SCIENCES

IMAGE

Compression, Coding and Protection of Images and Videos

Multimedia Security 1

Authentication and Data Hiding

Coordinated by
William Puech

iSTE

WILEY

Multimedia Security 1

SCIENCES

Image, Field Director – Laure Blanc-Feraud

Compression, Coding and Protection of Images and Videos,
Subject Head – Christine Guillemot

Multimedia Security 1

Authentication and Data Hiding

Coordinated by
William Puech

ISTE WILEY

First published 2022 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc.

ISTE Ltd
27-37 St George's Road
London SW19 4EU
UK

www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.wiley.com

Library of Congress Control Number: 2021948467

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN 978-1-78945-026-2

ERC code:
PE6 Computer Science and Informatics
PE6_5 Cryptology, security, privacy, quantum cryptography
PE6_8 Computer graphics, computer vision, multi media, computer games

Contents

Foreword by Gildas Avoine

Gildas AVOINE
Director of the CNRS Computer Security Research Network, INSA Rennes, University of Rennes, IRISA, CNRS, France

French academic and industrial research in cybersecurity is at the forefront of the international scene. While France cannot claim to have sovereignty over cybersecurity technologies, it undeniably possesses a wealth of skills, as French expertise covers all areas of cybersecurity.

Research in cryptography illustrates French excellence, but it should not overshadow other domains where French influence is just as remarkable, including formal methods for security, protection of privacy, security of systems, software and networks, security of hardware systems and multimedia data security, according to the classification proposed by the CNRS Computer Security Research Network (GdR).

The security of multimedia data is covered in this book. The evolution of our society from the written word to sound and image, with the notable arrival of the mobile phone and the democratization of the Internet has brought about new security needs. These are only the beginning of the transformation of our society, and the recent deployment of videoconferencing shows that research into the security of multimedia data is constantly confronted with new scientific challenges.

The complexity of the subject and its multidisciplinary dimension, which primarily combines signal processing and cryptography, are perfectly illustrated by the variety of

Multimedia Security 1,
coordinated by William PUECH.

subjects detailed throughout this book. The chapters thus reveal the scientific obstacles to be dealt with by the community, by anchoring them in real scenarios, such as the fraudulent copying of films, the deception of artificial intelligence or the spreading of doctored images on social media.

This book, made up of two volumes, is thus promised to become a reference in the field of multimedia data security, an introduction that is both exhaustive and in-depth that students, engineers and researchers will be able to appreciate through more than 600 pages enriched with numerous references. Everyone can indulge in their favorite kind of reading, whether linear or random.

Finally, I would like to thank all of the authors for their commitment to supporting the scientific community, and I would particularly like to thank William Puech for editing this edition of the book. William, alongside Patrick Bas and then Caroline Fontaine, is responsible for the theme of multimedia data security within the Computer Security GdR, thus allowing the entire cybersecurity community to better understand this fascinating subject.

Happy reading!

Foreword by Cédric Richard

Cédric RICHARD
Director of the CNRS GdR ISIS, Côte d'Azur Observatory,
University of Côte d'Azur, Nice, France

With the relentless increase in bandwidth and storage space, as well as the proliferation of mobile devices and the development of new standards, multimedia data is affecting our societies by changing the way that we access data and information. It is also changing our relationship to culture, by transforming interactions between individuals and their relationships with organizations. Multimedia activities are present in all major sectors of activity (security, health, telecommunications, etc.) and have supported their successive developments because of the common backbone they build, from information support to the application and user.

In this context, by protecting confidentiality and copyright, verifying integrity, analyzing and authenticating content, tracing copies and controlling access, particularly critical questions about multimedia data security are being asked. For example, the protection strategies implemented must take into account the specific needs of multimedia while meeting the requirements of the means of communication, thus establishing a compromise. A wrong approach can indeed lead to excessive coding of the data, or the alteration of their perceptual quality, and thus failure in the targeted security objectives.

As an interface discipline, the art of multimedia security is difficult!

Multimedia Security 1,
coordinated by William PUECH.

However, with this two-part book, William Puech and his co-authors take up the challenge brilliantly by painting an exhaustive and current panorama of multimedia security. They offer an in-depth analysis of authentication and hidden data embedding methods, biometric technologies and multimedia protection and encryption processes. Without giving in to an outdated formalism that could hinder the fluidity of their presentations, the authors captivate the reader by presenting the state of the art of each subject directly and in an illustrative way.

William Puech and the contributors to this book have provided considerable work to their French-speaking scientific communities of information, signal, image, vision and computer security, represented by the two appropriate French GdR groups of the CNRS. I would like to express all of my gratitude to them.

Preface

William PUECH
LIRMM, Université de Montpellier, CNRS, France

Nowadays, more than 80% of transmitted data on social media and archived in our computers, tablets, mobile phones or in the cloud is multimedia data. This multimedia data mainly includes images (photographs, computer-generated images), videos (films, animations) or sound (music, podcasts), but equally more and more three-dimensional (3D) data and scenes, for applications ranging from video games to medical data, passing through computer-aided design, video surveillance and biometrics. It is becoming necessary, urgent, not to say vital, to secure this multimedia data during its transmission or archiving, but also during its visualization. In fact, with everything digital, it is becoming increasingly easy to copy this multimedia data, to view it without rights, to appropriate it, but also to counterfeit it.

Over the last 30 years, we have observed an expansive development around multimedia security, both internationally and in France. In fact, at the French level, there are dozens of research teams in laboratories, but also a large number of industrials, who are focusing their activities on these aspects. This activity can also be found in several GdR (research groups) of the CNRS, but in particular the GdR ISIS (information, signal, image and vision) and the GdR computer security.

Multimedia security is a relatively new theme, as evidenced by the publication dates of the articles referenced in the various chapters of these two volumes. In fact, out of about 900 references, nearly 50% of them are less than 10 years old, and more than 35% are between 10 and 20 years old. Of course, let us not forget certain authors, such as Auguste Kerckhoffs (1835–1903) and Claude Shannon

Multimedia Security 1,
coordinated by William PUECH.

(1916–2001), without whom our community would not have advanced in the same way. The history of multimedia security really begins at the end of the 1990s, with the beginning of watermarking, steganography, but in a very timid manner, this being motivated by the digitization of content and the protection of rights holders. In 2001, motivated by the attack of September 11, research in steganalysis hidden signal detection and statistical detection became the top priority. Between 2000 and 2010, there was an international explosion in watermarking security. There were also major contributions in steganography and steganalysis. During this same decade, research into securing multimedia data by specific encryption was born with the aspects of selective or partial encryption and crypto-compression, while guaranteeing the preservation of international formats and standards. From 2010, new facets of multimedia data security have emerged with *forensics* aspects, as well as statistical approaches. There has also been a strong development in signal processing in the encrypted domain, as well as the tracing of traitors. In 2020, research in forensics and steganalysis has been gaining momentum, in particular with the emergence of machine learning, and especially with the exploitation and development of deep convolutional neural networks. The recent advances in this field have varied greatly, from steganography (GAN), adversarial methods, methods by content generation, to the processing of encrypted content, including the links between learning and information leakage, applications in biometrics and "real-life" content analysis.

This project of works began more than two years ago and has really meant a lot to me. In fact, at the French level, we have a certain strength in this field, and numerous gems that we have brought to light. Nothing could have been achieved without the support of the GdR ISIS and GdR computer security. It is largely because of these GdR that we have succeeded in tracking research activities in the field of multimedia security from a French point of view. The towns represented in these two works illustrate the richness and national diversity (Caen, Grenoble, La Rochelle, Lille, Limoges, Lyon, Montpellier, Paris, Poitiers, Rennes, Saint-Étienne and Troyes), because some of these cities, as we will see during our reading, are represented by several laboratories and/or universities.

As we will be able to see throughout these two volumes, even if they are grouped around multimedia security, the research themes are very broad and the applications varied. In addition, the fields cover a broad spectrum, from signal processing to cryptography, including image processing, information theory, encoding and compression. Many of the topics in multimedia security are a game of cat and mouse, where the defender of rights must regularly transform into a counter-attacker in order to resist the attacker.

The first volume primarily focuses around the authentication of multimedia data, codes and the embedding of hidden data, from the side of the defender as well as the attacker. Concerning the embedding of hidden data, it also addresses the aspects of invisibility, color, tracing and 3D data, as well as the detection of hidden messages in

images by steganalysis. The second volume mainly focuses on the biometrics, protection, integrity and encryption of multimedia data. It covers aspects such as image and video crypto-compression, homomorphic encryption, the embedding of hidden data in the encrypted domain, as well as the sharing of secrets. I invite the reader, whether they are a student, teacher, researcher or industrial to immerse themselves in these works, not necessarily by following the intended order, but going from one chapter to another, as well as from one volume to another.

These two volumes, even though they cover a broad spectrum in multimedia security, are not meant to be exhaustive. I think, and hope, that a third volume will complete these first two. In fact, I am thinking of sound (music and speech), video surveillance/video protection, camera authentication, privacy protection, as well as the attacks and counter-attacks that we see every day.

I would like to thank all of the authors, chapter managers, their co-authors, their collaborators and their teams for all of their hard work. I am very sorry that I have had to ask them many times to find the best compromises between timing, content and length of the chapters. Thank you to Jean-Michel, Laurent, Philippe ($\times 2$), Patrick ($\times 2$), Teddy, Sébastien ($\times 2$), Christophe, Iuliia, Petra, Vincent, Wassim, Caroline and Pauline! Thank you all for your openness and good humor! I would also thank the GdR ISIS and computer Security through Gildas and Cédric, but also Christine and Laure for their proofreading, as well as for establishing a connection with ISTE Ltd. I would also like to thank all of the close collaborators with whom I have worked for more than 25 years on the various themes that I have had the chance to address. PhD students, engineers, interns and colleagues, all of them will recognize themselves, whether they are in my research team (ICAR team) or in my research laboratory (LIRMM, Université de Montpellier, CNRS).

In particular, I would like to thank Vincent, Iuliia, Sébastien and Pauline for having accepted to embark on this adventure. Pauline, in addition to writing certain chapters, has been a tremendous collaborator for the advancement of this book. All of those responsible for the chapters have seen that, Pauline has been my shadow over the past two years, to ensure that these two works could see the light of day in 2021. Thank you Pauline! To conclude, I would like to warmly thank all of the members of my family, and in particular Magali and our three children, Carla, Loriane and Julian, whom I love very much and who have constantly supported me.

November 2021

1

How to Reconstruct the History of a Digital Image, and of Its Alterations

Quentin BAMMEY[1], Miguel COLOM[1], Thibaud EHRET[1], Marina GARDELLA[1], Rafael GROMPONE[1], Jean-Michel MOREL[1], Tina NIKOUKHAH[1] and Denis PERRAUD[2]

[1] *Centre Borelli, ENS Paris-Saclay, University of Paris-Saclay, CNRS, Gif-sur-Yvette, France*

[2] *Technical and Scientific Police, Central Directorate of the Judicial Police, Lyon, France*

Between its raw acquisition from a camera sensor and its storage, an image undergoes a series of operations: denoising, demosaicing, white balance, gamma correction and compression. These operations produce artifacts in the final image, often imperceptible to the naked eye but yet detectable. By analyzing those artifacts, it is possible to reconstruct the history of an image. Indeed, one can model the different operations that took place during the creation of the image, as well as their order and parameters.

Information about the specific camera pipeline of an image is relevant by itself, in particular because it can guide the restoration of the image. More importantly, it provides an identifying signature of the image. A model of the pipeline that is inconsistent across the whole image is often a clue that the image has been tampered with.

For a color version of all figures in this chapter, see www.iste.co.uk/puech/multimedia1.zip.

Multimedia Security 1,

coordinated by William PUECH.

Multimedia Security 1: Authentication and Data Hiding,
First Edition. William Puech.

However, the traces left by each step can be altered or even erased by subsequent processing operations. Sometimes these traces are even maliciously masked to make a forged image seem authentic to forensic tools. While it may be easy to deliberately hide the artifacts linked to a step in the processing of the image, it is more difficult to simultaneously hide several of the artifacts from the entire processing chain. It is therefore important to have enough different tests available, each of them focused on different artifacts, in order to find falsifications.

We will therefore review the operations undergone by the raw image, and describe the artifacts they leave in the final image. For each of these operations, we will discuss how to model them to detect the significant anomalies caused by a possible manipulation of the image.

1.1. Introduction

1.1.1. *General context*

The Internet, digital media, new means of communication and social networks have accelerated the emergence of a connected world where perfect mastery over information becomes utopian. Images are ubiquitous and therefore have become an essential part of the news. Unfortunately, they have also become a tool of disinformation aimed at distracting the public from reality.

Manipulation of images happens everywhere. Simply removing red eyes from family photos could already be called an image manipulation, whereas it is simply aimed at making an image taken with the flash on look more natural. Even amateur photographers can easily erase the electric cables from a vacation panorama and correct physical imperfections such as wrinkles on a face, not to mention the touch-ups done on models in magazines.

Beyond these mostly benign examples, image manipulation can lead to falsified results in scientific publications, reports or journalistic articles. Altered images can imply an altered meaning, and can thus be used as fake evidence, for instance to use as defamation against someone or report a paranormal phenomenon. More frequently, falsified images are published and relayed on social media, in order to create and contribute to the spread of fake news.

The proliferation of consumer software tools and their ease of use have made image manipulation extremely easy and accessible. Some software even go as far as to automatically restore a natural look to an image when parts of it have been altered or deleted. Recently, deep neural networks have made it possible to generate manipulated images almost automatically. One example is the site This Person Does

Not Exist[1], which randomly generates faces of people who do not exist while being unexpectedly realistic. The most surprising application is undoubtedly the arrival of deepfake methods, which allow, among other things, a face in a video to be replaced with the one of another person (face swapping).

1.1.2. *Criminal background*

These new possibilities of image manipulation have been exploited for a long time by governments, criminal organizations and offenders. Stalinist propaganda images can come to mind, in which certain characters who had become undesirable were removed from official photographs (Figure 1.1).

Figure 1.1. *An example showing how an image has been modified several times in a row, each person who had lost favor seeing their image removed from the photo. Only Joseph Stalin appears in all four photos*

Today, image manipulation can serve the interests of criminal or terrorist organizations as part of their propaganda (false claims, false events, masking of identification elements, addition of objects). Face swapping and deepfake techniques are also a simple way to undermine the image and privacy of public figures by placing them in compromising photos. The manipulation of images is also a means of exerting coercion, pressure or blackmail against a third party. These new image

1 Available at: www.thispersondoesnotexist.com.

manipulation techniques are also used by pedophiles to generate photographs that satisfy their fantasies. Manipulated images can also be used to cause economic harm to companies through disinformation campaigns. Administrative documents can be falsified in order to obtain official papers, a rental document or a loan from specialized organizations. Face morphing, whose objective is to obtain the photo of a visually "compatible" face from two faces, enables two users to share the same ID in order to deceive an identity check.

1.1.3. *Issues for law enforcement*

In the past, confessions, testimonies or photographs were enough to prove guilt. Technologies were not sufficiently developed to mislead investigators. Today, these methods are no longer sufficient and law enforcement authorities need innovative scientific tools to be able to present reliable evidence in court. As technology evolves rapidly, law enforcement agencies must continuously ensure scientific monitoring in order to keep up with the state-of-the-art technology, to anticipate and to have the most recent tools available to detect manipulation and other forms of cheating for malicious purposes. It is essential to maintain a high level of training for the experts responsible for authenticating the images. In fact, the role of the police, and in particular of the technical and scientific police, is to highlight any falsification in order to allow perpetrators to be sentenced, but also to exonerate the persons under judicial enquiry if they are innocent or if their crime cannot be proven. The role of the expert in image authentication is to detect any form of manipulation, rigging or editing aimed at distorting reality. They must be able to answer the following questions:

– Is the image real?

– Does it represent the real scene?

– What is the history of the image and its possible manipulations?

– What is the manipulated part?

– Has the image come from the device that supposedly took it?

In general, it is easier to conclude that an image is falsified than to say it is authentic. Detecting manipulation traces is getting harder over time, as new forgery methods are being developed. As a result, not finding any forgery traces does not prove the image's authenticity. The level of expertise of the forger should also be taken into account. In fact, the possible traces of manipulation will not be the same depending on whether the author is a neophyte, a seasoned photographer or a special effects professional. The author can also use so-called anti-forensic techniques aimed at masking traces of manipulation so that they become undetectable by experts; it is up to the expert to know these techniques and their weaknesses.

1.1.4. *Current methods and tools of law enforcement*

As technologies evolve over time, detection tools must also adapt. Particularly during the transition from film photography to digital images, the authentication methods that were mainly based on semantic analysis of the scene (visual analysis of defects, consistency of shadows and lighting, vanishing points) have been completed through structural and statistical analyses.

To date, available commercial tools are not helpful to the authentication of an image. Most of the time, experts need to design their own tools. This raises the concern of deciding on what ground results from such tools should be accepted as evidence in court. In order to compensate for this lack of objective and precise tools, the police recruits trainees, who participate in national projects (DEFALS challenge funded by the DGA and the French National Research Agency) or international projects (H2020 projects of the European Commission). The objective is to involve university researchers as well as industrialists and practitioners (forensic experts). In addition, experts develop good practice guides such as the "Best Image Authentication Practice Manual" within the framework of the ENFSI[2] to standardize and formalize analysis methodologies.

Digital images are an essential medium of communication in today's world. People need to be able to trust this method of communication. Therefore, it is essential that news agencies, governments and law enforcement maintain and preserve trust in this essential technology.

1.1.5. *Outline of this chapter*

Our objective is to recognize each step of the production chain of an image. This information can sometimes appear in the data accompanying the image, called EXIF (Exchangeable Image File Format), which also includes information such as the brand and model of the camera and lens, the time and location of the photograph, and its shooting settings. However, this information can be easily modified, and is often automatically deleted by social media for privacy reasons. Therefore, we are interested in the information left by the operations on the image itself rather than in the metadata. Some methods, like the one presented in Huh *et al*. (2018), offer to check the consistency of the data present in the image with its EXIF metadata.

Knowledge of the image production chain allows for the detection of changes.

A first application is the authentication of the camera model. The processing chain is specific to each device model; so it is possible to determine the device model by

2 ENFSI: European Network of Forensic Science Institutes.

identifying the processing chain, as implemented in Gloe (2012) where features are used to classify photographs according to their source device. More recently, Agarwal and Farid (2017) showed that even steps common to many devices, such as JPEG compression, sometimes have implementation differences that allow us to differentiate models from multiple manufacturers, or even models from the same manufacturer.

Another application is the detection of suspicious regions in an image, based on the study of the residue – sometimes called noise – left by the processing chain. This residue is constituted of all the traces left by each operation. While it is often difficult, or even impossible, to distinguish each step in the processing chain individually, it is easier to distinguish two different processing chains as a whole. Using this idea, Cozzolino and Verdoliva proposed to use steganography tools (see Chapter 5 entitled "Steganography: Embedding data into Multimedia Content") to extract the image residue (Cozzolino *et al*. 2015b). Treating this residue as a piece of hidden information in the image, an algorithm such as Expectation–Maximization (EM) is then used to classify the different regions of the image. Subsequently, neural networks have shown good performance in extracting the residue automatically (Cozzolino and Verdoliva 2020; Ghosh *et al*. 2019), or even in carrying out the classification themselves (Zhou *et al*. 2018).

The outline of this chapter arises from previous considerations. Section 1.2 describes the main operations of the image processing chain.

Section 1.3 is dedicated to the effect each step of the image processing pipeline has on the image's noise. This section illustrates how and why the fine analysis of noise enables the reverse engineering of the image and leads to the detection of falsified areas because of the discrepancies in the noise model.

We then detail the two main operations that lead to the final coding of the image. Section 1.4 explains how demosaicing traces can be detected and analyzed to detect suspicious areas of an image. Section 1.5 describes JPEG encoding, which is usually the last step in image formation, and the one that leaves the most traces. Similarly to demosaicing, we show how the JPEG encoding of an image can be reverse-engineered to understand its parameters and detect anomalies. The most typical cases are cropping and local manipulations, such as internal or external copy and paste.

Section 1.6 specifically addresses the detection of internal copy-move, a common type of manipulation. Finally, section 1.7 discusses neural-network-based methods, often efficient but at the cost of interpretability.

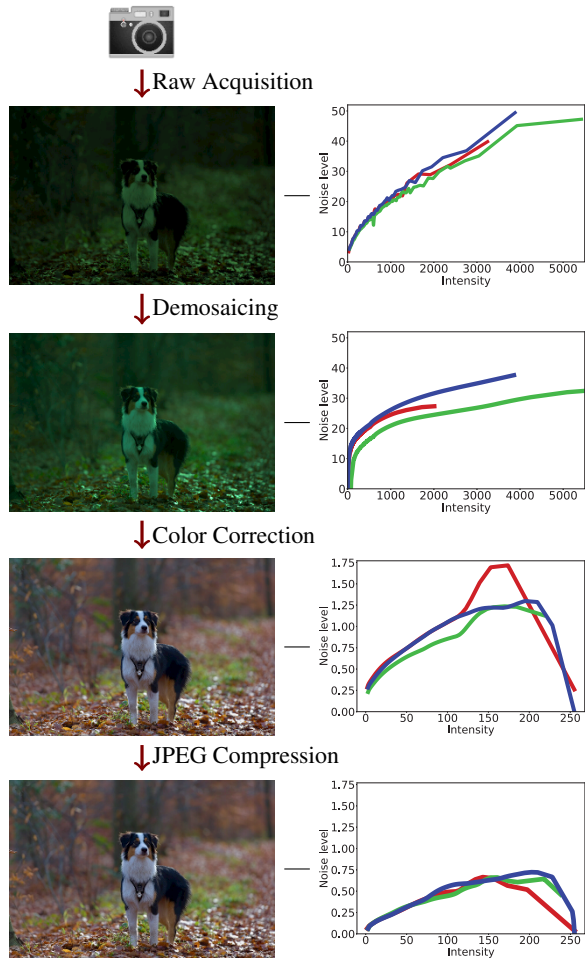


Figure 1.2. *Simplified processing pipeline of an image, from its acquisition by the camera sensor to its storage as a JPEG-compressed image. The left column represents the image as it goes through each step. The right column plots the noise of the image as a function of intensity in all three channels (red, green, blue). Because each step leaves a specific footprint on the noise pattern of the image, analyzing this noise enables us to reverse engineer the pipeline of an image. This in turn enables us to detect regions of an image which were processed differently, and are thus likely to be falsified*

1.2. Describing the image processing chain

The main steps in the digital image acquisition process, illustrated in Figure 1.2, will be briefly described in this section. Other very important steps, such as denoising, are beyond the scope of this chapter and will therefore not be covered here.

1.2.1. *Raw image acquisition*

The first step of acquiring a raw image consists of counting the number of incident photons over the sensor along the exposure time. There are two different technologies used in camera sensors: charge coupled devices (CCDs) and complementary metal-oxide-semiconductors (CMOS). Although their operating principles differ, both can be modeled in a very similar way (Aguerrebere *et al.* 2013). Both sensors transform incoming light photons into electronic charge, which interacts with detection devices to produce electrons stored in a potential light well. When the latter is full, the pixels become saturated. The final step is to convert the analog voltage measurements into digital quantized values.

1.2.2. *Demosaicing*

Most cameras cannot see color directly, because each pixel is obtained through a single sensor that can only count the number of photons reaching it in a certain wavelength range. In order to obtain a color image, a color filter array (CFA) is placed in front of the sensors. Each of them only counts the photons of a certain wavelength. As a result, each pixel has a value relative to one color. By using filters of different colors on neighboring pixels, the missing colors can then be interpolated.

Although others exist, almost all cameras use the same CFA: the Bayer array, which is illustrated in Figure 1.3. This matrix samples half the pixels in green, a quarter in red and the last quarter in blue. Sampling more pixels in green is justified by the human visual system, which is more sensitive to the color green.

Unlike other steps in the formation of an image, a wide variety of algorithms are used to demosaic an image. The most simple demosaicing algorithm is bilinear interpolation: missing values are interpolated by averaging the most direct neighbors sampled in that channel. As the averaging is done regardless of the image gradient, this can cause visible artifacts when interpolated against a strong gradient, such as on image edges.

To avoid these artifacts, more recent methods attempt to simultaneously take into account information from the three color channels and avoid interpolation along a steep gradient. For instance, the Hamilton–Adams method is carried out in three

stages (Hamilton and Adams 1997). First, it interpolates the missing green values by taking into account the green gradients corrected for the discrete Laplacian of the color already known at each pixel to interpolate horizontally or vertically, in the direction where the gradient is weakest. It then interpolates the red and blue channels on the pixels sampled in green, taking the average of the two neighboring pixels of the same color, corrected by the discrete Laplacian of the green channel in the same direction. Finally, it interpolates the red channel of blue-sampled pixels and the blue channel of red-sampled pixels using the corrected average of the Laplacian of the green channel, in the smoothest diagonal.

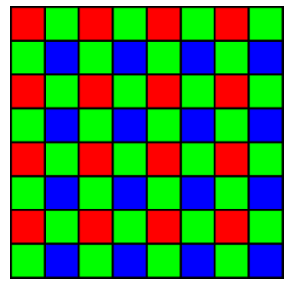


Figure 1.3. *The Bayer matrix is by far the most used for sampling colors in cameras*

Linear minimum mean-square error demosaicing (Getreuer 2011) suggests working not directly on the three color channels (red, green and blue), but on the pixelwise differences between the green channel and each of the other two channels separately. It interpolates this difference separately in the horizontal and vertical directions, in order to estimate first the green channel, followed by the differences between red and green, and then between blue and green. The red and blue channels can then be recovered by a simple subtraction. This method, as well as many others, makes the underlying assumption that the difference of color channels is smoother than the color channels themselves, and therefore easier to interpolate.

More recently, convolutional neural networks have been proposed to demosaic an image. For instance, demosaicnet uses a convolutional neural network to jointly interpolate and denoise an image (Gharbi *et al*. 2016; Ehret and Facciolo 2019). Even if these methods offer superior results to algorithms without training, they also require more resources, and are therefore not widely used yet in digital cameras.

The methods described here are only a brief overview of the large array of methods that exist for image demosaicing. This variety is increased by the fact that most industrial cameras do not disclose their often private demosaicing algorithm.

No demosaicing method is perfect – after all, it is a matter of reconstructing missing information – and produces some level of artifacts, although some produce much fewer artifacts than others. Therefore, it is possible to detect these artifacts to obtain information on the demosaicing method applied to the image, which is explained in section 1.4.

1.2.3. *Color correction*

White balance aims to adjust values obtained by the sensors, so that they match the colors perceived by the observer by adjusting the gain values of each channel. White balance adjusts the output using characteristics of the light source, so that achromatic objects in the real scene are rendered as such (Losson and Dinet 2012).

For example, white balance can be achieved by multiplying the value of each channel, so that a pixel that has a maximum value in each channel is found to have the same maximum value 255 in all channels.

Then, the image goes through what is known as gamma correction. The charge accumulated by the sensor is proportional to the number of photons incident on the device during the exposure time. However, human perception is not linear with the signal intensity (Fechner 1860). Therefore, the image is processed to accurately represent human vision by applying a concave function of the form $f_{k,\gamma} = ku^{\frac{1}{\gamma}}$, where γ typically varies between 1.8 and 2.2. The idea behind this procedure is not only to enhance the contrast of the image but also to encode more precisely the information in the dark areas, which are too dark in the raw image.

Nevertheless, commercial cameras generally do not apply this simple function, but rather a tone curve. Tone curves allow image intensities to be mapped according to precomputed tables that simulate the nonlinearity present in human vision.

