FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

FEUP

Integration and optimization of collusion secure fingerprinting in image watermarking

Fábio Haruo Touceira Takahashi

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Supervisor at FEUP: Maria Teresa Magalhães da Silva Pinto Andrade (Assistant Professor)

Supervisor at Fraunhofer SIT: Huajian Liu (Dr. Ing)

Supervisor at Fraunhofer SIT: Marcel Schäfer (Dipl. Math.)

26 October 2012

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | 2011/2012

A Dissertação intitulada

"Integration and Optimization od Collusion - Secure Fingerprinting in Image Watermarking"

foi aprovada em provas realizadas em 02-10-2012

o júri

Presidente Professor Doutor Jaime dos Santos Cardoso
Professor Auxiliar do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor José Manuel de Castro Torres
Professor Associado da Faculdade de Ciências e Tecnologia da Universidade Fernando Pessoa

Professora Doutora Maria Teresa Magalhães da Silva Pinto de Andrade
Professora Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Fábio Harvo Touceira Takahashi

Faculdade de Engenharia da Universidade do Porto

Abstract

In the recent years the exponential growth of the internet favoured the emergence of digital markets such as online stores. The advantages brought with this type market are several; From is the easiness of buying a digital format content without quality degradation, to the commodity of making it all from the distance of a few clicks without the necessity of the customer leave his home. Still, several problems appeared with this crescent expansion, more specifically regarding copyright management. The spread of peer-to-peer shared networks is a good example of illegal market on the internet, and the necessity of copyright protection has become an important issue regarding internet markets nowadays. Watermarking is a technique which consists in embedding in an imperceptible way to the human senses a message into a cover work. The objective of that mark may serve many purposes, for example as an authenticity proof (the usage of watermarking for authenticity purposes is the first known application of this technique) or a transaction watermarking, which is the application for which watermarking is used in this thesis. Transaction watermarking is a technique with the objective of discouraging illegal distribution of copyrighted multimedia work by individually marking each copy prior its distribution. The objective of marking each copy with a unique watermark message is to identify each user that holds a copy of a multimedia content. When a group of users compare their copies they can identify positions in their copies that disagree. By manipulation of their copies they can change those disagreeing positions and try to remove the watermark message by outputting another modified copy. This type of malicious manipulation is called collusion attack. In order to pre-empt collusion attacks, collusion-secure fingerprint codes are used. Fingerprinting messages are mathematically generated codes with a certain length m capable of being secure against a collusion of c users. The objective of this thesis is to integrate the fingerprinting watermarking into a set of images. One single image has a small watermark payload, and the fingerprinting codes are known by their large sizes, which are impossible to embed in one single image in most of the cases. There is thus the necessity of using a set of images to keep the applicability of the fingerprint codes. Accordingly, in this thesis a set of images will be used to embed a fingerprinting code. The proposed approach can be used for many types of multimedia resources which possess a large number of images like video-games or e-books for children. The implementation is made to meet real world application requirements and therefore it uses the container watermarking strategy, that consists in splitting the images in blocks with size N and embedding in the DWT low-frequency sub-band blocks only one bit. The achieved results demonstrates that the size of the images play fundamental role in the watermark robustness. Whereas larger image increases the payload, if they are resized to smaller ratio, more information is contained in each block of the image in comparison to the larger images with the same block size. Nevertheless, the robustness achieved in our work against most of common image post-processing operations and geometrical attacks enables the utilization of this strategy for a future application of this framework.

Acknowledgement

I would especially like to thank my supervisors, Dr. Ing. Huajian Liu of the Fraunhofer Institute SIT/CASED and Prof. Maria Teresa Andrade, of the Faculty of Engineering of University of Porto for their patience, insightful advice, and constructive feedback during the entire period of research work.
To other members of my thesis supervisory, evaluation and monitoring panel, Marcel Schäfer and Waldemar Berchtold, both of the Fraunhofer Institute SIT/CASED, a special thanks is also due to their guidance, helpful feedback and peerless dedication. I would to thank also to all the CASED members where this thesis was developed for the assistance provided when I needed.
From the Faculty of Engineering of University of Porto, I would like to thank to all my professors. Without their support and dedication in the last five years this work would never be accomplished.
I also would like to thank my parents João Takahashi and Maria Takahashi for their unconditional support during all these years. Their help and patience taught me to exert myself in order to accomplish success in life.
To all my friends that I have made in Brazil, Portugal and more recently in Germany I would like to thank them all for their fellowship in difficult times and for their support in all my decisions made during all these years.

Fábio Takahashi

Contents

List of Figures

List of Tables

Abbreviations and symbols

BER	Bit Error Rate
C-LPCD	Constrained Local Permutation with Cancellation and Duplication
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DRM	Digital Rights Management
DWT	Discrete Wavelet Transform
GRF	Gibbs Random Field
ID	Identification
FN	False Negative
FP	False Positive
HVS	Human Vision System
LPCD	Local Permutation with Cancellation and Duplication
MB	Megabytes
P2P	Peer-to-Peer
PC	Personal Computer
PSNR	Peak Signal-to-Noise Ratio
RBG	Red Blue Green
TV	Television

ε	fingerprinting codes error length
λ	Multiplicative parameter for the histogram region selection
ω	Watermark bit embedded
$\sum$	fingerprinting codes alphabet size
τ	Threshold for watermark detection
$B_{1,2}$	Region centres 1 and 2 for the watermarking embedding
c_o	maximum collusion size
c	number of colluders
$G_{1,2}$	Total region 1 and 2 for the watermarking embedding
H	Image Height
m	fingerprint code length
M	Number of bins selected for the neighbourhood B
n	number of users for a fingerprinting scheme
N	Size of the blocks for embedding
n_{blocks}	Total number of blocks used for embedding in one set of images
l_{blocks}	Number of bits that are repeated one more time than the minimum repetition
r	number of bins for an embedding region
R	Fingerprinting rate
r_{blocks}	Minimum number of repetitions of one embedded bit in the set of images
T	Threshold for watermark message embedding
T_{blocks}	Maximum number of blocks capable to be embedded in one image
U	Histogram interval suitable for watermark embedding
W	Image Width

Chapter 1

Introduction

Nowadays the effects caused by the fast expansion of the internet can be experienced worldwide. The information broadcasting has become faster and more efficient due to the improvements made in the communications field. Consequently, the internet has turned into a place where one can share any type of information within a few clicks to the whole internet community. In recent years, the popularization of the internet favoured the transition of the multimedia contents from analog to digital domain. With small effort, digital multimedia contents can be replicated arbitrary times without or with only little quality loss, since it is not only the owner of the content who could replicate but in general any person can. Although, these characteristics of the digital multimedia contents entail some issues such as the *copyright management*.

The illegal distribution of copyrighted material is a serious issue for the content owners who want to protect their works. With the appearing of several types of underground economies aiming at the illegal copyrighted material distribution, such as the peer-to-peer (P2P) file sharing networks, the necessity for assuring the copyright protection aroused. Considering this, several solutions appeared in order to stop the growth of the illegal distribution. Examples of protection applied in these types of work are the DRM[1] technologies that limit the utilization of the copyright content. Other category of solutions are given by the discouragement of the illegal distribution, and *transaction watermarking* is a technology applied with this purpose.

1.1 Motivation

Digital watermarking consists in integrating a watermark message that is imperceptible to the human ears or eyes into a digital work. In image files the watermarking consists in small changes into the image pixels, watermarks can be also embedded in audio content by modifying samples.

Transaction watermarking algorithms consist in marking each content with a unique watermark message for each work prior its distribution. The content owner who wishes to protect his work against illegal distribution embeds a kind of serial number into every single copy of the work

[1] The DRM shortening for **D**igital **R**ights **M**anagement is an access control class of technologies with the objective of limiting the usage of a copyrighted work/device after sale. These limitation can be of many types, *e.g.* copy protection.

identifying each customer who buys a copy of his work. If any illegally copy distributed on the internet is found, the content owner by extracting the watermark message can find the responsible user for the illegal distribution. The transaction watermarking schemes are also known as *tracing traitor* schemes and were first introduced by *Chor et al.* in [1].

The most straightforward case to achieve the unique watermark message would be assigning an ID to each customer who buys a copy of the copyrighted work. Thereafter, a message containing 30 bits would have 2^{30} possible combinations. The length of a code with this size would be enough to assign a copy to every single inhabitant in the European continent. For most of transaction watermarking applications the problem does not lie in satisfying the size of the message considering the number of users, if one just want to make sure that a different key is assigned to each user. As seen before with only 30 bits a large number of users can be identified. Even for several types of multimedia content (*e.g.* images, audio files) where the number of bits that is possible to embed is limited if compared to other types of content (*e.g.* e-books, movies), the transaction watermarking can be applied assuring the identification of enough users because the codes are not too long.

Nevertheless, transaction watermarking schemes are vulnerable against some specific types of attacks. For example, when two users compare their copies they are able to detect positions where their copies disagree. The users are able to modify these positions and output a tampered copy protecting themselves to be caught. This type of attack is called a *collusion attack*. The tampered copy can contain a watermark message that does not lead to any of the guilty users or even worse, it can lead to an innocent user.

In order to prevent such kind of attacks, the *fingerprinting codes* are an accepted solution. These codes are mathematically generated codes designed to be secure against collusion attacks. By using the fingerprinting codes the content owner is capable to trace back the responsible users with the minimum probability of accusing an innocent user. However, the utilization of the fingerprinting codes in many cases is limited due to the size. As already stated, images generally can be integrated with smaller watermark message due to its size when compared to other types of content. Recently, several approaches have been presented in order to reduce the size of the fingerprinting codes[2, 3]. Reducing the size of the collusion for which the fingerprinting code is secure against is one of the solutions. However, for mass market applications where the number of users is too large, the fingerprinting codes should be secure against larger collusions turning these codes impractical to most of the multimedia applications.

1.2 Goal

The main scope of this thesis is designing a new watermarking approach capable to be applied in images for a mass market scenario. As already stated an image has a very limited amount of data that can be embedded in it. The fingerprinting codes depend on the maximum collusion size that they have been designed to be resistant against, and if the maximum collusion size is large enough they achieve lengths that cannot be embedded even in more than one image. For example,

a fingerprinting code with a length of 4000 bits (500$bytes$) cannot be embedded efficiently into one image of 500$Kbytes$ because the payload[2] is too large for one single image. However, if a *set of images* instead of one single image is considered, enough payload can be achieved to embed the fingerprinting message.

While for one image scenarios no collusion resistant solution is available, sets of images are able to be integrated with a fingerprinting codes, enabling the application of the fingerprinting schemes for many types of commerce. A good example would be e-books for children. Books for children contains several images and the fingerprinting integration can be performed into all these images before the e-book distribution, analogously the same scenario exists for digital comics. Notwithstanding, the most prominent solution is the application for video-games. Video-games rank among one of the most illegally downloaded multimedia contents. In 2011 the rates of pirated copies download were 300% higher for the top 5 downloaded games compared to the top 5 pirated copies in 2008[3].

Whereas, in the literature shorter fingerprinting codes capable to be integrated in one image exists (a 2-secure fingerprinting approach is introduced in [2]), a practical solution for fingerprinting codes secure against larger collusions is yet to be found. While this thesis aims the possibility of integrating larger fingerprinting codes into sets of images other constraints should be regarded as well. For example, if these fingerprinting codes are applied into video games, the textures should be marked with the fingerprint fast enough to be delivered to the customer. In order to reduce the time for embedding the proposed scheme should be suitable to apply the *container watermarking* strategy[4].

[2] Payload stands for the amount of data that can be embedded into one content.
[3] In formation available at: http://drm.web.unc.edu/games/.
[4] This approach is introduced in [4].

Chapter 2

State Of The Art

In this chapter the concepts of watermarking for digital images will be discussed. In the first section a discussion about the general digital watermarking applications and its concepts is held followed by a further insight on its properties. Afterwards, the collusion secure fingerprinting concept will be introduced as an extension of transaction watermarking. Lastly, the robust hashing application in image identification is introduced.

2.1 Digital Watermarking

This section is devoted to the digital watermarking algorithms in general and their properties.

2.1.1 The watermarking model

In an elaborate way, digital watermarking consists in embedding imperceptible information (without the creation of *meta-data*[1]) into a specific multimedia content such as video, audio or image. The digital watermarking schemes can be separated into three different phases: *embedding*, *transmission* and *detection*.

Embedding phase

During the embedding phase the *watermark message* is incorporated on the original cover work. The information needed for the embedder is the original cover work, the watermark message and the secret key.

Transmission phase

The transmission phase can be understood as the period when the watermarked work was already distributed and is in use. The watermarking attacks happen during this stage.

Detection phase

The detection phase consists in extracting the embedded message from the watermarked

[1] Metadata can be referred as being the "data about data", *i.e.*, metadata is the additional information about a work. For example, in photographies the metadata can be the color depth, the dimensions of the image, the device used for the creation, the author etc.

work. In order to extract correctly the watermark message the secret key is necessary as mentioned before. The purpose for the secret key utilization is to increase the security of the watermarking scheme by allowing only the owner of the content that holds the secret key to extract correctly the information embedded.

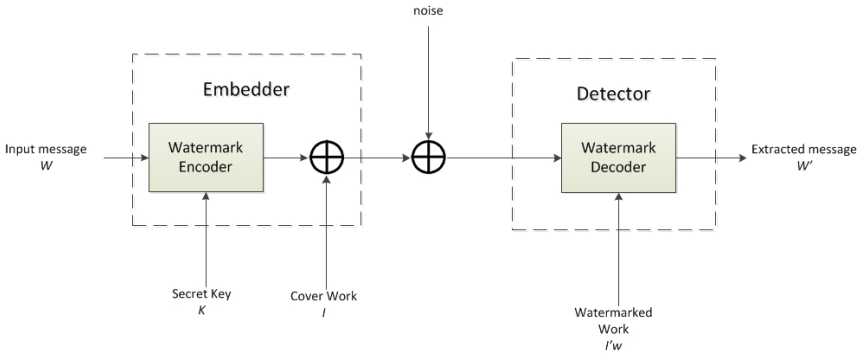


Figure 2.1: A *blind* watermarking scheme

Watermarking algorithms may be of two different natures: *blind* or *non-blind* [2].

Blind algorithms

The blind watermarking algorithm depicted in Figure 2.1 receives as input parameters the original cover work I, the watermark message W and the secret key K. The watermarked work is given by I'_W as the result of the input parameters $I \times W \times K \rightarrow I'_W$. For the watermark message extraction the detector should be fed with the watermarked work I'_w and the secret key K. The retrieved watermark message is given by

$$I'_W \times K \rightarrow W'$$

Where W' is the extracted message.

Non-blind algorithms

Non blind watermarking algorithms illustrated in Figure 2.2 are similar to the blind algorithms with the difference that the original cover work I is needed for detection. The embedding procedure is similar to the blind scheme receiving as input the cover work I, the watermark message W, and the key K. The difference of these two algorithms comes during the detection phase. In contrast to the blind algorithms, non-blind detectors need to be fed with the original cover work I in addition to the watermarked work I'_w and the secret key K. The extracted message is expressed by

$$I \times K \times I'_W \rightarrow W'$$

[2] *Langelaar et. al* in [5] address to blind and non-blind algorithms as *oblivious* and *non-oblivious* respectively.

For most of the watermarking applications blind scheme is used. Once the original cover work I is not provided during the detection, it makes the algorithm more secure. Moreover, in some applications the cover work might not be always available, turning this algorithm more practical. The secret key K is used with the purpose of increasing the watermarking algorithm security by ensuring that only the legitimate owner of the content is able to extract the watermark message. Therefore, watermark message should not be correctly extracted if the wrong secret key is provided.

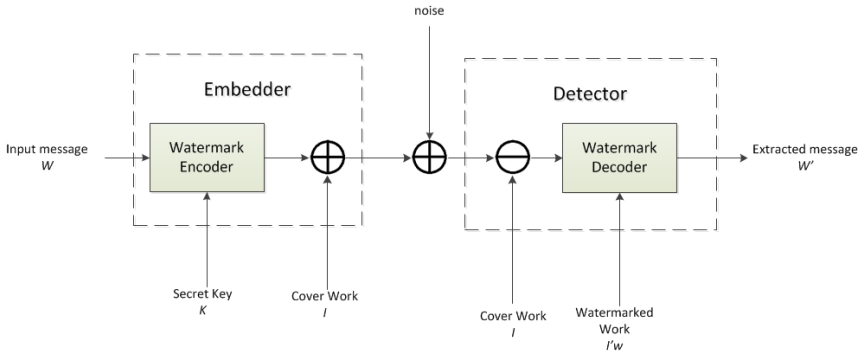


Figure 2.2: A *non-blind* watermarking scheme

2.1.2 Watermark Properties

The watermarking algorithms should be compliant with several requirements. There is no set of requirements to be met by all the watermarking schemes, however some requirements are common to most of the watermarking applications [6, 5].

2.1.2.1 Transparency

The transparency or perceptibility is a measurement for the visibility of the message on the watermarked work. If a human cannot distinguish between a watermarked work and the original, then the watermarking technique employed meets this requirement. The Peak Signal-to-Noise-Ratio (PSNR) is a commonly used distortion measurer and is given by Equation 2.1. This equation measures the dissemblance between the pixels the same positions of two images (the original and the attacked). For that reason the images should be of the same size, so the comparison can be computed for all the positions of the images . When the PSNR value tend to the infinity, it means that both images are highly alike. Therefore, the value of PSNR should be as high as possible. Usually for a watermarked content with a good transparency the PSNR exceeds the 40dB.

$$\text{PSNR} = 10\log[MN\frac{\max_I^2}{\sum_{m,n}(I(m,n)-\tilde{I}(m,n))^2}] \tag{2.1}$$

Although, because the PSNR is a difference distortion measurer some exceptions can be found, *e.g.* in the case of the *Stirmark* [3] random bending or pre-warping attacks for images the PSNR usually is lower than 20dB but it does not mean that the image quality is affected. Even after attacks of this type the image still presenting good visual quality. For these types of attack sometimes an evaluation based on correlation provides more reliable results. An example of correlation distortion measurer is given by the histogram similarity and is represented by Equation 2.2.

$$HS = \sum_{c=0}^{255} | h_I(c) - h_{\tilde{I}}(c) | \tag{2.2}$$

This equation corresponds to the dissimilarity of gray scale histograms. Where c corresponds to the bin counter, h_I is the histogram of the original work and $h_{\tilde{I}}$ is the histogram of the distorted work.

2.1.2.2 Robustness

A watermarking algorithm is called robust if after an attack the watermark message is still detectable. The attacks may be of several different types or even non-intentional data degradation such as compressions MPEG-4 for video, JPEG for images or MP3 for audio files. Although, robustness is not a requirement to be fulfilled by all the watermarking algorithms. Some watermarking applications requires a fragile watermark. Applications for authenticity proof for example requires fragile watermarks. In these applications, when the content is tampered whether by intentional or non intentional manipulations the watermark message is not retrievable, thus removing the legitimacy of the work.

2.1.2.3 Payload

The payload corresponds to the amount of input data message embedded into a watermarked work. Accordingly to the watermarking application, different payloads are necessary. In this case the difference between one bit of watermark message and one bit of real message can be noticed. In order to embed one bit of one real message it might be necessary to embed the bit more than once into the cover work. The number of repetitions *i.e.* the redundancy applied increases the length of the watermark message. The more redundancies the message has, the more reliable will be the detector response will be.

2.1.2.4 Security

The security of watermarking techniques can be equivalently understood as the security of encryption techniques. Accordingly with the Kerckhoff's assumption [7] one should assume that the method used to encrypt the data is known to an unauthorized party, *i.e.* the watermark embedder and detector, and the security must be located in the choice of the key. The watermarking is truly

[3] The Stirmark is a software introduced by *Petitcolas et al.* that consists in a benchmark for watermarking robustness testing. The tool employs several types of attacks, ranging from JPEG compression to geometrical distortions.

secure if even knowing the algorithm for embedding and extracting the watermark in the work it does not help any party to detect or remove it. This description is just a simple introduction to watermarking security requirements. Some confusion may be created between security and robustness concepts. Watermarking robustness is the capacity to retain the watermark message after post-processing manipulations, and afterwards extract the watermark message correctly. The watermarking security is undetectability of the watermark message even if the attacker knows the algorithm construction.

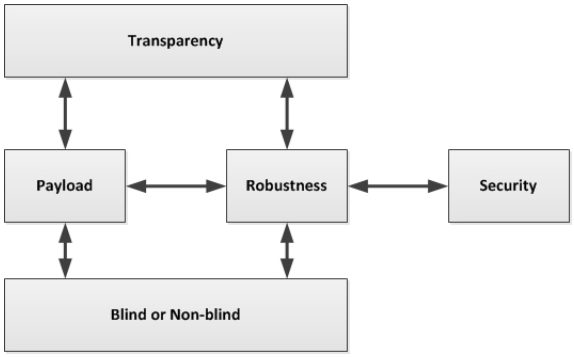


Figure 2.3: Interdependencies among the different watermarking requirements

Additionally, it is possible to notice an interdependence among these requirements. For example, more robust embedding methods generally cause direct influence in the transparency of the watermark message.

