coefficients
z_seq(z_seq<0) = 0;
indx =z_seq > MAX_LEVEL;
z_seq(indx) = MAX_LEVEL(indx);

% Converting z_seq into a bit stream
% Look up table reduced the time dramatically
b_stream = [''];
for n = 1:LN
    b = BT_LUT(z_seq(n)+1,:);
    b = b(9-BT_NUM(n):end);
    b_stream = [b_stream,b];
end

% Converting the bit stream into 14 bytes
lsb_out = bin2dec(reshape(b_stream,[14,8]));

return
```

D.4: The Code in the File crc16.m

```
% function crc = crc16 (data, create)
%
% Returns 16-bit Cyclic Redundancy Check (CRC) for input data.
% The CRC table will be recreated if 'create' string is provided
% after the input data.
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function crc = crc16 (data, create)

if nargin ==1
    create = 'no';
end

global CRC16TBL;

pol = uint16(0x1021); % Generator polynomial
crc = uint16(0xFFFF); % Initial value of CRC16
ref_in = false; % Should input data be reflected or not
ref_out = false; % Should the output value of CRC16 be reflected or not
```

```
xor_out = 0x0000; % Value to be XORed with the final value of CRC16

% Create CRC16TBL if it does not exist
if isempty(CRC16TBL) || strcmpi(create,'create')
    CRC16TBL = uint16(zeros(1,256));
    for n = 0:255
        reg = uint16(n) * 0x0100;
        for i  = 1:8
            bt = bitget(reg,16);
            reg = bitshift(reg,1);
                if bt
                    reg = bitxor(reg,pol);
                end
        end
        CRC16TBL(n+1) = reg;
    end
end

for n = 1:length(data)
    XR1 = uint8(bitshift(crc,-8));
    crc_index  = data(n);
    if ref_in
        crc_index = bin2dec(fliplr(dec2bin(crc_index,8)));
    end
    crc_index = bitxor(crc_index,XR1);
    XR2 = mod(bitshift(crc,8),0x10000);
    crc = CRC16TBL(crc_index+1);
    crc = bitxor(crc,XR2);
end

if ref_out
    crc = bin2dec(fliplr(dec2bin(crc,16)));
end

crc = bitxor(crc,xor_out);

return
```

D.5: The Code in the File dct2blk.m

```
% function blk_out = dct2blk(lsb_in)
%
% Converts the DCT coefficients stored in 14 bytes into
% 8x8 px block.
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018
```

```
function blk_out = dct2blk(lsb_in)

global PRM64;
global Q
global BT_NUM
global LN
global DCT_MAT
global ZGZG2M64

% Converting lsb_in to a bit stream
b_stream =(dec2bin(lsb_in(:),8));
b_stream = b_stream(:)';

% Getting DCT coefficients from the bit stream
z_seq=zeros(1,LN);
indx = 1;

% Break the bit stream according to BT_NUM
z_seq=char([]);
indx=1;
for n = 1:LN
    z_seq = [z_seq;b_stream(indx:indx+BT_NUM(n)-1)];
    indx = indx+BT_NUM(n);
end

% Converting bit steam into decimal
z_seq = bin2dec(z_seq)';

% Shifting AC coefficients
z_seq(2:LN) = z_seq(2:LN) - 2 .^ (BT_NUM(2:LN) -1);

% Expanding z_seq length to 64
z_seq=[z_seq,zeros(1,64-LN)];

% Converting zigzag sequence into matrix
dct_blk = z_seq(ZGZG2M64);

% Dequantising DCT coefficients
dct_blk = dct_blk .* Q;

% Finding inverse DCT coefficients
blk_out = DCT_MAT'*dct_blk*DCT_MAT;

return
```

D.6: The Code in the File mat2zig.m

```
% function zig = mat2zig(mat)
%
% Converts a square matrix into zigzag order.
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function zig = mat2zig(mat)

SZ = sqrt(length(mat(:)));
if mod(SZ,1) ~= 0
    disp('Input matrix must be square.');
    error;
end
mat=reshape(mat,[SZ,SZ]);

SZ = SZ(1);
zig = zeros(1,SZ^2);

inc_r = 1; % row increment
inc_c = -1; % columnt increment

indx = 1;

r = 1;
c = 1;
while 1
    zig(indx) = mat(r,c);
    indx=indx+1;
    r=r+inc_r;
    c=c+inc_c;
    if r < 1
        r = 1;
        inc_r = inc_r * -1;
        inc_c = inc_c * -1;
    end
    if c < 1
        c = 1;
        inc_r = inc_r * -1;
        inc_c = inc_c * -1;
    end
    if (r > SZ) || (c > SZ)
        break
    end
end
```

```
while 1
    r=r+inc_r;
    c=c+inc_c;

    if r > SZ
        r = SZ;
        c = c+2;
        inc_r = inc_r * -1;
        inc_c = inc_c * -1;
    end
    if c > SZ
        c = SZ;
        r=r+2;
        inc_r = inc_r * -1;
        inc_c = inc_c * -1;
    end
    zig(indx) = mat(r,c);
    indx=indx+1;
    if (r == SZ) && (c == SZ)
        break;
    end
end

return
```

D.7: The Code in the File set_global.m

```
% function set_global(key)
%
% Initiates the global variables needed for the encoding
% and the decoding functions of the fourth proposed technique.
% This function must be called before calling
% the files encode.m and decode.m .
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function set_global(key)

global PRM64;
global XPRM64;
global Q
global BT_NUM
global LN
global DCT_MAT
global BT_LUT
```

```
global M2ZGZG64
global ZGZG2M64
global CRC16TBL
global MAX_LEVEL

% Generating permutation matrices
rand('seed',key);
PRM64 = randperm(64);
XPRM64 = [PRM64;[1:64]]';
XPRM64 = sortrows(XPRM64)';
XPRM64 = XPRM64(2,:);
PRM64 = reshape(PRM64,[8,8]);
XPRM64 = reshape(XPRM64,[8,8]);