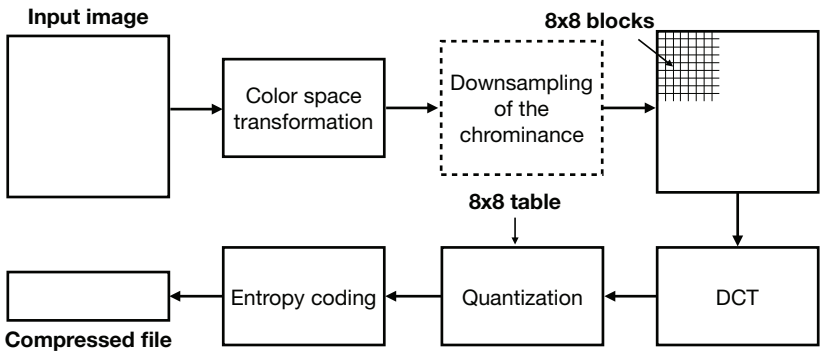


Figure 1.4. *JPEG compression pipeline*

1.2.4. *JPEG compression*

The stages of the JPEG compression algorithm, illustrated in Figure 1.4, are detailed below. The first stage of the JPEG encoding process consists of performing a color space transformation from RGB to YC_BC_R, where Y is the luminance component and C_B and C_R are the chrominance components of the blue difference and the red difference. Since the Human Visual System (HVS) is less sensitive to color changes than to changes in luminance, color components can be subsampled without affecting visual perception too much. The subsampling ratio generally applied is 4:2:0, which means that the horizontal and vertical resolution is reduced by a factor of 2. After the color subsampling, each channel is divided into blocks of 8×8 and each block is processed independently. The discrete cosine transform (DCT) is applied to each block and the coefficients are quantized.

The JPEG quality factor Q, ranging between 1 and 100, corresponds to the rate of image compression. The lower this rate, the lighter the resulting file, but the more deteriorated the image. A quantization matrix linked to Q provides a factor for each component of the DCT blocks. It is during this quantization step that the greatest loss of information occurs, but it is also this step that allows the most space in memory to be saved. The coefficients corresponding to the high frequencies, whose variations the HVS struggles to distinguish, are the most quantized, sometimes going so far as to be entirely canceled.

Finally, as in the example in Figure 1.5, the quantized blocks are encoded without loss to obtain a JPEG file. Each 8×8 block is zig-zagged and the coefficients are arranged as a vector in which the first components represent the low frequencies and the last ones represent the high frequencies.

Lossless compression by RLE coding (range coding) then exploits the long series of zeros at the end of each vector due to the strong quantization of the high frequencies, and then a Huffman code allows for a final lossless compression of the data, to which a header is finally added to form the file.

1.3. Traces left on noise by image manipulation

1.3.1. *Non-parametric estimation of noise in images*

Noise estimation is a necessary preliminary step to most image processing and computer vision algorithms. However, compared to the literature on denoising, research on noise estimation is scarce (Lebrun *et al*. 2013). Most homoscedastic white noise estimation methods (Lee 1981; Bracho and Sanderson 1985; Donoho and Johnstone 1995, 1994; Immerkær 1996; Mastin 1985; Voorhees and Poggio 1987; Lee and Hoppel 1989; Olsen 1993; Rank *et al*. 1999; Ponomarenko *et al*. 2007)

follow the same paradigm: they look for flat regions in the image and estimate noise in high frequencies, where noise dominates over signal.

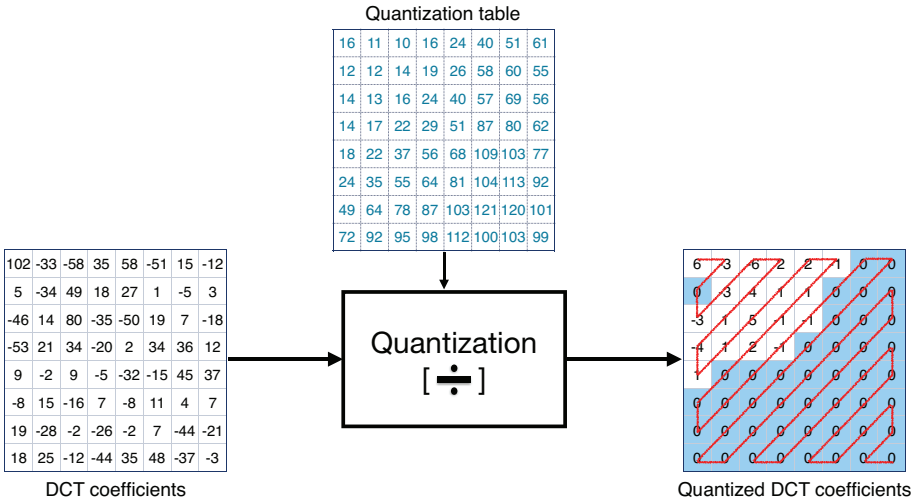


Figure 1.5. *An example of the impact of quantization on a DCT block. Each DCT coefficient is quantized by a value found in a quantization matrix. Rounding to the nearest integer results in many of the high-frequency coefficients being set to zero. Each block is zig-zagged to be encoded as a vector with a sequence of zeros*

We shall limit ourselves to discuss the method acknowledged as the best estimator for homoscedastic noise in the review (Lebrun *et al*. 2013), the Ponomarenko *et al*.'s method (Ponomarenko *et al*. 2007). This method computes the variance of overlapping 8×8 pixels blocks. To avoid the effects of textures and edges, blocks are sorted according to their low-frequency energy; only a small percentile (typically 0.5%) is used to select the blocks whose low- and medium-frequency energy is lowest. The final noise estimation is obtained by computing the median of the variances in the high frequencies of these blocks.

Homoscedastic white noise estimation algorithms can be adapted to estimate an arbitrary signal-dependent noise curve, as pointed out by Colom *et al*. (2014). However, after undergoing the complete camera processing chain detailed in section 1.3.2, noise depends not only on signal but also on frequency. A multi-scale estimation is needed in order to estimate highly correlated frequency-dependent noise (Lebrun *et al*. 2015). Following this observation, Colom and Buades extended Ponomarenko *et al*.'s method (Ponomarenko *et al*. 2007) to incorporate such a multi-scale approach (Colom and Buades 2013).

1.3.2. *Transformation of noise in the processing chain*

This section examines the way in which noise is affected at each step of the camera processing chain (see section 1.2). Noise curves obtained with the extended Ponomarenko *et al.*'s method (Colom and Buades 2013) along the processing chain (raw image, demosaicing, white balance, gamma correction and JPEG-encoding) are presented in Colom (2014) and compared to the temporal estimation.

Temporal estimations of noise curves are non-parametric, and good enough to be considered as ground-truth. Having ground-truth noise curves is an important issue when evaluating the performance of estimation methods. These temporal estimations are built by taking burst photos of the same scene, which consists of a calibration pattern with large flat zones (Figure 1.6), under constant lighting with a steady camera. Under these conditions, the variance of a pixel value can only be explained by noise. Thus, the noise curve obtained by computing the standard deviation of the temporal series yields the ground-truth noise curve. These noise curves depend on the camera used as well as the particular processing chain, including the ISO level and the exposure time.

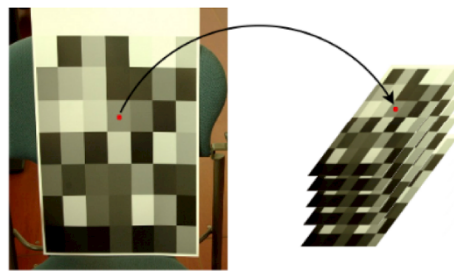


Figure 1.6. *Calibration model used for the construction of the temporal series*

1.3.2.1. *Raw image acquisition*

The value at each pixel generated by the process described in section 1.2.1 can be modeled as a Poisson variable, whose expectation is the real value of the pixel. The noise measured at the CCD or CMOS sensor has several components; Table 1.1 describes the main sources.

Figure 1.2 shows the noise curve obtained by temporal series (ground truth) and the estimation obtained from a single image computed using Ponomarenko *et al.*'s method (Colom and Buades 2013) with a simplified pipeline. Note that the estimate is accurate since all curves match. At this step, all channels have the same noise curve. As noise follows a Poisson distribution, the noise variance follows a simple linear relation $\sigma^2 = a + bu$, where u is the intensity of the ideal noiseless image, and a

and b are constants. Consequently, the noise curves are strictly increasing. Moreover, although the noise curves do not account for it, the noise characteristics reported above suggest that the noise is uncorrelated, that is, the noise at a certain pixel is not related to noise at any other pixel with the same signal intensity.

Type of noise	Description
Shot noise	Due to the physical nature of light. It describes the fluctuations in the number of photons detected due to their independent emission from each other.
Dark noise	Some electrons accumulate on the potential well as the result of a thermal cause. These electrons are known as dark current because they are present and will be detected even in the absence of light.
Photo response non-uniformity (PRNU)	It describes the way in which the individual pixels in the sensor array respond to uniform light sources. Due to variations in pixel geometry, substrate material, and micro-lenses, different pixels do not produce the same number of electrons from the same number of photons hitting them.
Readout noise	During the readout phase of the acquisition process, a voltage value is read at each pixel. This voltage is computed as a potential difference from a reference level which represents the absence of light. Thermal noise, inherent in the readout circuit, affects the output values.
Electronic noise	It is caused by the absorption of electromagnetic energy by the semiconductors of the camera circuits and the cross-talk phenomenon.

Table 1.1. *Description of the main sources of noise during the acquisition process*

1.3.2.2. *Demosaicing*

Demosaicing is presented in more detail in section 1.2.2. After this step, the noise at each pixel is correlated with its neighbors. After demosaicing, each channel has a different noise curve since channels are processed differently by the demosaicing algorithm.

In addition, the noise curves calculated using Ponomarenko *et al.*'s method and those obtained from the temporal series no longer match. This is due to the fact that

Ponomarenko *et al.*'s algorithm assumes white noise and estimates noise at high frequencies, which are affected by demosaicing. As the image processing chain is sequential, the temporal noise curves and those measured on a single image will no longer match after demosaicing.

1.3.2.3. *Color correction*

White balance increases the intensity range of the image. Since the weights are different for each color channel, as mentioned in section 1.2.3, the three noise curves are less correlated after this step. Then, gamma correction greatly increases the noise and the dynamic range of the image, due to the power law function. Furthermore, the noise curves are no longer monotonically increasing after this step. Indeed, if we denote γ the function applied during the gamma correction step, the asymptotic expansion around the intensity u yields $\gamma(u + n) = \gamma(u) + \gamma'(u)n$, where n is the noise at the intensity u.

1.3.2.4. *JPEG compression*

The dynamic range remains unchanged after JPEG compression. However, noise is reduced after JPEG compression due to the quantization of the DCT coefficients, in particular those corresponding to high frequencies. Furthermore, the curve estimated by Ponomarenko *et al.*'s algorithm (Colom and Buades 2013) differs more from the temporal series curves, because the noise estimation method estimates noise at high frequencies, which are altered or even destroyed by compression.

The noise present in JPEG images is the result of several transformations on the initial noise model, which initially follows a Poisson distribution. In the end, the final image's noise does not follow any predefined model, it instead depends on many unknown parameters that are set by each manufacturer. The only certainty we have is that noise is intensity dependent and frequency dependent. Therefore, it is preferable to use non-parametric models to estimate noise curves, so as to estimate the curves from the image itself.

1.3.3. *Forgery detection through noise analysis*

Image tampering, such as external copy–paste (splicing) or internal texture synthesis, can be revealed by inconsistencies in local noise levels, since noise characteristics depend on lighting conditions, camera sensors, ISO setting, and post-processing, as shown in section 1.3.2.

One of the unique features characterizing noise is the photographic response non-uniformity (PRNU), as presented in section 1.3.2. Chen *et al.* propose to detect the source digital camera by estimating the PRNU (Chen *et al.* 2008). According to the authors, the PRNU represents a unique fingerprint of image sensors and, hence, it would reveal altered images.

One of the most popular algorithms for detecting splicing using noise level traces is proposed by Mahdian and Saic (2009). It consists of dividing the image into blocks and estimating the noise level using wavelets in each block. Blocks are then merged into homogeneous regions, the noise standard deviation being the homogeneity condition. The output of this method is a map showing the segments of the image having a similar noise standard deviation.

A different approach is introduced in Pan *et al*. (2011), where the noise estimation is based on the kurtosis concentration phenomenon. The kurtosis of natural images across different frequency bands is constant. This allows for the estimation of noise variance when it follows an additive white Gaussian noise (AWGN) model. The method then segments the image into regions based on their noise variance using the k-means algorithm.

The method presented in Yao *et al*. (2017) makes use of the signal dependency of noise. Instead of a single noise level, it estimates a noise-level function. The image is segmented into edges and flat regions. It estimates the noise level on flat regions and the camera response function (CRF) on edges. Noise level functions are then compared and an empirical threshold is set to detect the salient curves. The main disadvantage of this method is that it assumes that the image has only undergone demosaicing.

We will now describe a recent method based on multi-scale noise analysis for the detection of tampering in JPEG-compressed images. After the complete camera processing chain, noise is not only signal dependent but also frequency dependent, which is mainly due to the correlation introduced by demosaicing and the quantization of DCT coefficients during JPEG compression. In this context, a multi-scale approach is necessary to capture the noise in the low frequencies. Indeed, when successive subscales are considered, the low frequencies become high frequencies due to the contraction of the DCT domain, which makes it possible to estimate the noise in low frequencies with the methods presented in section 1.3.1. One could argue that the noise contained in the low and medium frequencies could be directly estimated without considering subscales, but this is a risky procedure since these frequencies also contain part of the signal. This problem is avoided by the proposed method because, at each scale, the algorithm finds blocks having low variance in the low and medium frequencies to estimate the noise.

Consider the operator S that tessellates the image into sets of blocks of 2×2 pixels, and replaces each block by a pixel whose value is the average of the four pixels. We define the nth scale of an image u, and we denote it by S_n, as the result of applying n times the operator S to the image u.

The first step of the method consists of splitting the image into blocks of 512×512 with 1/2 overlap, which are called macro-blocks. Noise curves are estimated using

Ponomarenko *et al.*'s method (Colom and Buades 2013) in each of the three RGB channels for each macro-block, as well as for the complete image (global estimation), at scales 0, 1 and 2. For each scale and channel, the noise curves obtained from each macro-block are compared to the global estimation.

Ideally, a non-forged image should exhibit the same noise level function for all of its macro-blocks, as well as for the entire image. However, when estimating noise curves in the presence of textures, noise overestimation is likely to happen (Liu *et al.* 2006). Therefore, textured macro-blocks are expected to give higher noise levels, even when they are not tampered with. Thus, the global estimation obtained provides, in fact, a lower bound for the noise curves of the individual macro-blocks. Therefore, any macro-block with lower noise levels than the global estimation is suspicious, as it would be an indicator that the underlying region has a different noise model than the rest of the image.

For detection purposes, we consider the percentage of bins in the noise curve of the macro-block whose count is below the global estimation, independently for each RGB channel and for each scale. The geometric mean of percentages obtained for each channel provides a heat map, unique to each scale. These heat maps show macro-blocks with noise levels that are incompatible with the global estimation, at a given scale.

Different criteria can be considered to define detections. One possibility is to consider that a macro-block is detected if, at any scale, the geometric mean of the percentage of cells below the overall image curve is 100%. This means that the noise curve calculated by the macro-block, at a certain scale, is entirely below the noise curve of the overall image, for all three RGB channels.

The size of the macro-blocks may appear to be too big compared to other methods willing to detect forgeries by noise analysis. This choice is made in order to achieve S_2 with a reasonable number of bins to obtain an accurate enough estimation. Each sub-scale implies a reduction of the image by a factor of 2 in each direction. In this way, if the macro-blocks are 512×512 in S_0, in S_1, they are 256×256, and in S_2, the macro-blocks are 128×128.

Figure 1.7 shows an example of an image where an external copy–paste has been performed. The vase on the right has been cropped from the auxiliary image and pasted onto the original image. The results of applying the proposed method to this forged image are shown in Figure 1.8.

At S_0, the results do not show any different behavior in the tampered zone. If another scale is considered, S_1, we find that the manipulated area has lower noise levels than the rest of the image. In fact, the noise curves corresponding to macro-blocks containing the spliced region have about 80% of their bins below the

global noise curve, this percentage being slightly different for each channel. Finally, S_2 provides the strongest proof of falsification. Indeed, the noise curves corresponding to forged macro-blocks have all of their bins below the global estimation in all the three RGB channels.

(a) Original image (b) Auxiliary image (c) Falsified image

Figure 1.7. *Example of falsification: the vase in b) has been cut out and copied onto a), which gives c)*

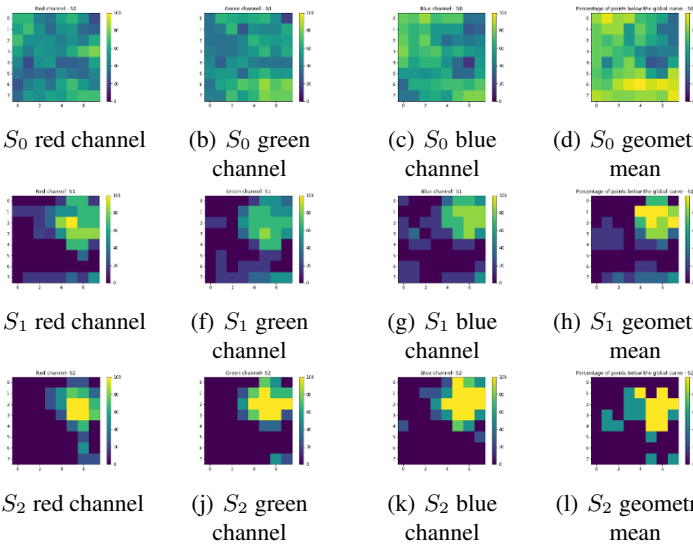
COMMENT ON FIGURE 1.7.– *The original image was taken with ISO 800 and exposure time 1/8 s. The auxiliary image was taken with ISO 100 and exposure time 1.3 s. Both images were taken with the same Panasonic Lumix DMC-FZ8 camera under the high-quality JPEG compression setting.*

This example illustrates the need for a multi-scale approach for noise inconsistency analysis applied to forgery detection.

To conclude, noise inconsistency analysis is a rich source for forgery detection due to the fact that forged regions are likely to present different noise models from the rest of the image. However, to exploit this, it is necessary to have algorithms that are capable of dealing with signal and frequency-dependent noise. The multi-scale approach is shown as an appropriate framework for noise inconsistency analysis.

1.4. Demosaicing and its traces

Image demosaicing, which will be presented in detail in section 1.2.2, leaves artifacts that can be used to find falsifications. The Bayer CFA (see Figure 1.3) is by far the most commonly used. Mosaic detection algorithms thus focus on this matrix, although they could be adapted to other patterns.



(a) S_0 red channel (b) S_0 green channel (c) S_0 blue channel (d) S_0 geometric mean

(e) S_1 red channel (f) S_1 green channel (g) S_1 blue channel (h) S_1 geometric mean

(i) S_2 red channel (j) S_2 green channel (k) S_2 blue channel (l) S_2 geometric mean

Figure 1.8. *Percentage of points below the global noise curve and geometric mean for each macro-block at S_0, S_1 and S_2*

1.4.1. *Forgery detection through demosaicing analysis*

Detecting demosaicing artifacts can answer two questions:

– Is it possible that a given image was obtained with a given device?

– Is there a region of the image whose demosaicing traces are inconsistent with the rest of the image?

Two different approaches can be used to study the demosaicing of an image. One could try to estimate the specific demosaicing algorithm that was used in the image. Such an estimation might prove that an image was not taken by a given camera, assuming the demosaicing method is different from the one used by the camera. Within an image itself, a region that was demosaiced differently than the global image is likely to be forged. Such an analysis could be justified by the large variety of demosaicing methods, however this variety also limits the potential detection. To establish or disprove the link between an image and a camera through the demosaicing algorithm would require us to know the algorithm used, which is rarely the case. Estimating the specific demosaicing algorithm used by a camera would require a large amount of images from said cameras; while theoretically possible, no such work has ever been attempted. The ability to detect regions of an image that have been demosaiced with a different algorithm than the rest of the image is only theoretical; in practice the estimation of a demosaicing method once again requires more data than can usually be provided by a small region of an image. Furthermore,

most demosaicing algorithms do not interpolate all regions in the same way. As a consequence, a reliable comparison of two estimations is difficult, as the same algorithm may have interpolated two regions differently. That being said, global learning-based methods can make use of the demosaicing algorithm to get information on an image, although this is just one feature learned and used among others, and not something that is conclusive by itself. For instance, Siamese-like networks such as Noiseprint (Cozzolino and Verdoliva 2020) implicitly learn some information on the demosaicing algorithm to detect forgeries. These methods will be presented in more detail in section 1.7.

A more promising approach is to directly detect the position of the Bayer matrix. Indeed, while the CFA pattern is almost always a Bayer matrix, the exact position of the matrix, that is, the offset of the CFA, varies. Detecting the position of the matrix therefore has two uses:

– we can compare the position of the Bayer matrix in the image to the one normally used by a specific device. If the positions do not correspond, then the image was either not taken by that device, or it was cropped in the processing;

– in the case of copy-move, both internal and external (splicing), there is a $\frac{3}{4}$ probability that the position of the Bayer matrix does not correspond between the original image and the pasted region. Therefore, detecting the position of the Bayer matrix, both globally and locally, can be used to find inconsistencies.

Most current demosaicing detection methods focus on this second idea, as local CFA inconsistencies give useful information on the image and can be found relatively easily in ideal conditions, that is, in uncompressed images, as we will now present.

1.4.2. *Detecting the position of the Bayer matrix*

Different methods make it possible to detect either the position of the Bayer matrix directly or inconsistencies of this matrix in the image.

1.4.2.1. *Joint estimation of the sampled pixels and the demosaicing algorithm*

In a pioneering paper on demosaicing analysis, Popescu and Farid (2005) propose to jointly estimate a linear model for the demosaicing algorithm and detect which pixels have been sampled in a given channel with an expectation-maximization (EM) algorithm. The demosaicing algorithm is estimated on pixels detected as interpolated (i.e. not sampled) as a linear combination of neighboring pixels in that channel. Sampled pixels are detected as pixels where the linear combination yields a result far from the correct value of the pixel. A pseudo-probability map of each pixel being sampled is then computed. Assuming the linear model is correct, sampled pixels will be correctly detected and there will be a strong 2-periodicity of the map, which can

easily be seen as a peak in the Fourier transform of the image. However, in a region which has been altered, the estimated linear model will no longer be correct, either because the demosaicing estimation appears differently or because there are no demosaicing traces left at all. The 2-periodicity peak will thus locally disappear, and can be detected as potential evidence of a forgery. This method can then potentially classify the used demosaicing algorithm, or detect the absence of demosaicing artifacts, sign of manipulations such as blurring or inpainting to hide data on the image. To detect a change in the position of the Bayer matrix, (González Fernández *et al*. 2018) use a DCT instead of a Fourier transform. This simple modification enables one to directly visualize a change of position as a local change of signs at the observed peaks, instead of a harder-to-visualize phase difference. Unfortunately, while these detections were possible when Popescu's article was first published, the demosaicing algorithms have become much more complex since then, both thanks to theoretical progress and increased computational power in cameras. Modern demosaicing algorithms process channels jointly and are strongly nonlinear, preventing an easy modelization of the demosaicing process.

1.4.2.2. *Double demosaicing detection*

Another method proposes to directly detect the CFA pattern used in the image (Kirchner 2010). In order to do this, the image is remosaiced and demosaiced in the four possible positions, with a simple algorithm such as bilinear interpolation. The reasoning is that demosaicing should produce an image closer to the original when it is remosaiced in the correct position. They then compare the residuals to detect which position of the CFA has been used. Since CFA artifacts are generally more visible in the green channel, they decide first the position of the sampled green pixels, before deciding between the remaining two positions with the red and blue channels, a paradigm that has been used in most publications since then. Their use of the bilinear algorithm limits them in the same way (Popescu and Farid 2005; González Fernández *et al*. 2018) due to the linearity and chromatic independence of the bilinear algorithm, which is not shared by most modern demosaicing algorithms. However, their method does not depend on the choice of algorithm, and could therefore provide very good results should the originally used demosaicing algorithm be known.

1.4.2.3. *Direct detection of the grid by intermediate values*

In order to break away from a specific algorithm, Choi *et al*. (2011) highlights that pixels are more likely to present extreme values locally in the channel in which they are sampled and, on the contrary, to take on intermediate values when they are interpolated (Choi *et al*. 2011). Therefore, they count the number of intermediate values in the four positions to decide which position is the correct one. The idea that pixels are more likely to take extreme values in their sampled channel is generally true with most algorithms, which makes this method produce good classification scores. However, the probability bias can be reversed when the algorithms make

heavy use of the high frequencies of other channels, which can lead to confident, but incorrect detection of certain regions of the image.

1.4.2.4. *Detecting the variance of the color difference*

Shin *et al*. (2017) attempts to avoid the assumption that color channels are processed independently. Instead of working separately with each channel, as was done until then, they work on the difference between the green and red channels, as well as between the green and blue channels. This reflects more accurately the operations done by many demosaicing algorithms, which first interpolate the green channel before using the green channel's information to interpolate the red and blue channels. They compute the variance of these differences in the four possible patterns on the two computed maps, and identify the correct pattern as the one featuring the highest variance, which is expected of the original pattern, whose pixels are all sampled instead of interpolated. Although the dependence of the color channels is hard-coded, the color difference is actually used in many current algorithms and represents a first step toward a full understanding of demosaicing artifacts.

1.4.2.5. *Detection by neural networks of the relative position of blocks*

More recently, Bammey *et al*. (2020) proposed to train a self-supervised convolutional neural network (CNN) to detect modulo-(2, 2) position of the blocks in the image. As CNNs are invariant to translation, they need to rely on image information to detect this position. Demosaicing artifacts, and to some extent JPEG artifacts, are the only relevant information a network can use to this end. As a result, training a network to detect this position will implicitly make it analyze demosaicing artifacts. This will thus lead to a local detection of the Bayer matrix's position. Erroneous outputs of the network are caused by inconsistencies in the image's mosaic, and can thus be seen as traces of forgery.

This method obtains better results than previous works, and can help further analyze the forgery as different kinds of forgeries will cause different artifacts. For instance, copy-move will cause a locally consistent shift in the network's output, whereas inpainting – usually performed by cloning multiple small patches onto the target area – may show each cloned patch detected with a different pattern. Other manipulations, such as blurring, or the copy-move of an image that features no mosaic – for instance due to downsampling – may locally remove the mosaic, and the output of the network will thus be noise like in the forged region. It is possible to achieve even better results with internal learning, by retraining the network directly on images to study. This lets the network adapt to different post-processing, most importantly to JPEG compression.

However, this method is more computationally intense than the other presented algorithms, especially when internal learning is needed. This makes it less practical to use when many images are to be analyzed.