2.1.3 Watermark Applications

Watermarking techniques may be used for many different purposes [6, 8]. As the main scope of this document will be transaction watermarking schemes other applications will be briefly introduced.

2.1.3.1 Broadcast monitoring

Broadcasting monitoring appeared with the objective of the advisertisers control if the advertisements that they have paid to be transmitted are really aired. This application appeared with an occurrence in late 90^{th} decade, when Japanese advertisers paid for commercials that have never been transmitted by the television operators. Broadcasting monitoring also can be used in order to protect the broadcasting of copyrighted material by pirate TV stations. Hence, the copyright protection may be used to protect any copyrighted material, and not only television advertisements.

2.1.3.2 Ownership proofing

A watermark message can be embedded into a multimedia content in order to mark it as a copyrighted material. The owner of the content can embed hidden data into the host content so that if an illegitimate user attempts to claim the rights of the same content, the ownership could be proved

by extracting the watermark message. The watermark extraction is possible only by the legitimate owner who possesses the secret key.

However, the ownership proofing application may arise other problems. If the false owner somehow removes the embedded message and with an embedder integrates his watermark, the content owner is no longer able to prove the ownership. In this type of applications it is crucial the utilization of a secret key in order to avoid the detection and removal of the watermark message by unauthorized parties.

2.1.3.3 Content Authentication

Watermarking for authentication purposes are used for proving a given content to be legitimate. If a watermarked content undergoes through any kind of intentional or unintentional manipulation and afterwards the data is extracted, the detector should be able to notice that the original data has been tampered. Usually these types of watermark messages are fragile, *i.e.* they are very susceptible to manipulations whether intentional or not.

2.1.3.4 Traitor tracing

Traitor tracing schemes [1] or transaction watermarking is one of the most prominent applications for digital watermarking. It consists in a distribution of copyrighted material scenario where all the copies are watermarked. The main difference between this application and watermarking for ownership proofing purposes is that all the watermarked copies receive a different watermark message. The objective of assigning to each copy a single watermark message is identifying the users and the possibility of tracing back the guilty user in case of illegal distribution. This type of watermarking technique is applied for example in online markets *e.g.* music stores or e-books. The process of watermarking is done when a costumer orders a copy of a given copyrighted material, the content is watermarked with an individual message as the last part of the transaction process prior to the distribution.

2.1.4 Image Watermarking

In this subsection the different approaches of image watermarking will be presented and the most common attacks to image watermarking schemes.

2.1.4.1 Image Watermarking Schemes

Image watermarking techniques may fall into two different domains: *space domain* and *frequency domain*. Each domain presents specific characteristics in how the watermark message is embedded and the resistance to most common types of attack may vary as well. On the scope of image watermarking, the spatial-domain techniques are more straightforward and simple methods to integrate the watermark message into an image. Whereas the transform domain algorithms are more complex to design and implement, the use of the transform domain algorithms allows to understand

the underlying information of the image, thus making it easier explore more efficiently the Human Visual System (HVS) characteristics. The usage of these types of algorithms allows to achieve better transparency while the robustness is increased. Therefore, the advantages brought by these algorithms, are far more advantageous and for that reason nowadays most of the watermarking algorithms are based on these techniques. The most common transforms used are the DFT, DCT and DWT. Some algorithms propose utilization of several transforms to improve the performance against a larger variety of attacks in watermarking schemes. These algorithms apply sequentially the transforms to a given content. Examples of these algorithms are presented in [9, 10].

Space-Domain Watermarking

All the methods of space domain watermarking consists in directly modifying the pixel values in order to embed the watermark message. Some methods use the spatial-domain histogram to embed the watermark message [11, 12], while others embed the watermark message by addition of a pseudo-random signal [13, 14, 15, 5]. The advantage of the methods based on spatial domain embedding lies on its simplicity as stated before, although the robustness of these methods is much weaker to the most simple post-processing operations such as compression or filtering, as well as geometrical attacks when compared to transform domain watermarking algorithms. Another drawback in the utilization of spatial domain techniques is the visual degradation. Usually spatial domain watermarking presents less transparency when compared to transform domain techniques because the embedding method does not use the HVS characteristics as efficiently as transform domain algorithms.

DCT-Watermarking

The DCT watermarking exploits the image frequency bands where the watermark message is embedded. The calculated DCT can be separated by different sub-bands and the watermark is embedded into the middle frequency sub-band. Embedding on the higher frequencies would turn the watermark message susceptible to attacks since this frequency is highly attenuated after simple noise addiction and image compression, on the other hand, embedding on the lower frequencies would turn the watermark more visible according to the HVS model. For that reason the embedding is performed on the middle frequency sub-band. The JPEG compression is also performed with the utilization of the DCT and a quantization table. The reason why JPEG format size is so reduced is due to the compression based on the perceptual model of images. Therefore the DCT watermarking is more robust against compression attacks because it anticipates the compression by embedding the watermark into the middle frequency DCT coefficients. The embedding procedure is made by splitting the image into 8×8 blocks and embedding the watermark into each block by changing the DCT coefficients of each block. [5, 16, 17]. After embedding the IDCT is computed to generate the watermarked image.

DFT-Watermarking

Another commonly employed transform is the DFT. In some approaches the watermark

message may be embedded into the phase of the DFT because of its importance when compared to the amplitude in terms of the intelligibility [18, 9] and its superior immunity to noise addition [5]. It means that in order to remove the watermark message from an image that has been marked in its DFT phase the image is severely degraded prior the watermark removal. In case the watermarking scheme is based on the DFT amplitude, the scheme usually presents a larger resistance to image shifting [19].

DWT-Watermarking

In order to use the characteristics of the HVS efficiently, the DWT-domain watermarking is a widely used transform, commonly used as computationally efficient version of frequency models for HVS [20]. The DWT can be computed resorting to two different methods.

The first method comes with a direct implementation for the wavelets used in signal processing which is the 1D-DWT. However, it is important to note that images are 2-dimension signals therefore they need to be remaped to an 1-dimension vector [22]. The remapping of the image is performed by zigzag scan in order to keep the neighbouring pixels close to each other.

The other method is using an extension of the single dimensional wavelet transform applied into a 2 dimensions domain, this method called 2D-DWT [24]. The simplest implementation of the 2D-DWT is achieved by using the Haar[4] wavelets. For example, an image I is split into blocks without overlapping X of size 2×2. The wavelet transform of the image is achieved by computing the transform seen on Equation 2.3 and for each input block X an Y block is outputted. The H matrix is given in 2.4, the resulting image I' is computed by simple geometrical transformation of the blocks X multiplied by the matrix H scaled by $\frac{1}{\sqrt{2}}$ in order to preserve the energy.

$$Y = HXH^T \tag{2.3}$$

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{2.4}$$

The Y block has four different coefficients. Each coefficient corresponds to one different wavelet frequency band as follows.

upper left: corresponds to the average of the four input pixels of X; The 2D low-pass filter result (Lo-Lo).

upper right: corresponds to the average of the horizontal gradient of X; The horizontal high-pass and vertical low-pass result (Hi-Lo).

lower left: corresponds to the average of the vertical gradient of X; The horizontal low-pass and vertical high-pass result (Lo-Hi).

[4] The Haar wavelets were introduced by *Alfréd Haar* and are also known as a special case of the Daubechies wavelet often mentioned as **Db2** wavelet

lower right: corresponds to the diagonal gradients X; The 2D high-pass filter result (Hi-Hi).

The figure Figure 2.4 shows the four different parts of the *Lena* 2D-DWT transform. The (Lo-Lo) part can be decomposed into a new series of wavelets generating k different levels as shown in Figure 2.5.

Figure 2.4: Lena figure decomposed in a single level DWT

The embedding of the watermark message is performed by embedding into the k^{th} level low frequency sub-band of the wavelet transform. Due to its characteristics of energy preservation, several approaches have been presented in recent years using the DWT method for embedding. Some of those approaches are based on the low-frequency coefficients modification by adding a pseudo random signal as the analogous form used for spatial domain embedding [5, 21]. One of the biggest advantages of using the wavelets is the topology of the low-frequency sub-band when compared to its original image. As result other watermarking approaches using the DWT embed the watermark message into the low-frequency sub-band histogram of the image [22, 23, 24].

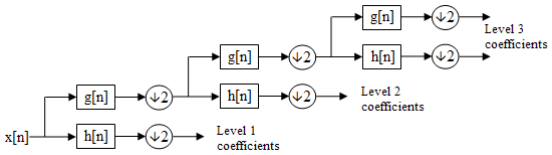


Figure 2.5: Decomposition of a signal into 3 different levels

2.1.4.2 Attacks on Image Watermarking Schemes

An image watermarking attack is understood as any kind of action that makes the watermarked image undergo through the unauthorized removal, hiding or detection of the watermarked message. Before introducing the different types of image watermarking attacks, an introduction of the different natures of attacks and attackers is made.

Unauthorized removal

Unauthorized removal prevents the owner of the content to retrieve the previously embedded

watermark by removing or destroying the watermark. There is two types of unauthorized removal attacks; **masking attacks** and **elimination attacks**. Masked watermarked images still being able to be identified by detectors although the watermark can not be retrieved. Some detectors can implement pre-emptive techniques for identifying masked works like rotation, cropping or scaling. An *elimination attack* consists in most of the cases post processing operations with the objective of degrading the image in order to remove the watermark.

Unauthorized embedding

Unauthorized embedding, is the act of embedding a second watermark in a previously watermarked work. By embedding a second watermark it does not mean that the first will not be detectable any longer. Depending on the embedding techniques both watermarks can still being detectable after an unauthorized embedding.

Unauthorized detection

The unauthorized detection is when one identifies the watermark in the work, however the watermark can be deciphered or not. The unauthorized detection is more relevant in *steganography* applications because the objective of this application is to mask the message. However, in some watermarking applications this attack could constitute a grave problem as well.

Both unauthorized removal and embedding attacks can be described as *active attacks* because the attack will modify the previous work. The situation of unauthorized detection is called an *passive attack* because it does not change the work.

The **attacker nature** is also a relevant aspect to be considered on attacks strategies. For example, an attacker may have different levels of knowledge. If an attacker has no knowledge about the algorithm construction, the attack strategy will be much more limited compared to an attacker that has knowledge about the watermarking algorithm, or an attacker that has in his possession a copy of the watermark detector. Attending to this fact, it is necessary to implement different watermarking strategies based on the knowledge of the attacker.

The attacker does not know anything

This is the most common case. When a attacker has only his own copy and does not know anything about the watermarking algorithm. In this case the attacker has only the opportunity of attacking the image in order to remove the watermark message.

The attacker has more than one watermarked copy

In this case, depending the watermarking application, the possibility of the attacker holds more than one copy is dangerous because by the comparison of the copies he can identify positions on the work where the copies disagree. This attack is commonly known as *collusion attack*.

The attacker knows the algorithm construction

This is a common attacker nature if the Kerckhoff assumption is considered. When the

attacker has the knowledge about the algorithm construction he can exploit the algorithm, find possible weaknesses, and attempt to tamper the cover work.

The attacker has a detector

This type of attacker is often referred as an *oracle* [25] or *black box*. It consists not in exploiting the algorithm itself but rather it consists in using the detector in order to remove the watermark message.

Bellow, some common attacks for images are listed. Attacks such as JPEG compression or low pass filtering will be omitted due to its direct understanding on the image processing context.

Copy attack

This attack was introduced by *Kutter et al.* [26], and as the name indicates, it consists in copying the embedded watermark to a second unwatermarked image. By analysing the watermarked image I_w, the attacker can estimate the unwatermarked image $\hat{I}$, and by performing the difference between the estimated unwatermarked image $\hat{I}$ and the watermarked, the attacker can find an estimate for the watermarking embedded. The next step consists in adding the estimated watermark previously obtained into the unmarked work I_u as seen in equation 2.6.

$$I_w - \hat{I} = \hat{w_{or}} \tag{2.5}$$

$$I_u + \hat{w_{or}} = I'_w \tag{2.6}$$

Splitting attack

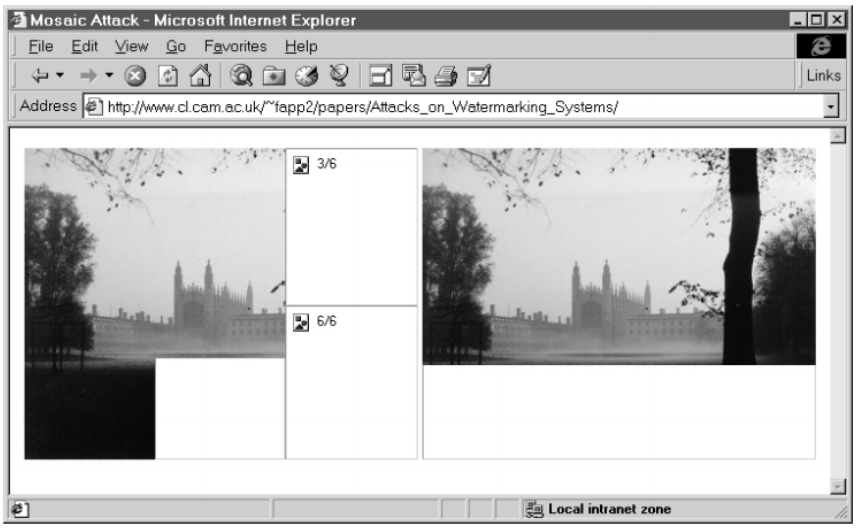


Figure 2.6: Splitting attack [27]. This attack is performed by splitting the image into several sub-images blocks and rendering them in a given sequence afterwards.

The splitting attack (or *mosaic attack*) introduced by *Petitcolas et al.* [27], does not remove the watermark message from the image since the image itself remains unchanged. However, the attack consists in splitting the original image into several sub-images (or mosaics) and rendering them in a web-browser in a given sequence. Web-browsers usually tend to align the images automatically, presenting the image with the same appearance as the original one. This attack is illustrated in Figure 2.6.

Desynchronization Attack

Amongst the most effective and target of many studies are the desynchronization attacks. These attacks are performed by geometrical manipulations such as re-sampling, scaling, cropping or rotating the image. A single geometrical distortion can render the watermark detector useless due to the difficulty of cope with these attacks. Whereas simple geometrical distortions such as cropping or rotating can affect the aestheticism of the attacked image, attacks based in a combination of several manipulations produce results with less visual impact. The pre-warping using the Markov random field (MRF) is an attack that lies in this category and consists in smooths distortions in different parts of the images. *Fenzi et al.* [28] explored the possibility of using the pre-warping of images for collusion-secure watermarking schemes, however the same algorithm can be used as an efficient attack strategy. Another example of desynchronization attack was introduced by *Petitcolas* and it is called *random bending attack* (RBA) or *Stirmark attack*. Differently from the pre-warping attack using the MRF, the RBA is based on global geometrical distortions on the image as depicted in Figure 2.7, however in both attacks the final result is still acceptable in case the original image is not available for comparison matters.

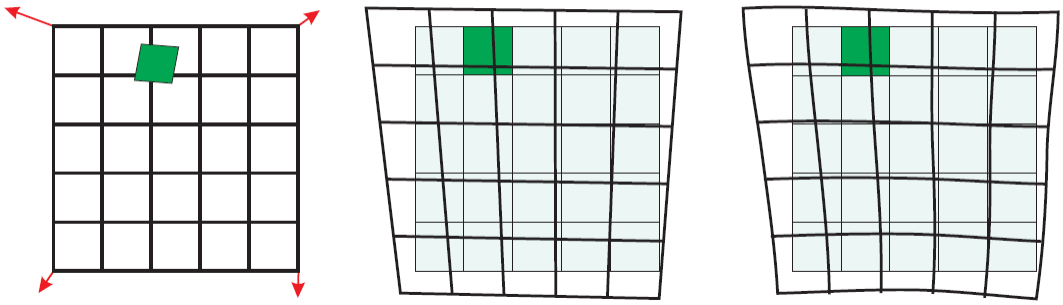


Figure 2.7: Random bending attack effect into one image

Ambiguity Attack

This attack can be understood as a copyright fight for the original cover work. The legitimate content owner integrates his watermark W into the original image I generating the watermarked work I_W. When a malicious user tries to tamper the watermarked work I_W by

removing the watermark W and embedding his own watermark message W_f a problem of copyright addressing appears. Attacks of this type are known as ambiguity attacks. The method used to avoid such kind of attacks is made by using *non-invertible* watermarking schemes. Non invertible watermarking schemes were introduced by *Craver and Memon* in [29], in their work the ambiguity attacks are referred as **SWICO** (Single-Watermarked-Image-Counterfeit-Original) attacks.

Gradient Descent Attack

In the gradient descent attack, the attacker has the help of a detector which is called *oracle* or *black box* [30] and knows how to read the responses outputted from the detector. The attacker uses the detector outputs to estimate the region where the watermark is not detectable any more. The basic idea is that the most abrupt output change corresponds to the shortest way out of the detection zone of the detection area. The example is depicted in Figure 2.8.

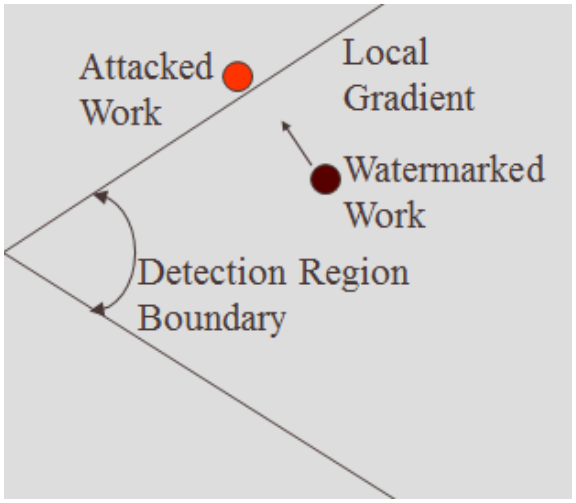


Figure 2.8: Gradient descent attack detection areas outputted by the detector

Sensitivity Attack

Attack strategies when the detector is available were introduced by *Cox et al.* in [31] and the sensitivity attack as an extension of those types of attack was introduced by *Linnartz and Dijk* in [32]. The attack strategy is based on encountering the shortest way out of the *detection region*. The detection region is the region where the watermark message still being detectable. Two assumptions are made about the detection region

1) the direction of a short path can be approximated by a normal vector ω_r to the surface of the detection region.

2) The normal vector ω_r is relatively constant over the detection region.

Considering these two previous assumptions the sensitivity attack is performed in two different steps: first, the watermarked image I_w is attacked in order to create the image I'_w. The

attacked image is generated by using different types of attacks such as averaging of the pixels, random modification of the pixels, etc. The normal vector ω_r is computed by using the images I_w and I'_w. Afterwards in order to remove the watermark message from the images the normal vector ω_r just has to be scaled and subtracted from the watermarked images in order to remove the watermark message. The attack is depicted in Figure 2.9.

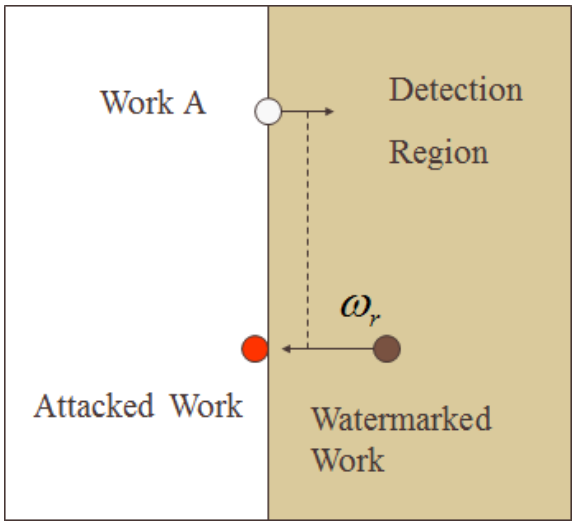


Figure 2.9: Sensitivity attack

Summary of Attack Types

The different types of watermarking attack can be split into different categories according to *Voloshynovskiy et Al.* [33] : *removal attacks*, *geometric attacks*, *cryptographic attacks* and *Protocol attacks*.