% Quantisation matrix (Luminance Quantisation table for Jpeg)
Q = [16 11 10 16 24 40 51 61;   ...
     12 12 14 19 26 58 60 55;   ...
     14 13 16 24 40 57 69 56;   ...
     14 17 22 29 51 87 80 62;   ...
     18 22 37 56 68 109 103 77;  ...
     24 35 55 64 81 104 113 92;  ...
     49 64 78 87 103 121 120 101;  ...
     72 92 95 98 112 100 103 99];

% Defining number of bits assigned for each coefficient
% bt_num_64 =  [8,[7,7],[6,6,6],[6,6,6,6]];
BT_NUM =  [8,[7,7],[6,6,6],[1:4]*0+5,[1:5]*0+4,[1:6]*0+3,[1:7]*0+2];
LN = length(BT_NUM);
MAX_LEVEL = ((2 .^ BT_NUM) -1);

DCT_MAT = dctmtx(8);

% Generatign look-up-table for binary values of
% the numbers from 0 to 255

BT_LUT = dec2bin([0:255]',8);

% Generating zigzag sequence for 8x8 matrix
M2ZGZG64 = mat2zig([1:64]);

ZGZG2M64  = zig2mat([1:64]);

% Calling crc16 function to create CRC16TBL
crc16(0,'create');

return
```

D.8: The Code in the File blk2dct.m

```
% function mat = zig2mat(zig)
%
% Converts a zigzag sequence into a square matrix.
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function mat = zig2mat(zig)

SZ = sqrt(length(zig));

if mod(SZ,1) ~= 0
    disp('Zigzag sequence length must have integer square root');
    error;
end

mat = zeros(SZ);

inc_r = 1; % row increment
inc_c = -1; % columnt increment

indx = 1;

r = 1;
c = 1;
while 1
    mat (r,c) = zig(indx);
    indx=indx+1;
    r=r+inc_r;
    c=c+inc_c;
    if r < 1
        r = 1;
        inc_r = inc_r * -1;
        inc_c = inc_c * -1;
    end
    if c < 1
        c = 1;
        inc_r = inc_r * -1;
        inc_c = inc_c * -1;
    end
    if (r > SZ) || (c > SZ)
        break
    end
end
```

```
while 1
    r=r+inc_r;
    c=c+inc_c;

    if r > SZ
        r = SZ;
        c = c+2;
        inc_r = inc_r * -1;
        inc_c = inc_c * -1;
    end
    if c > SZ
        c = SZ;
        r=r+2;
        inc_r = inc_r * -1;
        inc_c = inc_c * -1;
    end
    mat(r,c) = zig(indx);
    indx=indx+1;

    if (r == SZ) && (c == SZ)
        break;
    end
end

return
```

Appendix E: The Implemented Code for Tong's Technique

Appendix E contains a list of the implemented code for the Tong's technique. The main two functions in ***encode.m*** and ***decode.m*** call the function in the file: ***blk_map.m.***

E.1: The Code for the Encoding Stage of Tong's Technique

```
% function [I_enc]=encode(I_in)
%
% The encoding function for Tong et al. technique (2013).
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [I_enc]=encode(I_in)
I_SZ = size(I_in);
I_H = I_SZ(1);
I_W = I_SZ(2);

W_SZ =I_SZ/2;
W_H = W_SZ(1);
W_W = W_SZ(2);

if sum(mod(I_SZ,4)) ~= 0
      disp('Dimensions of the image must be multiple of 4');
      return
end

% Set 3 LSBs in the input image to 0
I_del = bitand(I_in,bin2dec('11111000'));

% generating the recovery bits which are 5 MSBs of the average of
% 2x2 px blocks in the input image
W1 = I_del(1:2:end,1:2:end)/4+I_del(1:2:end,2:2:end)/4+ ...
     I_del(2:2:end,1:2:end)/4+I_del(2:2:end,2:2:end)/4;

W1 = bitand(W1,bin2dec('11111000'));

% Shifting the bits of the watermark
```

```
W1 = bitshift(W1,-2);

% Shuffling (i.e. confusing) the recovery bits

[R_map,C_map] = blk_map(W_SZ);

% To enhance Shuffling, it is repeated once
for count = 1:2
      for r = 1:W_H
            W1(r,:)=circshift(W1(r,:),[0,R_map(r)]);
      end

      for c = 1:W_W
            W1(:,c)=circshift(W1(:,c),[C_map(c),0]);
      end
end

% Make a copy of the recovery bits and apply
% maximum-distance mapping on it.
W2 = circshift(W1,W_SZ/2);

% Count the number of on-bits (i.e. bit = 1) in each 2x2 px block
% in the encoded image

% Construct a look-up-table for the number of on-bits in a number
LUT = sum(dec2bin(0:255)-'0',2)';
bit_cnt = LUT(I_del(1:2:end,1:2:end)+1);
bit_cnt = bit_cnt+LUT(I_del(1:2:end,2:2:end)+1);
bit_cnt = bit_cnt+LUT(I_del(2:2:end,1:2:end)+1);
bit_cnt = bit_cnt+LUT(I_del(2:2:end,2:2:end)+1);

% Adding the on-bit count of the recovery bits
bit_cnt = bit_cnt +LUT(W1+1);
bit_cnt = bit_cnt +LUT(W2+1);

% Finding the first localisation (i.e. detection) bit
P1 = mod(bit_cnt,2);

% Storing the first localisation bit
W1 = W1 + P1;

% Finding the second localisation (i.e. detection) bit
P2 = ~P1;

% Storing the second localisation bit
W2 = W2 + P2;
```

```
% Storing the watermark data
I_enc = bitand(I_in,bin2dec('11111000'));
and_msk = bin2dec('00000111');