1.4.3. *Limits of detection demosaicing*

Recent methods proposed by Choi *et al*. (2011), Shin *et al*. (2017) or Bammey *et al*. (2020) are able to analyze the mosaic of images well enough for practical applications. It is now possible to detect, even locally, the position of the Bayer matrix. Detecting the presence of demosaicing artifacts is generally easy, even though their absence is not necessarily a sign of falsification because most modern demosaicing algorithms leave little to no artifacts on easy-to-interpolate regions. However, the range of images that can be detected remains limited. Demosaicing artifacts are 2-periodic, and they reside in the highest frequencies. As a result, they are entirely lost when the image is downsampled by a factor of at least 2. More generally, image resizing will also rescale the demosaicing artifacts; even though those might not always be lost, detection methods would need to be adapted to the new frequencies of the artifacts. JPEG compression is an even more important limitation. As compression mainly drops precision on the high-frequency components of an image, demosaicing artifacts are easily lost on compressed images. To date, even the best methods struggle to analyze CFA artifacts even at a relatively high compression quality factor of 95. Internal learning presented in Bammey *et al*. (2020) provides some degree of robustness to JPEG compression; however, demosaicing artifact detection remains limited to high-quality images, uncompressed or barely compressed, and at full resolution. This complements well the detection of JPEG compression, which we will now present.

1.5. JPEG compression, its traces and the detection of its alterations

In this section, we seek to determine the compression history of an image. We will focus on the JPEG algorithm, which is nowadays the most common method to store images. Most cameras use this format but others exist, such as HEIF, used in particular in Apple products since 2017. HEIF is also a lossy compression algorithm and therefore leaves traces; nevertheless, these traces are different from the ones produced by JPEG. As we will see, the analysis of the JPEG coding of an image makes it possible to detect local manipulations. For this, the methods take advantage of the structured loss of information caused by this step in the processing chain.

1.5.1. *The JPEG compression algorithm*

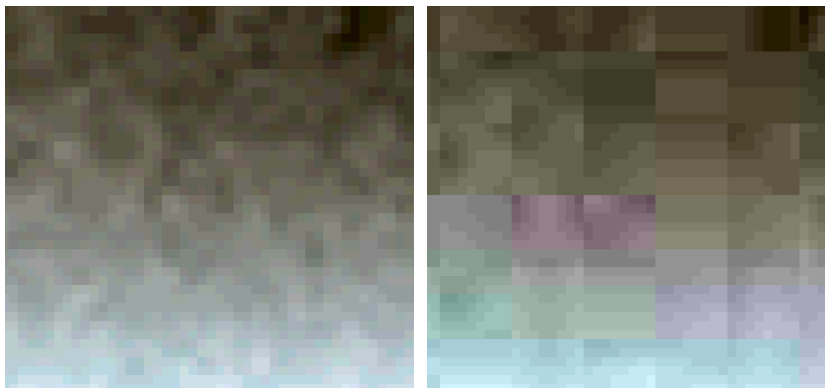
In JPEG encoding, the division of the image into 8×8 blocks and the application of a quantization step lead to the appearance of discontinuities at the edges of these blocks in the decompressed image.

Figure 1.9 shows the blocking effect that appears after JPEG compression. Contrast enhancement allows us to clearly see the 8×8 blocks. The greatest loss of

information is during the quantization step, explored in more detail in section 1.2.4. The blocking effect is due to quantization, depending on the Q parameter, applied on all 8×8 size blocks. Therefore, standard JPEG compression leaves two characteristic traces: the division into 8×8 non-overlapping blocks and the quantization, according to a quantization matrix, of the DCT coefficients. In other words, the two features to be detected from the image are:

1) the origin of the 8×8 grids;

2) the values of the quantization matrix.



(a) Uncompressed image (b) Compressed image with quality of 80

Figure 1.9. *Close-ups on an image before and after compression. The contrast has been enhanced to observe the JPEG artifacts, in particular the blocking effect, allowing us to see the edges of the 8 × 8 blocks*

In order to authenticate an image, the previous detections must verify that (1) the origin of the grid is aligned with the top left of the image; and (2) the quantization matrix calculated from the image is similar to the one in the header of the JPEG file. If the image is not in the JPEG format (providing the header file giving the quantization matrix), then estimating this information from the image itself is even more useful as an initial analysis.

Methods by Pevny and Fridrich (2008) make it possible to detect if an image has undergone a double compression, which creates an immediate argument against the authenticity of the image. Indeed, this would imply a duplicate in the processing chain of the image.

In the following sections, grid detection and quantization matrix estimation methods are illustrated. When no detection is made, the image may be classified as not having undergone JPEG compression.

1.5.2. *Grid detection*

In a JPEG-compressed image, the 8×8 blocks are created following a regular pattern starting at the pixel in the top left of the image and therefore coinciding with an original grid $(0, 0)$.

The aim of the method is to find the stage of separation in 8×8 blocks of the JPEG algorithm. This leads to having the position of the grid by giving its origin (this can vary if the image is cropped). If a grid is present, among the $8 \times 8 = 64$ different original possibilities, only one is correct.

Here, two families of methods are presented: methods based on block artifacts and methods based on the impact of quantization on the DCT coefficients.

1.5.2.1. *Compression artifacts*

These are methods based on the detectable traces left by compression. In their article, Minami and Zakhor propose a way of detecting JPEG grids with the aim of removing the blocking artifacts (Minami and Zakhor 1995). Later on, in Fan and de Queiroz (2003), the same ideas are used to decide whether an image has undergone JPEG compression, depending on whether traces are present or not. These methods use filters to bring out the traces of compression (Chen and Hsu 2008; (Li *et al*. 2009)). The simplest method calculates the absolute value of the gradient magnitude of the image (Lin *et al*. 2009), and others use the absolute value of derivatives of order 2 (Li *et al*. 2009). However, these two filters can have a strong response to edges and to textures present in the image and therefore can sometimes lead to faulty grid detections. To reduce the interference of details in the scene, a cross-difference filter, proposed by Chen and Hsu (2008), is more suitable. This filter, represented in Figure 1.10, amounts to calculating the absolute value of the result of a convolution of the image by a 2×2 kernel. The grid becomes visible because of the differentiating filter applied to the compressed image. The stronger the compression, the more this feature is present.

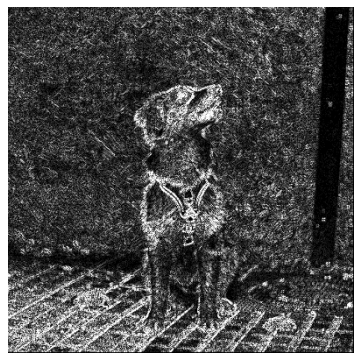
Recently, methods like the one proposed in Nikoukhah *et al*. (2020) have made these methods automatic and unsupervised thanks to statistical validation.

1.5.2.2. *DCT coefficients*

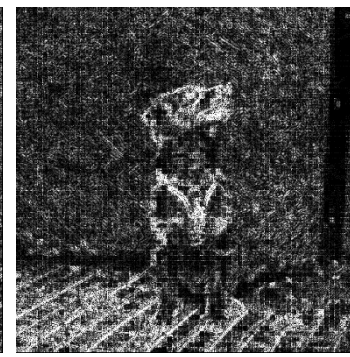
These are methods based on the impact of compression on the DCT coefficients. After quantization, the compression makes the size of the image file smaller by setting many of the DCT values to zero. As illustrated in Figure 1.5, the quantization leads to setting a lot of the high-frequency coefficients to zero. The values of the quantization matrix are generally larger in high frequencies. The stronger the compression, the more values are set to zero.

(a) Uncompressed image

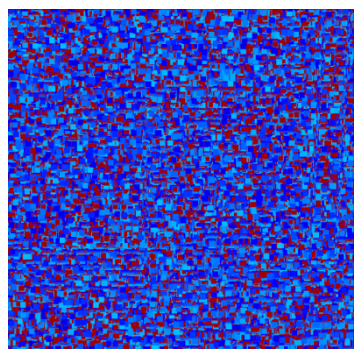
(b) Compressed image



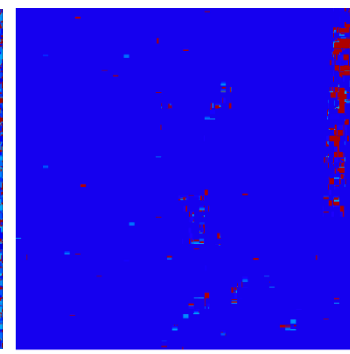
(c) Derivative filter applied to the uncompressed image



(d) Derivative filter applied to the compressed image



(e) Map of votes of the uncompressed image



(f) Map of votes of the compressed image

Figure 1.10. *Derivative filter and vote map applied to the same image without compression in a) and after JPEG compression of quality 80 in b)*

COMMENT ON FIGURE 1.10.– *The compressed image features a grid structure. The saturated zone on the right of the image hides any traces of compression. In the vote map, each pixel is associated with the grid for which it voted, in other words, the grid with the most zeros. For the compressed image, one color is dominant: it corresponds to the position* $(0, 0)$.

Based on the example of CAGI (Iakovidou *et al*. 2018), the ZERO method determines the origin of the grid by testing the 64 possibilities and selecting the one on which the DCT coefficients of the blocks has the most zeros (Nikoukhah *et al*. 2019). In other words, given an image, all of its pixels vote for the grid they think they belong to. In the event of a tie, the vote is not taken into account.

Figures 1.10(e) and 1.10(f) present the vote map: each pixel's color represents which of the 64 possible grids the pixel notes. Navy blue corresponds to the original grid $(0, 0)$, and red to a non-valid vote, in the event of a tie. At the top right of the image, the saturated zone is not used to detect JPEG traces since it does not contain any information.

The derivative filter presented in Figures 1.10(c) and 1.10(d) makes it possible to highlight the compression artifacts, and the vote map presented in Figures 1.10(e) and 1.10(f) is a colormap where each color is associated with a grid origin. In both cases, there is a clear difference between the image that has not undergone compression and the one which has undergone compression. In fact, these filters, which are part of the tools used by journalists and police experts today, lack a validation step. Indeed, as they are presented, users need to interpret them. It is important to understand why a filter detects one area rather than another. The goal would be to get a binary result.

In the case of an uncompressed image, no "vote" stands out significantly compared to the others. In the case of the compressed image in Figure 1.10(f), navy blue is dominant: it corresponds to position $(0, 0)$.

Whether it is the cross-difference or the pixel vote map, some areas remain difficult to interpret, therefore justifying the need for a statistical validation. For example, the saturated parts have no visible JPEG grid and therefore cannot be used to reach a decision.

1.5.3. *Detecting the quantization matrix*

The histogram of each of the 64 DCT coefficients makes it possible to determine the quantization step that corresponds to the associated value in the quantization matrix. Quantization has a very clear effect on the DCT coefficients histograms of an image, visible in Figure 1.11 before and after compression. DCT components

generally follow Laplacian distribution (Clarke 1985; Wallace 1992), except for the first coefficient that represents the average of the block.

The JPEG quantization step transforms each DCT coefficient into an integer, multiple of the quantization value (Fridrich *et al*. 2001). These integer values lead to real values for each pixel during compression, which are then rounded off to integer values. Due to the second rounding, the DCT coefficients of the image are no longer integer, but show a narrow distribution around the quantization values, as shown in Figure 1.11. The quantization value in Figure 1.11 is $q = 6$, and so the uncompressed coefficients are centered around the values $0, 6, -6, 12, -12$, and so on. Once a quantization model has been obtained for the DCT coefficients, forgery detection methods such as (Ye *et al*. 2007), look for inconsistencies in the histograms, after having established a stochastic model.

For example, Bianchi *et al*.'s method first estimates the quantization matrix used by the first JPEG compression, and then tries to model the frequencies of the histogram of each DCT coefficient (Bianchi *et al*. 2011).

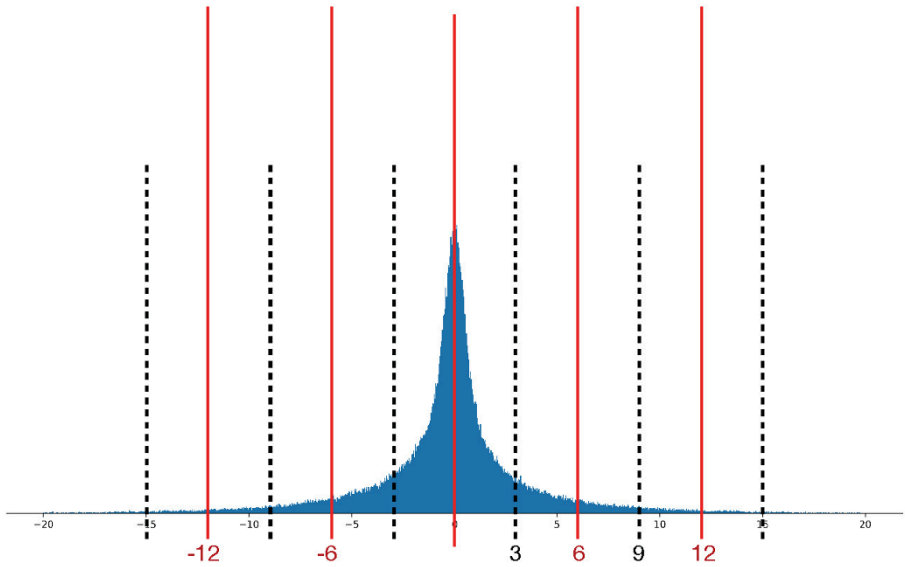
1.5.4. *Beyond indicators, making decisions with a statistical model*

Block artifacts, the number of zeros and the frequency interval of the histograms can be seen as compression detectors. However, a statistical validation is needed to determine whether the observations are indeed caused by compression or they are simply due to chance. This validation can be carried out by the *a contrario* approach (Desolneux *et al*. 2008).

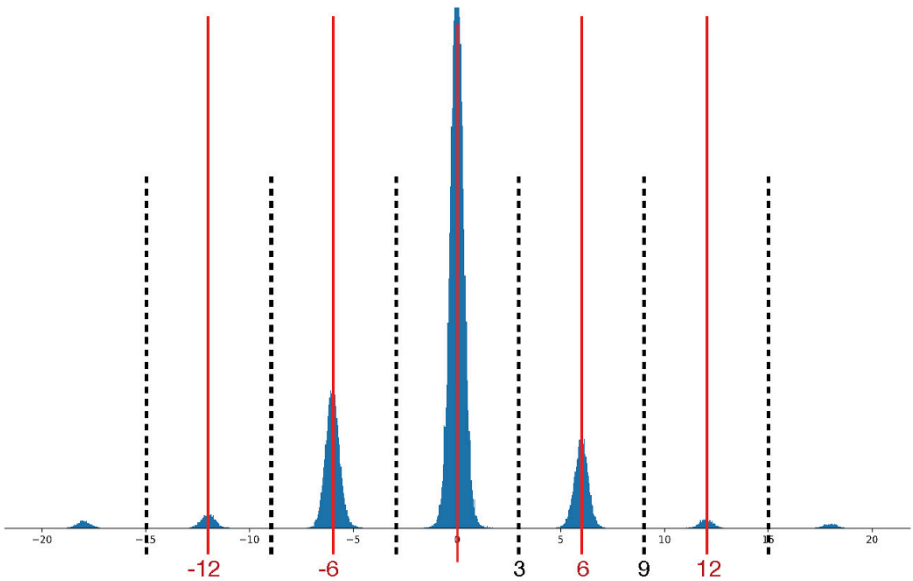
Applied to the whole image, these methods make it possible to know if an image has undergone JPEG compression, and if necessary, to know the position of the grid. The position of the grid origin indicates if the image has undergone a cropping after compression, as long as this cropping is not aligned with the initial grid, which can happen by chance in one out of 64 cases.

To verify an image, it is important to make the previous analysis methods local by checking the consistency of each part of the image with the global model. Several methods detect forgeries in areas having a different JPEG history than the rest of the image (Iakovidou *et al*. 2018; Nikoukhah *et al*. 2019).

Figure 1.12 illustrates a method that highlights an area where the JPEG grid origin is different from the rest of the image. In fact, the vote map in Figure 1.12(c) shows that it is already possible to visually distinguish the objects of the image having voted for a different grid than the rest of the image. A statistical validation automates the decision by giving a binary mask of the detection, as illustrated in Figures 1.12(e) and 1.12(f).



(a) Uncompressed image

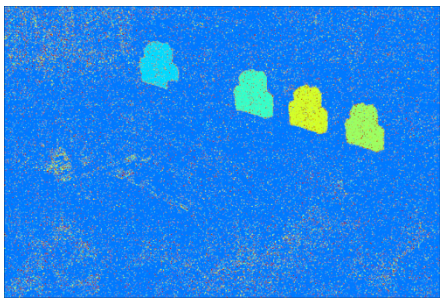


(b) Compressed image with quality of 93

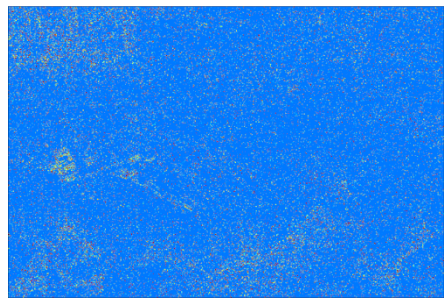
Figure 1.11. *Histogram of a DCT coefficient for an image before and after compression. There is a clear structure after quantization of the coefficients. The value of quantization is* $q = 6$

(a) An area has been copied several times in this image

(b) Original image



(c) Vote-map of the falsified image



(d) Vote-map of the original image

(e) The four falsified areas are detected automatically

(f) No detection is detected automatically

Figure 1.12. *In a), an area has been copied four times. The original image is shown in b)*

COMMENT ON FIGURE 1.12.– *In (c) and (d), the color indicates the origin of the grid of the JPEG blocks detected locally for the falsified and original images, respectively. The navy blue color corresponds to the detected main grid of origin* $(0, 0)$. *In (c), the areas whose block origin does not match the rest of the image are clearly visible. This detection is made automatic by the* a contrario *method, whose result can be seen in (e) and (f), where no anomaly is detected in the original image (f), while the falsified image (e) finds the four areas detected as being altered. The original and falsified images come from the database (Christlein* et al. *2012).*

Likewise, the quantization matrix can be estimated in order to know if it is consistent in each block of the image, and with the global quantization matrix which can be found in the associated header file, which allows the decompression of the image (Thai *et al*. 2017).

1.6. Internal similarities and manipulations

Finally, we will study the so-called internal manipulations, which modify an image by directly using parts of itself, like *inpainting* (Arias *et al*. 2011) and copy and paste.

Unlike other forgeries, these manipulations do not necessarily change residual traces of an image, because the parts used for the modification come from the same image. Therefore, specific methods are necessary for their detection.

The main difficulty in the detection of internal manipulations is the internal similarity of the image. A specialized database was created specifically to measure the rate of false detections between altered and authentic images, but with similar content in different regions (Wen *et al*. 2016).

The first methods are based on the study of Cozzolino *et al*. (2015a). Other methods use and compare key points, like those obtained with SIFT (Lowe 2004), which allows similar content to be linked. But this is often too permissive to detect copy and paste. This is why specialized methods, such as proposed by Ehret (2019), propose comparisons between descriptors to avoid the detection of similar objects, which are often distinguishable as shown in Figure 1.13. An example of copy and paste can be found in Figure 1.14.

Neural networks can also be used to detect copy-move manipulations, such as in Wu *et al*. (2018), where a first branch of the network detects the source and altered regions, while a second branch determines which of the two is the forgery, while other methods generally cannot distinguish the source from falsification.

(a) Real image, the two objects are similar but different

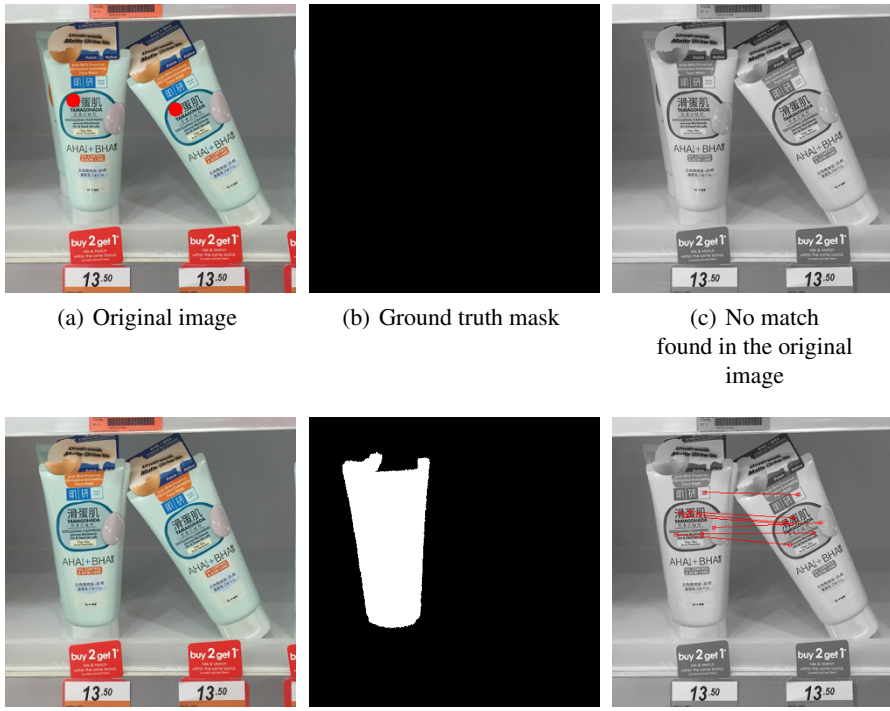
(b) Altered image with two copies of the same object

(c) A difference is visible on the patches of the two similar objects of the real image

(d) But when the image is altered, the patches of the two copies of the same object are identical

Figure 1.13. *The image in a) represents two similar, but different objects, while the image in b) represents two copies of the same object. Both images come from the COVERAGE database (Wen* et al. *2016)*

COMMENT ON FIGURE 1.13.– *The patches in (c) and (d) correspond to the descriptors used by Ehret (2019) associated with the look-at points represented by the red dots for the images that are authentic (a) and falsified (d), respectively. Differences are visible when the objects are only similar, whereas in the case of an internal copy–paste, the descriptors are identical. It is through these differences that internal copy–paste detection methods can distinguish internal copies from the presence of objects that would naturally be similar.*



(a) Original image (b) Ground truth mask (c) No match found in the original image

(d) Image altered by internal copy and paste (e) Ground truth mask of the altered image (f) Connections are found between the two identical copies

Figure 1.14. *Example of detection of copy–paste type modification on the images in Figure 1.13. The original and altered images are in (a) and (d), respectively, the ground-truth masks in (b) and (e), and the connections (Ehret 2019) between the areas detected as too similar in (c) and (f)*

1.7. Direct detection of image manipulation

To detect a particular manipulation, one must first be aware of the existence of this type of manipulation. As new manipulation possibilities are continually being created, it is necessary to continually adapt to new types of manipulation, otherwise the detection methods quickly become outdated. To break out of this cycle, several methods seek to detect manipulations without prior knowledge of their nature.

Recently, *generative adversarial networks* (GAN) have shown their ability to synthesize convincing images. A GAN is made up of two neural networks competing against each other: the first network seeks to create new images that the second one

fails to detect, while the second network seeks to differentiate original images from the ones generated by the first network.

Finally, the most common example concerns the use of automatic filters offered by image editing software such as Photoshop. Simple to use and able to produce realistic results, they are widely used. Neural networks can learn to detect the use of these filters or even reverse them (Wang *et al*. 2019), the training data can be generated automatically, but must deal with the immense variety of filters existing on this software.

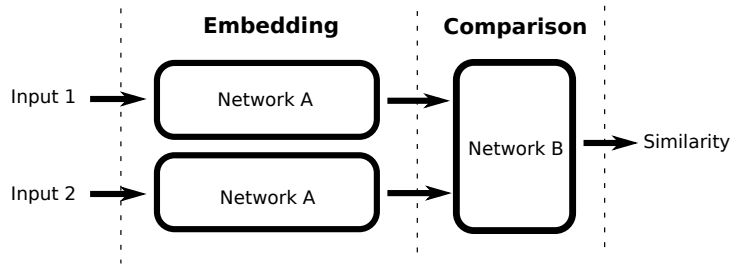


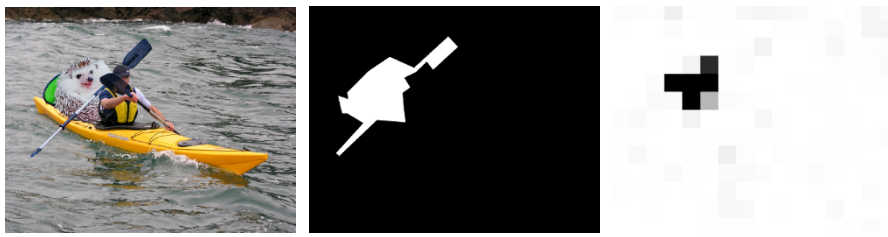
Figure 1.15. *Structure of the Mayer and Stamm (2019) network to compare the source of two patches. The same first network A is applied to each patch to extract a residue. These residues are then passed on to a network B which will compare their source and decide if the patches come from the same image or not*

Recently, Siamese networks have also been used for the detection of falsification (Mayer and Stamm 2019). They are bipartites, as shown in Figure 1.15. They consist of a first convolutional network that is applied independently to two image patches to extract hidden information from each, and then of a second network that compares the information extracted on the two patches to determine whether they come from the same picture. A big advantage of these methods is the ease of obtaining training data, since it is enough to have non-falsified images available and to train the network to detect whether or not the patches were obtained from the same picture. An example of detection with Siamese networks can be found in Figure 1.16.

1.8. Conclusion

In this chapter, we have described methods that analyze an image's formation pipeline. This analysis takes advantage of alterations made by the camera from the initial raw image to its final form, usually compressed JPEG. We have reviewed the transformations undergone by the raw image, and shown that each operation leaves traces. Those traces can be used to reverse engineer the camera pipeline, reconstructing the history of the image. It can also help detect and localize

inconsistencies caused by forgeries, as regions whose pipeline appears locally different than on the rest of the image. With that in mind, it is usually impossible to guarantee that an image is authentic. Indeed, a perfect falsification, which would not leave any traces, is not impossible, although it would require great expertise to directly forge a raw image – or revert the image into a raw-like state – and simulate a new processing chain after the forgery has been done. Falsifiers rarely have the patience nor the skills needed to carry out this task, however one cannot exclude that software to automatically make forged images appear authentic may emerge in the future.



(a) Falsified image (b) Ground truth (c) Detection of the Siamese network

Figure 1.16. *Example of modification detection with the Siamese network (Mayer and Stamm 2019)*

COMMENT ON FIGURE 1.16.– *The forged image comes from the database associated with Huh* et al. *(2018). The Siamese network gives a similarity score for each patch with a reference patch. The black areas in the Siamese network result correspond to patches that are incompatible with the reference patch.*

1.9. References

Agarwal, S. and Farid, H. (2017). Photo forensics from JPEG dimples. In *Workshop on Information Forensics and Security*. IEEE, Rennes.

Aguerrebere, C., Delon, J., Gousseau, Y., Musé, P. (2013). Study of the digital camera acquisition process and statistical modeling of the sensor raw data. Technical report [Online]. Available at: https://hal.archives-ouvertes.fr/hal-00733538.

Arias, P., Facciolo, G., Caselles, V., Sapiro, G. (2011). A variational framework for exemplar-based image inpainting. *International Journal of Computer Vision*, 93(3), 319–347.

Bammey, Q., Grompone von Giol, R., Morel, J.-M. (2020). An adaptive neural network for unsupervised mosaic consistency analysis in image forensics. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Bianchi, T., De Rosa, A., Piva, A. (2011). Improved DCT coefficient analysis for forgery localization in JPEG images. In *International Conference on Acoustics, Speech and Signal Processing*. IEEE, Prague.

Bracho, R. and Sanderson, A. (1985). Segmentation of images based on intensity gradient information. *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society Press, Amsterdam.

Buades, A., Coll, B., Morel, J.-M., Sbert, C. (2011). Self-similarity driven demosaicking. *Image Processing On Line*, 1, 51–56.