Most common image post-processing operations such as JPEG compression, filtering or noise addiction lies in the removal attacks category. The main objective of the attack is to remove the watermark message without concern about the watermarking algorithm. The result of these attacks are mainly damaging the image, however in some cases, the content may, even after the attack, be suitable for use. For example, in JPEG compression when an image is compressed with a high quality factor (*e.g.* Q95), no evident damage to most of the cases are noticed, but watermark messages from schemes using spatial domain algorithms may be removed.

The geometric attacks category. The main objective of such kind of attack is not the removal of the watermark message, but rather the masking of it by changing the location of the pixels in the image. Examples of attacks belonging to this category are rotation, scaling, cropping attacks, and the desynchronization attacks (Stirmark RBA or pre-warping MRF) as well.

Cryptographic attacks are attacks performed when the attacker knows the watermarking algorithm. With the knowledge about the embedding technique the attacker may try to

remove the watermark by exploring the scheme weaknesses. An example of cryptographic attack is the gradient descent attack.

Protocol attacks are different from the others attacks in a way that the main objective is not to destroy or remove the watermark message but rather to use the watermark message embedded into an image to mislead the content owner, *e.g.* copy attack.

2.2 Fingerprinting Codes Against Collusion Attacks

The following section will be devoted to the study of the collusion attacks and the importance of the fingerprinting codes applied as transaction watermarks.

2.2.1 Collusion Attacks

As seen before, transaction watermarking schemes consist in marking individually each copy prior to its distribution as the last part of the production process of a copyrighted work. The objective in marking each copy is to identify each customer that purchases a copy of a copyrighted work. If the work is distributed illegally on the internet the content owner is capable to trace back the distributor of the illegal copy. Although, in some cases the attacker might possess more than one copy, or a group of users might create a coalition to frame the watermarking scheme. The attacks based on the comparison of copies and finding the positions where the copies differ are called *collusion attacks*.

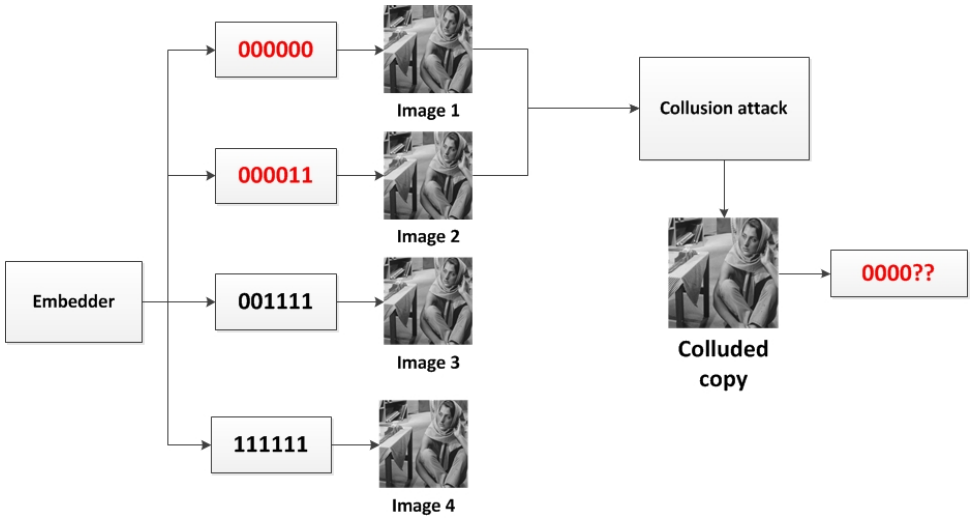


Figure 2.10: A collusion attack. The colluded image has the message '0000??' embedded

The figure Figure 2.10 depicts a scenario where the four images have different messages embedded. The visual aspect for all the images is the same. When the users 1 and 2 collude, by comparing their copies they are able to reveal and modify only the positions where their copies disagree. This condition receives the name of *marking assumption* [34]. By modifying the positions where their copies disagree they can output an image with a different message of their own

by flipping the detectable bits or inserting the unreadable '?' symbols. The unreadable symbol is the symbol that is interpreted by the detector as not being a '0' neither a '1' when the detector is based in a soft-decision evaluation. In some cases those symbols may not exist; If the detector provides hard-decisions it only outputs the values '1' or '0'. For the case shown in Figure 2.10 only the users 1 or 2 can be accused because the colluded copy possesses a part of the message that is common only to both users 1 and 2 which is the first 4 bits '0000'.

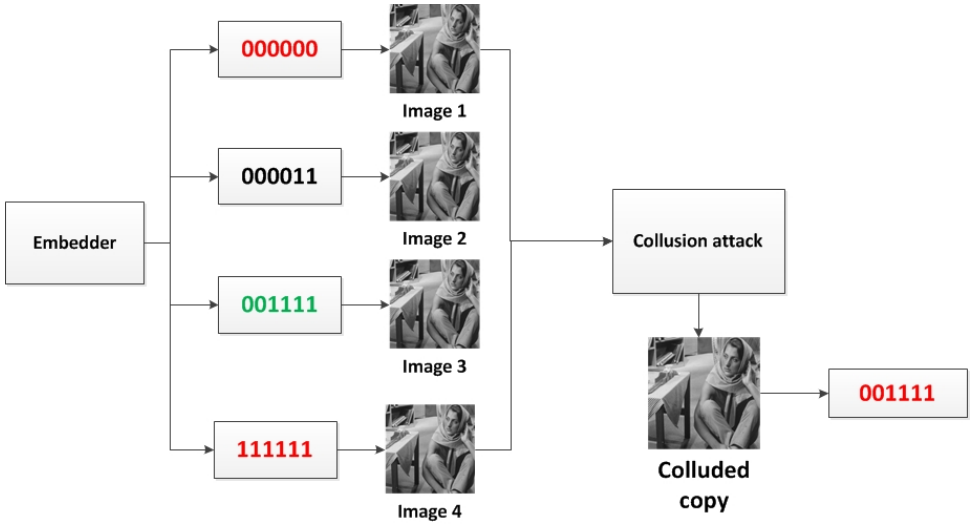


Figure 2.11: A collusion attack, outputting a copy accusing an innocent user.

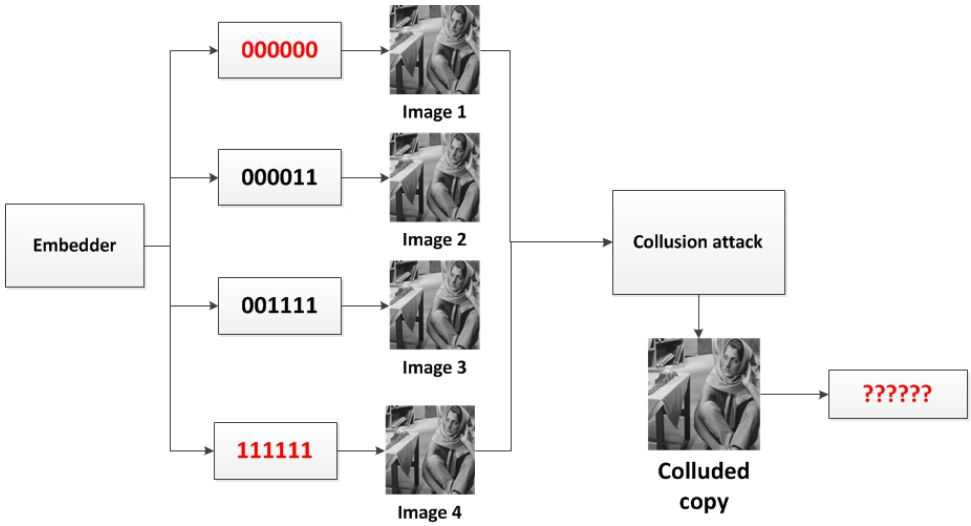


Figure 2.12: A collusion attack, no attacker is caught

In this same example, if the colluders 1 and 4 change all the positions where their copies disagree they are able to generate the sequence '001111' as seen in Figure 2.11. The messages of the users 1 and 4 are totally different and therefore they are able to identify all the watermarking positions. This scenario might lead the users to create a colluded copy framing an innocent user with a given probability. For the example on Figure 2.11 the user 3 is accused without being guilty.

This type of erroneous judgement is called *false positive error* FP and this error rate is designated by ε_1. Alongside with the chance of accusing an innocent user, there is also the hypothesis of not accusing any user. If the retrieved message does not permit any kind of judgement no guilty user can be caught. The image 2.12 illustrates the case when the users 1 and 4 collude outputting an image with all the message positions set to '?'. This type of error is called *false negative* FN and is represented by ε_2. Both the false positive and false negative rates should be very small, however an useful scheme must always have the rate ε_1 close to zero. In [35] the value for the FP rate is kept as $\varepsilon_1 \ll \varepsilon_2$. As the utilization of different watermark messages for each user is not enough to assure the creation of a scheme resistant against collusion attacks [34], the fingerprinting codes are used.

2.2.2 Collusion-Secure Fingerprinting for digital images

The importance of transaction watermarking schemes has already been presented and its applicability into the mass market products as in on-line stores are a tangible reality. Typically, transaction watermarks codes are randomly generated codes with a *Cyclic Error Control* CRC or *Error Correcting Codes* ECC block integrated on the messages. Though, as seen in the previous section, the utilization of transaction codes are not sufficient to assure that the scheme is secure against collusion attacks. Whereas transaction watermarking codes are based on randomly generated codes with an extra block for error control, fingerprinting codes are mathematically generated codes according to a probability distribution, without any error control block. In fact, the utilization of any error control code would attempt the recovery of a given watermark message, leading to one guilty user, whereas the fingerprinting codes are designed to catch more than one user in most of the cases.

Now in order to understand the fingerprinting schemes, some basic parameters must be understood. Those parameters are listed below.

Alphabet size

The alphabet size corresponds to the number of different characters for the fingerprinting characters. For example in a binary alphabet, which is the most common case the alphabet $\sum$ is defined by $\sum = \{0,1\}$. Therefore the alphabet size is two. In [35] Škoric reduced the alphabet size to a maximum of $q \leq 16$ characters due to the constraints of decoding complexity and perceptual quality parameters during the characters extraction.

Code Length

The length is the total number of characters from the alphabet $\sum$ that constitutes the fingerprint code and is usually denoted by the letter m.

Number of Users

The number of users is given by n and corresponds to the total number of possible fingerprinted copies. For mass market applications the number n is much larger if compared to *acceptance testing*[5] distribution for example.

Number of Colluders

The number of colluders usually is denoted by c. In order to represent the maximum number of colluders that the code is secure against the parameter c_o is adopted.

Error probability

A fingerprint code is said to be secure against a maximum collusion of size c_o with a tracing error probability of at most ε. Furthermore, regarding the tracing algorithm errors two different types of error rates are considered: the false positive rate FP, denoted by ε_1 is the rate for the number of accused innocent users. The other error rate considered is the false negative rate FN given by ε_2, and represents the rate of the tracing algorithm not catching any guilty user belonging to the collusion of c pirates as described previously.

Rate

The rate R is given by Equation 2.7. In the case of $c_0 = 1$, *i.e.* no collusion exists, the rate can be simplified to $R = \log|\sum|$ therefore $R = 1$. The reciprocal of the rate gives the length of the codes when compared to the trivial case.

$$R = \log_2 \frac{n}{m} \tag{2.7}$$

2.2.3 Collusion-Secure Fingerprinting Schemes

The first known fingerprinting scheme was introduced by *Boneh and Shaw* in [34]. With the development of this topic new schemes appeared such as the *Tardos* codes. Nowadays, of the main objectives for the fingerprinting codes is to reduce their length in order to embed them efficiently into most of the multimedia contents. For small collusions a fingerprinting code can be efficiently embedded into most of the contents because the code length is not too large. However, the challenge appears when the number of colluders increases, and consequentially the codes, that grow to way too large lengths turning the utilization of the fingerprints impractical for many multimedia contents. Hereunder some of the most important fingerprinting schemes are listed.

The Boneh And Shaw Approach

As discussed before, the *marking assumption* was introduced in [34] as the main condition to be fulfilled. The attackers cannot detect and alter the positions where their copies agree.

[5] Acceptance testing is the tests performed to a product prior to its distribution. Generally it consists in the creation of few copies distributed for some testers who will evaluate the product in order to check if it complies with the given requirements.

In case of altering those positions, it should be by performing an attack in order to remove the mark. Although, these attacks would probably degrade the work quality. In this same approach the codes are distributed over a binary alphabet, with n users and an error probability ε (in this approach the rates ε_1 and ε_2 are not decoupled, *i.e.* the chance of accusing an innocent user and accusing no one is the same) that is small. With the number of users n and the error ε it is possible to determine the number d of repetitions, *i.e.*, the number of ones or zeros necessary to represent one watermark bit. Each user will receive a watermarked copy with a different number of zeros and ones. For example, the first customer will receive a watermarked copy marked with '1' in all the positions, the second customer will receive a copy with all values set to '1' but the first d block, The same procedure is made for all the n customers. The last customer will receive a copy with all values set to '0' as can be seen in the table Table 2.1. Subsequently it is immediate to note that the length m is equal to $m = (N-1) \times d$ or it can be represented by the Equation 2.8.

$$m = O(c_0^4 \log(N/\varepsilon) log(1/\varepsilon)) \tag{2.8}$$

With the help of a secret key, the fingerprinting code undergoes through a permutation process in order to increase the security. The permutation process allows to avoid the colluders to identify any particular d-block. However this scheme presents a few drawbacks when compared to other approaches. First, the code length is too large for larger collusion sizes and the security is reduced when compared to other fingerprinting schemes; Because in this approach the chance of accusing an innocent user is the same of accusing no one, the probability of accusing an innocent user is high when compared to other schemes as well [34, 3].

A:	111111111
B:	000111111
C:	000000111
D:	000000000

Table 2.1: Fingerprinting codes for a number of 4 users and $d = 3$

Tardos Approach

Gábor Tardos introduced in [36] an approach based on probabilistic generated fingerprint codes.

Tardos proposed the generation of a matrix with the dimensions $n \times m$, in which the number of rows represents the number of users and the number of columns represents the code

length. The objective of this approach is to reduce the code length and the result achieved is given by Equation 2.9, that is ε-secure against a coalition of c_0 users.

$$m = O(c_0^2 \log(n/\varepsilon)) \tag{2.9}$$

The fingerprinting code denoted by $F_{nc\varepsilon}$ has the size m and follows the distribution over the pairs $f = (X, \sigma)$, where X is the input matrix $n \times m$ and σ is the accusation algorithm. The construction of the function $f = (X, \sigma)$ is made by picking the independent numbers $p_i = \sin^2 r_i$ distributed alongside $[t, 1-t]$ with $t = 1/(300c_0)$ for all $1 \leq i \leq m$ positions of the fingerprinting code $F_{nc\varepsilon}$. The value $r_i \in [t', \pi/2 - t']$ with $0 < t' < \pi/4$ and $sin^2 t' = t$. Afterwards, the positions X_{ji} of the matrix are selected independently from the alphabet $\sum = \{0, 1\}$ (in this case the alphabet is binary) and $P[X_{ji} = 1] = p_i$. Now, the accusation functions U_{ji} is computed by using the p_i values and the matrix X accordingly to Equation 2.10.

$$U_{ji} = \begin{cases} \sqrt{\frac{1-p_i}{p_i}} & \text{if} \quad X_{ji} = 1 \\ -\sqrt{\frac{p_i}{1-p_i}} & \text{if} \quad X_{ji} = 0 \end{cases} \tag{2.10}$$

the values p_i are independent and the matrix X is filled independently columnwise accordingly to the probability distribution. The process is made for all the columns until the matrix X is generated.
The function σ will accuse j on the pirated copy $y \in \{0, 1\}^m$ if the condition on Equation 2.11 verifies with $Z = 20c\log(1/\varepsilon)$ as the threshold parameter for the accusation algorithm.

$$\sum_{i=1}^{m} y_i U_{ji} > Z \tag{2.11}$$

During the algorithm construction Tardos proposes a lower bound of Equation 2.12 as derivation of Equation 2.9.

$$m = 100c_0^2 \log(1/\varepsilon) \tag{2.12}$$

Škoric Approach (Improved Tardos Fingerprinting)

On [35], Škoric proposed an improved scheme with a smaller lower bound for the Tardos approach. The constant 100 on Equation 2.12 was reduced to a constant equal $4\pi^2$. Tardos approach only considers the bits marked with a '1' as being informative, however, the bits marked with a '0' can be considered as carrying as much information as the '1' marked bits.

Consequently the accusation sum from Equation 2.11 can be altered since all the bits count. The new accusation sum is given by Equation 2.13.

$$S_j = \sum_{i=1}^{m} y_i U(X_{ji}, p_i) \tag{2.13}$$

The U pair can be defined by Equation 2.14. Due to the symmetry of the occurrences for $X = 1$ and $X = 0$ the distribution function verifies the $0 \longleftrightarrow 1$ symmetry. Then, the symmetry is set by doing $g_0(p) = -g_1(1-p)$. Where g_0 and g_1 are measure functions for the suspicion arising for observing y_i for a given X_{ji} and p_i.

$$U(X_{ji}, p_i) = \begin{cases} g_1(p_i), & \text{if} \quad X_{ji} = 1 \\ g_0(p_i), & \text{if} \quad X_{ji} = 0 \end{cases} \tag{2.14}$$

Škoric also decoupled the two different types of errors; the FP and FN errors. In his approach he assumes the case that $\varepsilon_1 \ll \varepsilon_2$ as the condition that is much worse accusing an innocent user. The results achieved with the decoupling of the errors ε_1 and ε_2 show that the tardos codes can be reduced to a lower bound of $m = 4\pi^2 c_0^2 \ln\varepsilon_1^- 1$. An improvement for larger collusions sizes $c_0 \geq 10$ were achieved as well. The Škoric modified length is given by Equation 2.15.

$$m = \frac{4}{\mu^2} c_0^2 \ln\varepsilon_1^{-1} \tag{2.15}$$

Where $\mu = \frac{2}{\pi}$. The equation 2.12 is reduced from 100 to approximately π^2.

Even though with the utilization of optimized schemes such as the improved Tardos scheme, the length of the fingerprint codes still being too long to be applied in single images. Facing this fact, two different solutions might be studied. The first solution is the reduction of the fingerprinting code lengths so the codes can be embedded into a single image. For the Tardos scheme for example, in order to decrease the code lengths the collusion size should be small and the error rate ε_1 should be relaxed. These constraints turn the application of fingerprints limited. Although, even with the limitations, schemes with this characteristics can be very useful in some specific cases *e.g.*, testing versions or pre-release copies. In those cases the number of users is very limited and a scheme using a reduced number of users and a small collusion is perfectly suitable. Some approaches for 2-secure fingerprinting schemes were presented. *Schäfer et al.* [2] presented a fingerprinting scheme with a zero FP rate for limited number of users and a collusion of 2 colluders.

The Tardos fingerprinting codes are too long for large collusions to be embedded into one single image, thus limiting its applicability in this type of content. However, a fingerprint code can be embedded in a set of images instead of one image only. By increasing the number of

images the payload is increased, and consequentially longer codes can be embedded into a set of images. Considering this scenario, fingerprinting for image set may be an useful solution to many applications where a set of image is available, *e.g.* video-games textures or e-books for children. In those two cases the number of images allows the embedding of a fingerprinting code that is capable to be secure against larger collusions. Typically, Video-game textures possess more than a couple of thousands of textures 2.2 making them the ideal candidates for this type of applications.

Name of the game	Number of Textures
Dungeon Siege 3	4186
Fallout 3	12261
Total War Shogun	6360

Table 2.2: Number of textures in some video-games

2.3 Robust Hashing for Images

When the watermark message, in this case a fingerprinting code is split and embedded into several images, the correct identification of each image issue assumes a critical role for the watermarking scheme. If the images are not identified in their right sequence, the code cannot be retrieved correctly. Attending to this fact, a method for identifying the images on the set and identifying them on the right order should be considered. The immediate solution is achieved by using the images names as an ID for each image. However, using the file name offers no security because it can be changed with no cost for the attackers and renders the detector useless. Other options are: using a database with the images on the set and by matching the images with the database images or even by the use of identification algorithms. The problem with these approaches is the high computational cost and the difficulty of implementation.

A faster method would be using the *cryptographic hashes* to identify the images on the set. Cryptographic hashes are algorithms that receives as input a certain block of data and returns a fixed size message. This algorithm presents good security characteristics because even slight modifications on the code can alter completely the message string. However, when the cryptographic hashes application are addressed into the image processing field, slight change can be understood as an attack. For example, a simple JPEG compression would be enough to completely modify the hash, thus rendering useless the hashing algorithm.