I_enc(1:2:end,1:2:end) = I_enc(1:2:end,1:2:end) + bitand(W1,and_msk);
W1 = bitshift(W1,-3);
I_enc(1:2:end,2:2:end) = I_enc(1:2:end,2:2:end) + bitand(W1,and_msk);

I_enc(2:2:end,1:2:end) = I_enc(2:2:end,1:2:end) + bitand(W2,and_msk);
W2 = bitshift(W2,-3);
I_enc(2:2:end,2:2:end) = I_enc(2:2:end,2:2:end) + bitand(W2,and_msk);

return
```

E.2: The Code for the Decoding Stage of Tong's Technique

```
% function [I_fxd,vld_msk]=decode(I_in)
%
% The decoding function for Tong et al. technique (2013).
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [I_fxd,vld_msk]=decode(I_in)
I_SZ = size(I_in);
I_H = I_SZ(1);
I_W = I_SZ(2);

W_SZ =I_SZ/2;
W_H = W_SZ(1);
W_W = W_SZ(2);

if sum(mod(I_SZ,4)) ~= 0
    disp('The dimensions of the input image must be multiple of 4');
    return
end

% Getting the watermark data

I_tmp = bitand(I_in, bin2dec('00000111'));
W1 = I_tmp(1:2:end,2:2:end);
W1 = bitshift(W1,3);
W1 = W1 + I_tmp(1:2:end,1:2:end);

W2 = I_tmp(2:2:end,2:2:end);
W2 = bitshift(W2,3);
```

```
W2 = W2 + I_tmp(2:2:end,1:2:end);

% Getting the first localisation (i.e. detection) bit
P1 = bitand(W1,bin2dec('00000001'));

% Getting the second localisation (i.e. detection) bit
P2 = bitand(W2,bin2dec('00000001'));

% Setting the localisation bits in the watermark to 0
W1=bitand(W1,bin2dec('00111110'));
W2=bitand(W2,bin2dec('00111110'));

% Shifting the bits in W1 and W2
W1 = bitshift(W1,2);
W2 = bitshift(W2,2);

% Count the number of on-bits (i.e. bit = 1) in each 2x2 px block
% in the encoded image

% Construct a look-up-table for the number of on-bits in a number
I_del=bitand(I_in,bin2dec('11111000'));
LUT = sum(dec2bin(0:255)-'0',2)';
bit_cnt = LUT(I_del(1:2:end,1:2:end)+1);
bit_cnt = bit_cnt+LUT(I_del(1:2:end,2:2:end)+1);
bit_cnt = bit_cnt+LUT(I_del(2:2:end,1:2:end)+1);
bit_cnt = bit_cnt+LUT(I_del(2:2:end,2:2:end)+1);

% Adding the on-bit count of the recovery bits
bit_cnt = bit_cnt +LUT(W1+1);
bit_cnt = bit_cnt +LUT(W2+1);

% Finding the first localisation (i.e. detection) bit for the input-image
blocks
P1x = mod(bit_cnt,2);

% Finding the second localisation (i.e. detection) bit for the input-
image blocks
P2x = ~P1x;

% Generate a matrix that determine the validity of the input-image blocks
vld = (P1==P1x) & (P2==P2x);

% Marking the invalid blocks in the W1 and W2 by storing -1 in them

W1 = double(W1);
W2 = double(W2);
```

```
W1(~vld) = -1000;
W2(~vld) = -1000;

% Remappig W2 using maximum-distance mappign
W2 = circshift(W2,W_SZ/2);

% Shuffling back the recovery bits in W1 and W2
[R_map,C_map] = blk_map(W_SZ);

% To enhance Shuffling, it is repeated once
for count = 1:2
    for c = 1:W_W
        W1(:,c)=circshift(W1(:,c),[-C_map(c),0]);
        W2(:,c)=circshift(W2(:,c),[-C_map(c),0]);
    end

    for r = 1:W_H
        W1(r,:)=circshift(W1(r,:),[0,-R_map(r)]);
        W2(r,:)=circshift(W2(r,:),[0,-R_map(r)]);
    end
end

% recovering the tampered blocks
I_fxd = bitand(I_in,bin2dec('11111000'));

for r=1:W_H
    for c=1:W_W
        if ~vld(r,c)
            if W1(r,c)>=0
                r1 = (r-1)*2 + 1;
                r2 = r1+1;
                c1 = (c-1)*2 + 1;
                c2 = c1+1;

                I_fxd(r1:r2,c1:c2) = W1(r,c);
            elseif W2(r,c)>=0
                r1 = (r-1)*2 + 1;
                r2 = r1+1;
                c1 = (c-1)*2 + 1;
                c2 = c1+1;

                I_fxd(r1:r2,c1:c2) = W2(r,c);
            end
        end
    end
end
```

```
% generating the validation mask matrix
vld_msk =  logical(zeros(I_SZ));
vld_msk(1:2:end,1:2:end) = vld;
vld_msk(1:2:end,2:2:end) = vld;
vld_msk(2:2:end,1:2:end) = vld;
vld_msk(2:2:end,2:2:end) = vld;

return
```

E.3: The Code in the File blk_map.m

```
% function [R_map,C_map]=blk_map(I_SZ)
%
% The function returns the values that are used to confuse
% the encoded image, the confusing is done by circular rotation
% of the rows and the columns of the image.
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [R_map,C_map]=blk_map(I_SZ)

% The width and height of the image
I_H = I_SZ(1);
I_W = I_SZ(2);

% The initial value (x0,y0) and the number of iterations (itr)
% are considered as the secret key.

x0 = 0.1;  % This is the value selected in the paper
y0 = 0.3;  % This is the value selected in the paper
itr = I_H; % As in the paper, the iteration number is the same
           % as the image size.