Chen, Y.-L. and Hsu, C.-T. (2008). Image tampering detection by blocking periodicity analysis in JPEG compressed images. In *10th Workshop on Multimedia Signal Processing*. IEEE, Cairns.

Chen, M., Fridrich, J., Goljan, M., Lukáš, J. (2008). Determining image origin and integrity using sensor noise. *IEEE Transactions on Information Forensics and Security*, 3(1), 74–90.

Choi, C.-H., Choi, J.-H., Lee, H.-K. (2011). CFA pattern identification of digital cameras using intermediate value counting. In *Multimedia Workshop on Multimedia and Security*. ACM, New York.

Christlein, V., Riess, C., Jordan, J., Riess, C., Angelopoulou, E. (2012). An evaluation of popular copy-move forgery detection approaches. *IEEE Transactions on Information Forensics and Security*, 7(6), 1841–1854.

Clarke, R.J. (ed.) (1985). Transform coding of images. *Astrophysics*. Academic Press, London and Orlando, FL.

Colom, M. (2014). Multiscale noise estimation and removal for digital images. PhD Thesis, University of the Balearic Islands, Palma.

Colom, M. and Buades, A. (2013). Analysis and extension of the Ponomarenko *et al.* method, estimating a noise curve from a single image. *Image Processing On Line*, 3, 173–197.

Colom, M., Buades, A., Morel, J.-M. (2014). Nonparametric noise estimation method for raw images. *Journal of the Optical Society of America A*, 31(4), 863–871.

Cozzolino, D. and Verdoliva, L. (2020). Noiseprint: A CNN-based camera model fingerprint. *IEEE Transactions on Information Forensics and Security*, 15, 144–159.

Cozzolino, D., Poggi, G., Verdoliva, L. (2015a). Efficient dense-field copy–move forgery detection. *IEEE Transactions on Information Forensics and Security*, 10(11), 2284–2297.

Cozzolino, D., Poggi, G., Verdoliva, L. (2015b). Splicebuster: A new blind image splicing detector. In *2015 IEEE International Workshop on Information Forensics and Security*. IEEE, Rome.

Desolneux, A., Moisan, L., Morel, J.-M. (2008). *From Gestalt Theory to Image Analysis*. Springer, New York.

Donoho, D.L. and Johnstone, I.M. (1994). Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3), 425–455.

Donoho, D.L. and Johnstone, I.M. (1995). Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, 90(432), 1200–1224.

Ehret, T. (2019). Robust copy-move forgery detection by false alarms control [Online]. Available at: https://arxiv.org/abs/1906.00649.

Ehret, T. and Facciolo, G. (2019). A study of two CNN demosaicking algorithms. *Image Processing On Line*, 9, 220–230.

Fan, Z. and de Queiroz, R.L. (2003). Identification of bitmap compression history: Jpeg detection and quantizer estimation. *IEEE Transactions on Image Processing*, 12(2), 230–235.

Fechner, G. (1860). *Elemente der Psychophysik*. Breitkopf and Hàrtel, Leipzig.

Fridrich, J., Goljan, M., Du, R. (2001). Steganalysis based on JPEG compatibility. *Multimedia Systems and Applications IV*, 4518, 275–281.

Getreuer, P. (2011). Zhang-Wu directional LMMSE image demosaicking. *Image Processing On Line*, 1, 117–126.

Gharbi, M., Chaurasia, G., Paris, S., Durand, F. (2016). Deep joint demosaicking and denoising. *ACM Trans. Graph.*, 35(6), 191:1–191:12.

Ghosh, A., Zhong, Z., Boult, T.E., Singh, M. (2019). Spliceradar: A learned method for blind image forensics. In *Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, Long Beach.

Gloe, T. (2012). Feature-based forensic camera model identification. In *Transactions on Data Hiding and Multimedia Security VIII*, Shi, Y.Q. (ed.). Springer, Berlin, Heidelberg.

González Fernández, E., Sandoval Orozco, A., García Villalba, L., Hernandez-Castro, J. (2018). Digital image tamper detection technique based on spectrum analysis of cfa artifacts. *Sensors*, 18(9), 2804.

Hamilton Jr, J.F. and Adams Jr, J.E. (1997). Adaptive color plan interpolation in single sensor color electronic camera. Document, US Patent, 5,629,734.

Huh, M., Liu, A., Owens, A., Efros, A.A. (2018). Fighting fake news: Image splice detection via learned self-consistency. In *European Conference on Computer Vision*. ECCV, Munich.

Iakovidou, C., Zampoglou, M., Papadopoulos, S., Kompatsiaris, Y. (2018). Content-aware detection of JPEG grid inconsistencies for intuitive image forensics. *Journal of Visual Communication and Image Representation*, 54, 155–170.

Immerkær, J. (1996). Fast noise variance estimation. *Computer Vision and Image Understanding*, 64(2), 300–302.

Kirchner, M. (2010). Efficient estimation of CFA pattern configuration in digital camera images. In *Media Forensics and Security Conference*. IS&T, San Jose.

Lebrun, M., Colom, M., Morel, J.-M. (2013). Secrets of image denoising cuisine. *Image Processing On Line*, 2013, 173–197.

Lebrun, M., Colom, M., Morel, J. (2015). Multiscale image blind denoising. *IEEE Transactions on Image Processing*, 24(10), 3149–3161.

Lee, J.-S. (1981). Refined filtering of image noise using local statistics. *Computer Graphics and Image Processing*, 15(4), 380–389.

Lee, J.-S. and Hoppel, K. (1989). Noise modeling and estimation of remotely-sensed images. In *12th Canadian Symposium on Remote Sensing Geoscience and Remote Sensing Symposium*. IEEE, Vancouver.

Li, W., Yuan, Y., Yu, N. (2009). Passive detection of doctored JPEG image via block artifact grid extraction. *Signal Processing*, 89(9), 1821–1829.

Lin, W., Tjoa, S., Zhao, H., Ray Liu, K. (2009). Digital image source coder forensics via intrinsic fingerprints. *IEEE Transactions on Information Forensics and Security*, 4(3), 460–475.

Liu, C., Freeman, W.T., Szeliski, R., Kang, S.B. (2006). Noise estimation from a single image. In *Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, Washington DC.

Losson, O. and Dinet, E. (2012). From the sensor to color images. In *Digital Color – Acquisition, Perception, Coding and Rendering*, Fernandez-Maloigne, C., Robert-Inacio, F., Macaire, L. (eds). ISTE Ltd, London and John Wiley & Sons, New York.

Lowe, D.G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2), 91–110.

Mahdian, B. and Saic, S. (2009). Using noise inconsistencies for blind image forensics. *Image and Vision Computing*, 27(10), 1497–1503.

Mastin, G.A. (1985). Adaptive filters for digital image noise smoothing: An evaluation. *Computer Vision, Graphics, and Image Processing*, 31(1), 103–121.

Mayer, O. and Stamm, M.C. (2020). Forensic similarity for digital images. *IEEE Transactions on Information Forensics and Security*, 15, 1331–1346.

Minami, S. and Zakhor, A. (1995). An optimization approach for removing blocking effects in transform coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(2), 74–82.

Nikoukhah, T., Anger, J., Ehret, T., Colom, M., Morel, J., Grompone von Gioi, R. (2019). JPEG grid detection based on the number of DCT zeros and its application to automatic and localized forgery detection. In *Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, Long Beach.

Nikoukhah, T., Colom, M., Morel, J.-M., Grompone von Gioi, R. (2020). Local JPEG grid detector via blocking artifacts, a forgery detection tool. *Image Processing On Line*, 10, 24–42.

Olsen, S.I. (1993). Estimation of noise in images: An evaluation. *CVGIP: Graphical Models Image Processing*, 55(4), 319–323.

Pan, X., Zhang, X., Lyu, S. (2011). Exposing image forgery with blind noise estimation. In *Multimedia Workshop on Multimedia and Security*. ACM, Buffalo.

Pevny, T. and Fridrich, J. (2008). Detection of double-compression in JPEG images for applications in steganography. *IEEE Transactions on Information Forensics and Security*, 3(2), 247–258.

Ponomarenko, N.N., Lukin, V.V., Zriakhov, M.S., Kaarna, A., Astola, J. (2007). An automatic approach to lossy compression of aviris images. In *International Geoscience and Remote Sensing Symposium*. IEEE, Barcelona.

Popescu, A. and Farid, H. (2005). Exposing digital forgeries in color filter array interpolated images. *Transactions on Signal Processing*, 53(10), 3948–3959.

Rank, K., Lendl, M., Unbehauen, R. (1999). Estimation of image noise variance. *IEEE Proceedings – Vision, Image and Signal Processing*, 146(2), 80–84.

Shin, H.J., Jeon, J.J., Eom, I.K. (2017). Color filter array pattern identification using variance of color difference image. *Journal of Electronic Imaging*, 26(4), 1–12.

Thai, T.H., Cogranne, R., Retraint, F., Doan, T. (2017). JPEG quantization step estimation and its applications to digital image forensics. *IEEE Transactions on Information Forensics and Security*, 12(1), 123–133.

Voorhees, H. and Poggio, T. (1987). Detecting textons and texture boundaries in natural images. In *International Conference on Computer Vision*. IEEE, London.

Wallace, G.K. (1992). The JPEG still picture compression standard. *IEEE Transactions on Consumer Electronics*, 38(1), xviii–xxxiv.

Wang, S.-Y., Wang, O., Owens, A., Zhang, R., Efros, A.A. (2019). Detecting photoshopped faces by scripting photoshop. In *International Conference on Computer Vision*. IEEE, Seoul.

Wen, B., Zhu, Y., Subramanian, R., Ng, T.-T., Shen, X., Winkler, S. (2016). Coverage – A novel database for copy-move forgery detection. In *International Conference on Image Processing*. IEEE, Phoenix.

Wu, Y., Abd-Almageed, W., Natarajan, P. (2018). Busternet: Detecting copy-move image forgery with source/target localization. In *European Conference on Computer Vision*. IEEE, Munich.

Yao, H., Wang, S., Zhang, X., Qin, C., Wang, J. (2017). Detecting image splicing based on noise level inconsistency. *Multimedia Tools and Applications*, 76(10), 12457–12479.

Ye, S., Sun, Q., Chang, E.-C. (2007). Detecting digital image forgeries by measuring inconsistencies of blocking artifact. In *International Conference on Multimedia and Expo*. IEEE, Beijing.

Zhou, P., Han, X., Morariu, V.I., Davis, L.S. (2018). Learning rich features for image manipulation detection. In *Conference on Computer Vision and Pattern Recognition*. IEEE, Salk Lake City.

2

Deep Neural Network Attacks and Defense: The Case of Image Classification

Hanwei ZHANG, Teddy FURON, Laurent AMSALEG and Yannis AVRITHIS

IRISA, University of Rennes, Inria, CNRS, France

Machine learning using deep neural networks applied to image recognition works extremely well. However, it is possible to modify the images very slightly and intentionally, with modifications almost invisible to the eye, to deceive the classification system into misclassifying such content into the incorrect visual category. This chapter provides an overview of these intentional attacks, as well as the defense mechanisms used to counter them.

2.1. Introduction

Deep neural networks have made it possible to automatically recognize the visual content of images. They are very good at recognizing what is in an image and categorizing its content into predefined visual categories. The vast diversity of the many images that are used to train a deep network allows it to recognize visual content with a high degree of accuracy and a certain capacity for generalization. From thousands of examples of images of animals, manufactured objects, places,

For a color version of all figures in this chapter, see www.iste.co.uk/puech/multimedia1.zip.

Multimedia Security 1,

coordinated by William PUECH.

Multimedia Security 1: Authentication and Data Hiding,
First Edition. William Puech.

people, elements of flora, etc., a deep neural network can almost certainly detect that an unknown image shows a dog, a cat, an airplane.

However, it is possible to intentionally modify these images so that the network is completely wrong in its classification. These modifications are made by an attacker whose goal is to deceive the classification, for example to pass off inappropriate content (child pornography) as something perfectly harmless. The big surprise is that these modifications are very small and are almost imperceptible to our eyes. These attacks take advantage of a certain vulnerability of deep networks, which can be quite easy to fool, as we will show in this chapter.

Passing off one piece of visual content for another is very problematic. Putting small pieces of paper of a particular shape and color in certain places on road signs prevents their automatic recognition by dashboard cameras in autonomous vehicles. Wearing a medallion decorated with a particular texture on clothing can make a person invisible to an algorithm detecting the presence of pedestrians. The examples multiply, and are sometimes funny, sometimes disturbing and sometimes dangerous when the decisions of the network puts lives at stake.

Adversarial images that are capable of deceiving classifiers are defined in section 2.2, and an overview of attacks intended to deceive a classifier whose technology is based on deep neural networks is provided in section 2.3. In response, many studies propose defenses, and section 2.4 aims to present them.

We will begin this chapter with a short presentation of the history of the field, and the vocabulary that we will be using. We will also present the main features of machine learning and image classification by deep neural networks.

2.1.1. *A bit of history and vocabulary*

This chapter deals with the vulnerabilities of deep neural networks, but in fact *all* machine learning algorithms have flaws and are vulnerable to intentional attacks. It was while researchers were working on automatic email classification in an attempt to separate *spam* from real messages that the first flaws were revealed. It was the work of Dalvi and his colleagues, and also Lowd and Meek in 2004, that showed that it was possible to deceive a linear classifier trained to detect spam (Dalvi *et al*. 2004; Lowd and Meek 2005). At that time, deep networks did not exist, and the techniques to choose from for machine learning processes relied on classifiers based on support-vector machines in particular.

Ten years later, the *adversarial machine learning* sector is gaining momentum, because at the moment the incredible power of deep networks is being revealed, but

at the same time they are very vulnerable. Around 2014, researchers were working on making images that could deceive a classifier based on a deep neural network. These images are called “adversarial images”.

Therefore, adversarial images are images that have been manipulated so that the network that classifies them is mistaken and assigns these images to an erroneous class. For example, if a image of a cat is shown, the network responds that this image is of an airplane. However, manipulation is *almost invisible* and by looking at the manipulated image, it looks exactly like a cat. The network itself is very confident in its decision that it is an airplane. Figure 2.1 illustrates this, where the American flag, when altered intentionally, is recognized as a vending machine, or even a sandal.

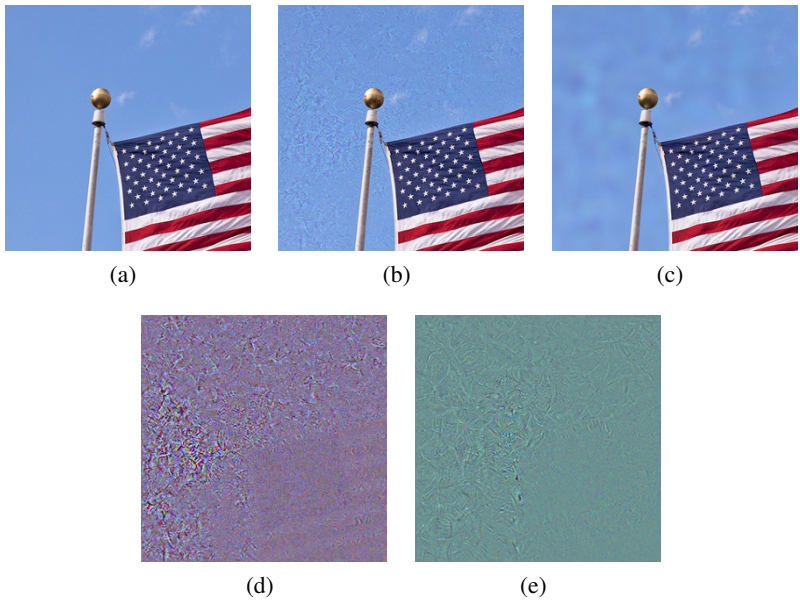


Figure 2.1. *The original image and adversarial images; the manipulations are almost invisible, the classification is incorrect*

COMMENT ON FIGURE 2.1.– *(a) The original image, classified correctly as a flag. (b) An adversarial image, created using the C&W method, classified as a vending machine by the network. (c) An adversarial image, created using the PGD_2 method, classified as a sandal by the network. (d) Distortion (greatly amplified to make it more visible) exists in image (b) and is created using C&W, making the original adversarial image. (e) Highly amplified distortion existing in image (c) and created using PGD_2, making the original adversarial image.*

Historically, one of the first studies into the many facets of vulnerability in learning algorithms was carried out by Barreno *et al*. (2006). In this seminal article, they discuss the vulnerability of artificial learning algorithms during the learning phases (they discuss poisoning) and the test phases (they discuss evasion), they distinguish between targeted attacks and non-targeted attacks, propose techniques for evaluating the power of these attacks and mention some defense mechanisms to avoid them. They also differentiate vulnerabilities according to the knowledge an attacker may have of the system they want to deceive, distinguishing "white box attacks" from "black box attacks". We will come back to these terms in section 2.3.

A very good historical perspective can be found in the article by Biggio and Roli (2018). Some excellent recent and complete developments are suggested by Serban and Poll (2018). We recommend reading these two publications.

2.1.2. *Machine learning*

Machine learning is part of the artificial intelligence field. Machine learning is an interdisciplinary field, with a mix of applied mathematics, statistics and algorithmics. It enables a computer to perform a task based on the careful examination of representative data. The set of rules that the machine must follow in order to perform a task is often either impossible to list "manually" because it is too complex (automatic translation for example), or defined, but leads to an exponential combination of behaviors given all possible input data (chess, go). Machine learning is specifically based on the analysis of huge amounts of data to estimate a model for performing the task. The more data and the more diverse the data, the better the model and the automatic fulfillment of the target task.

Machine learning has two phases. The learning or training phase first learns a model from a set of training data. The second phase applies the learned model to new data and therefore carries out the task. Sometimes, learning and application are intertwined in an effort to continuously improve the quality of the model.

The nature of the information available during the learning phase determines two types of approaches. Supervised learning uses data-label pairs, with the label being the responses we want the task to produce for each data item. It is then a question of classification of the data when the labels have discrete or categorical values, or of regression if they are continuous. On the other hand, non-supervised learning does not have labels. Since it is not always possible to label the large amounts of data, intermediate approaches have been designed where the degree of supervision is more or less high. There are so-called semi-supervised (the data are not all labeled), or even partially supervised (only some of the labels relevant to given data are provided)

approaches. There are also other forms of learning, such as reinforcement learning and transfer learning, but we will not go into them in detail here.

The fields of application are extremely varied, as are the tasks to be carried out: classification tasks, recognition tasks, translation tasks, grouping tasks, analysis tasks and prediction tasks; the list is almost endless. Learning applies to data of very different natures: symbolic, digital data, data of a continuous or discrete nature, graphs, trees, feature vectors, including images, sounds, texts and time series for example.

In this chapter, we focus on a specific type of data and the particular task of classifying images into predefined visual categories. The labels are associated with images, for example, airplane, boat, car, table, chair, building, person, dog, cat, ball, cutlery, daisy or tomato. We now consider a supervised learning environment. Once the model has been learned, it is a question of classifying new unknown images, without labels, into the right visual category or categories as best as possible.

Any machine learning process is built on a few fundamental concepts, whether technical or theoretical, that we will detail in this section. First is the concept of an objective function. This very generic term designates a function that reflects the performance of the model; in other words, its ability to perform the intended task. The training optimizes the parameters of the model in order to gradually maximize the objective function. This function is often a decreasing function of a global error linked to the incomplete performance of the task. The opposite of a mean squared error or a cross-entropy are two classic examples of objective functions. These functions are continuous with respect to the parameters of the model and make it possible to detect whether or not a small modification of the parameters improves the performance of the task.

The improvement of the learning is achieved by a gradual adjustment of the parameters of the model, so that the new values of these parameters increase the quality of the model observed through the objective function. Therefore, an optimization process is at work which, we hope, will find the optimum parameters without making the search for them too costly. The adjustment consists of a better positioning of a hyper-plane separating two families of data, for example.

After optimization, the learned model works well on the data used in training, which is normal. However, it is important that this model has generalization capabilities, so that, it can correctly process new and unknown data. Sometimes, when the model over-fits, it has little or no generalization capabilities. This must be avoided, so techniques such as cross-validation, regularization or random pruning could be used.

2.1.3. *The classification of images by deep neural networks*

This section describes what deep neural networks are and what the image classification task is. This section only provides an overview of these concepts and only presents the most important ones. We invite the reader to study the work by Goodfellow *et al.* (2016), which presents these concepts in much more detail and precision. In addition, this section introduces the mathematical notations needed to describe attacks and defenses later on.

In the red, green and blue color space, an image $\mathbf{I}$ of L lines and C columns is represented by a three-dimensional table in space $\{\mathbf{0}, \mathbf{1}, \ldots, \mathbf{255}\}^{\mathbf{3}\times\mathbf{L}\times\mathbf{C}}$. The pixels are integers between 0 and 255 (if coded on one byte). The output of the classifier is a class, that is, categorical data. The k possible categories (for example, airplanes, boats, cars, tables, chairs, buildings, people, dogs, cats) are ordered arbitrarily and the output of the classifier is an integer between 1 and k, denoted by $\hat{\ell}$.

The image classifying deep neural network here is schematically broken down into three levels. The first layer performs preprocessing that adapts the input image to the neural network. It often includes a sub-sampling of the image to a given size $r \times r$ (typically 224×224), and mainly reduces the dynamics of the pixels to the range $[0, 1]$ (a historical choice, but other choices are possible, like, for example reducing toward $[-1, 1]$). This can be done by dividing the pixel value by 255. More sophisticated transfer functions, which are sometimes different from one color channel to another, are also used. The output of this preprocessing is $\mathbf{x} = \mathsf{T}(\mathbf{I})$, traditionally noted as a column vector at $m = 3 \times r \times r$ components in $[0, 1]^m$.

The second layer is the neural network. A neuron is a small automaton that combines the data it receives from other neurons, and produces a value which is then transmitted to one or more neurons, which will each combine that value with the values received from other neurons and so on, which makes up an overall network. Neurons are often organized in layers, connected to each other, and it is the great multiplication of these layers that gives the term "deep". For example, there are networks made up of hundreds of interconnected layers, each made up of thousands of artificial neurons.

Therefore, a neuron is a small automaton that first operates a linear combination of the values received (from other neurons, for example), which are weighted by synaptic weights, and then added up. The value produced is then passed to an activation function, also called a thresholding function. Such functions introduce nonlinearity into the behavior of the neuron, which is essential. Sigmoid activation functions, hyperbolic tangent function, or Rectified Linear Unit (ReLU)-based functions are often used. These nonlinear functions are continuous, non-decreasing and are almost universally differentiable.

The output of the neural network is a real vector at k components called the logit vector: $\mathbf{z} = \mathsf{R}(\mathbf{x}, \boldsymbol{\theta}) \in \mathbb{R}^k$. The larger $z(j)$, the jth logit, is, the more likely the input data $\mathbf{x}$ belongs to class j. The $\boldsymbol{\theta}$ symbol is a "catch-all" parameter, representing the set of synaptic weights (of all neurons in all layers).

The third layer translates the logits into a probability vector $\mathbf{p} \in [0,1]^k$ s.t. $\sum_{j=1}^{k} p(k) = 1$. The value $p(j)$ is the probability that the input image is of class j. This translation is done with the function *softmax*, $\mathbf{p} = \mathsf{S}(\mathbf{z})$, defined by, $\forall\, 1 \leq j \leq k$:

$$p(j) = \frac{e^{z(j)}}{\sum_{i=1}^{k} e^{z(i)}} \qquad [2.1]$$

It is the gradual adjustment of the synaptic weights $\boldsymbol{\theta}$, parameters of the function $\mathsf{R}(\cdot)$, which forms the core of supervised learning, the first and the last layer being non-parametric. These weights are gradually adjusted so that the final value produced at the output of the classifier *ultimately* corresponds to the label associated with the input data.

The network is used in propagation mode when the input data gradually passes through it, and the network eventually produces the probability vector $\mathbf{p}$. The error between the output $\mathbf{p}$ and what should have been produced is measured. For a label ℓ associated with the input $\mathbf{x}$, the output is ideally a probability vector $\mathbf{p}_\ell^\star$ where $p_\ell^\star(j) = 1$ and where the other components of this vector are zero. The cross-entropy $h(\mathbf{p}, \mathbf{p}_\ell^\star)$ is a metric quantifying how $\mathbf{p}$ is different to $\mathbf{p}^\star$. The loss for the input $\mathbf{x}$ of the label ℓ is the number $\mathcal{L}(\mathbf{x}, \ell, \boldsymbol{\theta}) = h(\mathsf{S}(\mathsf{R}(\mathbf{x}, \boldsymbol{\theta})), \mathbf{p}_\ell^\star)$.

Backpropogation consists of tracing the error made by a neuron back through the network, from downstream to upstream, to its synapses and therefore to the upstream neurons. The gradient of the cross-entropy is calculated like this. This is greatly simplified by a chain calculus because the network is a composition of functions or layers. Therefore, the set of these weights $\boldsymbol{\theta}$ is updated iteratively by a gradient descent algorithm to decrease the cross-entropy. At iteration i:

$$\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathbf{x}_{j(i)}, \ell_{j(i)}, \boldsymbol{\theta}) \qquad [2.2]$$

where $\{\mathbf{x}_{j(i)}, \ell_{j(i)}\}$ corresponds to training data drawn at random at the ith iteration of the stochastic descent gradient and $\eta > 0$ is the learning rate.

It is this constant back and forth between propagation (calculated from $\mathsf{S}(R(\mathbf{x},\boldsymbol{\theta}))$) and backpropagation (calculated from $\nabla_{\boldsymbol{\theta}} h(S(R(\mathbf{x},\boldsymbol{\theta})),\mathbf{p}_\ell^\star)$), which ensures the convergence of the learning toward local minimum weights $\boldsymbol{\theta}$ of the cross-entropy calculated on the training datasets.

Once the learning is over, the test phase can start. During that phase, unknown data is checked against the model, which eventually produces a probability vector $\mathbf{p}$. The class predicted for the tested data is the one associated with the greatest probability observed in $\mathbf{p}$, that is:

$$\hat{\ell} = \mathsf{f}(\mathbf{x}) := \arg \max_{1\leq i\leq k} p(i) \qquad [2.3]$$

2.1.4. Deep Dreams

A loss $\mathcal{L}(\mathbf{x},\ell,\boldsymbol{\theta})$ is a continuous function with respect to parameters $\boldsymbol{\theta}$ of the network, but also to the input data $\mathbf{x}$. In equation [2.2], it is its gradient with respect to $\boldsymbol{\theta}$ that appears. What does the opposite of the gradient mean, with respect to $\mathbf{x}$, $-\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x},\ell,\boldsymbol{\theta})$? It is a three-dimensional table with the same dimension as $\mathbf{x}$, which indicates what tiny modification should be made to $\mathbf{x}$ to reduce the loss, that is, so that the input data is even better classified as belonging to the category ℓ.

This idea is at the root of *Deep Dreams* (Tual and Coutagne 2015; Wikipedia 2020), the psychedelic images, which are shown in Figure 2.2. They were built by increasing the structures recognized by a network in a given image and show us what allows it to predict the category ℓ.

These images where the recognized structures are amplified excessively are now part of the folklore of deep convolutional networks. However, this idea of calculating a gradient rather than a variable $\mathbf{x}$ is largely used to visually understand and interpret the decisions of neural networks. References in this field include the papers by Yosinski *et al.* (2015) and Simonyan *et al.* (2014).