Robust hashing algorithms share common features from cryptographic hashes and identification algorithms, in order to be robust against small distortions that the image can suffer but also secure and fast enough. In the past years the employment of image robust hashing algorithms increased significantly and several approaches have been proposed. *Xiang et Al.* proposed in [37] an image hashing algorithm based on the image histogram features. The scheme aimed to be robust against geometrical distortions by using the histogram in order to identify each image. This approach was revisited by *Xiang* in [23] and the DWT histogram was used in order to be secure

against a larger row of attacks. *Steinebach et Al.* proposed in [38] a fast robust to the most common image processing attacks hashing algorithm that achieve a low rate of false alarms.

Now, an example of a robust hashing algorithm will be illustrated. The algorithm described is the same used for the implementation of the proposed solution.

2.3.1 *Steinebach et Al* Approach

The following approach is the hashing algorithm presented in [38]. The algorithm consists in splitting the image into several blocks in order to create the hash.
The algorithm construction starts with the conversion of the images to gray scale values followed by a normalization to a fixed-size. The hash consists in a N bits string. The image will be split in N different blocks without overlapping. For each block the *mean value* $M_i, i = \{1,2,...,N\}$ is computed in order to generate the string $\{M_1, M_2..., M_N\}$. After the computation of all the M_i values the *median value* M_d of the string sequence is computed. Lastly the hash is normalized to a binary sequence given by

$$h(i) = \begin{cases} 0 & \text{if } M_i < M_d \\ 1 & \text{if } M_i \geq M_d \end{cases}$$

The figure Figure 2.13 illustrates the creation of a hash table. The example uses an $N = 256$. The current approach is proven robust against common post processing operations (*e.g.* JPEG compression, salt & pepper noise, low-pass filtering, etc.) and integrates pre-emptive measures against mirroring.

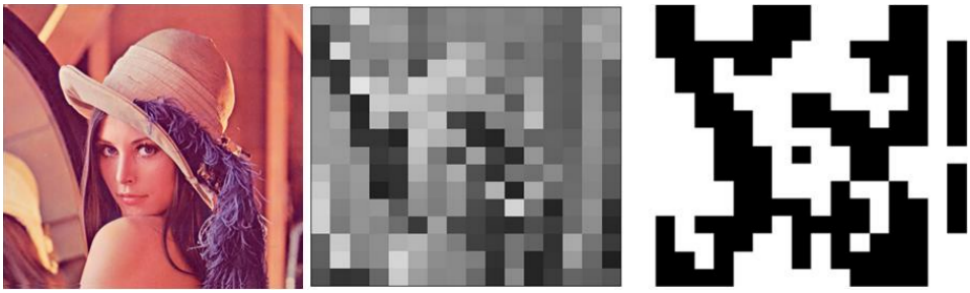


Figure 2.13: Steps of a hash creation. The image is split into several blocks, the mean value for each block is computed and by last, the image is normalized to a binary value using the median value.

Chapter 3

Proposed Watermarking algorithm for sets of images

In this chapter first the proposed watermarking algorithm scheme will be introduced. Hereafter, the integration of the fingerprinting codes and the utilization of the robust hashing algorithm for the proposed scheme are presented.

3.1 Proposed Robust Watermarking Algorithm

The presented watermarking algorithm is developed in order to be used with the container watermarking technique [4]. This strategy consists in a two phase watermarking procedure. First in the pre-processing stage, the cover work has its watermarking regions identified. The watermarking regions are specific parts of the cover work that are suitable for embedding the watermark message, *e.g.*, in an image it can be some parts of the image or even the whole content. In video watermarking these regions may be located in some specific frames. Afterwards, every embedding position is watermarked with the bits '0' and '1' and stored in a container. The second stage is based on the rendering of the watermarking regions into the final watermarked work. Upon request, the watermark message is generated. In this case, the watermark messages are collusion-secure fingerprint codes. The fingerprint code may have the value '0' or '1' and accordingly to the values, the final watermarked word is rendered with the files on the container. For example, an image has an embedding position $C(i),\ i = \{0,1\}$ where $C(0)$ represents a '0' embedded into the image and $C(1)$ represents a '1' embedded. If a '1' is requested, the embedder just has to take the container file $C(1)$. This embedding procedure strategy is faster when compared to the traditional strategies that perform all the embedding procedure upon the request of the message sequence, and turns the algorithm more suitable for real-time requirement applications, *e.g.* online stores.

The algorithm application uses several images to embed the fingerprinting code, *i.e.* the fingerprinting code will be split and embedded in among the images of the set. Let S be a set with s images defined by Equation 3.1. All the images on the set will be integrated with a part of the fingerprinting code to be embedded. Now let ω_S be the watermarking message to be embedded

in S with length m. Each image from the set S will be integrated with a part of the watermark ω_S accordingly to Equation 3.2, where ω_1 represents the part of the message that will be embedded into the image I_1 and so on. In a nutshell, the watermark will be split in several parts in order to be embedded in the images.

$$S = \{I_i \mid i = 1, 2, ..., s\} \tag{3.1}$$

$$\omega_S = \{\omega_i \mid i = 1, 2, ..., s\} \tag{3.2}$$

The watermarking embedding process will split each image of the set into square blocks with size N in which only one bit is embedded. The reason of embedding only one bit is to reduce the size of the container. For example, if only one bit is embedded into each block, it will be necessary to create two blocks in the container for each position that will hold the value '0' and '1'. Now if two bits are embedded into each block the possible sequences are '00', '01', '10' and '11', increasing thus the number of data for each block by 2^b where b is the number of bits embedded into each block. For more demanding applications of watermarking in images *e.g.* video-game textures, where the number of textures can exceed the thousands of images as seen in the table Table 2.2 the container size might be too large. Therefore, the container size will be at most the double of the size of the set, *i.e.* if the set has 100 Megabytes of data, then the size of the container will be at most 200 Megabytes.

For each squared block its DWT is computed and the low frequency sub-band histogram is extracted. The purpose of using the DWT transform is increasing the robustness against the most common signal processing operations *e.g.* JPEG compression, median filtering or high-pass filtering.

In order to embed and detect the watermark message correctly, it is necessary to identify each image in the set. Each image will hold an amount of data relative to different positions of the watermark message, therefore the need for identification of the images appear. The identification of the images will be made by using the robust hashing algorithm presented in the chapter 2.

3.1.1 Watermarking embedding

The following embedding algorithm presented takes into account the embedding in one image at a time, *i.e.*, an image I_x from the set $S = \{I_1, ..., I_x, ..., I_s\}$ has integrated in it a watermark message W belonging to the watermark code ω_S, where $\omega_S = \{\omega_1, ..., \omega_x, ..., \omega_s\}$.

From the original image is extracted its height and width in order to know how many blocks fit in it. The maximum number of bits to be embedded in an image is given by Equation 3.3. In case of the occurrence of *remaining regions* where a block cannot be fitted in, these parts are ignored for embedding and are only used for the reconstruction of the watermarked file. Remaining region is the name given to a part of an image where no block can be fitted, *i.e.* the dimensions of the square block $N \times N$ exceeds the available dimension on the remaining region. For example, if an image of size 90×150 pixels where N is set as 128 pixels block, the whole image is ignored.

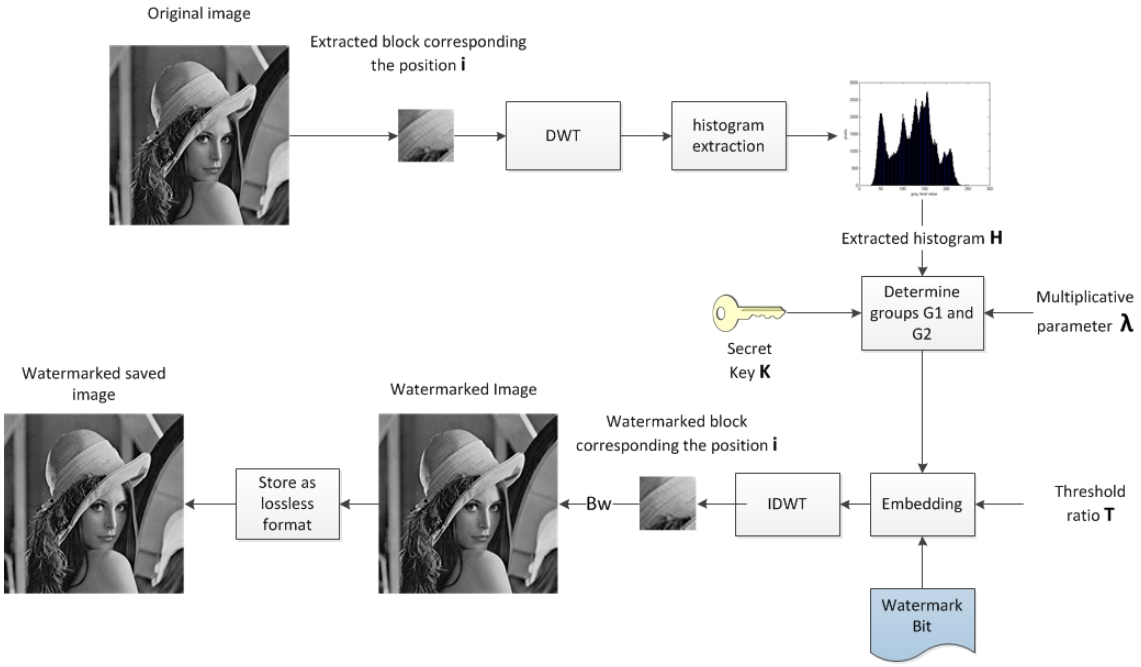


Figure 3.1: Block diagram for the watermarking embedding scheme

After the selection of the block its **first level** DWT is computed using the **Haar** Filter. Afterwards the histogram of the low frequency sub-band is extracted, since the watermark will be embedded only on this part of the image.

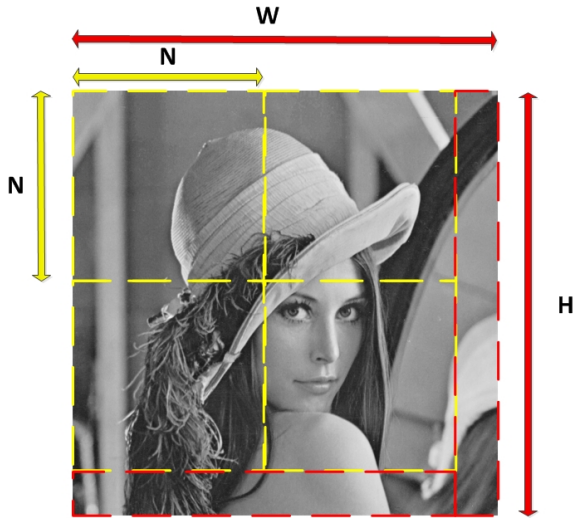


Figure 3.2: Selection of the blocks with size N in an image. The red region on the right side is not considered for embedding.

$$T_{blocks} = \left\lfloor \frac{W \times H}{N^2} \right\rfloor \tag{3.3}$$

After the DWT computation, the DWT low frequency sub-band block is given by $LL(u,v) =$

$\{ll(u,v) =| \ u = 1,2,3,...,R, v = 1,2,3,...,C\}$. Now, let H_{LL} be the histogram of the DWT low-frequency sub-band of a given block. The histogram is given by Equation 3.4. Where L is the number of bins in the histogram. The vector h_{LL} represents the count vector of the coefficients on each bin of the histogram.

$$H_{LL} = \{h_{LL}(i) \mid i = 1,2,3...L\} \tag{3.4}$$

The bin width for the histogram is given by

$$\text{Width} = \frac{\text{maxCoefficient} - \text{minCoefficient}}{L} \tag{3.5}$$

Where *MaxCoefficient* and *minCoefficient* are respectively the maximum and minimum values for the computed coefficients of the block $LL(u,v)$. Subsequently, the correspondent position on the histogram vector h_{LL} for each DWT coefficient is given by

$$i = \frac{\text{inputCoefficient} - \text{minCoefficient}}{\text{maxCoefficient} - \text{minCoefficient}}(L-1) \tag{3.6}$$

The embedding will consist in changing the value of the DWT coefficients in specific parts of the histogram in order to change the bins so the watermark is embedded. However, it is necessary to guarantee that after the IDWT computation the coefficients changed will be correctly computed also. In other words, if the number of bins L selected is too big, the changes of the coefficients for one bin to another represents adding a small constant that is the value given by Equation 3.5, whereas if a number of bins L selected is too small then the constant given in 3.5 is too big.

Now for example, if a histogram has $L = 500$ bins. If a coefficient belonging to the bin i is changed to the immediate neighbour bin $i+1$, when the IDWT is computed this change will be lost due to the rounding errors because the change is too slight. Therefore, in order to try to preserve the histogram shape, the value of L will be set to 256, that is the number of bins in the **gray scale** domain, that is the domain used for this approach.

Afterwards, the mean value $\overline{V}$ is calculated using the equation 3.6, and this shall be the reference point for the embedding region on the histogram. The reason for using the mean value is to ensure that the bins on the histogram have enough coefficients to embed the watermark [22, 37, 23, 24]. However using this condition in order to ensure having more samples is not necessarily fulfilled, in some specific cases and this issue is hereafter discussed on the next subsection.

The part of H_{LL} that is used for embedding and detection corresponds to a number of intervals centred in the mean value. The number of bins considered for embedding varies within the embedding range U given by

$$U = [(1-\lambda)\overline{V}, (1+\lambda)\overline{V}] \tag{3.7}$$

with λ ranging within $\lambda \in [0.4, 0.6]$. The parameter λ corresponds to a multiplicative parameter that gives the interval where the histogram is more consistent. Usually the bins located at

the ends of the histograms do not hold pixels or have fewer samples than bins located around the average value [37, 22, 24]. For this reason the parameter λ is used in order to avoid those bins. For this approach the parameter λ is set to be always 0.6 in order to achieve a larger B interval as possible without considering the ends.

Now let B_1 and B_2 be two different bins of the histogram belonging to the U interval. The two bins are selected based on a randomization factor provided by the secret key K. By selecting the bins using the key provided, the security of the algorithm is increased because the positions of the bins B_1 and B_2 are not known by the attackers if they do not know the secret key. Both bins B_1 and B_2 are the centres of two different regions located in the interval U. The size of the regions should be equal for both and the total size of them is given by

$$r = 2M+1, \quad M \in \mathbb{N} \wedge M \geq 0 \tag{3.8}$$

Where the parameter M corresponds to the number of bins selected in each direction of the center of one region. As the objective of the algorithm is to be resistant against geometrical attacks, one characteristic of those types of attacks should be considered that is the possibility of interpolation errors. In most of the geometrical attacks the pixels of the image undergo an interpolation to recreate the attacked image. This type of operation might change the pixels values and consequentially they will be located in a different bin than the original. In order to avoid this effect to prejudice the watermark, the number of bins for each region is augmented by an M factor. The works [37, 22, 24] use the same technique to increase the robustness of the embedding. In those papers the M factor indicates the number of bins to be selected to each region. In this work the factor M stands for the number of neighbouring bins on the left and the right for each selected bin in M. For example, if $M = 0$ it means that the regions centred in B_1 and B_2 will consist solely on the center bin. When a geometrical attack occurs, the coefficients that belong to those bins might be changed to neighbouring bins, thus removing the watermark. In order to increase the robustness at least one extra bin should be considered in each direction, *i.e.* M should be set to $M \geq 1$.

Both bins B_1 and B_2 and its neighbours belong to the regions G_1 and G_2 respectively. G_1 and G_2 are denoted by

$$G_i = \{h_{LL}(j) \mid j \geq B_i - M, j \leq B_i + M\} \quad j, M \in \mathbb{N} \quad i = \{1,2\} \tag{3.9}$$

The regions G_1 and G_2 should be disjoint. Additionally, another region called *disposal region* will be considered. Disposal region is the name given to the regions located on the surrounding of both groups G_1 and G_2 as illustrated in Figure 3.3. These regions are used when the coefficients of the bins of the groups G_1 or G_2 are reassigned for inside or outside of the groups. The implemented algorithm uses two disposal regions for each region, one at the left and the other at the right with the same size r from G_1 and G_2.

The watermark is embedded by modifying the ratio of the energies from the groups *G1* and *G2* as seen in Figure 3.4. The objective is to rearrange the energies of those groups in order to

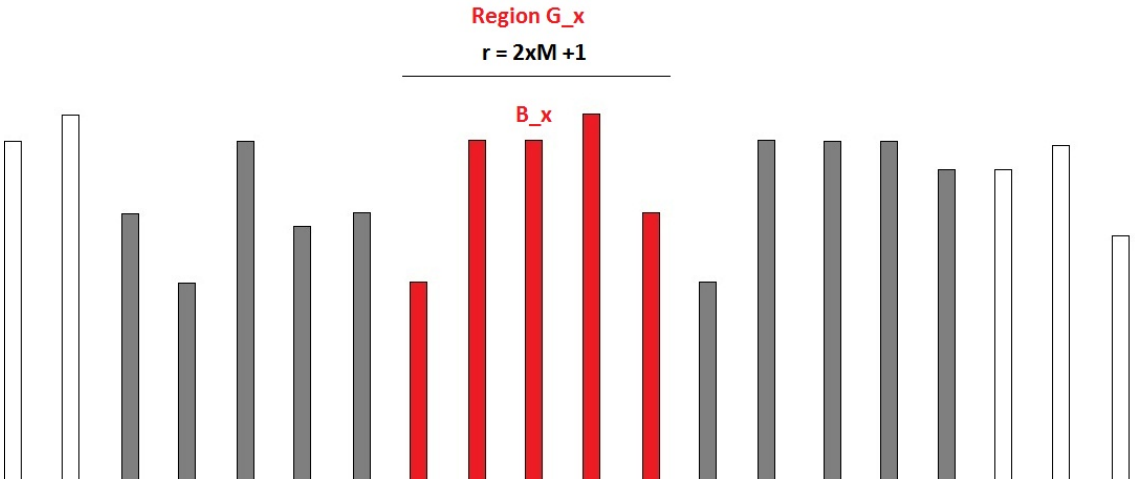


Figure 3.3: subgroup selected on the histogram. The bins marked in red represents an embedding region whereas the gray marked bins represents the *disposal regions*.

achieve the ratio r given by Equation 3.10.

$$ratio = \begin{cases} \log(\frac{G1}{G2}) \geq \log(T) & if \quad \omega(i) = 1 \\ \log(\frac{G1}{G2}) < \log(T) & if \quad \omega(i) = 0 \end{cases} \tag{3.10}$$

Henceforth, *log* will be denoted to represent the basis 10 logarithm, *i.e.* $\log_{10}$. T is the threshold for embedding, and T should be a positive number larger than 1, $T > 1$, $T \in \mathbb{R}$ and $\omega(i)$ corresponds to the bit of the i^{th} position of the message to be embedded. For example, if $\omega(i) = 0$, the ratio is computed by performing the summation of the bins in G_1 and G_2 and dividing them afterwards. If the ratio is smaller than $\frac{1}{T}$ then no operation is needed, otherwise it will be necessary to modify the coefficients from the disposal groups of G_2 assigning the coefficients to G_2, and at the same time modify the coefficients from G_1 to its disposal groups until the ratio is equal or smaller than $\frac{1}{T}$.