% Initializing row and column shuffling matrices
max_SZ = max([I_H,I_W]);
R_map = zeros([1,max_SZ]);
C_map = zeros([1,max_SZ]);

% Initialising the chaotic sequence
for n  = 1:itr
    x1 = 1 - 2 * y0^2;
    y1 = cos(6 * acos(x0));
    x0=x1;
    y0=y1;
end
```

```
for count  = 1:max_SZ
    x0=x1;
    y0=y1;

    x1 = 1 - 2 * y0^2;
    y1 = cos(6 * acos(x0));

    x=x1;
    x=x*10^10;
    x=abs(floor(x));
    x=mod(x,I_W);

    y=y1;
    y=y*10^10;
    y=abs(floor(y));
    y=mod(y,I_H);

    R_map(count) = y;
    C_map(count) = x;
end

R_map = R_map(1:I_H);
C_map = C_map(1:I_W);

return
```

Appendix F: The Implemented Code for Dadkhah's Technique

Appendix F contains a list of the implemented code for the Dadkhah's technique. The main two functions in ***encode.m*** and ***decode.m*** call the functions in the files: ***blk_map.m, loc_bits.m, and set_global.m***. The file ***set_global.m*** must be called before calling the ***encode.m*** and ***decode.m*** files in order to set the global variables used by them.

F.1: The Code for the Encoding Stage of Dadkhah's Technique

```
% function [I_enc]=encode(I_in,key)
%
% The encoding function for Dadkhah et al. technique (2014).
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [I_enc]=encode(I_in,key)
I_SZ = size(I_in);
I_H = I_SZ(1);
I_W = I_SZ(2);

W_SZ =I_SZ/2;
W=uint8(zeros(W_SZ));
W_H = W_SZ(1);
W_W = W_SZ(2);

if sum(mod(I_SZ,8)) ~= 0
    disp('Dimensions of the image must be multiple of 8');
    return
end

I_enc = bitand(I_in,bin2dec('11111100'));

N = (I_W*I_H)/(4*4);

map = blk_map(N,key);

blk_num = 0;
```

```
for r = 1:4:I_H
    for c = 1:4:I_W

        blk_num=blk_num+1;

        blk1 = I_enc(r:r+1,c:c+1);
        blk2 = I_enc(r:r+1,c+2:c+3);
        blk3 = I_enc(r+2:r+3,c:c+1);
        blk4 = I_enc(r+2:r+3,c+2:c+3);

        % generating localisation bits and storing them in a matrix
        loc_bts = [loc_bits(blk1),loc_bits(blk2); loc_bits(blk3), ...
        loc_bits(blk4)];

        % finding the row and column numbers for current block
        % in the watermark
        rw = floor((r-1)/2) + 1;
        cw = floor((c-1)/2) + 1;

        % storing the localisation bits in the current block
        % in the watermark
        W(rw:rw+1,cw:cw+1) = W(rw:rw+1,cw:cw+1)+ loc_bts;

        % finding the recovery bits of the current block
        % recoery bits are 5 MSBs of the average of each 2x2 px block
        and_msk = bin2dec('11111000');
        rec_bts1 = bitand(round(sum(sum(blk1))/4),and_msk);
        rec_bts2 = bitand(round(sum(sum(blk2))/4),and_msk);
        rec_bts3 = bitand(round(sum(sum(blk3))/4),and_msk);
        rec_bts4 = bitand(round(sum(sum(blk4))/4),and_msk);

        % finding the row and column numbers for destination block
        % in the watermark
        rwd = (floor((map(blk_num)-1)/(W_W/2)) * 2) +1;
        cwd = (mod((map(blk_num)-1),(W_W/2)) * 2) +1;

        % store the localisation bits in a matrix
        rec_bts = [rec_bts1,rec_bts2; rec_bts3, rec_bts4];

        % storing the recovery bits in the destination block
        % in the watermark
        W(rwd:rwd+1,cwd:cwd+1) = W(rwd:rwd+1,cwd:cwd+1)+ rec_bts;
    end
end

% encrypting the watermark by by XORing it with a random sequence
```

```
% In the paper, the random sequence depends also on block number
% However, for simplicity the random sequence in the code
% depends only on the secret key.

rand('seed',key)
seq = randi([0,255],[W_H,W_W]);

W = bitxor(W,seq);

% embedding the watermark in 2 LSBs of the image

and_msk = bin2dec('00000011');

I_enc(1:2:end,1:2:end) = I_enc(1:2:end,1:2:end) + bitand(W,and_msk);
W = bitshift(W,-2);
I_enc(1:2:end,2:2:end) = I_enc(1:2:end,2:2:end) + bitand(W,and_msk);
W = bitshift(W,-2);
I_enc(2:2:end,1:2:end) = I_enc(2:2:end,1:2:end) + bitand(W,and_msk);
W = bitshift(W,-2);
I_enc(2:2:end,2:2:end) = I_enc(2:2:end,2:2:end) + bitand(W,and_msk);

return
```

F.2: The Code for the Decoding Stage of Dadkhah's Technique

```
% function [I_fxd,vld_msk,rec_fail]=decode(I_in,key)
%
% The decoding function for Dadkhah et al. technique (2014)
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [I_fxd,vld_msk,rec_fail]=decode(I_in,key)

I_SZ = size(I_in);
I_H = I_SZ(1);
I_W = I_SZ(2);

W_SZ =I_SZ/2;

W=uint8(zeros(W_SZ));
W_H = W_SZ(1);
W_W = W_SZ(2);

if sum(mod(I_SZ,8)) ~= 0
    disp('Dimensions of the image must be multiple of 8');
```

```
    return
end

% retreive the watermark from the 2 LSB of the image
and_msk = bin2dec('00000011');
I_tmp = bitand(I_in,and_msk);
W = W+I_tmp(2:2:end,2:2:end);
W = bitshift(W,2);
W = W+I_tmp(2:2:end,1:2:end);
W = bitshift(W,2);
W = W+I_tmp(1:2:end,2:2:end);
W = bitshift(W,2);
W = W+I_tmp(1:2:end,1:2:end);