This raises questions about the quantity $+\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x},\ell,\boldsymbol{\theta})$. Added to the variable $\mathbf{x}$, it decreases the probability $p(\ell)$: this perturbation erases the typical structures of the class ℓ and the resulting image is not as well recognized as being in class ℓ. In the same way, a perturbation $-\nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x},\ell',\boldsymbol{\theta})$ with $\ell' \neq \ell$ increases the probability that the image is in class ℓ', which can lead to a wrong classification. This is, in fact, the basic idea for generating white box adversarial images.

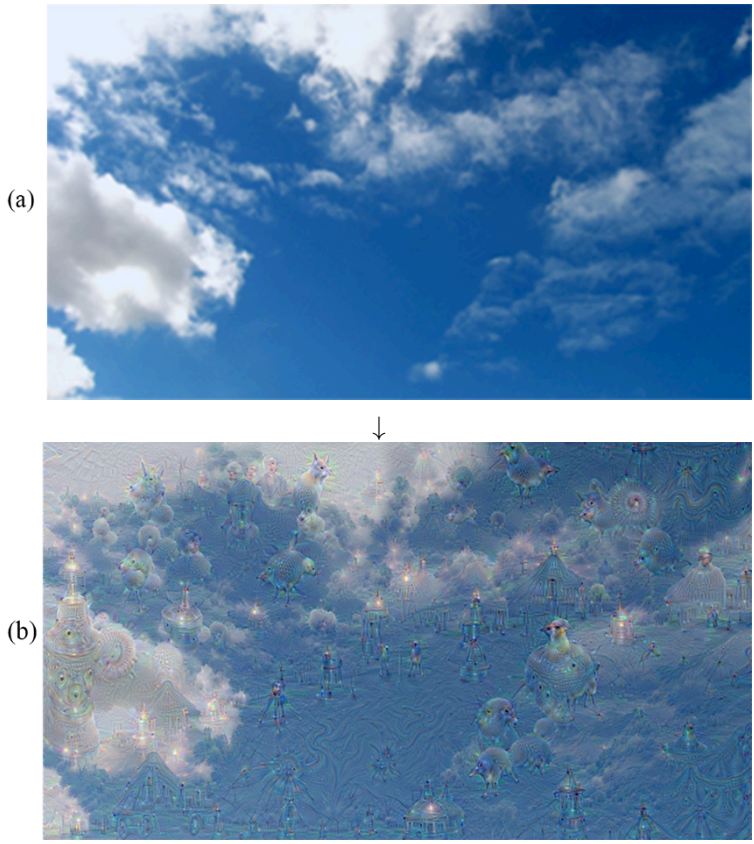


Figure 2.2. *Illustration of the Deep Dreams process applied to the original image (a) (source: (Wikipedia 2020))*

2.2. Adversarial images: definition

Let $\mathrm{f} : \mathcal{R}^m \rightarrow \{1, \cdots, k\}$ be a classifier mapping a vector of pixels (forming an image) to a discrete label, giving the class that this vector belongs to, among k possible classes. For an image $\mathbf{x} \in \mathcal{R}^m$ and a target label $\ell \in \{1, \cdots, k\}$, an adversarial perturbation $\mathbf{r}$ is produced by resolving the following optimization problem:

$$\begin{aligned} &\min \|\mathbf{r}\|_p \\ &\textit{such that } \mathsf{f}(\mathbf{x}+\mathbf{r}) = \ell \\ &\qquad (\mathbf{x}+\mathbf{r}) \textit{ is an image} \end{aligned} \qquad [2.4]$$

This equation fits with targeted and non-targeted attacks, as well as white box or black box attacks.

Targeted attacks are those where the attacker wants to see the attacked image classified exactly in the category ℓ (e.g. the attacker wants an image of a dog to be classified as an image of a cat). *Non-targeted attacks* are those where the attacker wants an attacked image $\mathbf{x} + \mathbf{r}$ to be classified in whichever class, as long as ℓ is different from the class $\mathsf{f}(\mathbf{x})$ that it belongs to (e.g. the attacker wants an image of a dog to be classified as anything, except a dog).

White box attacks consider the scenario where the attacker knows everything about the classifier network. They can thus imitate it in their garage, assemble it and test an attack, then once operational, they can deploy it. *Black box attacks* consider the opposite, that the attacker does not know the network details. On the contrary, they have a model of the classifier in their garage. They cannot "open" it and see how it works, but they can use it as an oracle; in other words, submit images to it and observe its predictions as many times as they want to. Certain articles consider "gray boxes", that is, contexts where the attacker only has partial knowledge of the network. We will come back to all of this later in section 2.3.2.

Equation [2.4] contains a distortion term to be minimized. It is frequently the magnitude of the distortion that is measured, often according to the norm L_2, L_1, L_∞ (Goodfellow *et al.* 2014a; Carlini and Wagner 2017), since they are quite intuitive, and, when the measure gives a very weak distortion, it is then often almost invisible to the eye. However, these metrics do not reflect our perception of images and small distortions can sometimes be very visible. Also, it is clear that we want to measure the distortion according to another metric, rather than one based on the way our visual system works from a neurological and psychological point of view (Wang *et al*. 2004; Sharif *et al*. 2018; Fezza *et al*. 2019). Unfortunately, this metric is much more complex to calculate. Therefore, the L_p norms are often favored in practice.

Equation [2.4] defines adversial images via an optimization problem. The attack is the process implemented to find the solution to this problem. The variable $\mathbf{x}$ existing in a space of a very large dimension m. Finding this solution is difficult since the function $\mathsf{f}(\cdot)$ has no explicit form. It is likely that an attack actually finds an approximate solution; in other words, it finds a perturbation $\mathbf{r}$ of greater distortion than the bare minimum, given by the solution of equation [2.4]. *So a first criterion to evaluate the quality of an attack is the distortion it produces*.

A second criterion is given by the second line of equation [2.4]. An attack achieves its goal (targeted or non-targeted) if $\mathsf{f}(\mathbf{x}+\mathbf{r}) = \ell$. But this problem is so difficult to resolve that an attack can sometimes fail, failing to produce an adversial image $(\mathbf{x}+\mathbf{r})$. *So the second criterion to evaluate the quality of an attack is the probability of its success*.

These two criteria are deeply linked by a trade-off. It is easy to develop an attack that always succeeds: this attack replaces $\mathbf{x}$ by a whole other image $\mathbf{x}'$, whose predicted class is ℓ, but the distortion $\|\mathbf{x}' - \mathbf{x}\|_p$ is huge and is clearly visible to the naked eye. It is easy to develop a zero distortion attack: it is the attack which uses the untouched $\mathbf{x}$, but the probability of success is zero (unless $\mathbf{x}$ is immediately misclassified by the network). These two extreme attacks are pointless, but they illustrate this trade-off. The probability of success is an increasing function of distortion.

The third criterion is the complexity of the algorithm, measured by its memory consumption or by the required computation time. The algorithms listed below are often iterative, and counting the number of iterations that are needed to make an adversarial image of good visual quality is a valuable indicator. The lower the complexity, the faster the attack, but then the probability of success is often very low or the distortion is very high.

Before presenting different algorithms attacking networks by producing adversarial images, let us come back to the last term of equation [2.4]. It is said that $(\mathbf{x}+\mathbf{r})$ $is\ an\ image$. Let us see what this means and what it involves.

This definition, which is based on $\mathbf{x} = \mathsf{T}(\mathbf{I})$ and not on image $\mathbf{I}$, is historical. It reflects the fact that the community working on computer vision and those working on neural networks are not interested in the preprocessing layer because there is nothing to learn or train. So the condition $(\mathbf{x}+\mathbf{r})$ $is\ an\ image$ simply means that $\mathbf{x}+\mathbf{r} \in [0,1]^m$, just like $\mathbf{x}$. It would be possible to come to a "real" digital image (with values of pixels between 0 and 255) by simply applying the inverse preprocessing $\mathsf{T}^{-1}(\cdot)$.

In reality, from our point of view, things are not that simple. We have described the preprocessing as being part of the classifier. So $\mathbf{x}$ is an internal variable that the classifier does not have access to. However, the aim of the attacker is to attack the input image $\mathbf{I}$, and not $\mathbf{x}$. In addition, the method of first finding $\mathbf{x}+\mathbf{r}$, and then applying the inverse preprocessing to form an image is sometimes not possible, because there is no such image such that its preprocessing gives $\mathbf{x}+\mathbf{r}$. We will come back to this later in this chapter.

2.3. Attacks: making adversarial images

This section gives a quick overview of techniques used to produce adversarial images. We are not exhaustive, we will describe attacks which are, in a way, exemplary

of what the literature offers, very vast literature that is rapidly expanding. All of the techniques presented here are based on equation [2.4], which they enrich in a number of ways.

We mainly describe the case where the network is perfectly known to the attacker (*white box attack*) and the attacks are not targeted (*untargetted attacks*). We will extend this framework at a later stage.

2.3.1. *About white box*

2.3.1.1. *The attacker's objective function*

In this scenario, the attacker has access to the probability vector $\mathbf{p}$, calculated by the network. With the original image being from class ℓ_g, the vector $\mathbf{p}$ must move away from $\mathbf{p}^\star_{\ell_g}$ (the image is less recognized as being from class ℓ_g). In an attack targeting the class ℓ, $\mathbf{p}$ must get closer to $\mathbf{p}^\star_\ell$. Like supervised learning, the attacker also works with an objective function defined via cross-entropy:

$$\mathcal{J}(\mathbf{x}, \ell) = h(\mathbf{p}, \mathbf{p}^\star_\ell) - h(\mathbf{p}, \mathbf{p}^\star_{\ell_g}) \quad [2.5]$$

$$= \log(p(\ell_g)) - \log(p(\ell)) \quad [2.6]$$

Decreasing the objective function amounts to increasing the predicted probability for the class ℓ and decreasing that of the original class ℓ_g. Note that a perturbation makes the image adversarial if $\mathcal{J}(\mathbf{x} + \mathbf{r}, \ell) < 0$.

The definition of the objective function is more difficult for non-targeted attacks. Decreasing only $h(\mathbf{p}, \mathbf{p}^\star_{\ell_g})$ is not enough. The first trick is to target the most probable class, other than ℓ_g. Hence an objective function:

$$\mathcal{J}(\mathbf{x}) = \log(p(\ell_g)) - \max_{\ell \neq \ell_g} \log(p(\ell)) \quad [2.7]$$

There are other objective functions in the literature. We will see, for example, the DeepFool attack, which detects that a predicted high probability class is not necessarily an easier class to reach.

2.3.1.2. *Two big families*

The core of equation [2.4] is formed by minimizing a distortion and successfully deceiving the system. Also, the algorithms producing adversarial images are divided into two families, according to whether they set themselves the objective of never exceeding a distortion whose maximum value is specified, or the objective of succeeding in producing adversarial images that will all be able to deceive the system, without limiting the distortion (although the minimum is sought). Let us characterize these two families before listing the algorithms.

2.3.1.2.1. Distortion objective

All of the algorithms producing adversarial images included in this family aim to maximize the probability of success while not exceeding a fixed distortion. This distortion is not necessarily an explicit parameter of the attack, but it is determined by some of its parameters. This is generally expressed by:

$$\begin{aligned} &\min \mathcal{J}(\mathbf{x}+\mathbf{r}) \\ &such\ that\ \|\mathbf{r}\|_p \leq \epsilon \end{aligned} \quad [2.8]$$

where $\mathcal{J}$ is the lossy function of the attacker and ϵ is the maximum distortion allowed.

The performance of this type of attack is measured by their probability of success $P_{suc} = \mathcal{P}(\mathsf{f}(\mathbf{x}+\mathbf{r}) \neq \ell_g)$, which of course depends on the value given to ϵ. If the attack does not succeed for a given value of ϵ, then this value can be increased and the algorithm starts again with this new, larger ϵ. Note that the ϵ, having finally made it possible to create an adversarial image, is not necessarily minimal.

2.3.1.2.2. Success target

In this family, algorithms aim for success and always produce an adversarial image at the cost of arbitrary, but minimal, distortion. This is expressed by:

$$\begin{aligned} &\min \|\mathbf{r}\|_p \\ &such\ that\ \mathcal{J}(\mathbf{x}+\mathbf{r}) < 0 \end{aligned} \quad [2.9]$$

It is the minimum distortion expectation that characterizes the performance of these algorithms once the network is fooled.

These two families of attacks use a development limited to the first order of the objective function as a basic principle:

$$\mathcal{J}(\mathbf{x}+\mathbf{r}) = \mathcal{J}(\mathbf{x}) + \mathbf{r}^\top \nabla_{\mathbf{x}} \mathcal{J}(\mathbf{x}) + o(\|\mathbf{r}\|) \quad [2.10]$$

The distortions $\mathbf{r}$ which decrease the objective function are therefore positioned toward the opposite of the gradient $-\nabla_{\mathbf{x}} \mathcal{J}(\mathbf{x})$. This approximation, being local, is only valid for the distortions of low amplitude.

From the value of the objective function, a backpropagation process is initiated, which goes back to the vector representing the image while keeping the synaptic weights unchanged. Then, it is this vector that modifies the original image so that the system is eventually fooled. The perturbation is, therefore, a function of the gradient. It is worth noting that because of auto-differentiation (Goodfellow *et al.* 2016), the calculation of the gradient is automatic. On the other hand, the complexity of this calculation is just double that of the propagation. Let us now list the main algorithms belonging to these two families.

2.3.1.3. *Distortion objective: main attacks*

2.3.1.3.1. FGSM

The first algorithm that uses the gradient to create a perturbation and produces an adversarial image is the one proposed by Goodfellow *et al.* (2014a). It is FGSM, which stands for the *Fast Gradient Sign Method*. This method is very simple and depends on a perturbation calculated as:

$$\mathbf{y} = \mathbf{x} + \mathbf{r} = \mathbf{x} - \epsilon \operatorname{sign} \nabla_{\mathbf{x}} \mathcal{J}(\mathbf{x}) \qquad [2.11]$$

It is this perturbation that minimizes the objective function to the first order for the constraint $\|\mathbf{r}\|_\infty = \epsilon$.

The characteristic elements of attacks whose objective is distortion are found here (see equation [2.8]). By studying the gradient of the objective function $\mathcal{J}$, and by calculating the opposite, it is then possible to determine how to modify the vector at the input of the network to decrease $\mathcal{J}$ and hopefully lead to its misclassification. The value of ϵ controls the maximum distortion allowed. This method is very simple and very quickly creates adversarial images that can sometimes mislead the classifier. However, it is a bit rough, since it only uses one observation of the gradient to determine which perturbation to apply.

2.3.1.3.2. I-FGSM

It is simple to refine FGSM by having it observe the gradient multiple times as the perturbation is created. So I-FGSM (Kurakin *et al*. 2016) is the iterative version of FGSM. Contrary to what equation [2.11] allows, the perturbation is not directly calculated. I-FGSM initializes $\mathbf{y}_0 := \mathbf{x}$ and then iterates by increasing the inverse of the gradient each time by α. The recurrence is therefore:

$$\mathbf{y}_{i+1} := \operatorname{proj}_{B_\infty[\mathbf{x};\epsilon]}(\mathbf{y}_i - \alpha \operatorname{sign} \nabla_{\mathbf{x}} \mathcal{J}(\mathbf{y}_i)) \qquad [2.12]$$

where $\operatorname{proj}_{\mathcal{A}}$ is the estimation on the region $\mathcal{A}$ (in the minimum sense of the Euclidean space) followed by a term-to-term threshold to stay in the Hypercube $[0, 1]$.

Here, the region $\mathcal{A}$ is the ball $B_\infty[\mathbf{x}; \epsilon]$ of L_∞ norm, center $\mathbf{x}$ and radius $\epsilon > \alpha$. Therefore, the first iterations remain inside the ball and the projection is not active, then the iterations calculate perturbations that get out of the hyper ball and the projection brings them back to its surface. This iterative approach is also known as the *Basic Iterative Method* (BIM) (Papernot *et al*. 2018).

I-FGSM and BIM carry out targeted or non-targeted attacks, and it all depends on the definition of the objective function.

2.3.1.3.3. PGD_2

Projected Gradient Descent is also an iterative method, but it projects the gradient on a ball of norm L_2, and not on ball of norm L_∞ (Madry *et al*. 2017). Therefore:

$$\mathbf{y}_{i+1} := \text{proj}_{B_2[\mathbf{x};\epsilon]}(\mathbf{y}_i - \alpha\eta(\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{y}_i))) \quad [2.13]$$

where $\eta(\mathbf{x}) := \mathbf{x}/\left\|\mathbf{x}\right\|$, which is a normalization, according to the norm L_2.

The ball L_2 used for the projection is $B_2[\mathbf{x};\epsilon]$, center $\mathbf{x}$ and radius ϵ. Once again, the attack does not end when $\mathbf{y}_i$ touches the ball $B_2[\mathbf{x};\epsilon]$ for the first time. It continues and seeks to minimize the objective function while remaining on the ball.

2.3.1.3.4. M-IFGSM

Iterative approaches progress along the gradient at a fixed pace, symbolized by α (see equations [2.12] and [2.13]). Adjusting this value is difficult: if it is too small the algorithms will not progress and find an adversarial image because the number of iterations is limited; if it is too large the algorithms will progress quickly, but then the gradient cannot be followed finely, which can create fluctuations.

Approaches such as M-IFGSM (Dong *et al*. 2018) incorporate a progressive adaptation mechanism for the pace: during the first iterations, it is an advantage to progress rapidly along the gradient. On the other hand, later, it is better to progress in small steps to better follow the gradient and reach a local minimum.

2.3.1.4. *Success goal: main attacks*

Techniques in this family are typically more expensive. The discovery of a near adversarial image is guaranteed if the complexity is not limited.

2.3.1.4.1. L-BFGS

Szegedy *et al*. (2013) discuss the problem of creating adversarial images using a Lagrangian formulation. Distortion is no longer a constraint, but is integrated into the objective function:

$$\ell(\mathbf{r}) := \mathcal{J}(\mathbf{x}+\mathbf{r}) + \lambda * \left\|\mathbf{r}\right\|^2 \quad [2.14]$$

For a given value of $\lambda > 0$, the minimization of the objective function without constraint is carried out using the numerical method BFGS (Broyden– Fletcher–Goldfarb–Shanno). It is an iterative gradient descent method.

A strong value for λ means that the minimum $\mathbf{r}^\star$ of the objective function is not an adverse perturbation, because it has given too much weight to the Euclidean distortion. On the contrary, a value that is too low gives a minimum $\mathbf{r}^\star$, making $\mathcal{J}(\mathbf{x}+\mathbf{r})$ very negative and causing a big distortion. This is illustrated by Figure 2.3. Therefore, it is necessary to do a binary search to find an adequate Lagrange multiplier. This means that the program carrying out the attack has two layered iterative loops, which explains its great complexity.

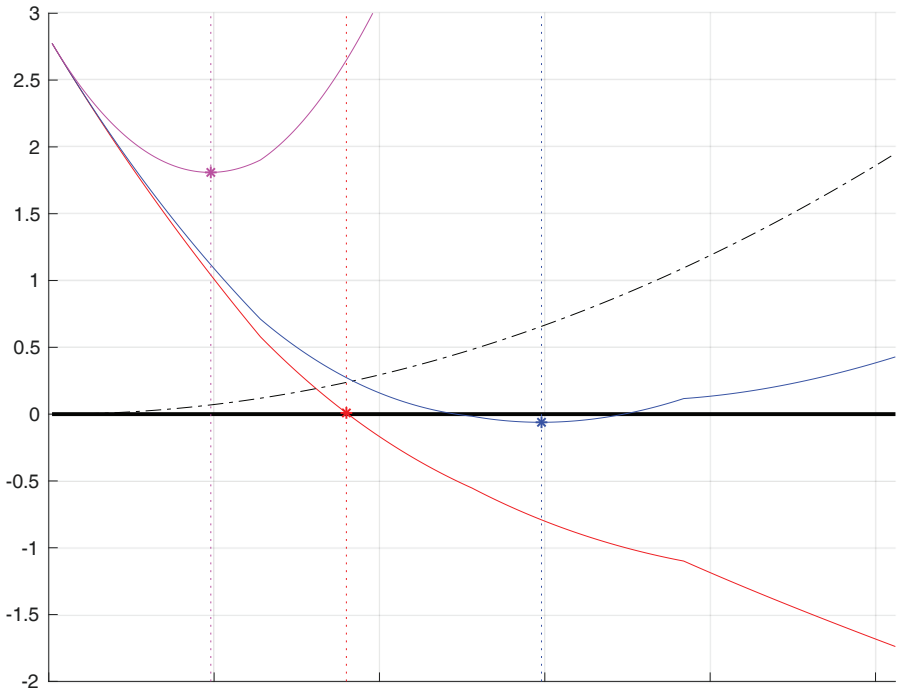


Figure 2.3. *Illustration of L-BFGS in 1D*

COMMENT ON FIGURE 2.3.– *The perturbation* $\mathbf{r}$ *is collinear with the gradient of the objective function. The abscissa is the norm of* $\mathbf{r}$*. The objective function* $\mathcal{J}(\mathbf{x}+\mathbf{r})$ *is given in red. It disappears and becomes negative when* $\mathbf{r}$ *is strong enough to be adverse. The distortion* $\|\mathbf{r}\|^2$ *is given as black dotted lines, and the function* $\mathcal{J}(\mathbf{x}+\mathbf{r})+\lambda * \|\mathbf{r}\|^2$ *is given in blue and magenta for two values of* λ*. The first value is too small (in blue): the minimum is "far after" the red asterisk; the*

perturbation is adverse, but of great distortion. The second value is too big (in magenta): the minimum is before the red asterisk; the perturbation is not adverse.

2.3.1.4.2. C&W

The very well-known attack by Carlini and Wagner (2017), noted as C&W later, follows this idea. However, it also deals with the constraint that $\mathbf{x} + \mathbf{r}$ must remain in $[0, 1]^m$ by a change of variable replacing $\mathbf{x} + \mathbf{r}$ by $\sigma(\mathbf{w})$, where $\mathbf{w} \in \mathbb{R}^n$ and $\sigma(\cdot) : \mathbb{R} \to [0, 1]$ is the sigmoid function applied component by component. In addition, for a given λ, C&W uses the numerical method Adam (Kingma and Ba 2015), to find the minimum of an objective function in $\mathbb{R}^m$:

$$\ell(\mathbf{w}) := [\mathcal{J}(\sigma(\mathbf{w})) + \mu]_+ - \mu\lambda \left\|\sigma(\mathbf{w}) - \mathbf{x}\right\|^2 \qquad [2.15]$$

where $\mu > 0$ is a margin and $[x]_+ := x$ if $x > 0$, 0 if not.

When $\mathcal{J}(\sigma(\mathbf{w})) < -\mu$, the first term becomes zero and the distortion takes $\sigma(\mathbf{w})$ back toward $\mathbf{x}$, as illustrated by Figure 2.4. This can cause fluctuations around the margin. Again, a binary search is needed to find a good value of λ, hence great complexity.

2.3.1.4.3. DDN

Decoupling Direction and Norm (Rony *et al*. 2019) is an iterative attack, very similar to PGD_2, seen here before. The formulation of DDN is:

$$\mathbf{y}_{i+1} := \text{proj}_{S_2[\mathbf{x};\rho_i]}(\mathbf{y}_i + \alpha\eta(\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{y}_i))) \qquad [2.16]$$

Here, the projection is carried out on the sphere $S_2[\mathbf{x}; \rho_i]$ of radius ρ_i and center $\mathbf{x}$, even though $\mathbf{y}_{i+1}$ is inside. The main difference with the PGD_2 formulation given by equation [2.13] is that the radius of this sphere changes from one iteration to another. This radius at iteration i is obtained by calculating $\rho_i = (1-\gamma)\|\mathbf{y}_i - \mathbf{x}\|$ when $\mathbf{y}_i$ is an adverse vector. When this is not the case, then $\rho_i = (1+\gamma)\|\mathbf{y}_i - \mathbf{x}\|$, with $\gamma \in (0, 1)$.

2.3.1.5. *Other attacks*

Other attacks are in the same style, but with variations, either on the definition of the objective function, or on the definition of the distortion. Finally, this overview of attacks ends with the description of a few techniques that take quite different paths to achieve their goals. They are separate because it is not easy to arrange them in one of the two families presented above.

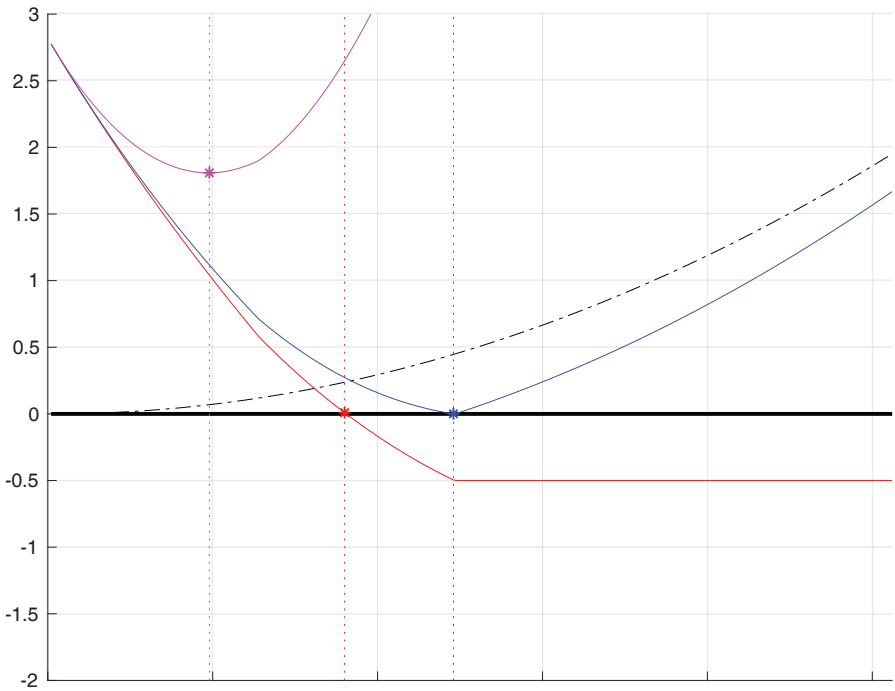


Figure 2.4. *Illustration of C&W in 1D. The perturbation* $\mathbf{r}$ *is collinear to the gradient of the objective function. This is the same configuration as in Figure 2.3, except the margin* $\mu = 0.5$ *for the threshold of equation [2.15]. Notice its effect: the blue minimum is closer to the red asterisk*

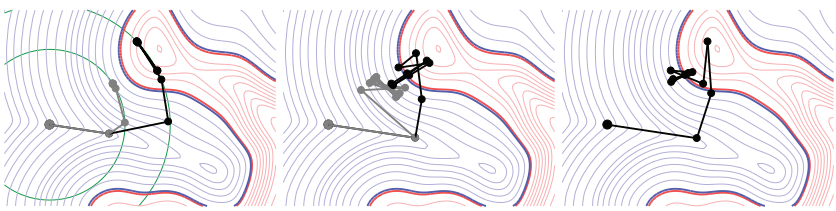


Figure 2.5. *Illustration, in two dimensions, of adverse attacks on a binary classifier*

COMMENT ON FIGURE 2.5.– *From left to right:* PGD_2*, C&W, DDN. The regions associated with the two classes are in red and blue. The level lines indicate the predicted probabilities. The objective is to find an adverse point in the red area,*

which is as close as possible to the starting point $\mathbf{x}$*. In gray (respectively black), are the paths taken for PGD*$_2$ *(Kurakin* et al. *2016) (radius of the green circle) or for a parameter* λ *for C&W (Carlini and Wagner 2017).*

2.3.1.5.1. DeepFool

It is a non-targeted white box attack that uses a more sophisticated objective function. In equation [2.7], a non-targeted attack uses the objective function in order to assign the most likely class for the classifier to the adversarial image under construction, apart from the original class ℓ_g. So, the objective function has a positive, but low, value in $\mathbf{x}$. It seems easier to make it negative.