When a coefficient from a disposal group is modified to a group itself, its new value depends on the position of the coefficient given by Equation 3.11. In other words, it depends if the disposal group is located at the right or at the left of the group. If a value from one of the groups G_1 or G_2 is modified, its new value is given by Equation 3.12.

$$b' = B_i + sign(b - B_i) \left\lfloor \frac{|\, b - B_i \,| - M}{2} \right\rfloor \tag{3.11}$$

The value b is the bin index from the disposal group located on the left or the right side of the group G_1 or G_2. Its value after reassignment to the groups G_1 or G_2 is denoted by b'.

$$\begin{cases} b' = b + k(M+1) \quad , k = \{-1, 1\} & \text{if } b = B_i \\ b' = b + sign(b - B_i)(|\, b - B_i \,| + M + k) \quad , k = \{0, 1\} & \text{otherwise} \end{cases} \tag{3.12}$$

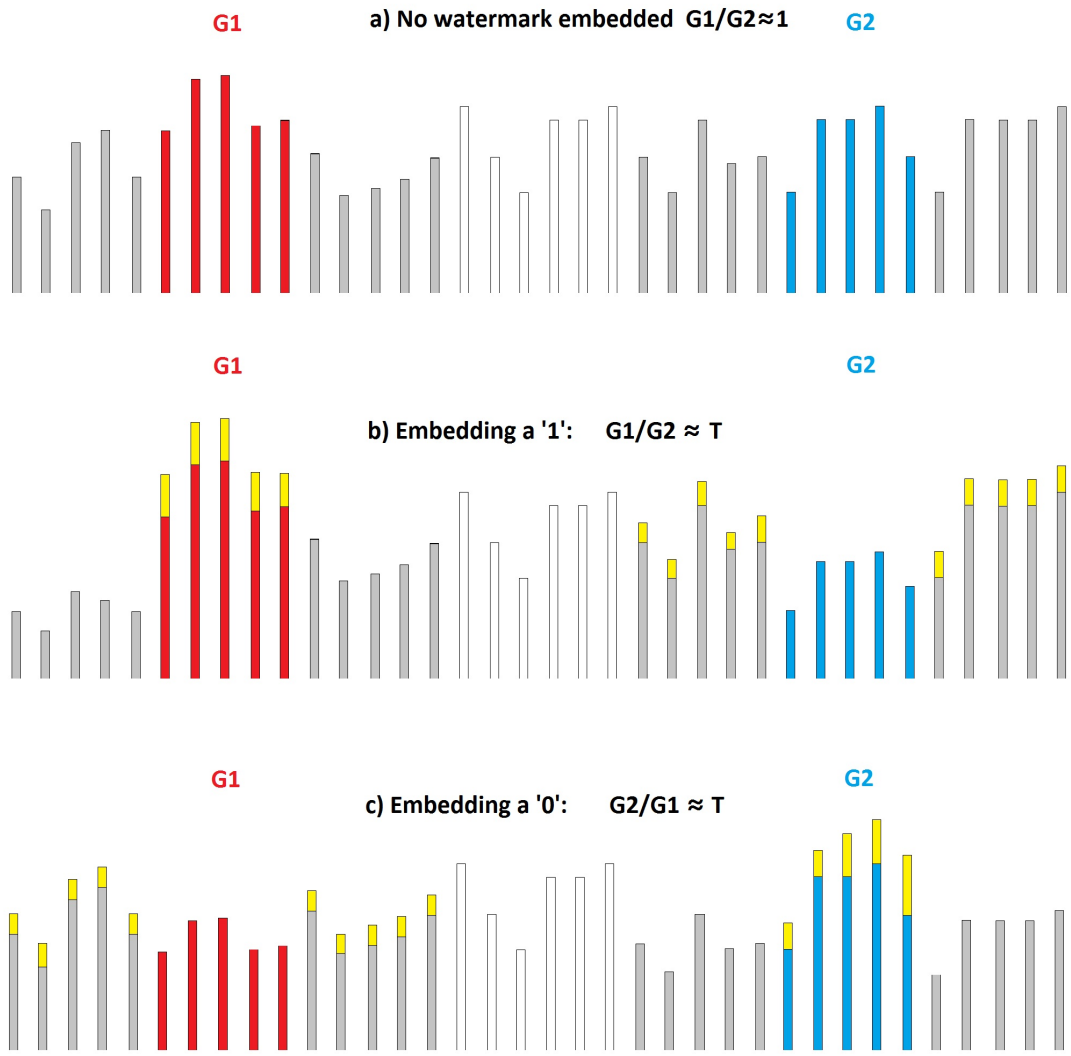


Figure 3.4: Illustration of a set of bins belonging to the embedding region. Illustration a) represents the case when there is no watermark embedded *i.e.* $\frac{G1}{G2} \approx 1$. The illustration b) represents the case when a '1' is embedded, thus increasing G_1 and decreasing G_2, assuring the condition $\frac{G1}{G2} \approx T$. In the illustration c) is depicted the case when a '0' is embedded into the block making $\frac{G2}{G1} \approx T$

Now b is a bin index of the group G_1 or G_2, and b' represents the bin index of b after modification to one of the disposal groups, and the value of k is randomly selected from the set.

The number of modified coefficients from the groups G_1 and G_2 are denoted by the parameters δ_1 and δ_2. Those parameters are non-deterministic *i.e.*, they are not selected prior to the embedding. Nevertheless, these parameters can used to know how many coefficients were changed in order to embed the bit. The calculation method is given by the equations

$$\delta_1 + T\delta_2 \geq TG_2 - G_1 \quad if \quad \omega(i) = 1$$

$$\delta_2 + T\delta_1 \geq TG_1 - G_2 \quad if \quad \omega(i) = 0$$

After the watermark embedding the groups G_1 and G_2 will be modified to G_1' and G_2' respectively. The modified groups G_1' and G_2' are given by

$$G_1' = G_1 + \delta_1$$

$$G_2' = G_2 - \delta_2$$

if a '1' is embedded. If a '0' is embedded then the groups G_1' and G_2' are given by

$$G_1' = G_1 - \delta_1$$

$$G_2' = G_2 + \delta_2$$

After embedding, the IDWT for the block is computed generating the watermarked block.

3.1.2 Watermarking detection

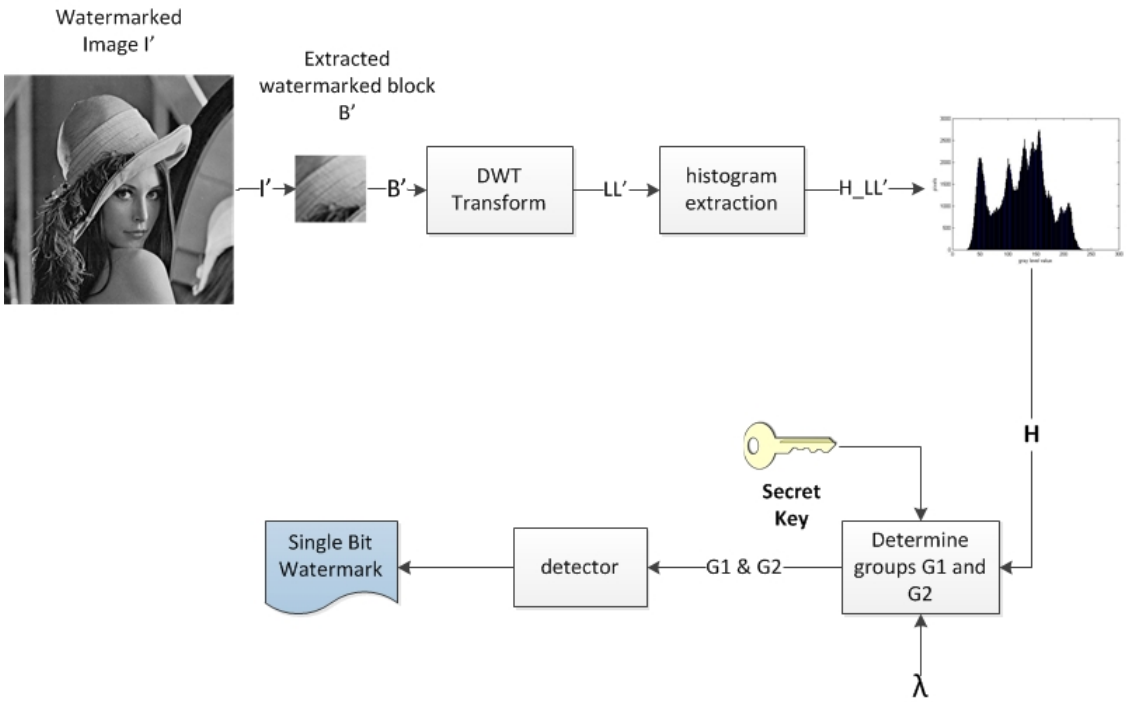


Figure 3.5: Watermark detection procedure.

The watermarking detection is performed receiving as input parameters the watermarked image I', the secret key K and the block size N as illustrated in Figure 3.5. First, the number of bits that are embedded in the image is estimated by using the dimensions of the watermarked image I' and the block N. Therefore, the dimensions of the watermarked image, even after an attack should be the same as the original image. Once again, the number of blocks in the image is calculated

by using the Equation 3.3. Afterwards, the detection algorithm is performed blockwise computing the first level DWT for each block and extracting the ratio. The ratio is computed using the same method as the embedding. First the centres B_1 and B_2 of the regions G_1 and G_2 are selected with the assistance of the same secret key used for embedding in order to set the randomization seed correctly for the selection of B_1 and B_2. The summation of the bins of regions G_1 and G_2 is computed, and finally the ratio is computed.

$$\omega'(i) = \begin{cases} 1 & if \quad \log(\frac{G_1}{G_2}) \geq \tau \\ 0 & if \quad \log(\frac{G_1}{G_2}) < -\tau \end{cases} \tag{3.13}$$

The value $\omega'_x(i)$ corresponds to the i^{th} bit in the sequence of the image I'_x, where $i = \{1, ..., T_{blocks}\}$. The detection process is repeated for all the T_{blocks} of the watermarked image until the watermark ω'_x is extracted. The parameter τ is the decision threshold. Usually the value of the threshold τ is $\tau \geq 0$.

After extracting all the bits in all the images of the set S the message ω'_S is reconstructed accordingly to Equation 3.2.

3.1.3 Algorithm improvements

After the implementation and testing of the algorithm presented in the previous section, some tests were performed in order to evaluate the performance of the proposed scheme. In order to increase the performance of the algorithm some improvements were made. These improvements are listed bellow.

3.1.3.1 Blocks selection constraint

According to [22, 37], the mean value $\overline{V}$ of an image is used as the center of the interval U for embedding because it represents the region where the bins have more occurrences of the DWT coefficients, and consequentially the embedding is more robust. However, in this approach the images are split into several blocks of size N, and the chance of selecting blocks that are not suitable for embedding, *e.g.* background only, exists. The blocks that are not suitable for embedding are called *bad blocks*.

Now, an example in how the blocks can be bad for embedding even though the whole image is a good candidate for embedding is depicted. In Figure 3.6 an image having a good histogram, depicted in Figure 3.7 is shown. The DWT histogram of the whole image is more or less constant around the average value $\overline{V}$ making the image a good candidate for embedding. Differently, the block histogram depicted by 3.8 is not a good candidate for embedding, because around the mean value there are several bins with a small value of coefficients and at the same time other bins, more precisely located near the end of the histogram, belong to a region of the image that might represent a *local background*. Local background regions are described by regions on the blocks that can be interpreted as the background of the blocks. For example, in Figure 3.6 the block is characterized by having a large homogeneous region. This homogeneous region of the block is

understood as a local background, though in the whole image context this region does not belongs to the background of the image.

By modifying a large number of coefficients the watermark may become visible in the image because it creates visual artifacts in this local background. For that reason this types of blocks should be avoided in order to keep the transparency.

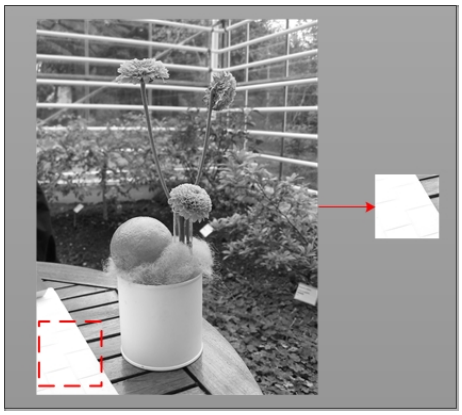


Figure 3.6: Grayscale image and one extracted block of size 256x256 pixels

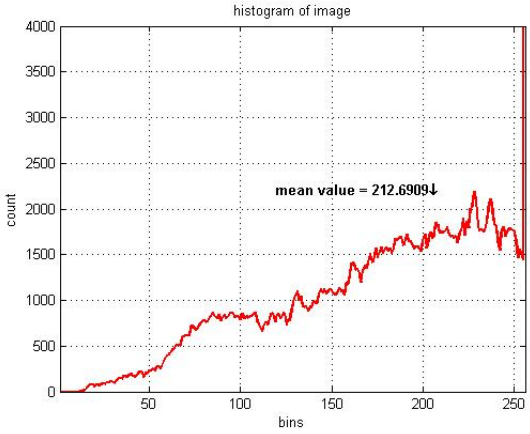


Figure 3.7: Extracted first level low frequency sub-band histogram of the image 3.6

When the bins that have just a few coefficients or no coefficients are called *bad bins* [37]. The bad bins occurrence is more common around the block ends, although in some particular cases the occurrence of the bad bins may be around the average value. For example, considering one square block of size N^2 where half of its coefficients are located on the bin with the value 0 and the other half on the bin with value 255, the mean value is given by $mean = \frac{(0\times\frac{N^2}{2})+(255\times\frac{N^2}{2})}{N^2} \approx 128$. All the bins on the histogram except the bin '0' and '255' are bad bins of the histogram and are not suitable for embedding. This effect is depicted in Figure 3.9 where most of the values are situated in the ends of the histogram. As expected, the image histogram does not have enough coefficients around the average value as illustrated in the histogram depicted in Figure 3.10. The same effect

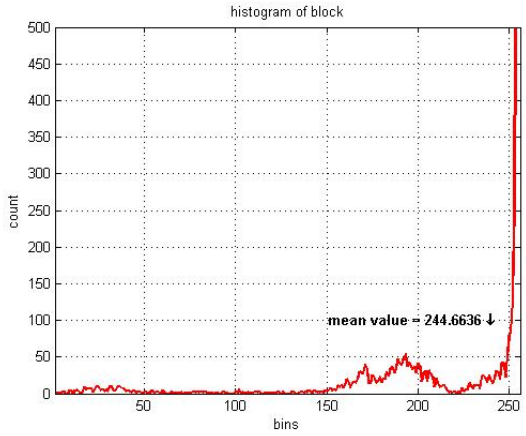


Figure 3.8: Extracted first level low frequency sub-band histogram of the block 3.6

is equally observed for a selected block in Figure 3.11. Therefore, in the presence of cases where the block does not have enough coefficients around the average value, the block will be discarded.

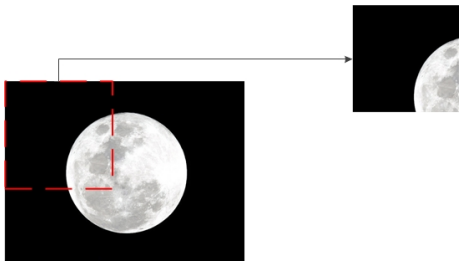


Figure 3.9: Picture of a moon and an extracted block with size $N = 256$.

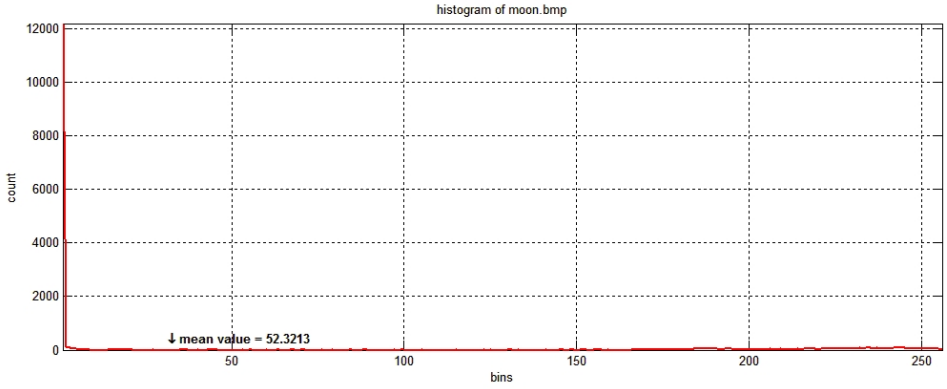


Figure 3.10: Histogram of the figure Figure 3.9.

In order to avoid the bad blocks, an evaluation of the histogram is made to check if the block is suitable for embedding prior to the embedding. The constraint of the block hold the background mostly and the constraint of having *bad bins*. First, in order to reduce the probability of embedding on the background, the bins of the U interval is analysed. The analysis is made by performing the following steps.

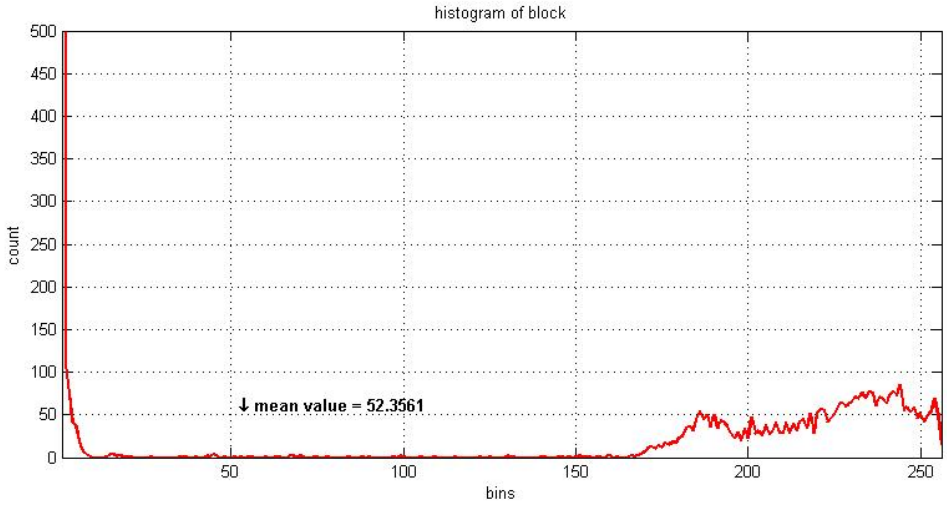


Figure 3.11: Histogram of the block shown in the figure Figure 3.9.

Extract the *U* interval

First the U interval is determined by accordingly to the interval defined by Equation 3.7.

Check the weight of each bin in *U*

The whole block contains N^2 coefficients, although since only the low frequency sub-band is used the number of coefficients available is reduced to $\frac{N^2}{4}$. The weight of each bin of U is computed by dividing the occurrence of the bins by the value $\frac{N^2}{4}$ in order to obtain the weight. The number of coefficients of the bins of the U interval is also computed.

Remove the bins that constitute a narrow and tall region

A narrow and tall region is defined as being a small set of neighbouring bins (this approach selected between 5 to 20 consecutive bins) that have more or less the same coefficient occurrence but have much more than the surroundings bins. The method to measure the coefficient occurrence is due to the weight of each bin. If some region with this characteristic is found in the interval U, the bins are not considered for embedding.

Compute the average of coefficients remaining on the interval

If the number of coefficients is less than half of the original value. The block is not considered as being good for embedding.

Now, in order to avoid the embedding in a interval with too many bad bins, the interval U should not have more than a certain number of bad bins. The number of bad bins and the minimum number of coefficient in each bin that will make the block be discarded for embedding were computed based on empirical tests. The maximum number of bad bins allowed in the interval was 10% of its total size, whereas the minimum number of samples in a non-bad bin is set to 10. If an interval presents more than the maximum number of bad bins allowed, the block is also discarded.

If both requirements for bad bins and background are met the region is selected as being proper for embedding.

3.1.3.2 Possibility of sharing the disposal region

The case of sharing the disposal region was analysed in order to measure the robustness of the algorithm if a shared region is allowed between the groups. Considering the histogram H_{LL}, the histogram contains L different bins and the interval U for embedding contains K bins, $K < L$, $K \in [(1-\lambda)\times\overline{V},(1+\lambda)\times\overline{V}]$. For the selection of the two centres of regions B_1 and B_2, these bins are selected based on the randomization factor provided by the secret key K. Now two different scenarios for the selection of the regions G_1 and G_2 will be regarded:

The disposal region can be shared between the two neighbours

In this case, the centres B_1 and B_2 are selected based on the randomization factor provided by the secret key attending to the only constraint that is $\mid B_1 - B_2 \mid \geq (2r)$. This means that the only constraint applied is that the G_1 and G_2 are disjoint separated by two times the size of the region r. In fact, the minimum distance allowed for these groups is in the case where they share the same disposal region, as illustrated in Figure 3.12.

The disposal region cannot be shared between the two neighbours

Now for this scenario, the centres B_1 and B_2 are selected in the same way of the previous scenario but now attending to the constraint $\mid B_1 - B_2 \mid \leq (3\times r)$. In fact, now in addition to the regions G_1 and G_2 being disjoint, the disposal regions of both regions G_1 and G_2 are also disjoint. This approach is illustrated in Figure 3.4*a)*.

Now, let U be an interval with i bins. If an attacker who knows the algorithm tries to pick randomly two different bins the chance of he selecting the right bins are higher when the disposing regions are also disjoint because less r bins are available once he pick the first bin. Therefore, using the shared disposal region provides an improved security. On the other hand, when the disposal regions are shared a coefficient that has been moved from one group to the disposal region, could be reassigned again to belong to the other region. This situation is undesirable because it might lead to visible artifacts on the watermarked image and the robustness could be reduced due to the rounding errors.

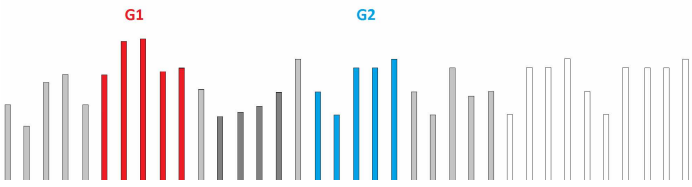


Figure 3.12: The groups G_1 and G_2 share one disposal region. The total number of bins occupied in this case is 26

A set of 10 images with variable size was tested in order to verify which of the two solutions performed out well. The test consisted in comparing the ratio differences in two occasions: when

the blocks were watermarked before saving and after saving the image and detecting the ratio again. If the difference of the ratios is small it means that the robustness is good, otherwise it means that rounding errors are influencing the watermark robustness. In Figure 3.13 it is possible to observe that the results achieved by using the disjoint disposing regions provides a higher robustness. The selected solution was using disjoint regions.