% encrypting the watermark by by XORing it with a random sequence
% In the paper, the random sequence depends also on block number
% However, for simplicity the random sequence in the code
% depends only on the secret key.

rand('seed', key);
seq = randi([0,255],[W_H,W_W]);
W = bitxor(W,seq);


% Setting 2 LSBs in the input image to 0
I_tmp = bitand(I_in,bin2dec('11111100'));

N = (I_W*I_H)/(4*4);

map = blk_map(N,key);

% Note: the authentication process is done for all of 4x4 px blocks
% i.e.: if any 2x2 block is tampered then all 4x4 block is tampered

% performing first level authentication which depends on the
% localisation bits that are stored in eachblock, -1 is stored
% in the watermark block if it is tampered
blk_num = 0;

% vld is block validation matrix, tampered blocks = 0
vld = logical(zeros(W_SZ));
for r = 1:4:I_H
    for c = 1:4:I_W

        blk_num=blk_num+1;

        % getting the 2x2 px blocks from the input image
```

```
        blk1 = I_tmp(r:r+1,c:c+1);
        blk2 = I_tmp(r:r+1,c+2:c+3);
        blk3 = I_tmp(r+2:r+3,c:c+1);
        blk4 = I_tmp(r+2:r+3,c+2:c+3);

        % generating localisation bits and storing them in a matrix
        loc_bts = [loc_bits(blk1),loc_bits(blk2); loc_bits(blk3), ...
        loc_bits(blk4)];

        % finding the row and column numbers for current block
        % in the watermark
        rw = floor((r-1)/2) + 1;
        cw = floor((c-1)/2) + 1;

        % getting the localisation bits matrix from the watermark
        wloc_bts = W(rw:rw+1,cw:cw+1);
        wloc_bts = bitand(wloc_bts,bin2dec('00000111'));

        % Compare generated loacalisation bit to the localisation bits
        % in the watermark.

        vld(rw:rw+1,cw:cw+1) = min(min(wloc_bts == loc_bts));
    end
end

blk_num = 0;
% Second level authentication based on recovery bits matching
for r = 1:4:I_H
    for c = 1:4:I_W

        blk_num=blk_num+1;

        % finding the row and column numbers for current block
        % in the watermark
        rw = floor((r-1)/2) + 1;
        cw = floor((c-1)/2) + 1;

        % Skip if the current 4x4 px block is tampered
        if ~vld(rw,cw)
            continue;
        end

        % Finding row and column numbers for the destination block
        % in the watermark
        rwd = (floor((map(blk_num)-1)/(W_W/2)) * 2) +1;
        cwd = (mod((map(blk_num)-1),(W_W/2)) * 2) +1;
```

```
        % Skip if the destination block is tampered
        if ~vld(rwd,cwd)
            continue;
        end

        % Check if the recovery bits stored in the destination block are
        % equal to the generated ones for the current block.

        % Getting the 2x2 px blocks from the input image
        blk1 = I_tmp(r:r+1,c:c+1);
        blk2 = I_tmp(r:r+1,c+2:c+3);
        blk3 = I_tmp(r+2:r+3,c:c+1);
        blk4 = I_tmp(r+2:r+3,c+2:c+3);

        % Finding the recovery bits of the current block.
        % Recoery bits are 5 MSBs of the average of each 2x2 px block.
        and_msk = bin2dec('11111000');
        rec_bts1 = bitand(round(sum(sum(blk1))/4),and_msk);
        rec_bts2 = bitand(round(sum(sum(blk2))/4),and_msk);
        rec_bts3 = bitand(round(sum(sum(blk3))/4),and_msk);
        rec_bts4 = bitand(round(sum(sum(blk4))/4),and_msk);

        % Store the localisation bits of the current block in a matrix.
        rec_bts = [rec_bts1,rec_bts2; rec_bts3, rec_bts4];

        % Getting the recovery bits from the watermark in
        % the destination block.
        wrec_bts = bitand(W(rwd:rwd+1,cwd:cwd+1),bin2dec('11111000'));

        % Compare the generated recovery bits to the stored ones.
        vld(rw:rw+1,cw:cw+1) = min(min(wrec_bts == rec_bts));
    end
end

% Generating a matrix to indicate block-recovery failure
rec_fail = logical(zeros(I_SZ));

% Recovering Tampered blocks
I_fxd=bitand(I_in,bin2dec('11111100'));
blk_num = 0;
% Second level authentication based on recovery bits matching
for r = 1:4:I_H
    for c = 1:4:I_W

        blk_num=blk_num+1;

        % finding the row and column numbers for current block
```

```
        % in the watermark
        rw = floor((r-1)/2) + 1;
        cw = floor((c-1)/2) + 1;


        % Skip if the current 4x4 px block is not tampered
        if vld(rw,cw)
            continue;
        end

        % Finding row and column numbers for the destination block
        % in the watermark
        rwd = (floor((map(blk_num)-1)/(W_W/2)) * 2) +1;
        cwd = (mod((map(blk_num)-1),(W_W/2)) * 2) +1;

        % Skip if the destination block is tampered and update
        % failure matrix
        if ~vld(rwd,cwd)
            rec_fail(r:r+3,c:c+3) = true;
            continue;
        end

        % Getting the recovery bits from the watermark in the
        % destination block.
        wrec_bts = bitand(W(rwd:rwd+1,cwd:cwd+1),bin2dec('11111000'));

        % Recover the tampered block
        I_fxd(r:r+3,c:c+3) = imresize(wrec_bts,2,'linear');
    end
end

% generating the validatoin mask matrix by resizing the validation matrix
vld_msk =  logical(zeros(I_SZ));
vld_msk(1:2:end,1:2:end) = vld;
vld_msk(1:2:end,2:2:end) = vld;
vld_msk(2:2:end,1:2:end) = vld;
vld_msk(2:2:end,2:2:end) = vld;

return
```

F.3: The Code in the File blk_map.m

```
% function output = blk_map(N,seed)
%
% The mapping for the blocks in Dadkhah et al. technique (2014)
% The output is 1 by N matrix that has the corresponding
% mapped block for the blocks from 1 to N
```