With DeepFool, Moosavi-Dezfooli shows that this reasoning is incorrect. The ease of making the objective function negative certainly depends on its initial value, but also on its gradient. At the first order, according to equation [2.10], the minimum distortion necessary in the L_2 norm is achieved when $\mathbf{r} \propto -\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}, \ell)$ with:

$$\|\mathbf{r}\| = \frac{\mathcal{J}(\mathbf{x}, \ell)}{\|\nabla_{\mathbf{x}}\mathcal{J}(\mathbf{x}, \ell)\|} \qquad [2.17]$$

It is best to target the ℓ class that requires the least distortion. This is illustrated by Figure 2.6. But this formula is only an approximation at the first order. In addition, it must be estimated for all (or part of) the classes, except for the original class ℓ_g.

2.3.1.5.2. ILC

The *Iterative Least-likely Class* (ILC) (Papernot *et al*. 2018) proposes another alternative objective function. It is possible that the class assigned to the attacked image is sometimes semantically close to the original class ℓ_g. An image of a swallow taken for an image of a sparrow seems more insignificant to us than if this same image of a swallow is taken for an image of a car. Thus, ILC prefers to target the least likely class for the original image.

2.3.1.5.3. JSMA

Papernot *et al.* (2016b) propose a targeted attack for low distortions in norm L_0. The attack finds out which pixels play an important role in the classification. This approach, called the *Jacobian-based Saliency Map Attack* (JSMA), estimates the Jacobian matrix of the function $\mathbf{x} \to \mathbf{p}$. This calculation determines which elements of $\mathbf{x}$ have the most influence, not only to increase the predicted probability of the targeted class $p(\ell)$, but also to decrease the predicted probabilities of all of the other classes.

Few pixels have this property, but modifying them is extremely effective in deceiving the classifier. However, they must be modified with a large amplitude,

which produces “salt and pepper noise” in the image. This modification is often very visible, but can pass for an error in the coding of the photo.

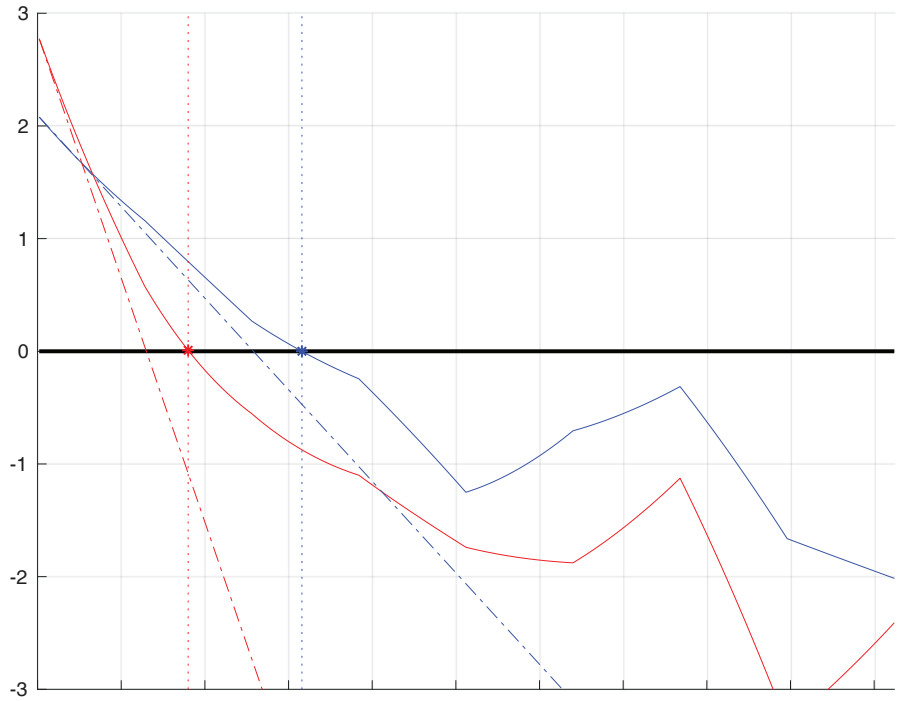


Figure 2.6. *Illustration of DeepFool*

COMMENT ON FIGURE 2.6.– *The blue and red curves correspond to the objective function $\mathcal{J}(\mathbf{x}, \ell)$ for two classes ℓ_1 and ℓ_2, which are different when the perturbation $\mathbf{r}$ is collinear with their gradient. The abscissa corresponds to the $\mathbf{r}$ norm. Notice that $p(\ell_g) = 0.8$, $p(\ell_1) = 0.1$, $p(\ell_2) = 0.05$. As $p(\ell_1) > p(\ell_2)$, it seems interesting to target the class ℓ_1: the blue objective function starts from lower down. This is an error because this one is canceled “later” than that of the class ℓ_2 in red. To find out, DeepFool calculates the gradient in $\mathbf{x}$, which amounts here to approaching the objective function by its tangent in $\|\mathbf{r}\| = 0$ (dotted).*

This technique is remarkable since it is quite fascinating to note that changing the value of a few pixels, or even a single pixel, is enough to lead to a misclassification.

2.3.1.5.4. Universal attacks

Moosavi-Dezfooli *et al*. (2017) have shown that it is possible to create a unique adversarial perturbation that works whatever the image proposed to the network. To do this, they repeatedly apply the DeepFool algorithm (see section 2.3.1.5.1) to all of the images in the training set until a particular perturbation causes the misclassification of a large part of these images. More formally, their approach looks for the perturbation $\mathbf{r}$, bounded by ϵ, such that:

$$\begin{aligned} &\min \|\mathbf{r}\|_p \leq \epsilon \\ &\text{such that } \mathcal{P}_{\mathbf{x}\sim P_{data}}(\mathsf{f}(\mathbf{x}+\mathbf{r}) \neq l_g) \geq 1-\delta \end{aligned} \quad [2.18]$$

where δ indicates the proportion of images from the training set that have become adversarial images and belong to the sample P_{data} of all of the images.

Generally, the algorithm succeeds in finding multiple adversarial samples, which are often visually very different from each other, thus facilitating universal attacks.

2.3.1.5.5. Geometric attacks

So far, attacks change pixels' value additively: $\mathbf{y} = \mathbf{x} + \mathbf{r}$. They are sometimes called value-metric attacks. Geometric attacks do not change the value of the pixels but their position, by slight rotations and local translations. An optical flow applies a displacement field to the pixels of the original image: the pixel at position (k, l) is moved to position $(k, l) + \Delta(k, l)$ in the adversarial image.

Xiao *et al*. (2018) sought to optimize this optical flow by observing the variations in classification probabilities. The objective function integrates $\mathcal{J}(\mathbf{x}, \ell)$ and a part regularizing the optical flow so that it generates small continuous movements. Again, a BFGS-type numerical method is used. The adversarial images often seem perfect.

2.3.1.5.6. Generative Adversarial Network attacks

Generative Adversarial Networks (GAN) (Goodfellow *et al.* 2014b) form a class of machine learning algorithms that learn to estimate a probability distribution from samples submitted to them. The learned distribution forms a model that the network can then use to generate new samples that are completely synthesized, but that will belong to this same distribution. By consuming a very large collection of images of the faces of existing people, a generative network can then synthesize new artificial, but realistic, faces.

GAN are made of two distinct parts, a *generator* that learns the distribution and generates a new sample, and a *discriminator* that estimates whether the sample it observes comes from the generator or directly from the learning set, and then notifies the generator. These two parts compete against each other, in that the generator tries

to create an artificial sample that the discriminator will not be able to distinguish from a real sample. The discriminator, therefore, forces the generator to improve the quality of the synthesis.

From this general idea, it then seems natural to use these GAN to produce adversarial images. The generator creates adversarial images that nevertheless appear normal to our eyes. Baluja and Fischer (2017), for example, trained a generator to create images, which mislead a particular network by modifying them via a residual network (He *et al*. 2016).

We are, therefore, very far from the gradient calculation mechanism of the first attacks. The advantage of that generative approach is the almost instantaneous speed to produce an adversarial image. But there are many drawbacks. The learning time is very long. This approach therefore only makes sense if the attacker has a large number of adversarial images to create. In addition, a learned network only targets a given class and is only valid against a particular classifier.

2.3.2. *Black or gray box*

The black box model is much more strict than the white box model. The attacker does not know anything about the targeted network. At this point, it is impossible to calculate gradients and therefore impossible to apply the techniques mentioned so far. Nevertheless, the attacker can use the targeted network as an oracle and observe the way in which it labels an image that is proposed.

The gray box and black box models assume that the attacker has much less information at their disposal. For the gray box model, we suppose that the attacker knows some elements of the targeted network. For example, that the network uses a pre-trained model made available off the shelf, but with defense mechanisms that are secret. The attacker can then partly reproduce the behavior of the targeted network to set up their attacks.

2.3.2.1. *Two concepts about the black box*

If the "black box" means observing the inputs/outputs of a system, what are these outputs? Some believe that the output that the attacker has access to is the predicted probability vector $\mathbf{p}$. Others believe that the output is the predicted class $\ell = \mathtt{f}(\mathbf{x})$.

The nature of these outputs makes a big difference, as noted in the article by Ilyas *et al*. (2018). The predicted class $\mathtt{f}(\mathbf{x})$ is a constant piecewise function. Almost certainly, the attacker does not see if a small amplitude perturbation $\mathbf{r}$ is going in the right direction, since it does not necessarily change the output. This is not the case for the predicted probability vector.

2.3.2.2. *Output = probability vector*

In this case, there is no big difference with white box attacks. The attacker calculates an objective function, like in section 2.3.2.1, and seeks a perturbation that minimizes it. The only difference is that unfortunately, the gradient is no longer available. The attacker then uses so-called zero order numerical methods, such as differential evolution algorithms (Storn and Price 1997), sometimes called genetic algorithms. These algorithms randomly take distortions, calculate their objective functions, select the distortions which have obtained the lowest values and recombine them with random mutations. This selection–recombination–mutation cycle is repeated.

In this way, the *One pixel attack* (Su *et al*. 2017) is the counterpart of the JSMA attack, where the pixels (even *the* pixel) to be modified are found thanks to the genetic algorithm. Likewise, Engstrom's work (Engstrom *et al*. 2017) is the black box counterpart of the geometric attacks by Xiao *et al*. (2018) in white box. Likewise, Zhao *et al*. (2017) involve generators of adversarial distortions, like in Baluja and Fischer (2017), where the discriminator is a black box classifier.

An alternative to genetic algorithms is estimating the gradient of the objective function in certain directions:

$$\frac{\partial \mathcal{J}(\mathbf{x})}{\partial x(i)} \approx \frac{\mathcal{J}(\mathbf{x} + h\mathbf{e}_i) - \mathcal{J}(\mathbf{x} - h\mathbf{e}_i)}{2h} \qquad [2.19]$$

To estimate the gradient, we have to calculate this deviation for all of the pixels $1 \leq i \leq m$, which is costly. Chen *et al*. (2017) show that this is not necessary. They apply a stochastic gradient descent where the directions $\mathbf{e}_i$ are iteratively drawn at random. This attack is called *zeroth-order optimization* (ZOO). Let us quote another attack belonging to this category, designed by Narodytska and Kasiviswanathan (2017).

2.3.2.3. *Output = predicted class*

Szegedy *et al*. (2013) are among the first to realize that adversarial images designed to attack one specific network are also adversarial for another network. But it was Papernot *et al.* (2016a) who first explored the transfer properties of attacks by studying gray box and black box attacks. Let us also cite the work of Liu *et al*. (2016) as a notable article on this subject. A similar phenomenon, traditional in machine learning, is well known: it is possible, to a certain extent, to transfer what has been learned by one network to another.

Papernot *et al.* (2016) rely on the observation of the proportion of adversarial images deceiving the first system, which also succeed in deceiving the second. To be more precise, they distinguish transfers between learning systems built on the same

fundamental principle from transfers between systems built on different principles (a transfer between a deep network and an SVM-based system for example).

The results of their study (Papernot *et al.* 2016a) show that the transfers are possible and easy between neural systems. On the other hand, although always possible, the transfer of attacks is more difficult between systems built on models that cannot be derived, mathematically speaking, like those built from SVMs, nearest neighbors or decision trees (this is opposed, for example, to neural networks, for which it is easy to calculate the gradient).

This observation then makes it possible to attack black box learning systems. By multiplying the requests to the targeted classifier, the attacker can build a model: a new classifier is trained to imitate the black box in the sense that the outputs of this surrogate classifier must ultimately be identical to those of the targeted black box. Then, the attacker uses this new model, but in a white box, to forge adversarial images, with the hope that they also deceive the black box network because of the transfer property.

2.4. Defenses

There are just as many defenses as there are attacks. This section provides an overview. Schematically, we can distinguish three families of defenses:

– *Reactive techniques*: these strategies are based on preprocessing, carried out before feeding the network. These block the images if adversarial content is detected or filter and clean the images, hoping to remove the adversarial perturbation.

– *Proactive techniques*: these strategies build networks that are inherently stronger to adversarial attacks. This category includes, for example, approaches which incorporate many adversarial images into the learning phase.

– *Obfuscation techniques*: these strategies hide or obfuscate the important parameters that an attacker needs to produce adversarial images.

Another viable view distinguishes whether the defense is an add-on module connected to the network (and therefore the classifier works with or without defense), or whether the defense is an integral part of the network resulting in a radical transformation of the classifier.

2.4.1. *Reactive defenses*

This family groups together the techniques which detect the adversarial nature of an image and/or apply a preprocessing to the images submitted to the network, to eliminate what makes them adversarial from their content.

Detection techniques introduce an additional class. This class is not necessarily labeled "adversarial images", but simply "unknown class". The fundamental theory is that the goal of the attacker is not to target this class. The attack fails if this class is the result of classification. The detection is sometimes justified as follows: (1) the images are points in a large-dimensional space $\mathbb{R}^m$ concentrated along manifolds; (2) attacks push these images out of their manifold. The detectors learn to distinguish these manifolds by collecting statistics, calculated either in the image domain, or in the hidden layers.

The preprocessings filter the images in order to remove the adversarial distortion, without altering the visual content. Here, the images are never rejected, we hope they are cleaned and therefore harmless. In theory, filtering amounts to projecting an image onto the manifolds of the natural images mentioned above. Once again, some techniques filter the images before classification, others filter the representations that travel in the networks.

In reality, reactive techniques mix preprocessing and detection. It is possible to build a detector from preprocessing by thresholding the quantity of the filtered noise of the image. Therefore, we list examples of reactive defenses without clear distinction.

2.4.1.1. *Learn about the manifold of natural images*

This is the issue for many defenses. The advantage is that this learning only consumes original images. Thus, the defense is not biased toward one or more specific attacks. *MagNet* (Meng and Chen 2017) employs autoencoders to project the image and bring it closer to the manifold of natural images. Variants use sparse representations of image patches like *D3* (Moosavi-Dezfooli *et al*. 2018), estimates based on mixtures of Gaussians (Ghosh *et al*. 2018), or deep generator networks like *PixelDefend* (Song *et al*. 2017) or *Defense-GAN* (Samangouei *et al*. 2018). More uncommon, Dubey *et al*. (2019) search the Internet for the images most similar to the query, then decide on its class by a majority vote on the predictions of the similar images.

2.4.1.2. *Interaction with the classifier*

A simple method of detection is *feature squeezing* (Xu *et al*. 2017). Many simple filters are applied to degrade or simplify the image, hence the name "squeezer" (compression, slight blur filter), before submitting it to the network. Then, the deviations at the output of the network are observed with respect to the predicted probability vector $\mathbf{p}$ given for the original image. Any significant deviation suggests that the tested image is adversarial. Guo *et al*. (2017) and Liang *et al*. (2018) developed approaches that are very close one another. *SafetyNet* (Lu *et al*. 2017) is based on the analysis of active neurons in the classifier. They encode typical activation patterns of a deep layer of a network processing clean images, and compare this description to the current one when an unknown image is processed.

This comparison is made via a radial kernel support vector machine. Bypassing this defense forces the attacker to integrate the response of all of the rectification units in the network into their attack, which is difficult in practice. This defense works well, even when the network is big. A more recent version of this defense idea is called *Network Invariance Checking* (Ma *et al*. 2019).

2.4.2. *Proactive defenses*

Proactive defenses aim to improve the intrinsic strength of models.

2.4.2.1. *Reducing the amplitude of gradients*

If the network function experiences strong gradients, then a very small perturbation is needed to greatly modify the output of the network. This explains the vulnerability of the network to attacks. Reducing the amplitude of the gradients makes the network stronger.

One of the very first approaches is "distillation", which is originally a technique for transferring what has been learned by a large network to a smaller network (Hinton *et al*. 2015). In very broad terms, the distillation trains the small network, not with the labels of the images, but with the probability vectors predicted by the large network, which are more informative than simple labels. Papernot *et al.* (2016c) rely on distillation, but apply this transfer on the same network architecture. Therefore, the first version of the network is trained on labels, and the second is trained on the knowledge learned from the first. This "autotransfer" is made at a high temperature in the *softmax* function, which reduces the amplitude of the gradients of the network function. Nevertheless, Carlini and Wagner designed attacks that made the distillation defenses fail (see the C&W attack, in section 2.3.1.4).

Gu and Rigazio (2014) suggest training networks with a new constraint: each layer must be "contracting", in the sense of a Lipschitzian function (Tsuzuku *et al*. 2018). This is incorporated during the training by a penalty, which aims to reduce the variation of its response to perturbations it receives as an input. Overall, this increases the strength of the network and requires the applied distortion to be significantly stronger for an attack to be successful.

2.4.2.2. *Adversarial training*

Learning with more data allows a network to generalize better, to refine the boundaries between classes in the representation space. This classic trick is done by adding quasi copies of images that have undergone small translations or rotations.

The idea is the same here, by improving learning with adversarial images. The network therefore learns that, despite the perturbation, such-and-such an image is indeed in such-and-such a category. The principle is simple, but the implementation

is difficult. An attack targets a network, which during the learning process is, by definition, unpredictable. Each time the synaptic weights are updated, the adversarial versions of the training images must be recalculated. The attack must therefore be super fast. This is how Goodfellow *et al.* (2014a) proceed thanks to the simplest of attacks: FGSM.

This idea leads to the concept of robust optimization by a *min–max* formulation, where the learning process has an objective:

$$\min_{\boldsymbol{\theta}} \sum_{j} \max_{\mathbf{r}_j | \|\mathbf{r}_j\| < \epsilon} \mathcal{L}(\mathbf{x}_j + \mathbf{r}_j, \ell_j, \boldsymbol{\theta}) \qquad [2.20]$$

with $\{\mathbf{x}_j, \ell_j\}$ training data.

In a way, the training tries to get all of the images in the ball with center $\mathbf{x}_j$ and radius ϵ to be classified as $\mathbf{x}_j$. We can also cite various works exploring these same ideas (Huang *et al*. 2015; Madry *et al*. 2017; Tramèr *et al*. 2017). Let us quote the approach by Lee *et al*. (2017) again, where a generative network creates adversarial images, which feed a classifier carrying out adversarial learning.

Many gray areas remain in adversarial training. The robustness provided is sometimes disputed. The network is more robust against simple attacks, but still just as vulnerable to more complex attacks. The robustness is obvious on the training images, but it does not generalize well. There is a price to pay: the network is more robust against attacks but less precise on the original images. The consensus is not yet established with certainty because the adversarial training is difficult to carry out. Many variations exist, gradually increasing the quantity of adversarial images, while increasing the strength of the attacks from a very large number of original images. All of this is costly, with the benefits not always outweighing the extra costs.

2.4.3. *Obfuscation technique*

Creating adversarial images in white box relies most often on the utilization of the gradient of the differentiable objective function. Introducing strong nonlinearities makes the network non-differentiable and prevents the calculation of a gradient. This family of techniques was explored by Goodfellow *et al.* (2014a) (see Buckman *et al*. (2018)). Athalye *et al*. (2018) have explored this subject and show the ineffectiveness of this approach: in white box, nothing forces the attacker to use the gradient of the objective function. It can modify the network and replace any nonlinearity with a smoother function.

Obfuscation becomes more serious when it is based on the insertion of a secret key in the classifier, like in cryptography. This is the only way to prevent white box

analysis. The attacker knows all of the details of the network except one secret high entropy parameter. This is difficult to combine with machine learning and often requires re-training the whole network, or part of it, each time a secret key is taken (Shumailov *et al*. 2018; Taran *et al*. 2020).

Another possibility is to make the classifier random. For each call to the network, the final prediction depends on a random value that the attacker cannot know. This could be a slight modification to the input image, or modifications within the network: Dhillon *et al*. (2018) suggest randomly suppressing certain neurons (those which react weakly) and to increase, in proportion, the importance of the reaction of the conserved neurons.

2.4.4. *Defenses: conclusion*

We have mentioned several defenses; there are many others, sometimes simple variations, and sometimes more original contributions too. In general, the evaluation of their effectiveness leaves a lot to be desired. Many of them are evaluated on very small sets of tests, can only withstand a particular class of attacks without being clearly perceptible, or are even too expensive to be usable in practice. There is not yet a rigorous protocol to assess the quality of a defense technique making a network more robust to adversarial attacks. It is very difficult to compare the respective merits of different defensive strategies.

The addition of defense strategies sometimes leads to a reduction in the quality of the networks: the classification performance on the natural images (not attacked) of a network with defense is worse than without a defense. This observation is disputed because nothing completely implies such tension (although some theoretical papers claim otherwise).

Some approaches take a more formal point of view and try to guarantee the robustness of the network, as long as the distortion remains below a boundary, whose value must be calculated. Still at an early stage, we nevertheless cite the promising studies (Huang *et al*. 2017; Katz *et al*. 2017; Sinha *et al*. 2017; Wong and Kolter 2017; Raghunathan *et al*. 2018; Ruan *et al*. 2018). Another contribution, written in French, tackles the same topic (Bazille *et al*. 2019)

2.5. Conclusion

This chapter has provided an overview of attack and defense techniques, involving the vulnerabilities of machine learning systems based on deep neural networks for image recognition tasks. This field of research is very active and the work is increasing every day.

They aim to make attacks more and more imperceptible (Zhang *et al* to appear), even when examined through sophisticated psycho-visual metrics (Fezza *et al*. 2019). They also aim to be faster in order to make defenses based on robust learning (Zhang *et al*. 2019).

Some works aim to better understand the causes of attacks and the reasons for the success or failure of defenses. Instead, these works explore problems linked to the distribution of data in high-dimensional spaces, the effects of thresholding functions and the theoretical guarantees that a network can offer.

Of course, attacks and defenses are not limited to just images, and some work explores the creation of adversarial videos (Jiang *et al*. 2019; Wei *et al*. 2019), audio (Carlini and Wagner 2018; Qin *et al*. 2019), texts (Behjati *et al*. 2019) and time series (Fawaz *et al*. 2019), in an attempt to understand the relationships between vulnerabilities and multimodal data (Park *et al*. 2019), or even explore the problems of deceiving *malware* (Martins *et al*. 2020).

Additionally, other works consider different tasks, such as similarity search (Amsaleg *et al*. 2017), clustering, feature selection, embedding, hashing, similarity learning and outlier detection.

Much remains to be understood, it is very encouraging.

2.6. References

Amsaleg, L., Bailey, J., Barbe, D., Erfani, S.M., Houle, M.E., Nguyen, V., Radovanovic, M. (2017). The vulnerability of learning to adversarial perturbation increases with intrinsic dimensionality. In *Workshop on Information Forensics and Security*. IEEE, Rennes, 1–6.

Athalye, A., Carlini, N., Wagner, D. (2018). Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples [Online]. Available at: https://arxiv.org/abs/1802.00420.

Baluja, S. and Fischer, I. (2017). Adversarial transformation networks: Learning to generate adversarial examples [Online]. Available at: https://arxiv.org/abs/1703.09387.

Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D. (2006). Can machine learning be secure? In *AsiaCCS*. ACM, Taipei, 16–25.

Bazille, H., Fabre, E., Genest, B. (2019). Certification formelle des réseaux neuronaux profonds : un état de l'art en 2019. *AI and Defense 2019 – Artificial Intelligence and Defense*, 1–10 [Online]. Available at: https://hal.archives-ouvertes.fr/hal-02350253.

Behjati, M., Moosavi-Dezfooli, S., Baghshah, M.S., Frossard, P. (2019). Universal adversarial attacks on text classifiers. In *ICASSP*. IEEE, Brighton, 7345–7349.

Biggio, B. and Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84, 317–331.

Buckman, J., Roy, A., Raffel, C., Goodfellow, I. (2018). Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*. ICLR, Vancouver.

Carlini, N. and Wagner, D.A. (2017). Towards evaluating the robustness of neural networks. In *Symposium on Security and Privacy*. IEEE, San Jose.

Carlini, N. and Wagner, D.A. (2018). Audio adversarial examples: Targeted attacks on speech-to-text. In *IEEE Symposium on Security and Privacy Workshops*. IEEE, San Francisco, 1–7.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., Hsieh, C.-J. (2017). ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Workshop on Artificial Intelligence and Security*. ACM, New York, 15–26.

Dalvi, N.N., Domingos, P.M., Mausam, Sanghai, S.K., Verma, D. (2004). Adversarial classification. In *KDD*. ACM, Seattle, 99–108.

Dhillon, G.S., Azizzadenesheli, K., Lipton, Z.C., Bernstein, J., Kossaifi, J., Khanna, A., Anandkumar, A. (2018). Stochastic activation pruning for robust adversarial defense [Online]. Available at: https://arxiv.org/abs/1803.01442.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., Li, J. (2018). Boosting adversarial attacks with momentum. In *Conference on Computer Vision and Pattern Recognition*. IEEE, Salt Lake City, 9185–9193.

Dubey, A., van der Maaten, L., Yalniz, Z., Li, Y., Mahajan, D. (2019). Defense against adversarial images using web-scale nearest-neighbor search. In *Conference on Computer Vision and Pattern Recognition*. IEEE, Long Beach, 8767–8776.

Engstrom, L., Tran, B., Tsipras, D., Schmidt, L., Madry, A. (2017). A rotation and a translation suffice: Fooling CNNs with simple transformations [Online]. Available at: https://arxiv.org/abs/1712.02779.

Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P. (2019). Adversarial attacks on deep neural networks for time series classification. In *IJCNN*. IEEE, Budapest, 1–8.

Fezza, S.A., Bakhti, Y., Hamidouche, W., Déforges, O. (2019). Perceptual evaluation of adversarial attacks for CNN-based image classification. In *Eleventh International Conference on Quality of Multimedia Experience*. QoMEX, Berlin, 1–6.

Ghosh, P., Losalka, A., Black, M.J. (2018). Resisting adversarial attacks using Gaussian mixture variational autoencoders [Online]. Available at: https://arxiv.org/abs/1806.00081.

Goodfellow, I.J., Shlens, J., Szegedy, C. (2014a). Explaining and harnessing adversarial examples [Online]. Available at: https://arxiv.org/abs/1412.6572.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014b). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2672–2680.

Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press [Online]. Available at: http://www.deeplearningbook.org.

Gu, S. and Rigazio, L. (2014). Towards deep neural network architectures robust to adversarial examples. In *International Conference on Learning Representations*. ICLR, Banff.

Guo, C., Rana, M., Cisse, M., van der Maaten, L. (2017). Countering adversarial images using input transformations [Online]. Available at: https://arxiv.org/abs/1711.00117.

He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*. IEEE, Las Vegas, 770–778.