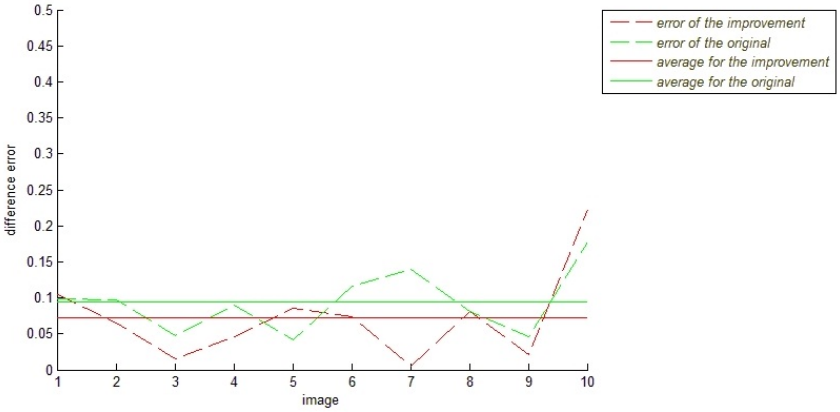


Figure 3.13: Results for the sharing the disposal region test.

3.2 Integration of the fingerprinting codes

The integration of the fingerprint is performed by embedding fingerprint bits into the blocks of the image set. The fingerprint should be split in several parts accordingly to Equation 3.2 and integrated into one image at a time. The embedding procedure starts when a copy of a set of images is requested. The embedding procedure is depicted in Figure 3.14 and is described bellow.

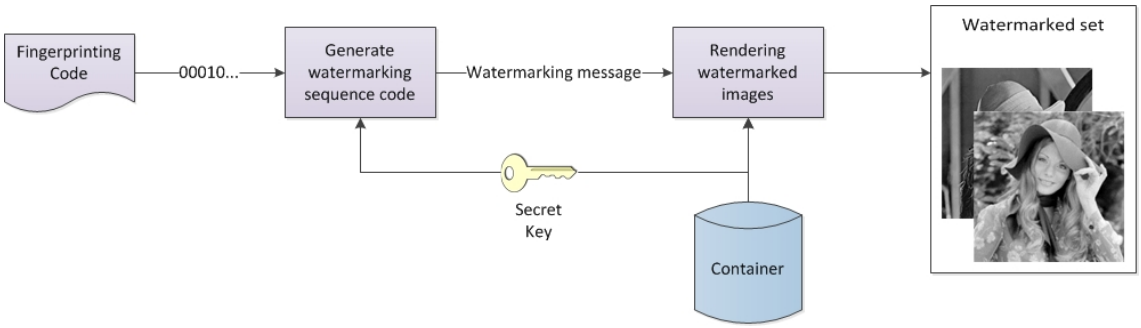


Figure 3.14: Process of rendering the fingerprinted image set upon request with a fingerprinting code.

Generate the watermark message

In order to generate the watermark message, first the fingerprint should be generated. The length of the fingerprint should be at most the same of the total number of block in the set of images. Although the number of blocks in the set of images may exceed the length m of

the fingerprinting code. Attending to this fact, a redundancy can be applied. For example, if a fingerprint code ω with length $m = 6$ is integrated in a set of images containing two images and each image is capable to be integrated with 6 bits. The total number of blocks will be 12, and therefore each fingerprinting bit can be repeated once. In this approach, the method used to generate the repetition of the bits is made using the randomization based on the secret key as illustrated in Figure 3.15. The generated watermark message ω is then integrated with the first 6 bits in the first image and the remaining 6 bits in the second image. This phase of generating the watermark sequence consists in the first stage of the watermark embedding upon the request of the content and when the fingerprint is generated.

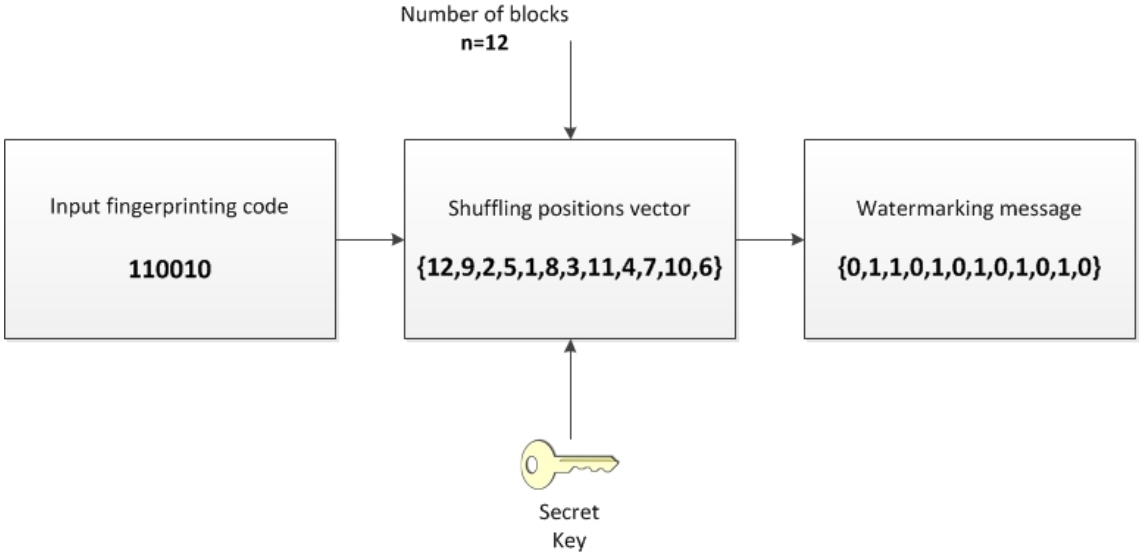


Figure 3.15: process of shuffling the fingerprint code and assigning to **n** blocks. In this example the fingerprint code is of length $m = 6$ and the total number of blocks is given by $n = 12$. Each fingerprint bit can be repeated once.

Rendering the Watermark

The embedding of the shuffled watermarking message is made during the rendering phase. The rendering phase consists in get on the container the specific block marked as '0' or '1' for each position given by the watermarking message. The whole process of rendering the image is depicted in Figure 3.14.

The sequence of the images should always be the same for embedding and detection in order to detect the watermark bits on the right order. The robust hashing program identifies the watermarked images and then the detector can perform the detection of each image in the right order.

3.2.1 Number of blocks in an image set

Let S be a set of images with $S = \{I_i \mid i = 1,...,s\}$ where I_i is the i^{th} image of the set. The size of the set, *i.e.*, the number of images is denoted by s. For each image I_i there is a number j_i

of possible blocks to embed the watermark message. The blocks in each image are denoted by $s_i = \{b_i(j) \mid j = 1, ..., T_{blocks_i}\}$, where $b_i(j)$ represents the block of the i^{th} image in its j^{th} position. Finally the total number of blocks in one set is given by

$$n_{blocks} = \sum_{i=1}^{s} \sum_{j=1}^{T_{blocks_i}} 1 \tag{3.14}$$

Since only one bit will be embedded per block, the length of the watermark message will be the same as the number of blocks n_{blocks}. The number of blocks, *i.e.* the length of the watermarking message should be at least the same as the length m of fingerprints. Let m be the length of a fingerprinting code for a group of n users secure against a number of c colluders. If $m > n_{blocks}$ then it is not possible to embed a fingerprinting code into the set.

For the case that $m \leq n_{blocks}$ it is possible to embed the fingerprinting code into the set. If n_{blocks} exceeds m, repetitions for each watermark bit will be embedded. The number of repetitions is denoted by Equation 3.15. The number of blocks left to create the repetitions may not be sufficient to repeat all the fingerprint code bits unless the parameter n_{blocks} is at least $n_{blocks} \geq 2m$. Since the repetition factor may not be extended to all the bits of the fingerprint code, another parameter called *remaining blocks* (l_{blocks}) is calculated. The parameter l_{blocks} corresponds the number of fingerprinting bits that can be repeated one more time beyond n_{blocks}. This parameter can be described as seen on Equation 3.16. For example, if a fingerprint code with size $m = 100$ is given and the image set has $n_{blocks} = 231$ blocks suitable for embedding. Every fingerprint code can be embedded twice ($R_{blocks} = \lfloor 231/100 \rfloor = 2$). However, $mod(231/100) = 31$ blocks remain unused. In order to use all the blocks available on the set the first 31 bits of the fingerprinting code will be replicated another time.

$$R_{blocks} = \left\lfloor \frac{n_{blocks}}{m} \right\rfloor \tag{3.15}$$

$$l_{blocks} = mod\left(\frac{n_{blocks}}{m}\right) \tag{3.16}$$

The equation that shows the relations among all the parameters, number of blocks n_{blocks}, repetitions r_{blocks}, remaining blocks l_{blocks} and fingerprinting code length m is given by

$$n_{blocks} = m r_{blocks} + l_{blocks} \tag{3.17}$$

3.2.2 Hash database generation

The hash database is constituted by the hashes of the images and its blocks. For each image its hash is computed and then stored in the database. Afterwards, the blocks of the image are evaluated in order to check its suitability for embedding. If the block is suitable for embedding the hash is stored, otherwise there is no need to store the hash and the block is not used for embedding. The

process is done successively until all the hashes for the images and according blocks are stored in the database.

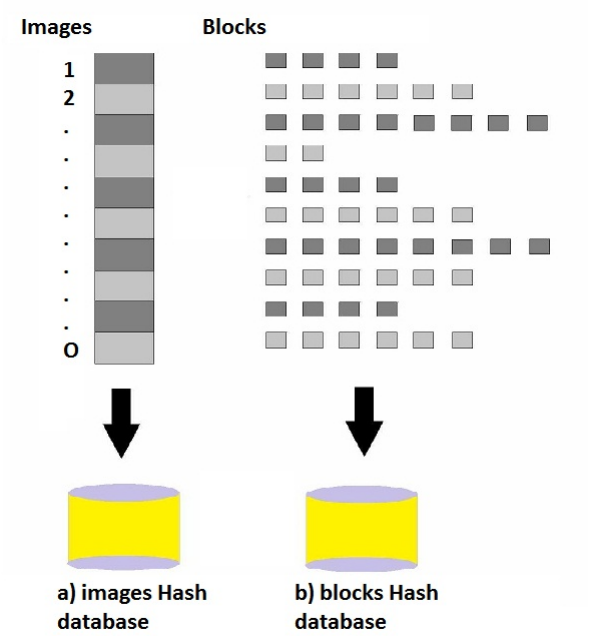


Figure 3.16: The hash tables creation for images and blocks.

The reason for using the hash of both images and blocks is simple. The detector should be able to know which blocks have a watermark embedded, since that during the container creation some blocks may be unused because they are not suitable for embedding. Hence the hashes of the blocks are used for detection. Another important aspect is that some blocks belonging to different images may have similar hashes. In order to avoid confusion, the hashes of the images are used also.

Chapter 4

Implementation

This chapter will be devoted to the implementation of the algorithm proposed on the past chapter. A description of the implementation and the tools used is made for the algorithm and the tools to be used on the tests and some of the selected attacks introduced in chapter 2 will be studied more in detail.

4.1 Watermarking Program Implementation

The implementation for the proposed algorithm was made using C++ programming and the library *OpenCV* [1]. Another two additional implementations developed by *Dr. Ing. Huajian Liu* member of *Fraunhofer SIT* were used; A 2D-DWT library implementation in C++, and the *robust hashing tool* were used for the algorithm proposed in chapter 3.

The implemented watermarking algorithm, when in the embedding mode, outputs the watermarked image and a file with the ratios of the blocks. Whereas in detection mode, the watermarking algorithm outputs the message containing '0' or '1' and the detector responses. Typically when a watermarking program used for embedding, at least the secret Key K and the message ω with length m is provided. This implementation beyond the parameters mentioned before needs two more input parameters; one is the block size N, and another the embedding strength T. The block size is of size $N = 128n \times 128n$ with, $n = \{1,2,3,4\}$. The embedding strength T is variable accordingly to the values $T = \{2,3,4,6,16,30\}$.

The correct input parameters needed for the detection process are: the block size N, the key K and the message length m. A false committal of these parameters implicate detection errors, i.e. the detector is not able to read the message correctly. In order to increase the security of the algorithm, the block size is provided as a part of the secret key.

The images used were converted to a lossless format (*Windows bitmap*) and only the gray scale channel was used. For the tests many different sizes of images were used as seen on appendix A.

[1] OpenCV is the shortening for **Open** Source **C**omputer **V**ision and is a library of programming functions for real time computer vision. The utilization for this library on the thesis scope is to use in more simple applications for image processing operations

The high resolution images used were downloaded from the CASED repository of images[2]. Other images used were downloaded from the Signal and Image Processing Institute website,SIPI[3].

4.2 Selected Attacks

Now, the implemented attacks will be presented. As discussed previously, the watermarking for image set scheme will be destined to works that after an attack to the images, the aestheticism should be preserved. In this testset an attacked image is considered as suitable for using when after some common removal and geometrical attacks the appearance still similar to the watermarked. The removal attacks, as already discussed in chapter 2, are based on degradation of the image in order to remove the watermark message, *e.g.* JPEG compression, brightness adjustment or Gaussian filtering. The selected removal attacks for the presented scheme are listed in the table Table 4.1.

Type of attack	Attack parameters
JPEG compression	$Q = 80$ [a]
Contrast adjustment	± 20
Bright adjustment	± 40
Gamma correction	$\gamma = 0.9$
Gaussian Blur filtering	2 pixel radius [b]
Median filtering	2 pixel radius
Sharpen filtering	2 pixel radius

[a] Q represents the quality factor for JPEG compressions.

[b] This corresponds to the Window size. In this case a window with dimensions 5×5 was used.

Table 4.1: Set of removal attacks applied into the selected images

The second category of attacks performed was geometrical attacks. Accordingly to chapter 2, geometrical attacks aims not to remove the watermark message but instead, altering the message, that it is no longer readable by changing the pixels. The most common geometrical attacks are *rotation*, *cropping* and *scaling*. However, these attacks have the disadvantage of heavily modifying the visual aspect of the image. Two constraints are imposed for the geometrical attacks: the first consists in preserving the dimensions of the watermarked image, and the second consists in preserving the visual aspect of the images are used to select the attacks. From the three most straightforward attacks, rotation, cropping and scaling only the rotation and scaling are considered. Two other attacks based on combination of geometrical distortions were selected; the random bending attack (RBA) and pre-warping using the Markov Random Fields (MRF). The random bending attack was performed by the *Stirmark tool* software. As seen on chapter 2, this attack was introduced by *Petitcolas et al.* in [27].

[2] This thesis was developed in CASED **C**enter for **A**dvanced **Se**curity Research **D**armstadt facilities.

[3] http://sipi.usc.edu/database/database.php?volume=misc

Type of attack	Attack parameters
Light rotation	angle $= 0.5^{o}$
Rescaling	$85\% \rightarrow 117.65\%$ [a]
RBA	$Q = 90$ [b]
MRF pre-warping	$d_{\max} = 4 \quad , N = 8$ [c]

[a] After scaling, the image was rescaled back in order to preserve the original size.

[b] The attack is combined with a JPEG compression.

[c] The value for the maximum displacement is given by $d_{\max}$ and the displacement matrix has the size N.

Table 4.2: Set of geometrical attacks applied into the selected images

4.2.1 Attacks implementation

The majority of the image enhancement attacks were performed using the software *Irfanview*[4]. The following attacks were performed using the *Irfanview tool*: JPEG compression, brightness and contrast adjustment, gamma correction, Gaussian blur, sharpening, rotation and scaling. The median filtering attack was implemented using the *Mathworks*® *Matlab* software.

4.2.2 Random Bending Attack

As introduced previously the random bending attack is a specific attack available in the Stirmark tool for image benchmarking. The attack consists in a three step geometrical modification of the input image [27, 39]. The algorithm construction was presented in [28] as:

- The first step consists in applying the following transform

$$\begin{matrix} x' \\ y' \end{matrix} = \begin{matrix} t_{10} + t_{11}\mathrm{x} + t_{12}\mathrm{y} + t_{13}\mathrm{xy} \\ t_{20} + t_{21}\mathrm{x} + t_{22}\mathrm{y} + t_{23}\mathrm{xy} \end{matrix}$$

  Where x' and y' are the new coordinates and x and y the original coordinates. The matrix $T = \{t_{ij}\}$ is a general transformation matrix.

- The second step of the transform is a pixel shifting based on the position of the pixels relatively to the dimensions $M \times N$ of the image. The displacement should be maximum ($d_{\max}$) at the center of the images and none at the borders.

$$x'' = x' + d_{\max}\sin(y'\frac{\pi}{M})$$

$$y'' = y' + d_{\max}\sin(x'\frac{\pi}{N})$$

[4] The *Irfanview* tool is a freeware image viewer with many editing options available, such as image compression, filtering and compression

- The last operation consists in a random local displacement for each pixel position

$$x''' = x'' + d_{\max}\sin(2\pi f_x x'')\sin(2\pi f_y y'')\text{rand}_x(x'', y'')$$

$$y''' = y'' + d_{\max}\sin(2\pi f_x x'')\sin(2\pi f_y y'')\text{rand}_y(x'', y'')$$

Afterwards the geometrically distorted image undergoes a JPEG compression factor before finishing the attack.

4.2.3 pre-warping using MRF

A C++ implementation of the pre-warping using the Markov random fields was performed within the framework of this thesis. The proposed approach made in [28] was intended to be applied into collusion-secure schemes. Although the very same approach can be used as a geometrical attack as well. For that reason a transcoding of the Matlab program of [28] had to be implemented, so that in this thesis the MRF could be used as an attack.

4.2.3.1 The MRF

As mentioned before the algorithm uses the Markov random fields characteristics to generate the displacement to be applied into the images. In order to introduce the MRF the labelling outlook will be introduced as the mathematical representation. Two types of elements are considered for the labelling; a set of sites and a set of labels.

Let $\mathfrak{S}$ be a discrete set of sites m , $\mathfrak{S} = \{1,...,m\}$. A site is represented as either a point or a region in the euclidian space. The set of labels represent the status of the regions contained in $\mathfrak{S}$ and is represented by $\mathfrak{L}$. The labelling problem consists in labelling each label from $\mathfrak{L}$ to a site $\mathfrak{S}$. Naturally, all the sites are connected by different sites, *i.e.* regions or points. The surrounding sites of a site in $\mathfrak{S}$ are called neighbours. The neighbourhood system is defined by $\mathfrak{N} = \{\mathfrak{N}_i \mid \forall i \in \mathfrak{S}\}$, where $\mathfrak{N}_i$ is the set of neighbours for the i^{th} site. From the neighbouring system two properties are verified:
1) The site cannot be neighbour to itself.
2) the neighbouring relationship should be symmetric.

Lastly, the clique concept is introduced and is defined by c. A clique is a subset of sites in S, and it is either a single site $c = \{i\}$, or a group of neighbouring sites $c = \{i, i'\}$. Each single site clique is denoted by $\{C_1, C_2, C_3....\}$ and the collection of cliques are represented by C as being $C = C_1 \bigcup C_2 \bigcup C_3 ... C_m$.

Now, $F = \{F_1, ..., F_m\}$ is a family of random variables defined on the set S in which each random variable f_i assumes a value in $\mathfrak{L}$. The family F receives the name random field. For any f on F defined as $F = \{F_1 = f_1, F2 = f_2, ..., F_m = f_m\} = f$, the family F is called a Markov Field on S only if the following two parameters are satisfied:

- $P(f) > 0, \forall f \in \mathbb{F}$, this is known as *positivity*.

- $P(f \mid f_{\mathfrak{S}-\{i\}}) = P(f_i \mid f_{\mathfrak{N}_i})$, this property is called *Markovianity*.