```
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function output = blk_map(N,seed)

if mod(N,2) ~= 0
    disp ('The input value must be multiple of 2');
    return;
end

rand('seed',seed);
R2 = randperm(N/2)+N/2;

% The following code ensure that if A is mapped to B then
% B will not be mapped to A
map = [R2;1:N/2]';
map=sortrows(map)';
R1 = map(2,:);
R1 = circshift(R1,[0,floor(N/4)]);

output = [R2,R1];

return
```

F.4: The Code in the File loc_bits.m

```
% function output = loc_bits(input)
%
% Returns the authentication bits for 2x2 block in
% Dadkhah et al. technique (2014)
% The bits are returned as a decimal number
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function output = loc_bits(input)

global PARITY;

SV = svd(input); % calculate singular values

FL1 = floor(SV(1));
FL2 = floor(SV(2));

if FL1~=SV(1) | FL2~=SV(2) % one of the SVs is not integer
```

```
    A1 = floor((SV(1)-FL1)*100) > 50;      % Condition is not clear
    A2 = floor((SV(2)-FL2)*100) > 50;      % in the paper.
    A3  = (PARITY(FL1+1) == 0) & (PARITY(FL2+1) == 0);

    if A1 & A2 & A3
        output=0;
    elseif A1 & ~A2 & A3
        output=4;
    elseif A1 & A2 & ~A3
        output=2;
    elseif ~A1 & A2 & A3
        output=1;
    elseif ~A1 & ~A2 & A3
        output=6;
    elseif A1 & ~A2 & ~A3
        output=5;
    elseif ~A1 & A2 & ~A3
        output=3;
    elseif ~A1 & ~A2 & ~A3
        output=7;
    end
else % all SVs are integers
    B1 = (PARITY(FL1+1) == 0) & (PARITY(FL2+1) == 0);
    B2 = (SV(1) <= 50) & (SV(2) <= 50);
    B3 = ~B1;
    output = B1*4 + B2*2 + B3;
end

return
```

F.5: The Code in the File set_global.m

```
% function set_global()
%
% Initiates the global variables needed for the encoding
% and the decoding functions of Dadkhah et al. technique.
% This function must be called before calling
% the files encode.m and decode.m .
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function set_global()

global PARITY;

PARITY=zeros(1,1024);
```

```
for indx = 1:1024
    PARITY(indx) = mod(sum(bitget((indx-1),1:10)),2);
end
return
```

Appendix G: The Implemented Code for the Experimental Evaluation

Appendix G presents the code used for the experimental evaluation of the proposed and the referenced techniques.

G.1: The Folder Structure and File Description of the Implemented Code

The folder structure and the contained files in the main code folder ***My_Code*** are shown in Figure G.1, the code is organised in such a way it could be reused for the different techniques.

Each technique has its own subfolder, which contains a subfolder called ***coder,*** The ***coder*** subfolder contains the main encoding and decoding files along with any functions they need. In each technique's folder there are two files: ***test_kodak.m*** which is used in finding the experimental evaluation using Kodak image database, and the other file is ***test_resolution.m*** which is used to find the encoding and decoding time for different image resolutions.

The ***images*** folder contains the images used by the code, such as ***lighthouse.png*** and ***peppers.png***. It also contains some subfolders that contain different formats and sizes of the Kodak database images.

The folder ***shared*** contains some functions files that are used by the files: ***test_kodak.m*** and ***test_resolution.m***. The files are: ***pfa.m*** which is used to calculate the PFA, ***pfr.m*** which is to calculate the PFR, ***rect_tmpr.m*** which is used to tamper the images.

```
My_Code/
├── Dadkhah_2014/
│   ├── coder/
│   │   ├── blk_map.m
│   │   ├── decode.m
│   │   ├── encode.m
│   │   ├── loc_bits.m
│   │   └── set_global.m
│   ├── test_kodak.m
│   └── test_resolution.m
├── images/
│   ├── kodak512/
│   ├── kodak516/
│   ├── kodak_grey/
│   ├── kodak_rgb/
│   ├── lighthouse.png
│   └── peppers.png
├── shared/
│   ├── pfa.m
│   ├── pfr.m
│   └── rect_tmpr.m
├── Technique1/
│   ├── coder/
│   │   ├── decode.m
│   │   └── encode.m
│   ├── test_kodak.m
│   └── test_resolution.m
├── Technique2/
│   ├── coder/
│   │   ├── decode.m
│   │   └── encode.m
│   ├── test_kodak.m
│   └── test_resolution.m
├── Technique3/
│   ├── coder/
│   │   ├── decode.m
│   │   └── encode.m
│   ├── test_kodak.m
│   └── test_resolution.m
├── Technique4/
│   ├── coder/
│   │   ├── blk2dct.m
│   │   ├── crc16.m
│   │   ├── dct2blk.m
│   │   ├── decode.m
│   │   ├── encode.m
│   │   ├── mat2zig.m
│   │   ├── set_global.m
│   │   └── zig2mat.m
│   ├── test_kodak.m
│   └── test_resolution.m
└── Tong_2013/
    ├── coder/
    │   ├── blk_map.m
    │   ├── decode.m
    │   └── encode.m
    ├── test_kodak.m
    └── test_resolution.m
```

Figure G.1: Folder structure for the implemented code in the experimental evaluation.

G.2: The Code in the File test_kodak.m

The code in the ***test_kodak.m*** file that corresponds to the first technique will be listed here, the differences for other techniques are highlighted inside the code.