Hinton, G., Vinyals, O., Dean, J. (2015). Distilling the knowledge in a neural network [Online]. Available at: https://arxiv.org/abs/1503.02531.

Huang, R., Xu, B., Schuurmans, D., Szepesvári, C. (2015). Learning with a strong adversary [Online]. Available at: https://arxiv.org/abs/1511.03034.

Huang, X., Kwiatkowska, M., Wang, S., Wu, M. (2017). Safety verification of deep neural networks. In *International Conference on Computer Aided Verification*. CAV, Heidelberg, 3–29.

Ilyas, A., Engstrom, L., Athalye, A., Lin, J. (2018). Black-box adversarial attacks with limited queries and information. *Proceedings of Machine Learning Research*, 80, 2142–2151.

Jiang, L., Ma, X., Chen, S., Bailey, J., Jiang, Y. (2019). Black-box adversarial attacks on video recognition models. In *ACM Multimedia*. ACM, Nice, 864–872

Katz, G., Barrett, C., Dill, D.L., Julian, K., Kochenderfer, M.J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*. CAV, Heidelberg, 97–117.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization [Online]. Available at: https://arxiv.org/abs/1412.6980.

Kurakin, A., Goodfellow, I., Bengio, S. (2016). Adversarial examples in the physical world [Online]. Available at: https://arxiv.org/abs/1607.02533.

Lee, H., Han, S., Lee, J. (2017). Generative adversarial trainer: Defense to adversarial perturbations with GAN [Online]. Available at: https://arxiv.org/abs/1705.03387.

Liang, B., Li, H., Su, M., Li, X., Shi, W., Wang, X. (2018). Detecting adversarial image examples in deep neural networks with adaptive noise reduction. *IEEE Transactions on Dependable and Secure Computing*, 18(1), 72–85.

Liu, Y., Chen, X., Liu, C., Song, D. (2016). Delving into transferable adversarial examples and black-box attacks [Online]. Available at: https://arxiv.org/abs/1611.02770.

Lowd, D. and Meek, C. (2005). Adversarial learning. In *KDD*. ACM, Chicago, 641–647.

Lu, J., Issaranon, T., Forsyth, D. (2017). Safetynet: Detecting and rejecting adversarial examples robustly. In *International Conference on Computer Vision*. IEEE, Venice, 446–454.

Ma, S., Liu, Y., Tao, G., Lee, W., Zhang, X. (2019). NIC: Detecting adversarial samples with neural network invariant checking. In *26th Annual Network and Distributed System Security Symposium*. NDSS, San Diego.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks [Online]. Available at: https://arxiv.org/abs/1706.06083.

Martins, N., Cruz, J.M., Cruz, T., Abreu, P.H. (2020). Adversarial machine learning applied to intrusion and malware scenarios: A systematic review. *IEEE Access*, 8, 35403–35419.

Meng, D. and Chen, H. (2017). Magnet: A two-pronged defense against adversarial examples. In *SIGSAC Conference on Computer and Communications Security*. ACM, Dallas, 135–147.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., Frossard, P. (2017). Universal adversarial perturbations. In *CVPR*. IEEE, Hawaii, 86–94.

Moosavi-Dezfooli, S.-M., Shrivastava, A., Tuzel, O. (2018). Divide, denoise, and defend against adversarial attacks [Online]. Available at: https://arxiv.org/abs/1802.06806.

Narodytska, N. and Kasiviswanathan, S. (2017). Simple black-box adversarial attacks on deep neural networks. In *Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, Honolulu, 1310–1318.

Papernot, N., McDaniel, P., Goodfellow, I. (2016a). Transferability in machine learning: From phenomena to black-box attacks using adversarial samples [Online]. Available at: https://arxiv.org/abs/1605.07277.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A. (2016b). The limitations of deep learning in adversarial settings. In *European Symposium on Security and Privacy*. IEEE, Saarbrücken.

Papernot, N., McDaniel, P., Wu, X., Jha, S., Swami, A. (2016c). Distillation as a defense to adversarial perturbations against deep neural networks. In *European Symposium on Security and Privacy*. IEEE, Saarbrücken.

Papernot, N., Faghri, F., Carlini, N., Goodfellow, I., Feinman, R., Kurakin, A., Xie, C., Sharma, Y., Brown, T., Roy, A., Matyasko, A., Behzadan, V., Hambardzumyan, K., Zhang, Z., Juang, Y.-L., Li, Z., Sheatsley, R., Garg, A., Uesato, J., Gierke, W., Dong, Y., Berthelot, D., Hendricks, P., Rauber, J., Long, R. (2018). Technical report on the CleverHans v2.1.0 adversarial examples library [Online]. Available at: https://arxiv.org/abs/1610.00768.

Park, J.S., Rohrbach, M., Darrell, T., Rohrbach, A. (2019). Adversarial inference for multi-sentence video description. In *CVPR. Computer Vision Foundation/IEEE*. Long Beach, 6598–6608.

Qin, Y., Carlini, N., Cottrell, G.W., Goodfellow, I.J., Raffel, C. (2019). Imperceptible, robust, and targeted adversarial examples for automatic speech recognition. *Proceedings of Machine Learning Research*, 97, 5231–5240.

Raghunathan, A., Steinhardt, J., Liang, P. (2018). Certified defenses against adversarial examples [Online]. Available at: https://arxiv.org/abs/1801.09344.

Rony, J., Hafemann, L.G., Oliveira, L.S., Ayed, I.B., Sabourin, R., Granger, E. (2019). Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *Conference on Computer Vision and Pattern Recognition*. IEEE, Long Beach, 4322–4330.

Ruan, W., Huang, X., Kwiatkowska, M. (2018). Reachability analysis of deep neural networks with provable guarantees [Online]. Available at: https://arxiv.org/abs/1805.02242.

Samangouei, P., Kabkab, M., Chellappa, R. (2018). Defense-gan: Protecting classifiers against adversarial attacks using generative models [Online]. Available at: https://arxiv.org/abs/1805.06605.

Serban, A.C. and Poll, E. (2018). Adversarial examples: A complete characterisation of the phenomenon. *CoRR* [Online]. Available at: https://dblp.org/rec/journals/corr/abs-1810-01185.bib.

Sharif, M., Bauer, L., Reiter, M.K. (2018). On the suitability of l_p-norms for creating and preventing adversarial examples [Online]. Available at: https://arxiv.org/abs/1802.09653.

Shumailov, I., Zhao, Y., Mullins, R.D., Anderson, R. (2018). The taboo trap: Behavioural detection of adversarial samples. *CoRR* [Online]. Available at: http://arxiv.org/abs/1811.07375.

Simonyan, K., Vedaldi, A., Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations*. ICLR, Banff.

Sinha, A., Namkoong, H., Duchi, J. (2017). Certifying some distributional robustness with principled adversarial training [Online]. Available at: https://arxiv.org/abs/1710.10571.

Song, Y., Kim, T., Nowozin, S., Ermon, S., Kushman, N. (2017). Pixeldefend: Leveraging generative models to understand and defend against adversarial examples [Online]. Available at: https://arxiv.org/abs/1710.10766.

Storn, R. and Price, K.V. (1997). Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optimization*, 11, 342–359 [Online]. Available at: https://doi.org/10.1023/A:1008202821328.

Su, J., Vargas, D.V., Kouichi, S. (2017). One pixel attack for fooling deep neural networks [Online]. Available at: https://arxiv.org/abs/1710.08864.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R. (2013). Intriguing properties of neural networks [Online]. Available at: https://arxiv.org/abs/1312.6199.

Taran, O., Rezaeifar, S., Holotyak, T., Voloshynovskiy, S. (2020). Machine learning through cryptographic glasses: Combating adversarial attacks by key-based diversified aggregation. *EURASIP Journal on Information Security 2020*, 10.

Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., McDaniel, P. (2017). Ensemble adversarial training: Attacks and defenses [Online]. Available at: https://arxiv.org/abs/1705.07204.

Tsuzuku, Y., Sato, I., Sugiyama, M. (2018). Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. *Advances in Neural Information Processing Systems*. Curran Associates Inc., New York, 6541–6550.

Tual, M. and Coutagne, G. (2015). On a testé pour vous deep dream, la machine à "rêves" psychédéliques de Google [Online]. Available at: https://www.lemonde.fr/pixels/article/2015/07/09/on-a-teste-pour-vous-deep-dream-la-machine-a-reves-psychedeliques-de-google_4675562_4408996.html.

Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P. (2004). Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 600–612.

Wei, X., Liang, S., Chen, N., Cao, X. (2019). Transferable adversarial attacks for image and video object detection. In *IJCAI*. Macao, 954–960.

Wikipedia (2020). DeepDream [Online]. Available at: https://fr.wikipe-dia.org/ wiki/ DeepDream.

Wong, E. and Kolter, J.Z. (2017). Provable defenses against adversarial examples via the convex outer adversarial polytope [Online]. Available at: https://arxiv.org/abs/1711.00851.

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., Song, D. (2018). Spatially transformed adversarial examples [Online]. Available at: https://arxiv. org/abs/1801.02612.

Xu, W., Evans, D., Qi, Y. (2017). Feature squeezing: Detecting adversarial examples in deep neural networks [Online]. Available at: https://arxiv.org/abs/ 1704.01155.

Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H. (2015). Understanding neural networks through deep visualization. In *Deep Learning Workshop, International Conference on Machine Learning*. ICML, Beijing.

Zhang, H., Avrithis, Y., Furon, T., Amsaleg, L. (2019). Walking on the edge: Fast, low-distortion adversarial examples. *IEEE Transactions on Information Forensics and Security*, 16, 701–713.

Zhang, H., Avrithis, Y., Furon, T., Amsaleg, L. (2020). Smooth adversarial examples. *EURASIP Journal on Information Security*, 15 [Online]. Available at: https://doi.org/10.1186/s13635-020-00112-z.

Zhao, Z., Dua, D., Singh, S. (2017). Generating natural adversarial examples [Online]. Available at: http://arxiv.org/abs/1710.11342.

3

Codes and Watermarks

Pascal LEFÈVRE[1], Philippe CARRÉ[1] and Philippe GABORIT[2]

[1] *XLIM, CNRS, University of Poitiers, France*

[2] *XLIM, CNRS, University of Limoges, France*

In this chapter, we propose to analyze the principle of digital watermarking of images through the prism of the use of error-correcting codes in a very specific framework, namely the so-called robust watermark. The process of watermarking can be studied according to the formality of the transmission of information in a possibly noisy channel. These error-correcting codes have demonstrated their value in this type of application, which is why they were quite naturally introduced in the context of watermarking. In this chapter, classical Hamming error-correcting codes as well as Bose-Chaudhuri-Hocquenghem (BCH) codes and Reed–Solomon (RS) codes are associated with random and packet error structures. In order to illustrate the impact and benefit of using error-correcting codes, we deploy a simple use case based on by index modulation watermarking. In particular, we discuss the differences in behavior regarding robustness, according to the attack and the code used. We conclude the chapter by introducing a more original code and show how a specific code can respond to a particular problem, here the problem of cropping.

3.1. Introduction

Over the years, several watermarking paradigms have emerged due to the protection requirements of many applications. It is very easy to modify or falsify

For a color version of all figures in this chapter, see www.iste.co.uk/puech/multimedia1.zip.

Multimedia Security 1,

coordinated by William PUECH.

Multimedia Security 1: Authentication and Data Hiding,
First Edition. William Puech.

images because of public access to image processing software. Checking the integrity and authenticating an image are therefore important issues. In this context, the idea of the fragile watermark is to insert a pattern so that any modification of the image changes the pattern. The modified regions can therefore be detected by analyzing this pattern. The disadvantage of this watermarking paradigm is that it is not possible to distinguish between content modified inadvertently or with good intentions, and content that has been maliciously modified. For example, the oldest methods classify a compressed image in the category of falsified images, yet its semantics have not changed. Examples of work can be found in previous studies (Yeung and Mintzer 1997; Wolfgang and Delp 1999; Khan *et al*. 2014; Bravo-Solorio *et al*. 2018; Shehab *et al*. 2018). A similar paradigm is semi-fragile watermarking. Work on semi-fragile watermarking (Lu *et al*. 2003; Ho and Li 2004; Maeno *et al*. 2006) has been developed in order to distinguish malicious modifications from those that are not.

These two approaches clash with the principle of robust watermarking which aims to resist image modifications. In fact, so-called robust techniques aim to resist all types of image modifications in order to extract information without error. The error-correcting code tool will be positioned in this context of robust watermarking, since the function of the code is to correct the errors and therefore remove the influence of the attacks on the image. Therefore, first of all, we propose to introduce robust watermarking, which is the framework this chapter is centered around: the error-correcting codes aim to improve robustness in the face of various attacks.

In the following, after recalling the context of robust watermarking (section 3.2) and index modulation (section 3.3), we present section 3.4, classical Hamming error-correcting codes as well as BCH codes and RS codes. In sections 3.5 and 3.6, we discuss a simple use case based on watermarking by index modulation applied to color images. In section 3.7, we analyze the differences in behavior concerning robustness according to the attack undergone and the code used. This chapter concludes (section 3.8) with the introduction of a more original code, and we show how a particular code can answer a particular problem.

3.2. Study framework: robust watermarking

In this chapter, we will concentrate on so-called robust watermarking, which constitutes the classical application framework for the use of codes in watermarking. Many applications, such as copyright protection, require the perfect extraction of a mark inserted in an image that has been modified. In practice, a digital watermarking method will "survive" an attack to a certain extent and it is believed that when an image is too deteriorated, the extraction of the mark loses its value.

The robust watermarking paradigm allows a three-step scenario: inserting the mark into a host image, the transmission of the marked image in a noisy channel (noisy due

to an attack on the watermarked image) and finally, the detection step. We detail each step in this section. Note that this formulation through the idea of noisy channels gives full meaning to the introduction of codes. In order to simply explain the use of error-correcting in a watermarking framework, we describe the classical outline of insertion and detection.

We show the classical diagram of insertion of a mark in Figure 3.1. Image $\mathcal{X}$, chosen as the host of the mark, can be of different types, that is, in grayscale or in color (defined in the RGB color space or other color spaces). Depending on the methods, image $\mathcal{X}$ can be represented in a space other than the original spatial domain (frequency coefficients, wavelet coefficients). From the transformed image $\mathcal{X}'$, coefficients noted as $\mathcal{C}_{\mathcal{X}}$ are chosen with the extraction function Extr (random selection of the insertion sites, decomposition of the entire image) because of the secret key k.

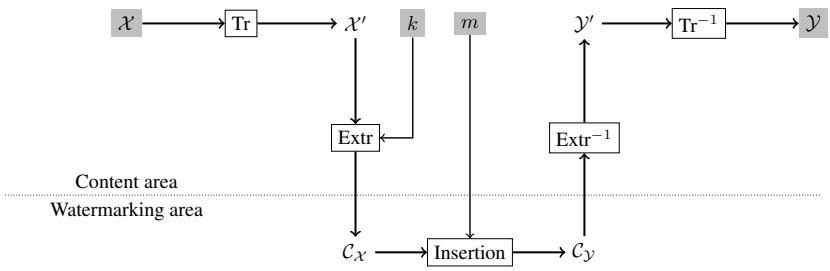


Figure 3.1. *Classical diagram of watermark insertion*

Then, the message m is inserted by modifying $\mathcal{C}_{\mathcal{X}}$ in the watermarking area, which is only accessible with the key k to cut off access to this area. The coefficients $\mathcal{C}_{\mathcal{Y}}$ are then reintegrated into the transformed image and we get a marked image $\mathcal{Y}$ with the inverse transformed Tr^{-1}.

In the second step of this scenario, the image marked $\mathcal{Y}$ is transmitted through a noisy channel $\mathcal{C}$ (Figure 3.2). The noise n represents the possible different attacks on a channel (e.g. compression and noise). Here, we see the parallel that can be drawn with the framework for analyzing the transmission of information.

Without any other "preparation" of the message, the robustness of the information hidden in front of a given attack depends on several criteria.

At the detection stage (Figure 3.3), we access the watermark channel because of the extraction function Extr and the key k of image $\mathcal{Z}$. Then, we apply the planned detection method to the coefficients $\mathcal{C}_{\mathcal{Z}}$. If the power of the attack is reasonable, the estimate m' is the same as m.

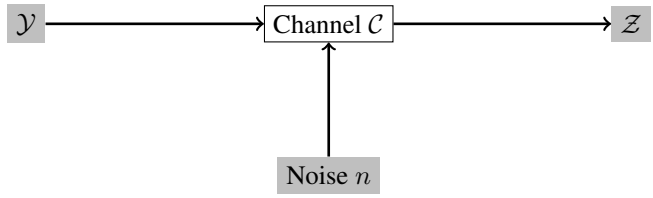


Figure 3.2. *Diagram of the transmission of an image in a channel affected by noise*

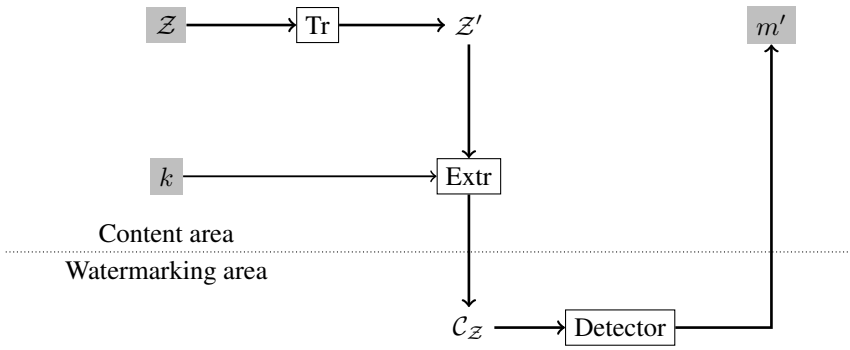


Figure 3.3. *Classical diagram of detection of a mark*

The insertion area can be chosen according to the attack that the mark must resist. Changing the representation of the image makes it possible to take advantage of certain properties of resistance to certain attacks. For example, naively, resistance to an attack of the filtering type can be obtained by modifying the frequency coefficients corresponding to the low frequencies. Another essential point to consider is the synchronization of the mark; in other words, the choice of the insertion sites in an image. If the attacks we are facing have an influence on the position of the coefficients, it is necessary to take this into account at the insertion (and detection) stage. In some cases, the choices for insertion space and synchronization are not that different. For example, in the spacial domain, inserting information on pixels representing the edge areas (choice of coefficients according to a fixed property) can be equivalent to inserting information in high frequencies.

For the following, in order to more precisely illustrate the impact of the use of error-correcting codes on a watermarking method, we propose the use of a quantization strategy called index modulation (or QIM) (Chen and Wornell 1999b) for its performance when combined with error-correcting codes.

3.3. Index modulation

In 1999, Chen and Wornell proposed index modulation (or the *QIM method*) to perform watermarking by quantization. This algorithm is known for its ease of implementation and its low cost of calculation, but also for its fragility in the face of scaling attacks. Pérez-González *et al*. (2005) suggest an improvement called *Rational Dither Modulation* to better resist this last attack. Chen and Wornell have also proposed variations, such as *Dither Modulation (DM) QIM* and *Distortion Compensated (DC) QIM* as well as an associated theoretical study. The DMQIM method is an adaptation of the QIM method which moves the quantification cells with a random vector (Zamir and Feder 1994; Chen and Wornell 1999b). The modified values are more difficult to identify and make access to the watermarking channel more difficult. The DCQIM method is composed of a processing step added after the QIM quantification in order to improve the invisibility/robustness compromise. The authors also propose a vector variation called *Spread Transform Dither Modulation,* which consists of quantifying the projection of the sample vectors on a direction axis. Chen and Wornell proposed a second extension of the QIM method called *Lattice QIM* (Moulin and Koetter 2005), a method that we have chosen to use in this chapter for its regular structure (Euclidean network) and its robustness potential, previously associated with error-correcting codes. There are many contributions using the QIM method and its variations (e.g. Eggers *et al*. (2000a); Bas *et al*. (2003); Wang and Lin (2004); Oostveen *et al*. (2004); Li and Cox (2007); Abdul *et al*. (2013)).

In the next part of this chapter, we propose to study the *Lattice QIM* (LQIM) method.

3.3.1. *LQIM: insertion*

The quantization space of this method is a Euclidean lattice ($\mathbb{Z}^L$) of dimension L where a host sample $x \in \mathbb{R}^L$ is transformed into an element y from this network. The scalar QIM method is the special case $L = 1$.

The sample x is made by selecting coefficients from the image. To insert a bit of information, L pixel values are selected in the image. For example, a strategy of random selection of these values can be adopted. These insertion sites are secret and only accessible to the sender and the receiver because of a secret key.

To insert a bit of information $m \in \{0, 1\}$, x is quantized by y belonging to one of the following subsets called *cosets* denoted by Λ_0 and Λ_1 defined as:

$$\begin{cases} \Lambda_0 = \Delta\mathbb{Z}^L - \dfrac{\Delta}{4} \\ \Lambda_1 = \Delta\mathbb{Z}^L + \dfrac{\Delta}{4} \end{cases} \qquad [3.1]$$

thanks to the following quantization function Q_m:

$$y = Q_m(x, \Delta) = \left\lfloor \frac{x}{\Delta} \right\rfloor \Delta + (-1)^{m+1} \frac{\Delta}{4} \quad [3.2]$$

with Δ the quantization step of the method.

Figure 3.4 is an example of quantization in dimension $L = 2$ of vector x. For any cross or circle with center y_m, the dotted diamonds define the boundaries of each quantization cell. On insertion, x is transformed into the nearest y_m in Euclidean distance.

3.3.2. *LQIM: detection*

On reception of an image, which has been marked and then modified, the message can be detected by extracting the coefficients used for insertion using the secret key. To extract the information bit m, the coset closest to the associated z vector is calculated to determine the estimate $\hat{m}$ such that:

$$\begin{aligned} \hat{m} &= \arg \min_{m \in \{0,1\}} d(z, \Lambda_m) \\ d(z, \Lambda) &= \min_{y \in \Lambda} \|z - y\|_2 \end{aligned} \quad [3.3]$$

with the Euclidean norm, $\|.\|_2$.

The detection equation [3.3] is illustrated with Figure 3.4.

In this section, we have described the insertion and detection part of Figures 3.1 and 3.3. In the case of a real application, the transformation Tr and the method of extracting coefficients or synchronization must also be selected. A simple example is the insertion in the spatial domain with random extraction of pixels. In section 3.4, we propose to tackle the integration of error-correcting codes more specifically in digital watermarking strategies.

3.4. Error-correcting codes approach

Error-correcting codes are powerful tools in information theory. They are used to resolve problems to do with signal transmission in noisy channels (not reliable). Robust digital watermarking can be seen as a transmission issue. In this chapter, we intend to review how error-correcting codes can be integrated into watermarking strategies in order to improve robustness against certain attacks. Depending on the

attack encountered, the errors produced may have a particular structure and therefore one error-correcting code may be more effective than another. The next part of this section is dedicated to a brief overview of the BCH (Bose and Ray-Chaudhuri 1960) and RS (Reed and Solomon 1960) codes, which are Hamming codes (Hamming 1950).

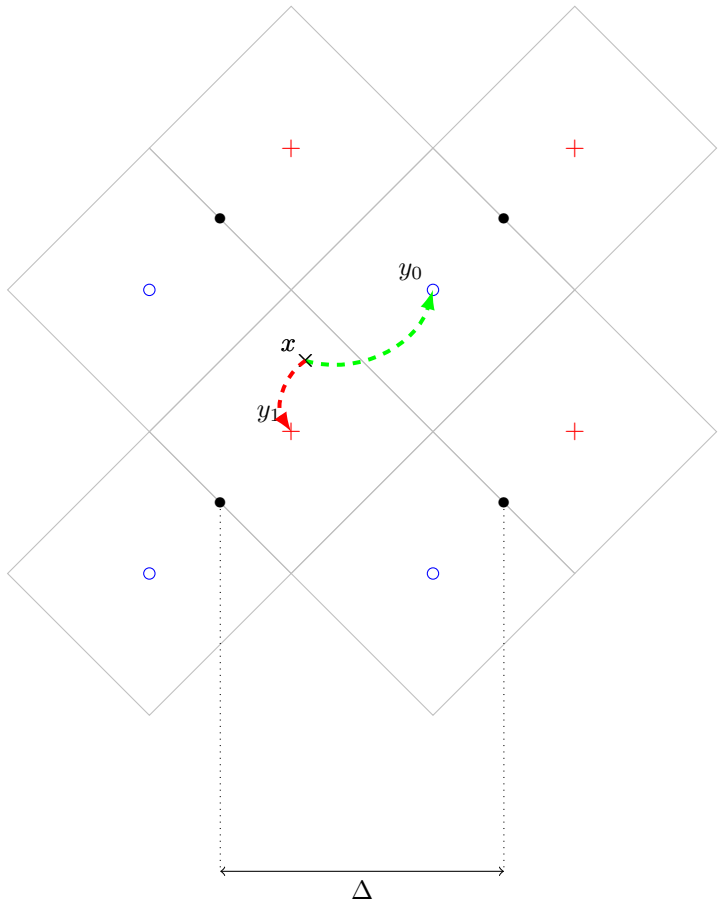


Figure 3.4. *Representation of the quantization space (or Euclidean network) in dimension L = 2. The symbol + represents bit 1 (coset Λ_1) and ∘ represents bit 0 (coset Λ_0)*

3.4.1. *Generalities*

When data are transmitted in a channel, transmission errors can occur. The role of error-correcting codes is to correct these transmission errors. To protect this information, a code corrector adds repetition to a message so that errors can be detected and corrected after transmission. In Figure 3.5, we present a diagram illustrating the different stages of transformation and transmission of a message.

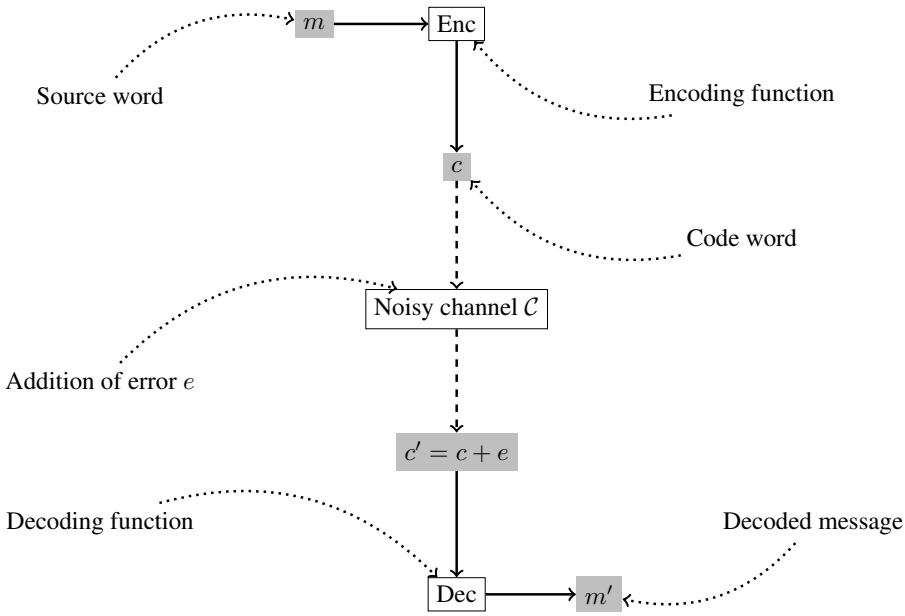


Figure 3.5. *The different stages allowing the reliable transmission of a message in a channel*

An encoding function is an injective application ϕ defined as:

$$\phi\colon \{0,1\}^k \to \{0,1\}^n$$

with k being the dimension of the code and n being the length of the code.