The $f_{\mathfrak{S}-\{i\}}$ is the set of labels at the sites in $\mathfrak{S} - \{i\}$, whereas $f_{\mathfrak{N}_i} = \{f'_i \mid i' \in \mathfrak{N}_i\}$ stands for the set of labels in the neighbourhood on the i^{th} site. The sets of variables in F are considered as a Gibbs random field (GRF) on the sites $\mathfrak{S}$ regarding the neighbouring system $\mathfrak{N}$ only if the configuration $P(f)$ obeys a Gibbs distribution

$$P(f) = Z^{-1} e^{-T^{-1} U(f)} \tag{4.1}$$

The Z is a normalization parameter constant and is defined as $\sum_{f \in \mathbb{F}} e^{T^{-1} U(f)}$, whereas T is the temperature constant and $U(f)$ is the energy function. The energy function is given by the sum of the clique potentials denoted by Equation 4.2, while $P(f)$ is the probability function for the occurrence of happening a given configuration f.

$$U(f) = \sum_{c \in C} V_c(f) \tag{4.2}$$

4.2.3.2 The pre-warping MRF

In the implemented algorithm the random field is the variable F and the set $\mathfrak{S}$ is the pixels of the image. The value assigned to each variable on the field represents the displacement for a pixel, *i.e.*, each random variable f is denoted by two components x and y. The random variables relation with the random fields F is represented by

$$F_i \longleftrightarrow f_i = (f_x, f_y) \in L \times L$$

As seen previously an MRF is determined by the Gibbs distribution and the neighbourhood system is already defined. The neighbourhood considered is the first order neighbourhood of adjacent pixels for each given pixel (x, y).

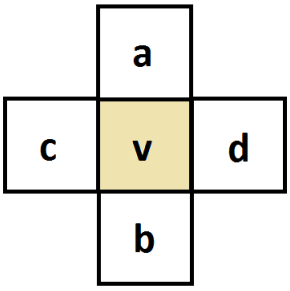


Figure 4.1: The neighbourhood for the pixel v.

The potential function is represented by a bivariate normal distribution

$$V_{((x,y),(\tilde{x},\tilde{y}))} = \frac{1}{2\pi\sigma_x\sigma_y}\exp\left(-[\frac{(f_x - f_{\tilde{x}})^2}{2\sigma_x^2} + \frac{(f_y - f_{\tilde{y}})^2}{2\sigma_y^2}]\right) \tag{4.3}$$

The variables f_x and f_y are the components of the displacement vector $f_{(x,y)}$ associated with the pixel (x,y) shown in Figure 4.1 as v , whereas $(\tilde{x},\tilde{y})$ is one of the points belonging to the neighbourhood of the pixel v in Figure 4.1. Analogously to f_x and f_y, the values $f_{\tilde{x}}$ and $f_{\tilde{y}}$ are the components of the displacement vector $f_{(\tilde{x},\tilde{y})}$. The components of the standard deviation σ are given by $\sigma = (\sigma_x, \sigma_y)$. The MRF in this approach is a minimization problem to the energy for the equation 4.3. The objective is to only create the displacement field, and in order to achieve this field the equations 4.1 and 4.3 are used. The algorithm initializes the displacement field with random values assigned to each pixel (x,y). The values for each pixel in the image are independent of the other pixels and they are represented by

$$L_1 \times L_2, \quad L_1 = \{f \in \mathbb{Z} : -c_1 \leq f \leq c_1\} \quad \text{and} \quad L_2 = \{f \in \mathbb{Z} : -c_2 \leq f \leq c_2\}$$

The values for c_1 and c_2 are calculated based on the dimensions of the input image. The initial random field is a noisy version of the final displacement field. The final displacement field is achieved by iterating the process until the minimum energy exceeds a minimum threshold. Afterwards, the displacement field is applied to the image using an interpolation method.

Figure 4.2: A pre-warping attack example. The image on left the is watermarked image and the image on the right side is the watermarked image after a pre-warping attack.

The algorithm presented in [28] introduced an improvement based on the intelligibility of the warped images. The application of the pre-warping applies small distortions that, even though

being local, they might cause many visible undesired effects. The Figure 4.2 shows the application of a pre-warping attack using the MRF without any correction. As it is possible to notice, the image on the right is very similar to the one on the left however, the details on the bars that look twisted affects severely the image quality.

These undesired effects is visible in specific shapes of the images. In this case the bars look twisted, but also in some other cases round shaped objects may look elliptical-shaped after the attack. In order to avoid these effects a correction of the warped image using a edge detector is made. The edge detector used is the *Rothwell*. A C++[5] implementation of the Rothwell detector was used in order to implement the improvement. The objective of the algorithm is to identify regions where the image cannot be altered such as round regions or bar shaped regions. The advantage of the Rothwell detector in comparison to other edge detectors is a higher accuracy in detecting L shaped forms as illustrate in Figure 4.3, whereas more simple filters such as Sobel or Prewitt do not possess this characteristic.

Figure 4.3: The rothwell edge detection.

The result with the image corrections is shown in Figure 4.4. It is possible to notice the correction made by pre-empting the distortion of the bars.

[5] http://www.marathon.csee.usf.edu/edge/edge_detection.html

Figure 4.4: A pre-warping attack correction demonstration. The image prior the correction is presented on the left, and the corrected image is presented on the right side.

Chapter 5

Evaluation and Results

This chapter will be devoted to the presentation of the final results achieved with the implementation of the algorithm proposed within this thesis. First, a discussion on the results obtained for the transparency achieved face to the algorithm characteristics is presented. Afterwards, the results regarding the robustness is presented considering the attacks discussed on the previous chapter.

5.1 Transparency Results

Test number	average PSNR
1	53.94
2	53.96
3	53.24
4	53.97
5	53.82
6	55.00
7	49.95
8	54.93
9	52.99
10	52.45

Table 5.1: Average PSNR for each watermarked set

The transparency results are given by the PSNR of the watermarked images compared to the original. Because only a small percentage of the pixels is modified, the watermark message barely causes any difference of the watermarked image when compared to the original. The PSNR average for the watermarked set when compared to the original is over $50dB$ for a embedding threshold of $T = 3$ as illustrated on table 5.3. For the test 7 the embedding threshold was $T = 6$ and the PSNR for the watermarked set was $49.95dB$. The results shown on [22] presents a more reduced PSNR which is around $45dB$. The average value the differences for a watermarked image and the original are smaller than the decimal *i.e.* $Av_o - Av_w \leq 0.01$. On the table 5.2 it is possible to see the results for some of the images. The images described on the table are available in appendix B.

Features	Lena	Baboon	Plane	Drop	Peppers
Av_o	124.11	129.69	179.13	103.26	120.46
Av_w	124.10	129.69	179.13	103.27	120.46
PSNR	53.71	52.15	55.68	52.33	52.12

Table 5.2: PSNR and average of the pixels for different images marked with $T = 3$. Average PSNR for each test image set

Considering the attacks introduced in the previous chapter, the PSNR was measured for one test. The results show that the images suffer a severe degradation in most of the cases. The geometrical attacks present the lower PSNR averages. The explanation for the pre-warping MRF attack and the Stirmark RBA is that the pixels of the images are changed from their original positions. This measure of similarity cannot be fully applied to these two attacks once that the aestheticism still being preserved as previously seen. However, both the rotation and the scaling attacks present a low PSNR as well. The image degradation is mainly due to the interpolation errors when the images are manipulated to recover the original size.

Attack name	$PSNR_{av}$
Watermarked	53.94
Gaussian blurring	28.18
Brightness adjustment	20.47
Contrast adjustment	34.61
Gamma correction	28.47
JPEG compression	31.73
Median filtering	25.83
Sharpen filtering	29.56
Pre-warping	16,91
Stirmark RBA	15,21
Rotation	17,08
Scaling	17.11

Table 5.3: Average PSNR for each watermarked set

5.2 Attacks results

The first tests implemented have different results for different sets of images. The objective for realizing several tests was to measure the payload considering the size N for the blocks and the number of images on the sets. The table 5.4 shows the number of blocks suitable for embedding in each set. The embedding strength threshold picked was $T = 3$ similarly to the approaches given in [22, 24].

The tests consisted in attacking the images accordingly to the selected attacks selected on chapter 4. The detection threshold τ shown in Equation 3.13 selected was $\tau = \log(1.5)$. The different block size parameters selected were $N = \{256, 384, 512\}$.

test	block size (N)	number of blocks
1	256	68
2	384	17
3	512	17
4	256	2744
5	384	1249
6	512	665
7	512	665
8	256	818
9	384	313
10	512	168

Table 5.4: Number of blocks possible to embed for each test with different block sizes

The number of blocks varies accordingly to the number of images on the set and the size N of the blocks. The set '6' and '7' were tested with the same parameters, with the difference that the embedding threshold for the set '7' was $T = 6$. Furthermore, on the table 5.4 it is possible to notice from the tests '2' and '3' that even though the block size is different, the number of blocks is the same. The explanation for this is illustrated by Figure 3.2. The image set used on the tests '1','2' and '3', favours the embedding using the block size 512 or 256 because smaller regions are left without being used.

Removal Attacks

Henceforth, all the results shown on the tables correspond to the bit error rate (BER). The table 5.5 illustrates the results concerning the removal attacks described on table 4.1. The JPEG compression attack and the Brightness adjustment attacks presents a detection rate in almost all of the cases. The worst cases are due to both the median filtering and contrast enhancement attacks that presents an BER of around 40% in some cases. By comparing the tests '6' and '7' it is possible to evaluate the influence of the threshold T increasing. A larger robustness to almost all the selected attacks is achieved by doubling the embedding threshold T. The results of the tests using a larger block size are more satisfactory. However, as seen on table 5.4, the payload in some cases is drastically reduced.

Geometrical Attacks

The geometrical attacks implemented differ from the attacks performed in [24] in such a way that the attacks in this thesis considered should preserve the aspect of the image, *i.e.* after the attack the size should remain the same and the aestheticism should be degraded the

Attack name	Tests numbers									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Gaussian blur	0.088	0.000	0.000	0.122	0.087	0.086	0.069	0.137	0.133	0.113
Brightness adjustment	0.000	0.000	0.000	0.017	0.011	0.006	0.006	0.036	0.003	0.000
Contrast adjustment	0.353	0.191	0.176	0.206	0.184	0.167	0.151	0.238	0.185	0.160
Gamma correction	0.176	0.176	0.059	0.146	0.118	0.110	0.097	0.143	0.105	0.102
JPEG $Q = 80$	0.000	0.000	0.000	0.101	0.065	0.054	0.050	0.066	0.048	0.042
Median Filtering	0.412	0.294	0.176	0.325	0.250	0.227	0.151	0.364	0.345	0.333
Sharpen Filtering	0.074	0.000	0.000	0.099	0.066	0.063	0.054	0.111	0.089	0.065

Table 5.5: Removal attack bit error rate for the removal attacks using $\tau = \log(1.5)$.

less as possible. Therefore the scaling attacks and rotation should be applied twice to guarantee that the attacked image returns to its original size. The pre-warping attack only applies the distortion, whereas the Stirmark RBA is combined with geometrical distortion and compression. As it is possible to observe, the pre-warping MRF attack achieves a the most successful rate of detection in these set of attacks. The most effective attack is the Stirmark RBA, that excludes completely the watermark message by destroying around 70% of the watermark message in some cases. The figure Figure 5.3 illustrates the detector responses for the Stirmark RBA with compression. The responses obtained with the pre-warping MRF attack, also illustrated in 5.3b, shows a detector response where the distribution of the '0' responses are clearly separated from the '1' distribution. In contrast to the pre-warping distribution, the Stirmark RBA distribution generates a Gaussian shaped distribution because both '0' and '1' responses overlaps with each other. The same effect of the RBA is verified with the scaling and rotation attacks.

5.3 Results Improvement

The results previously illustrated considers an detection threshold $\tau = \log(1.5)$. The parameter τ implies the existence of a small region R that is given by

$$R =]-\log(\tau), \log(\tau)[$$

Attack name	Tests numbers									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Pre-warping MRF	0.044	0.000	0.000	0.337	0.199	0.141	0.119	0.103	0.048	0.038
Stirmark RBA	0.588	0.574	0.353	0.668	0.650	0.641	0.437	0.585	0.566	0.238
Light rotation	0.515	0.412	0.294	0.551	0.492	0.467	0.327	0.566	0.534	0.506
Scaling	0.588	0.574	0.412	0.486	0.432	0.287	0.270	0.482	0.478	0.457

Table 5.6: Geometrical attack bit error rate for the removal attacks using $\tau = \log(1.5)$.

If the detector outputs a value $V \in R$, the detector shall not output any response, *i.e.* not a '0' nor a '1' will be outputted, but instead the output is the *quote mark* symbol ?. The threshold used on the previous section created an interval $R =]-\log(1.5), \log(1.5)[\Leftrightarrow R =]-0.176, 0.176[$ where no response '0' or '1' was generated. Now, the tests will consider the parameter $\tau' = \log(1.0)$ and the new region R' will be given by $R' =]-0.000, 0.000[$. This means that all the values should be considered (excluding when the detector outputs the 0.0 value). The explanation given to do that is that all the values should be considered because many of the bits embedded should be weakened but they should preserve the embedded value even though the value is too weak to be detected with a larger threshold. The approaches presented in [24, 22] use this detection threshold to detect the watermark message.

The results obtained with the new detection threshold τ' improved significantly compared to the previous value. The table 5.7 illustrated the results where comparatively to the results obtained previously it is possible to notice a better performance for the algorithm. Most of the errors are reduced to a scale of $\pm 5\%$. The most effective attacks still being the median filtering and contrast adjustment. However even these errors present a reduced rate when compared to the previous results.

Attack name	**Tests numbers**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Gaussian blur	0.000	0.000	0.000	0.086	0.054	0.053	0.034	0.061	0.056	0.054
Brightness adjustment	0.000	0.000	0.000	0.011	0.008	0.006	0.003	0.127	0.003	0.000
Contrast adjustment	0.184	0.176	0.118	0.206	0.113	0.111	0.109	0.131	0.102	0.099
Gamma correction	0.118	0.059	0.059	0.146	0.81	0.069	0.068	0.060	0.058	0.046
JPEG $Q = 80$	0.000	0.000	0.000	0.062	0.056	0.050	0.022	0.029	0.016	0.012
Median Filtering	0.176	0.059	0.059	0.250	0.162	0.100	0.090	0.164	0.137	0.125
Sharpen Filtering	0.015	0.000	0.000	0.054	0.045	0.045	0.032	0.042	0.042	0.042

Table 5.7: Removal attack bit error rate for the removal attacks using $\tau = \log(1)$.

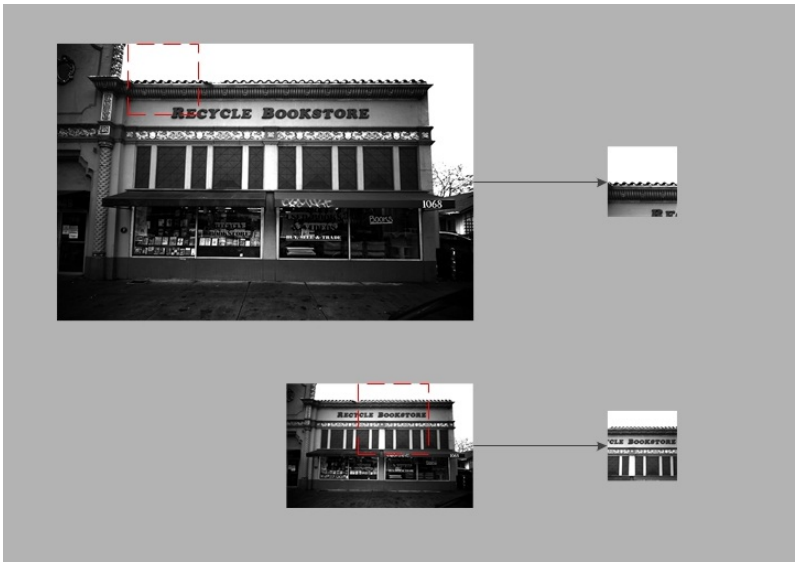
The results for the geometrical attacks improvement were greatly improved as seen in the table 5.7 when compared to the previous results on 5.6. Now the Stirmark RBA can be reduced to a rate of $\pm 15\%$ of error, whereas the pre-warping algorithm can be generally reduced to much smaller error rates. The results presented for the rotation and scaling attacks are much smaller as well.

Attack name	**Tests numbers**									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Pre-warping MRF	0.029	0.000	0.000	0.246	0.121	0.101	0.098	0.053	0.026	0.018
Stirmark RBA	0.353	0.235	0.176	0.454	0.442	0.380	0.243	0.322	0.275	0.137
Light rotation	0.309	0.235	0.118	0.333	0.269	0.266	0.179	0.322	0.249	0.226
Scaling	0.382	0.294	0.253	0.269	0.215	0.140	0.136	0.226	0.182	0.179

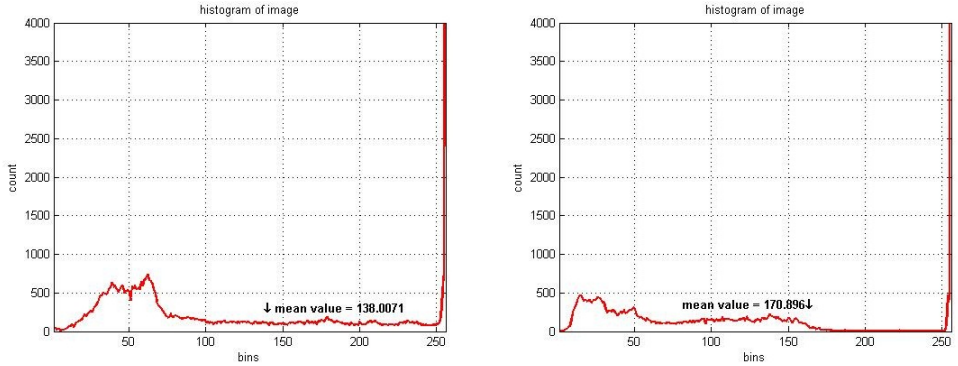
Table 5.8: Geometrical attacks bit error rate for the removal attacks using $\tau = \log(1)$.

With the analysis of the results obtained it is possible to notice that the tests results **8-10** are generally better than the results achieved in **4-6**. The differences on these tests are especially due

to the images used for embedding. While the tests **4-6** uses images of different sizes, including in high resolution, the tests **8-10** employs reduced size images.



(a) The image on bottom was resized to 40% of its original size shown on top. The extracted blocks have the same size N



(b) The two histograms for the blocks in the figure above. The histogram on the left belongs to the smaller image, the one on the right side belongs to the larger image.

Figure 5.1: Influence of using larger images for embedding.

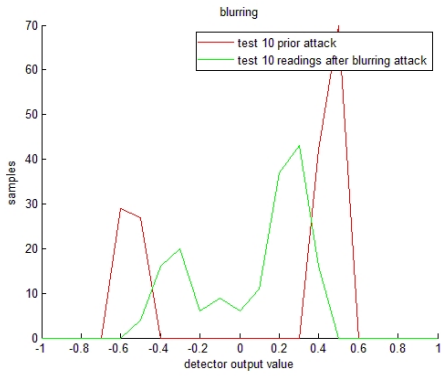
Accordingly to the algorithm presented on chapter 3, a constraint for selecting the blocks was applied in order to exclude blocks that are do not contain much information, *i.e.* mostly constituted by background colors intensities. The smaller the block is, the bigger is the probability of selecting blocks that are prone to be *local backgrounds*. Local backgrounds, was defined as any kind of region that when considered the whole image, its pixels does not represents a background region, but if this same region constitutes the largest part of one block the background effect is created. Therefore, once the proposed approach is based on blocks, the histograms should be evaluated based on its local characteristics.

The figure Figure 3.6 seen before illustrates these cases, of local backgrounds. Now, addressing the problem to the results by analysing the figure Figure 5.1a,both blocks have the same size but the block of the smaller image holds more information than the block from the larger image.Now, by analysing the first level DWT low frequency sub-band histograms of both blocks, illustrated by the figure Figure 5.1b it is possible to notice the difference of both histograms. The histogram belonging to the block of the smaller image (on the left) has a stable neighbourhood around the mean value whereas the histogram from the block of the bigger image (oh the right side) has a more unstable neighbouring region. Both regions possess enough samples to be suitable for embedding, however the difference is that the region of the smaller image is more stable.

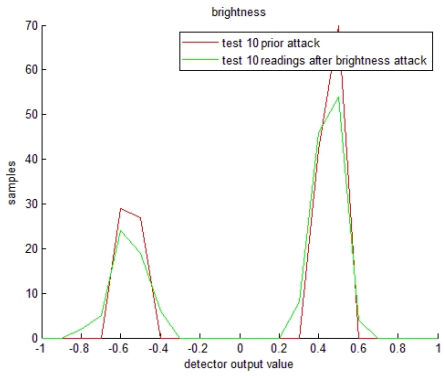
Satisfactory results were achieved when the block size is big enough to hold enough information about the images. However, with the increasing of the block size the number of blocks available is reduced. Recalling the reason why one bit is embedded into each block, was because of the applicability of this approach with the container watermarking strategy. If one bit is embedded into each block, only two pre-watermarked blocks of each block should be stored on the container. As seen before that with the increasing of the number of bits the number of blocks could increase significantly. Another issue that should be considered is the blocks size and the remaining region. Due to the requirement for block size to be of a standard size N, several images might have some regions discarded or in some cases, when the block size exceeds the image size, the whole image can be discarded.