```
% Testing Kodak database images for the first technique
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

clear all;
close all;
warning ('off','all');

display(strftime ("Start at: %H:%M:%S", localtime (time ())))
fflush(stdout)

T0 = tic;

% adding paths for shared code
addpath('coder/');
addpath('../shared/');

% adding image directory
img_dir = '../images/kodak512/';

% Note:
% The directory '../images/kodak512/' is used for all techniques
% except for the second one where the folder '../images/kodak516/'
% is used.

% load image processing package (Required for Octave)
pkg load image
pkg load signal

% reading image list
img_lst = dir([img_dir,'*.png']);
img_count = length(img_lst);

% Creating results directory
DT = clock;
DT = floor(DT(2:end));
DT = sprintf( '%02d', DT );
r_dir = ['results',DT];
mkdir(r_dir);
```

```
% creating the results files
f_psnr = fopen([r_dir,'/kodak_psnr.csv'],'w');
f_pfa = fopen([r_dir,'/kodak_pfa.csv'],'w');
f_pfr = fopen([r_dir,'/kodak_pfr.csv'],'w');
f_t_enc = fopen([r_dir,'/kodak_t_encoding.csv'],'w');
f_t_dec = fopen([r_dir,'/kodak_t_decoding.csv'],'w');

key = 12345;

ratio = [0,0.05,0.1,0.15,0.2,0.24,0.25,0.3,0.35,0.4,0.45,0.49,0.5];

fprintf(f_psnr,["Ratio",sprintf(',%.2f' , ratio),"\n"]);
fprintf(f_pfa,["Ratio",sprintf(',%.2f' , ratio),"\n"]);
fprintf(f_pfr,["Ratio",sprintf(',%.2f' , ratio),"\n"]);
fprintf(f_t_enc,["Ratio",sprintf(',%.2f' , ratio),"\n"]);
fprintf(f_t_dec,["Ratio",sprintf(',%.2f' , ratio),"\n"]);

for img_num = 1:img_count

    fprintf(f_psnr, img_lst(img_num).name);
    fprintf(f_pfa, img_lst(img_num).name);
    fprintf(f_pfr, img_lst(img_num).name);
    fprintf(f_t_enc, img_lst(img_num).name);
    fprintf(f_t_dec, img_lst(img_num).name);

    I = imread([img_dir,img_lst(img_num).name]);

    % Selecting the image used in tampering which is the next one
    % in the list and for the last one the first one is selected.

    if img_num == img_count;
        tmpr_img = 1;
    else
        tmpr_img = img_num+1;
    end

    I_tmpr = imread([img_dir,img_lst(tmpr_img).name]);

    for rat_indx = 1:length(ratio)

        % Finding the width and height the tampered area
        tmpr_w = sqrt(ratio(rat_indx));
        tmpr_h = tmpr_w;
        tmpr_ratio = [tmpr_h,tmpr_w];

        % Note:
        % The previous tmpr_w and tmpr_h are used for the techniques:1,2,
```

```
        % and Dadkhah's technique while for other techniques
        % the values are:
        % tmpr_w = sqrt(ratio(rat_indx)/2);
        % tmpr_h = tmpr_w*2;

        t0_enc = tic;
        I_enc = encode(I,key);
        T_ENC = toc(t0_enc);

        [I_mod,tmpr_msk] = rect_tmpr(I_enc,tmpr_ratio,I_tmpr);

        t0_dec = tic;
        [I_fxd,vld] = decode(I_mod,key);
        T_DEC = toc(t0_dec);

        PSNR = psnr(I,I_fxd);
        PFA = pfa(tmpr_msk,vld)*100;
        PFR = pfr(tmpr_msk,vld)*100;

        fprintf(f_psnr,",%.4f", PSNR);
        fprintf(f_pfa,",%.4f", PFA);
        fprintf(f_pfr,",%.4f", PFR);
        fprintf(f_t_enc,",%.4f", T_ENC);
        fprintf(f_t_dec,",%.4f", T_DEC);
    end

    % Adding new-line to the files
    fprintf(f_psnr,"\n");
    fprintf(f_pfa,"\n");
    fprintf(f_pfr,"\n");
    fprintf(f_t_enc,"\n");
    fprintf(f_t_dec,"\n");
end

%closing the files
fclose(f_psnr);
fclose(f_pfa);
fclose(f_pfr);
fclose(f_t_enc);
fclose(f_t_dec);

% Reading files the results files
PSNR_mat = dlmread ([r_dir,'/kodak_psnr.csv'],',',1,1);
PFA_mat = dlmread ([r_dir,'/kodak_pfa.csv'],',',1,1);
PFR_mat = dlmread ([r_dir,'/kodak_pfr.csv'],',',1,1);
T_ENC_mat = dlmread ([r_dir,'/kodak_t_encoding.csv'],',',1,1);
T_DEC_mat = dlmread ([r_dir,'/kodak_t_decoding.csv'],',',1,1);
```

```
% Finding the mean values
PSNR_av = mean(PSNR_mat);
PFA_av = mean(PFA_mat);
PFR_av = mean(PFR_mat);
T_ENC_av = mean(T_ENC_mat);
T_DEC_av = mean(T_DEC_mat);

% Storing the mean values in a file
f_av = fopen([r_dir,'/kodak_average.csv'],'w');
fprintf(f_av,["Ratio",sprintf(',%.2f' , ratio),"\n"]);

fprintf(f_av,["PSNR",sprintf(',%.4f' , PSNR_av),"\n"]);
fprintf(f_av,["PFA*100",sprintf(',%.4f' , PFA_av),"\n"]);
fprintf(f_av,["PFR*100",sprintf(',%.4f' , PFR_av),"\n"]);
fprintf(f_av,["T_ENC",sprintf(',%.4f' , T_ENC_av),"\n"]);
fprintf(f_av,["T_DEC",sprintf(',%.4f' , T_DEC_av),"\n"]);
fclose(f_av);