The increasing of the threshold T provides a larger resistance against most of the attacks and the threshold can be augmented to the infinity. However, the probability of appearing visible artifacts in the watermarked image increases. The reason is because a large number of coefficients might be addressed from one region to another.

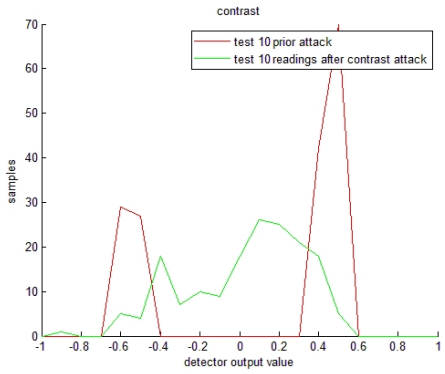
The main scope for this approach is to employ the watermarking scheme into the image-set fingerprinting. It has been discussed before the large size of the fingerprinting codes usually constitutes a barrier to its application into the image domain. However, with this thesis it has been possible to see that if the set is large enough, a larger number of bits can be embedded. The table 2.2 shows that the size of the sets for video-games present a large number of textures which makes, video-games a suitable candidate for this algorithm. This algorithm presents another improvement relatively to the approaches presented in [24, 22] that is the preservation of the mean value. These approaches requires that the content owner embed the same number of '0' and '1' in order to preserve the mean value, however, the proposed approach can be extended to be applied to embed several bits regardless of the sequence without modifying the mean value.



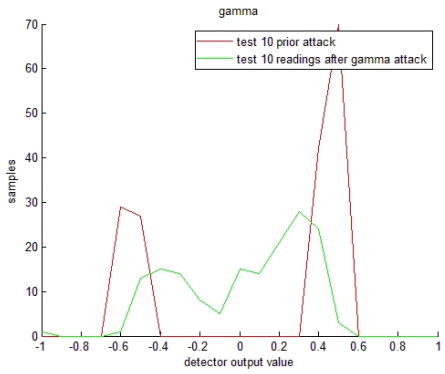
(a) Gaussian blur attack distributions for the detector responses



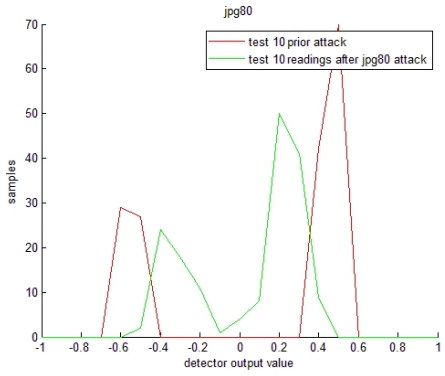
(b) Brightness adjustment attack distributions for the detector responses



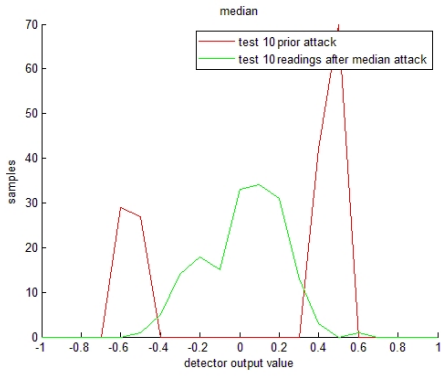
(c) Contrasts adjustment attack distribution for the detector responses



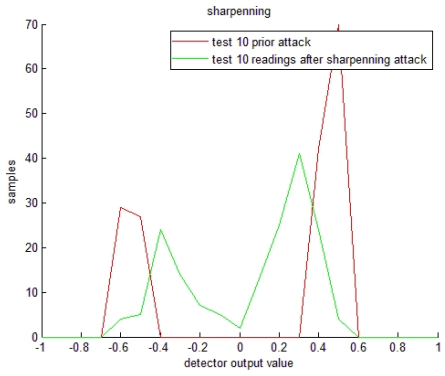
(d) Gamma correction attack distribution for the detector responses



(e) JPEG compression attack distribution for the detector responses

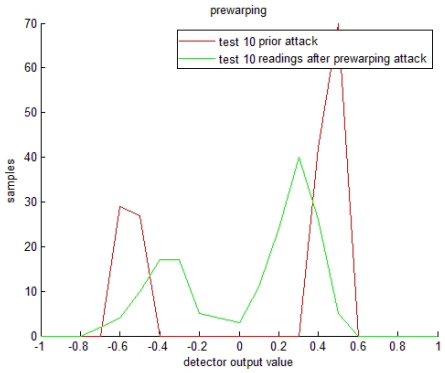


(f) Median attack distribution for the detector responses

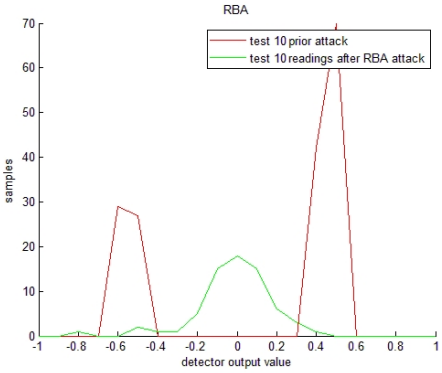


(g) Sharpen filtering distribution for the detector responses

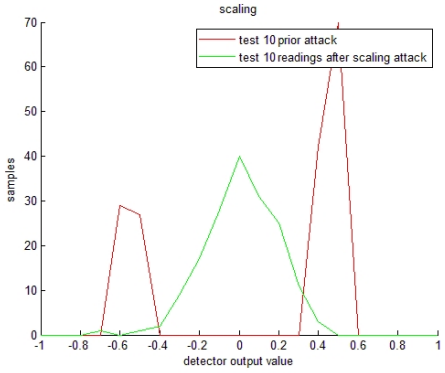
Figure 5.2: fig:Detector responses for the removal attacks on the test 10.



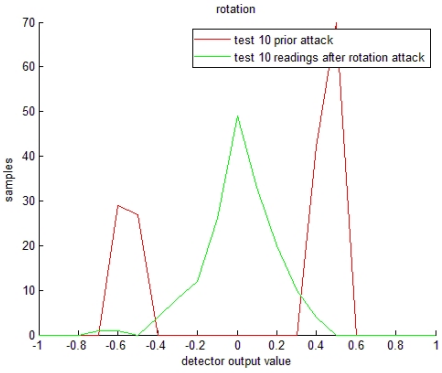
(a) Pre-warping attack distribution



(b) Stirmark random bending attack distribution



(c) Scaling attack distribution



(d) Rotation attack distribution

Figure 5.3: fig:Detector responses for the geometrical attacks on the test 10.

Chapter 6

Conclusion and Future work

As concluding remarks, the main outcome for this thesis was the introduction of the first watermarking approach for sets of images. One of the main challenges of the scientific community nowadays regarding fingerprinting codes is to reduce the size of these codes for larger collusions in order to turn the fingerprinting codes applicable for images. Notwithstanding, the current length capable to be achieved for fingerprinting codes still being able to be used when these fingerprints are meant to be integrated in sets of images. Therefore, this approach presents itself as an ideal candidate to be applied in contexts where the cover work consists in more than one image, such as video games textures, e-books for children and digital comics.

The results presented in Table 5.7 and Table 5.8 show that when the selection of the blocks holds enough information, the embedding achieves large robustness to the most common attacks and some of the geometrical attacks for a detection threshold of $\tau = 1$. Nevertheless, there are several parameters that are yet to be optimized such as the decision parameters to check if the blocks are suitable for embedding, the optimal size N for the blocks and the embedding strength T. The results show that, as expected, a larger embedding strength T results in more robust watermarks, however it is yet to decide the maximum ratio to be used for this strength because visual artifacts appear with the increasing of the strength. In order to optimize the algorithm, decision making methods can be applied in a future improvement, such as the gradient descent. The utilization of a sliding window for the block N and the method of the gradient descent could be a good strategy to find a proper position in the image for the embedding positions. Other alternative methods to be exploited are the utilization of the image processing techniques in order to find regions of the images more suitable for embedding and setting be block in these positions.

The size of the set of images used for the testset is much more reduced than the number of images that should be available for a video game scenario. Therefore, for a video game application the number of blocks and consequentially the payload would increase drastically compared to the total of blocks seen on 5.4 allowing the embedding of much larger fingerprinting codes.

This approach also inflicts less influence in the mean value $\overline{V}$ for the watermarked image when compared to the approaches [24, 22]. This means that the probability of wrongly selecting the mean value is decreased when the flooring function is used for selecting the centre bins B_1 and

B_2.

The results show also that a viable solution to achieve larger robustness would be to sort the images based on their resolutions into k different levels. It has been seen in 5.1a that the robustness of the algorithm does not lie only on the size of the blocks, *i.e.* number of samples. A higher robustness is achieved when the histogram is more stable. This means that even having large blocks in high resolution images does not ensure a robust embedding if the histogram is not stable enough around the mean value.

The current approach proposes embedding only one bit into each block. In this way, by using the container strategy the container total size would be at most the double of the total data of all the images of the set. If the number of bits embedded per block is increased for 2 instead of only 1, the new size for the container would be at most four times the original size of the data. Despite the increasing of the container size, the fingerprint length would be of the double compared to the length proposed for this approach. The increasing of the container can applied in case the number of images available is smaller. For example, e-books for children and digital comics usually contains many images but not as many as video-games. Therefore, for these applications increasing the size of the container could be a reasonably strategy for future applications.

Appendix A

In this annex the tables with the information of the images used for the algorithm tests are presented. The tables Table A.1, Table A.2 and Table A.3 represent three different sets used for the tests. The table Table A.1 consists only in small sized images and is used for the tests 1 to 3. The table Table A.2 uses the same images from Table A.1 in addition to high resolution images and is used for the tests 4 to 7. Lastly, the table A.3 consists in the same images from A.2, but the high resolution images are resized 40% of their original size and some other images are added.

number of test	Image size in Mega pixels	number of images
1-3	1.048	1
	0.262	13
	0.066	4

Table A.1: Set of Images used for the tests 1-3

number of test	Image size in Mega pixels	number of images
4-7	14.033	2
	12.000	2
	8.958	1
	8.872	1
	7.990	2
	6.016	1
	5.947	2
	5.252	1
	4.9152	15
	3.871	3
	3.773	1
	3.763	1
	3.508	1
	3.146	6
	1.920	7
	1.229	1
	1.048	1
	0.262	13
	0.066	4

Table A.2: Set of Images used for the tests 4-7

number of test	Image size in Mega pixels	number of images
8-10	2.841	2
	2.430	2
	2.401	1
	2.373	1
	2.340	1
	2.021	1
	1.813	1
	1.796	1
	1.619	7
	1.219	4
	1.204	2
	1.064	1
	1.048	1
	0.995	15
	0.784	6
	0.764	1
	0.762	1
	0.711	1
	0.637	6
	0.389	7
	0.262	13
	0.249	1
	0.066	4

Table A.3: Set of Images used for the tests 8-10

Appendix B

In this annex some figures used for this thesis writing are illustrated. Some of the images were renamed, all images were resized for displaying.

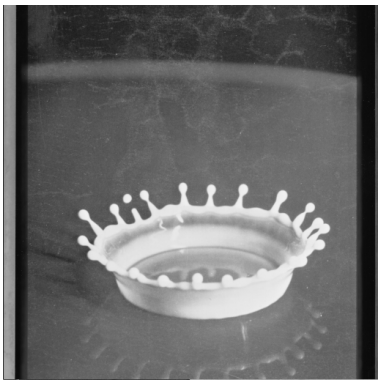


Figure B.1: Drop.bmp

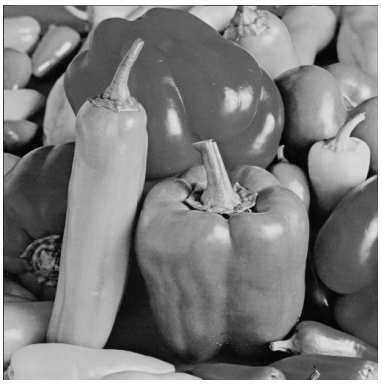


Figure B.2: peppers.bmp

Figure B.3: Lena.bmp

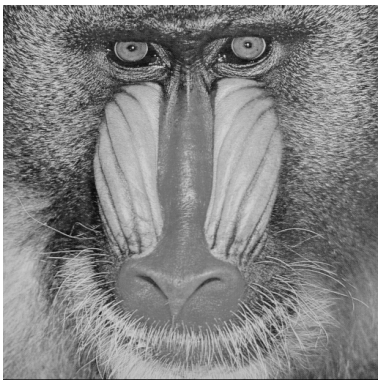


Figure B.4: Baboon.bmp

Figure B.5: plane.bmp

References

[1] Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors, 1994.

[2] Marcel Schäfer, Waldemar Berchtold, Sascha Zmudzinski, and Martin Steinebach. Zero false positive 2-secure fingerprinting watermarking based on combining hamming distance conditions and parent pair search. In *Proceedings of the 12th ACM workshop on Multimedia and security*, MM&Sec '10, pages 169–174, New York, NY, USA, 2010. ACM.

[3] Gábor Tardos. Capacity of collusion secure fingerprinting: a tradeoff between rate and efficiency, 2010.

[4] Martin Steinebach, Enrico Hauer, and Patrick Wolf. Efficient watermarking strategies. In *Proceedings of the Third International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution*, pages 65–71, Washington, DC, USA, 2007. IEEE Computer Society.

[5] G. C. Langelaar, I. Setyawan, and R. L. Lagendijk. Watermarking digital image and video data. a state-of-the-art overview. *Signal Processing Magazine, IEEE*, 17(5):20–46, 2000.

[6] I. J. Cox, M. L. Miller, and J. A. Bloom. Watermarking applications and their properties. In *Information Technology: Coding and Computing, 2000. Proceedings. International Conference on*, pages 6–10.

[7] A. Kerckhoffs. "la cryptographie militaire". *J. Sci. Militaires*, 9:5–38, Jan. 1883.

[8] Ingemar Cox, Matthew L. Miller, and Jeffery A. Bloom. *Digital watermarking*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[9] J.J.K.O. Ruanaidh, W.J. Dowling, and F.M. Boland. Phase watermarking of digital images. In *Image Processing, 1996. Proceedings., International Conference on*, volume 3, pages 239 –242 vol.3, sep 1996.

[10] Joseph J.K. ÓRuanaidh and Thierry Pun. Rotation, scale and translation invariant spread spectrum digital image watermarking. *Signal Process.*, 66(3):303–317, May 1998.

[11] Shumei Wang, Yong Fan, and Ping Yu. A watermarking algorithm of gray image based on histogram. In *Image and Signal Processing, 2009. CISP '09. 2nd International Congress on*, pages 1 –5, oct. 2009.

[12] Li Xin-Wei, Guo Bao-Long, Li Lei-Da, and Shen Hong-Xin. A new histogram based image watermarking scheme resisting geometric attacks. In *Information Assurance and Security, 2009. IAS '09. Fifth International Conference on*, volume 1, pages 239 –242, aug. 2009.

[13] Technische Informatik Und, Germano Caronni, H. H. Brüggemann, and W. Gerhardt-häckl. Assuring ownership rights for digital images, 1995.

[14] I. Pitas. A method for signature casting on digital images. In *Image Processing, 1996. Proceedings., International Conference on*, volume 3, pages 215–218 vol.3.

[15] Joshua R. Smith and Barrett O. Comiskey. Modulation and information hiding in images. pages 207–226. Springer-Verlag, 1996.

[16] I.J. Cox, J. Kilian, F.T. Leighton, and T. Shamoon. Secure spread spectrum watermarking for multimedia. *Image Processing, IEEE Transactions on*, 6(12):1673 –1687, dec 1997.

[17] Y.J. Song and T.N. Tan. Comparison of four different digital watermarking techniques. In *Signal Processing Proceedings, 2000. WCCC-ICSP 2000. 5th International Conference on*, volume 2, pages 946 –950 vol.2, 2000.

[18] M. Hayes. The reconstruction of a multidimensional sequence from the phase or magnitude of its fourier transform. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 30(2):140 – 154, apr 1982.

[19] Xiu mei Wen, Wei Zhao, and Fan xing Meng. Research of a digital image watermarking algorithm resisting geometrical attacks in fourier domain. In *Computational Intelligence and Security, 2009. CIS '09. International Conference on*, volume 2, pages 265 –268, dec. 2009.

[20] M. Barni, F. Bartolini, V. Cappellini, A. Lippi, and A. Piva. A dwt-based technique for spatio-frequency masking of digital signatures, 1999.

[21] Xiangui Kang, Jiwu Huang, Yun Q Shi, and Yan Lin. A dwt-dft composite watermarking scheme robust to both affine transform and jpeg compression. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(8):776 – 786, aug. 2003.

[22] Shijun Xiang and Hyoung-Joong Kim. Geometrically invariant image watermarking in the dwt domain. In *Proceedings of the 8th international conference on Information security applications*, WISA'07, pages 76–90, Berlin, Heidelberg, 2007. Springer-Verlag.

[23] Shijun Xiang. Histogram-based perceptual image hashing in the dwt domain. In *Multimedia Information Networking and Security (MINES), 2010 International Conference on*, pages 653 –657, nov. 2010.

[24] Liyun Wang, Hefei Ling, Fuhao zou, and Zhengding Lu. Real-time compressed- domain video watermarking resistance to geometric distortions. *IEEE MultiMedia*, 19(1):70–79, January 2012.

[25] Ilaria Venturini. Counteracting oracle attacks. In *Proceedings of the 2004 workshop on Multimedia and security*, MM&Sec '04, pages 187–192, New York, NY, USA, 2004. ACM.

[26] Martin Kutter, Sviatoslav Voloshynovskiy, and Alexander Herrigel. The watermark copy attack. pages 371–380, 2000.

[27] Fabien A. P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn. Attacks on copyright marking systems. In *Proceedings of the Second International Workshop on Information Hiding*, pages 218–238, London, UK, 1998. Springer-Verlag.

[28] Michele Fenzi, Huajian Liu, Martin Steinebach, and Roberto Caldelli. Markov random fields pre-warping to prevent collusion in image transaction watermarking. In *Proceeding of the Seventh IASTED International Conference on Signal Processing, Pattern Recognition and Applications (SPPRA 2010), Innsbruck, Austria*. ACTA Press, Feb 2010. (Best Paper Award winner).

[29] Scott Craver and Nasir Memon. On the invertibility of invisible watermarking techniques, 1997.

[30] Ton Kalker. Watermark estimation through detector observations. In *in Proceedings IEEE Benelux Signal Processing Symposium 98*, pages 119–122, 1998.

[31] Ingemar J. Cox and Jean paul M. G. Linnartz. Public watermarks and resistance to tampering. In *In International Conference on Image Processing (ICIP'97*, pages 3–6. IEEE, 1997.

[32] Jean paul M. G. Linnartz and Marten Van Dijk. Analysis of the sensitivity attack against electronic watermarks in images. pages 258–272. Springer Verlag, 1998.

[33] S. Voloshynovskiy, S. Pereira, T. Pun, J. J. Eggers, and J. K. Su. Attacks on digital watermarks: Classification, estimation-based attacks, and benchmarks. *IEEE Communications Magazine*, 39:118–126, 2001.

[34] D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *Information Theory, IEEE Transactions on*, 44(5):1897–1905, 1998.

[35] B. Skoric, T. U. Vladimirova, M. Celik, and J. C. Talstra. Tardos fingerprinting is better than we thought. *IEEE Transactions on Information Theory*, 54(8):3663–76, 2008. 10115028 Tardos fingerprinting collusion attacks Tardo hard-coded constants false positive false negative error rates statistical properties reasonable assumptions Gaussian-distributed stochastic variables.

[36] Gabor Tardos. Optimal probabilistic fingerprint codes. In *In 35th ACM STOC*, pages 116–125. ACM Press, 2003.

[37] Shijun Xiang, Hyoung-Joong Kim, and Jiwu Huang. Histogram-based image hashing scheme robust against geometric deformations. In *Proceedings of the 9th workshop on Multimedia & security*, MM&Sec '07, pages 121–128, New York, NY, USA, 2007. ACM.

[38] Martin Steinebach, Huajian Liu, and York Yannikos. Forbild: efficient robust image hashing. volume 8303, page 83030O. SPIE, 2012.

[39] Nazim Fates and Fabien A. P. Petitcolas. Watermarking scheme evaluation tool. In *Proceedings of the 2000 International Conference on Microelectronic Systems Education*, MSE '00, pages 328–, Washington, DC, USA, 2000. IEEE Computer Society.