T_total = toc(T0);

disp(['Total time = ',num2str(floor(T_total/60)), ...
' Min ',num2str(mod(T_total,60)),' Sec.'])
```

G.3: The Code in the File test_resolution.m

```
% The code for testing different image resolutions
% This code is the same for all techniques
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

clear all;
close all;

warning ('off','all');

% adding path for common codes
addpath('coder/');
addpath('../shared/');

% load image processing package (Required for Octave)
pkg load image
pkg load signal

img_dir = '../images/';
```

```
% Creating results directory
DT = clock;
DT = floor(DT(2:end));
DT = sprintf( '%02d', DT );
r_dir = ['res_results',DT];
mkdir(r_dir);

% creating the results files
f_t_enc = fopen([r_dir,'/res_t_enc.csv'],'w');
f_t_dec = fopen([r_dir,'/res_t_dec.csv'],'w');

I = imread([img_dir,'lighthouse.png']);
I_tmpr = imread([img_dir,'peppers.png']);

key = 1234;

res = 120*[1:10];

fprintf(f_t_enc,["Image Size",sprintf(',%d' , res .^ 2),"\n"]);
fprintf(f_t_dec,["Image Size",sprintf(',%d' , res .^ 2),"\n"]);

t_enc = zeros(1,length(res));
t_dec = zeros(1,length(res));
i = 1;
for r = res

    % scaling images
    Ix = imresize(I, [r,r]);
    Ix_tmpr = imresize(I_tmpr, [r,r]);

    t0_enc = tic;
    I_enc = encode(Ix,key);
    t_enc(i) = toc(t0_enc);

    ratio=[.45,.45];
    [I_mod,tmpr_msk] = rect_tmpr(I_enc,ratio,Ix_tmpr);

    t0_dec = tic;
    [I_fxd,vld] = decode(I_mod,key);
    t_dec(i) = toc(t0_dec);

    i=i+1;
end

% Saving data to files
fprintf(f_t_enc,["Time (sec)",sprintf(',%0.4f' , t_enc),"\n"]);
fprintf(f_t_dec,["Time (sec)",sprintf(',%0.4f' , t_dec),"\n"]);
```

```
% Closing files
fclose(f_t_enc);
fclose(f_t_dec);
```

G.4: The Code in the File pfa.m

```
% function PFA = pfa(tmpr_msk,vld_msk)
%
% Probability of false acceptance
%
% tmpr_msk: Tampered area mask (tampered pixel = 1)
% vld_msk: Validity mask image returned by tamper recovery decoding
%          functions,valid (i.e. untampered) pixels = 1.
%
% PFA: Probability of false acceptance
% = number of tampered blocks detected as valid/number of tampered blocks
%
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function PFA = pfa(tmpr_msk,vld_msk)

SM = sum(sum(tmpr_msk));
if SM == 0
    PFA = 0;
else
    PFA = sum(sum(vld_msk & tmpr_msk))/SM;
end

return
```

G.5: The Code in the File pfr.m

```
% function PFR = pfr(tmpr_msk,vld_msk)
%
% Probability of false rejection
%
% PFR = number of
% tmpr_msk: Tampered area mask (tampered pixel = 1)
% vld_msk: Validity mask image returned by tamper recovery decoding
%           functions, valid (i.e. untampered) pixels = 1.
%
% PFR: Probability of false rejection
% = number of valid blocks detected as tampered / number of valid blocks.
%
```

```
% Author: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function PFR = pfr(tmpr_msk,vld_msk)

SM = sum(sum(~tmpr_msk));
if SM == 0
    PFR = 0;
else
    PFR = sum(sum(~vld_msk & ~tmpr_msk))/SM;
end

return
```

G.6: The Code in the File rect_tmpr.m

```
% function [I_out,msk] = rect_tmpr(I_in,ratio,I_tmpr)
%
% Tampering the central rectangular region of the input image.
% I_in: The input image.
% ratio: The hight and the width ratios of the tampering rectangle
%        with respect to the image dimensions.
% I_tmp: The image used for tampering (default is 0).
% msk: A logical matrix where the tampered area 1.
%
% Auther: Mahmoud Alnaanah (malnaanah@gmail.com)
% Update: 19 July 2018

function [I_out,msk] = rect_tmpr(I_in,ratio,I_tmpr)

I_SZ = size(I_in);
I_H = I_SZ(1);
I_W = I_SZ(2);

if sum(mod(I_SZ,2))~=0
    error('Input image dimensions must be multiple of 2.');
end

r1 = floor((I_H/2)*(1-ratio(1))) + 1;
r2 = r1 + round(I_H*ratio(1)) -1;

c1 = floor((I_W/2)*(1-ratio(2))) + 1;
c2 = c1 + round(I_W*ratio(2)) -1;

if nargin == 3
    if sum(size(I_tmpr) == I_SZ) ~= 2;
```

```
        I_tmpr = imresize(I_tmpr,I_SZ);
    end
end

if nargin==2
    I_tmpr = zeros(I_SZ);
end

%  Adjust the tampering image pixel level so that the minimum difference
%  between it and the tampered image is 5.

MIN_DF = 5;

DF = double(I_tmpr)-double(I_in);

indx = DF <= MIN_DF  &  DF >= 0;
DF(indx) = -DF(indx) + MIN_DF;

indx = DF >= -MIN_DF & DF < 0;
DF(indx) = -DF(indx) - MIN_DF;

indx = DF < -MIN_DF | DF > MIN_DF;
DF(indx) = 0;

I_new = double(I_tmpr) + DF;

indx = I_new > 255;
I_new(indx) = I_new(indx) - 2 * MIN_DF;

indx = I_new < 0;
I_new(indx) = I_new(indx) + 2 * MIN_DF;

I_tmpr = uint8(I_new);

I_out = I_in;
msk = logical(zeros(I_SZ));

I_out(r1:r2,c1:c2)=I_tmpr(r1:r2,c1:c2);
msk(r1:r2,c1:c2) = true;

return
```