ϕ designs a code of parameters (k, n). So a word (from the source) $m \in \{0,1\}^k$ has a code word for an image $c = \phi(m) \in \{0,1\}^n$. The encoding function ϕ is designed by Enc() (Figure 3.5). A code corrector is a set $C = Im(\phi)$.

When a code word c' is transmitted in a channel, it may contain errors. To correct these, a basic algorithm includes comparing the word received with the code words; in other words, the elements of C, such that:

$$C = Im(\phi) = \{\phi(m) \mid m \in \{0,1\}^k\}$$

to find $c \in C$ which is closest to c', in the sense of Hamming distance.

The word c' is then decoded through the code word c.

Hamming distance is the number of distinct symbols between two words c_1 and c_2. It is defined by:

$$d_H(c_1, c_2) = \#\{i \mid (c_1)_i \neq (c_2)_i\}$$

We also define *Hamming weights* as the number of symbols, that are not zero, of a word c. It is defined by:

$$w_H(c) = \#\{i \mid c_i \neq 0\}$$

Note that this basic algorithm takes a long time to finish because the number of comparisons made is exponential (complexity in $\mathcal{O}(n2^k)$). If k is small, the decoding time is acceptable, but this becomes impractical when k is larger. When decoding, we implicitly assume that when the number of errors is low enough, the number of code words closest to c' is equal to 1 (perfect codes).

Another important basic concept about correcting codes is the idea of *minimal distance* which represents the smallest distance between two code words of a C code. It makes it possible to characterize the decoding power. A minimum distance code d is written as (n, k, d) and is capable of correcting t errors, such that:

$$t = \frac{d-1}{2}$$

There are codes with more structure, such as linear codes. A code is said to be *linear* if there is a matrix $G \in \mathcal{M}_{n,k}(GF(2))$ with k rows and n columns with coefficients in $\{0,1\}$ of rank = k, such that:

$$\forall m \in \{0,1\}^k, \phi(m) = G \times m$$

The linearity of a code gives image C a vector subspace structure of $\{0,1\}$. The minimal distance of a linear code d is therefore equal to the smallest non-zero

Hamming weight of a code word of C. For a given linear code, it can be hard work to determine its minimal distance. However, there is an increase in this called the *Singleton bound*:

$$d \leq n - k + 1$$

When $d = n - k + 1$, the parameter code (n, k, d) is a code known as *maximum distance separable*(MDS).

G is called the *generator matrix of* ϕ. A linear code is also associated with the concept of *control matrix*. *Control matrix* $H \in \mathcal{M}_{n-k,n}(GF(2))$ of the code C is defined as:

$$m \in C \iff Hm = 0$$

The main advantage of the linearity of a code is being able to build faster decoding algorithms. We have introduced the most important aspects of the basic concepts on corrective codes and give an overview of the BCH and RS codes in sections 3.4.3 and 3.4.4. First, we propose to study a particular construction widely used in watermarking: code concatenation.

3.4.2. *Codes by concatenation*

The robustness of a watermarking algorithm can be improved by using the principle of error correction by concatenation (Figures 3.6–3.8). In the context of watermarking, it is quite simply a question of using two concatenated codes: a so-called external coding or "outer coding", which will be a simple coding by repetition, and an internal coding or "inner coding", which will be a classical code like Hamming code, BCH or RS. During decoding, the repetition of the mark makes it possible to reduce the level of error (and can also provide certain properties such as robustness to cropping attacks) and the associated classical code will have the function of correcting the errors in order to be able to decide whether the signature received and reconstructed is valid or not. In fact, an error-correcting code cannot show its potential when the error level is too high, and therefore this repetition of the encoded signature makes it possible to improve the error rate of the channel and bring it to an acceptable level. In some cases, this allows interaction on Watermark channels at very high error levels ($0.1 < \text{BER} < 0.5$).

Figure 3.6 shows this principal of construction of the mark that we are going to insert from the signature to be hidden. The repeat setting is adaptive, depending on the size of the image and the signature.

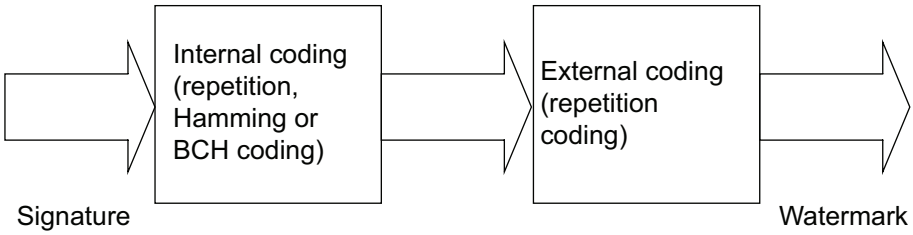


Figure 3.6. *Strategy by "concatenation codes"*

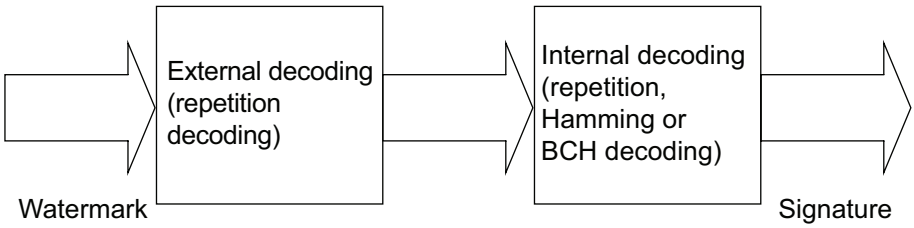


Figure 3.7. *Decoding in the case of the code by "concatenation codes"*

Obviously on reception, the opposite procedure is applied to decode the signature from the mark, as shown in Figure 3.7.

Figure 3.8 illustrates the complete diagram associated with the use of codes in the context of watermarking. First of all, a signature is created from a secret key. Then this signature is encoded through the internal encoding (in the example we use a simple 4-bit repetition code). Then the external coding is applied by repetition of the encoded signature, with a repetition parameter that depends on the size of the image or on the watermarking strategy (e.g. ROI encoding).

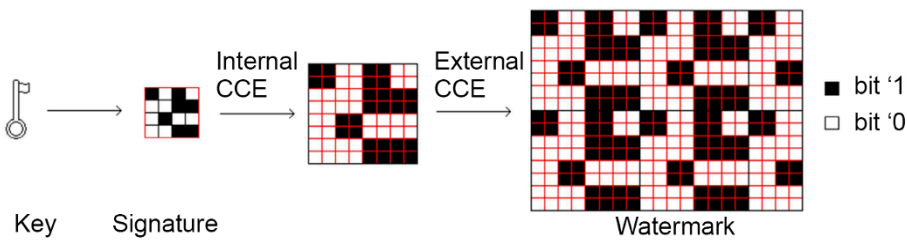


Figure 3.8. *Illustration of the creation of a mark by code concatenation*

Note that we are talking about repeating code because a repeating code simply consists of building the mark from the signature by repeating each code word n times. Decoding simply consists of an averaging operation for each code word received, in order to distinguish a 0 from a 1.

3.4.3. *Hamming codes*

Hamming codes are linear block codes. For an integer $m > 1$, we have the following representation for binary Hamming codes in the form $(n, k, d) = (2^m - 1, 2^m - 1 - m, m)$.

For m = 3, we have Hamming codes of parameters $(7, 4, 3)$. These Hamming codes code size 4 data bits in blocks of length 7 (Hamming code word). The additional 3 bits are parity bits. Each of these 3 bits represents the parity of 3 of the 4 data bits, and no parity bit represents the same data bits twice. All parity bits are even parity. The Hamming error-correcting code $(7, 4, 3)$ can correct a 1-bit error in each of the Hamming code words.

For a linear code (n, k, d), the matrix of generator G is a $k \times n$ matrix, for which the vector space generated is the code given. A parity-check matrix for a linear code (n, k, d) is a $(n - k) \times n$ H matrix, for which $Hx = 0$ for any style of code x.

The generator matrix G used in the making of Hamming codes is made up of I, the identity matrix and the parity-check matrix A, where $G = [I|A]$.

The process of building the code is as follows.

Let d_1, d_2, d_3 and d_4 be bits of data, and a Hamming code (7,4,3) can define parity bits p_1, p_2 and p_3 such that:

$$p_1 = d_1 \oplus d_2 \oplus d_4$$
$$p_2 = d_1 \oplus d_3 \oplus d_4$$
$$p_3 = d_2 \oplus d_3 \oplus d_4$$

We can now build the generator matrix G from the Hamming code:

$$G = \begin{pmatrix} 1\,1\,1\,0\,0\,0\,0 \\ 1\,0\,0\,1\,1\,0\,0 \\ 0\,1\,0\,1\,0\,1\,0 \\ 1\,1\,0\,1\,0\,0\,1 \end{pmatrix}$$

where the first, second and fourth bits are parity bits, respectively, and the third, fifth, sixth and seventh bits are data bits, respectively. This is specified by the following elements:

$$x_1 = x_3 \oplus x_5 \oplus x_7$$
$$x_2 = x_3 \oplus x_6 \oplus x_7$$
$$x_4 = x_5 \oplus x_6 \oplus x_7$$

with (x_1, x_2, x_4) and (x_3, x_5, x_6, x_7) denoting parity and data bits, respectively.

From the side of the recipient, we have the parity-check matrix $H = [A^T|I]$ given by h_1, h_2 and h_3, where:

$$h_1 = x_4 \oplus x_5 \oplus x_6 \oplus x_7$$
$$h_2 = x_2 \oplus x_3 \oplus x_6 \oplus x_7$$
$$h_3 = x_1 \oplus x_3 \oplus x_5 \oplus x_7$$

If there is an error (assuming one at most), $(h_1 h_2 h_3)$ will be the binary representation of indications of incorrect bits. A binary Hamming code has a parity-check matrix, which has m column vectors. For the code with parameters $(7, 4, 3)$, the parity-check matrix is:

$$H = \begin{pmatrix} 0\,0\,0\,1\,1\,1\,1 \\ 0\,1\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,1\,0\,1 \end{pmatrix}$$

When we multiply the code word received by the parity-check matrix, we get the corresponding parity ranging from 000 to 111. These three bits give us the location of the error. The word 000 indicates that there was no transmission error, while non-zero words (001 through 111) indicate the location of the error relative to the seven received code word bits. We can then correct at most $t = \lfloor (d-1)/2 \rfloor$ errors because the minimum Hamming distance between our code words is 7 – 4 = 3. We can therefore correct 1 error at most.

As soon as we know the location of the error, we can simply reverse the corresponding bit to correct the error. Next, we remove the parity bits from positions one, two and four to extract the message. The Hamming codes are perfect error correction codes 1. In other words, any word received including one error at most is

decoded correctly and the code has the smallest possible size of all the codes, which correct a single error.

The Hamming codes that we used can correct an error in each code word and, additionally, they allowed us to simply introduce the concepts related to the use of the codes. However, it is necessary to study other types of codes that correct more errors. We have selected BCH codes, which are explained in section 3.4.4.

3.4.4. *BCH codes*

3.4.4.1. *Cyclic block codes*

Let C be a linear block code over a finite field F of length n. C is called a cyclic code, if for each code word $c = (c_1, ..., c_n)$ of C, the word $(c_n, c_1, ..., c_{n-1})$ of F^n obtained by a cyclic shift to the right of the components is also a code word of C.

BCH codes are codes in cyclic blocks, such that for any positive integer $m \geq 3$ and t with $t \leq 2^{m-1} - 1$, there is a BCH code of length $n = 2^m - 1$ able to correct t errors and of dimension $k = n - m \times t$.

To go even further, we define some basic terminology.

– Irreducible polynomial: we call a polynomial $f(x)$ irreducible over $K[x]$ if it has no proper divisors in the field $K[x]$.

– Primitive polynomial: the polynomial $f(x)$ is primitive if it is irreducible of degree $n > 1$ and it is not a divisor of $1 + x^m$ for all $m < 2^n - 1$.

– Galois field: for each primitive polynomial of order p^m, there is a unique finite field of order p^m denoted by $GF(p^m)$. $\alpha \in GF(2^r)$ is primitive if $\alpha^m \neq 1$ for $1 \leq m \leq 2^r - 1$ and each non-zero word in $GF(2^r)$ can be expressed as a power of α.

In Table 3.1, the construction of $GF(2^4)$ using the primitive polynomial $h(x) = 1 + x + x^4$ is illustrated, where each vector is indicated as a power of β.

– Minimal polynomial: the minimal polynomial of α is the polynomial $K[x]$ of smallest degree admitting for root $m_\alpha(x)$ where $m_\alpha(x)$ is irreducible in K. If $f(x)$ is any polynomial above K such that $f(\alpha) = 0$, then $m_\alpha(x)$ is a factor of $f(x)$ and $m_\alpha(x)$ is unique. $m_\alpha(x)$ is also a factor of $1+x^{2^r-1}$. α is an element of $F = GF(2^r)$ and the order of α is the smallest positive integer m such that $\alpha^m = 1$ and $\alpha \in GF(2^r)$ is a primitive element if there is order $2^r - 1$.

3.4.4.2. *An example of construction: BCH (15,7,5)*

BCH error-correcting codes of length $2^r - 1$ $(r \geq 4)$ are defined by $g(x) = m_\beta(x)m_{\beta^3}(x)$. The generator polynomial $gp(x) = m_1(x)m_3(x)$ is of degree $2r$ and

the code has dimension $n - 2r = 2^r - 1 - 2r$. β is a primitive element in $GF(2^4)$ made with $p(x) = 1+x+x^4$, $m_1(x) = 1+x+x^4$ and $m_3(x) = 1+x+x^2+x^3+x^4$. Therefore, $gp(x) = m_1(x)m_3(x) = 1+x^4+x^6+x^7+x^8$ is the generator polynomial of BCH code $(15, 7, 5)$ which, in turn, gives us the generator matrix G:

$$G = \begin{pmatrix} 1\,0\,0\,0\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,1 \\ 0\,0\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,1\,0 \\ 0\,1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0 \\ 1\,0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0 \\ 0\,1\,1\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1 \\ 1\,1\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0 \end{pmatrix}$$

Word	Polynomial in x	Power of β
0000	0	–
1000	1	$\beta^0 = 1$
0100	x	β^1
0010	x^2	β^2
0001	x^3	β^3
1100	$1 + x = x^4$	β^4
0110	$x + x^2 = x^5$	β^5
0011	$x^2 + x^3 = x^6$	β^6
1101	$1 + x + x^3 = x^7$	β^7
1010	$1 + x^2 = x^8$	β^8
0101	$x + x^3 = x^9$	β^9
1110	$1 + x + x^2 = x^{10}$	β^{10}
0111	$x + x^2 + x^3 = x^{11}$	β^{11}
1111	$1 + x + x^2 + x^3 = x^{12}$	β^{12}
1011	$1 + x^2 + x^3 = x^{13}$	β^{13}
1001	$1 + x^3 = x^{14}$	β^{14}

Table 3.1. *Galois field GF(2^4)*

H is the parity check matrix for error correcting codes BCH $(15, 7, 5)$. It is defined by:

$$H = \begin{pmatrix} \beta^0 & \beta^0 \\ \beta & \beta^3 \\ \beta^2 & \beta^6 \\ \vdots & \vdots \\ \beta^{2^r-2} & \beta^{3(2^r-2)} \end{pmatrix}$$

By using Table 3.1, we obtain the following parity-check matrix:

$$H = \begin{pmatrix} 1 & 1 \\ \beta & \beta^3 \\ \beta^2 & \beta^6 \\ \beta^3 & \beta^9 \\ \beta^4 & \beta^{12} \\ \beta^5 & 1 \\ \beta^6 & \beta^3 \\ \beta^7 & \beta^6 \\ \beta^8 & \beta^9 \\ \beta^9 & \beta^{12} \\ \beta^{10} & 1 \\ \beta^{11} & \beta^3 \\ \beta^{12} & \beta^6 \\ \beta^{13} & \beta^9 \\ \beta^{14} & \beta^{12} \end{pmatrix}$$

The decoding algorithm will then be constructed as follows:

1) calculate the syndrome $wH = [s_1, s_3] = [w(\beta), w(\beta^3)]$;

2) if $s_1 = s_3 = 0$, there are no errors;

3) if $(s_1)^3 = s_3$, a single error in position i, where $s_1 = \beta^i$;

4) if $s_3 = 0$ and $s_1 \neq 0$, two errors in positions $(i + 5)\%15$ and $(i + 10)\%15$, where $s_1 = \beta^i$;

5) constructing a quadratic equation:

$$x^2 + s_1 x + (\frac{s_3}{s_1} + s_1^2) = 0$$

BCH codes are designed for random errors. For packet errors, it can be more effective to use RS codes.

3.4.5. *RS codes*

3.4.5.1. *Principle of RS codes*

RS codes (Reed and Solomon 1960) are non-binary cyclic correcting codes whose symbols are made of m bits, where $m > 2$.

RS codes (n, k, d) of symbols containing m bits exist for all n, k and d satisfying the following conditions:

$$0 < k < n < 2^m + 2$$

where k is the number of data symbols (in other words, the size of the code) to encode and n is the number of symbols in a code word (in other words, the size of the code):

$$(n, k, d) = (2^m - 1, 2^m - 1 - 2t, 2t + 1)$$

where d is the minimal Hamming distance between the code words transmitted and received; in other words, the number of different symbols between the two words and t the capacity of the code correction. RS codes can correct any combination of up to t where t is given by:

$$t = \lfloor (d - 1)/2 \rfloor$$

3.4.5.2. *Bounded distance decoding algorithms*

Bounded distance decoding algorithms can correct up to t erroneous symbols. One of the algorithms (Justesen and Høholdt 2004) is illustrated in this section.

For a word received $r = (r_1, r_2, ..., r_n)$:

1) solve the following linear system:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{l_0} & r_1 & r_1x_1 & \dots & r_1x_1^{l_1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{l_0} & r_2 & r_2x_2 & \dots & r_2x_2^{l_1} \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{l_0} & r_n & r_nx_n & \dots & r_nx_n^{l_1} \end{bmatrix} \begin{bmatrix} Q_{0,0} \\ Q_{0,1} \\ Q_{0,2} \\ \vdots \\ Q_{0,l_0} \\ Q_{1,0} \\ Q_{1,1} \\ \vdots \\ Q_{1,l_1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad [3.4]$$

with $l_0 = n - 1 - t$ and $l_1 = n - 1 - t - (k - 1)$;

2) write:

$$Q_0(x) = \sum_{j=0}^{l_0} Q_{0,j}x^j$$

and:

$$Q_1(x) = \sum_{j=1}^{l_1} Q_{1,j}x^j$$

3) calculate:

$$g(x) = -\frac{Q_0(x)}{Q_1(x)}$$

4) if $g(x) \in F[x]$ then the corrected code word is $cc = g(x_1), g(x_2), \dots, g(x_n)$ if not, decoding fails.

3.4.5.3. *Performance against packet errors of RS codes*

RS codes are particularly useful against burst noise and they exhibit lower performance against errors that are random. This is illustrated in the following example.

Consider a $(n, k, d) = (40, 11, 30)$ RS code, where each symbol is made up of $m = 6$ bits, as shown in Figure 3.9. As $d = 30$ indicates that this code can correct any $t = 14$ symbol error in a block of 40. Consider the presence of a burst of noise

lasting 60 bits that disturbs 10 symbols, as shown in Figure 3.9. The RS $(40, 11, 30)$ error correcting codes can correct any 14 symbol errors using the bounded distance decoding algorithm regardless of the type of error the attack induced.

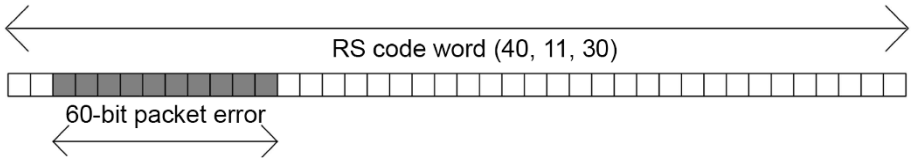


Figure 3.9. *Performance of RS codes against packet errors*

The code corrects by blocks of 6 bits and replaces the whole symbol by the correct one without taking into account the number of corrupted bits in the symbol, that is, it treats an error of 1 bit in the symbol in the same way as it treats an error of 6 bits of the symbol, replacing them with the correct 6 bit symbol.

This gives the RS codes a tremendous burst noise advantage over binary codes. In this example, if the 60-bit noise disturbance can occur in a random fashion rather than as a contiguous burst, it could affect many more than 14 symbols, which is beyond the capability of the code. This shows the advantage of using RS codes to correct burst errors and their inability to correct random errors beyond a certain limit.

Any attack that affects the watermarked image in a random fashion might make the watermark decoding difficult for images watermarked using RS codes. In the watermarking channel, the errors, characterized by the different attacks, occur in random or burst manner. Depending on the placement of the watermark in an image and the use of error correcting codes, the robustness of the signature can be increased against the attacks.

Note that for RS codes, conventional bounded distance decoding algorithms correct up to $t = \lfloor (n-k)/2 \rfloor$ symbol errors, as shown in the example above. By using list decoding, Sudan (1997) and later Guruswami and Sudan (1998) have shown that the RS's error correction capability can be improved to $t_S = n - \sqrt{2kn}$ and $t_{GS} = n - \sqrt{nk}$, respectively, which makes then more effective against burst errors or errors in bulk.

To conclude, the integration of corrective codes in a digital watermarking strategy consists of inserting a code word in an image. Against a given attack, the question will be to choose the correction code that provides the best correction performance by observing the structure of the errors linked to this attack (block or random error, for example). However, using "classical" codes is not always the best strategy. For example, another error structure has recently been studied, associated with codes

called *rank-metric codes* that are best for correcting this error structure. The works of Lefèvre *et al*. (2018, 2019) are the first contributions introducing rank-metric codes to improve the robustness of digital watermarking. These codes are different to classical Hamming codes because the metric corresponds to the rank of a matrix on a finite field and has the mathematical properties of a distance. We discuss these codes at the end of this chapter.

3.5. Contradictory objectives of watermarking: the impact of codes

The aim of robust watermarking is to optimize three properties in particular: the maximum amount of information that an image can contain, the invisibility of the mark (and preservation of the quality of the host content) and the robustness of the mark to image changes and, in some cases, security. This is why in recent years we have defined watermarking according to these four characteristics. However, in general, when one of these properties is improved, the others are degraded.

Watermarking requires a mark to be *invisible* to the naked eye and does not degrade the host content. To achieve this, the distortions between host content and marked content must be low enough. Ideally, a mark should be imperceptible to the human visual system (HVS), which is the subject of Chapter 4 of this volume, entitled "Invisibility". This can also be invisible to a machine, in other words, a statistical invisibility, but it is a property that relates more to steganography than to digital watermarking. For the HVS, the perceptual approach of digital watermarking of images makes it possible to create insertion distortions in the areas of the image most sensitive to visual change. In Chapter 4, we discuss how we can take into account HVS sensitivity to minimize psycho-visual distortions to the insertion in the specific framework of color images. Concretely, the addition of an error correction code in the construction of the inserted mark does not have a direct impact on this idea of invisibility. It is obvious that the burying strategy is the main process influencing this invisibility. However, the introduction of a code that allows an increase in robustness can make it possible, in certain cases, to limit the "insertion force" parameter needed for the correct decoding of the signature, such as the parameter of quantification in the example that serves as a common theme. And so, we limit visual degradation indirectly.

The *capacity* is the maximum amount of useful information that the image to be marked can contain with respect to a given insertion method. In general, we always use binary symbols, but it is possible to encode the information on only two symbols. The more information we insert, the more the number of modified coefficients increases and the more the distortion increases. The quality of the marked image decreases.

To measure this amount, we can calculate the embedding rate in the spatial domain, such that:

$$\mathrm{ER}(\mathcal{X}, \mathcal{Y}) = \frac{l}{|\{(i,j) \mid \mathcal{Y}[i,j] \neq \mathcal{X}[i,j], 1 \leq i \leq m, 1 \leq j \leq n\}|} \quad [3.5]$$

with 1 being the bit number of the message, and the denominator is the number of modified pixels for image $\mathcal{X}$ of size $m \times n$.

The unit of this measurement is bit per pixel. We can also make the same measurement in a transformed space (bit per modified coefficients). At first glance, the insertion of an encoding step will limit this capacity since we are adding repetition. This is even true with the code concatenation process. However, *at first glance*, the use of codes makes it possible to get to the maximum capacity level for a given robustness and level of attack.

Finally, the *robustness* of a watermark is its ability to preserve the message it transmits against a distortion (or attack) on the marked image. In practice, an attack has parameters that indicate its "force"; in other words, how much the noise signal damages the host signal containing the mark. There are many types of attacks. The so-called desynchronization attacks change the location of the mark's insertion sites. For example, we have the geometric modifications of the image, such as translation or rotation. Image cropping and re-framing also desynchronizes the mark by changing the image size. In this case, it is necessary to adapt the insertion method of the mark. In other words, a simple vector quantization on the pixels of the image cannot prevent a desynchronization of coefficients.

There are different ways to measure the robustness of a mark. The most well-known example is the calculation of an error rate (*bit error rate* (TEB or BER)), between the original message and the one estimated by the receptor of the marked image:

$$\mathrm{BER} = \frac{|\{i \mid m_i \neq m'_i\}|}{l} \quad [3.6]$$

with $m' = (m'_1, \ldots, m'_l)$ the detected message.

Another well-known way is to calculate the correlation between m and m'.

Depending on the needs and analysis criteria of each, an image error rate can be defined (*Image Error Rate* (IER)) and is defined as:

$$\mathrm{IER} = \begin{cases} 0 \text{ if BER} = 0 \\ 1 \text{ otherwise} \end{cases} \quad [3.7]$$

which makes it possible to measure the performance of a correction code without a decoding algorithm, for example. In the case of a binary code corrector, the number of detected errors must not exceed the correction rate of the code chosen. So IER = 0 means decoding took place without errors. If we are dealing with non-binary symbols, we are using a symbol error rate, which is the ratio of erroneous non-binary symbols.

In practice, when an attack is powerful enough to delete a mark, the detected BER is 0.5, which corresponds to a random binary sequence, but when the distortions generated by this attack are less significant, the BER tends toward 0. However, in certain situations, the BER is close to 1, which means that almost all the bits have been inverted. In this case, the detected information is not destroyed (since it is "inverted"), in that the Shannon entropy does not vary.

The *security* of a watermarking scheme is as important a subject as the three properties previously discussed. It is inspired by security in cryptography to protect multimedia content. However, there are important differences between these two research areas (Cox *et al*. 2006). Being able to determine a level of security requires a specific study of the application for which a watermarking scheme is intended. In the context of robust watermarking, the aim of a watermarking method is to insert an invisible signal in a host image while maintaining acceptable image quality. Here, the security analysis involves evaluating the problem of an attacker (illegitimate recipient) accessing or even modifying the watermarking channel (Barni *et al*. 2003). A secure watermarking method can produce, for example, marked images whose access to the mark channel is inaccessible without a key, which is only known to the sender and the legitimate recipient. In this chapter, we do not discuss security analysis.

3.6. Latest developments in the use of correction codes for watermarking

The first contributions on (semi-)fragile watermarking using correction codes appeared in the 2000s. Lee and Won suggested using parity bits as a message of the mark, whose bit positions are mixed thanks to a random sequence (or insertion key). The imperceptibility of the mark is ensured by modifying only the least significant bits (Lee and Won 2000). Their algorithm makes it possible to restore the modified zones. Sun and Chang (2002) have introduced a semi-fragile method of authentication using correction codes. They also propose inserting parity bits of code words in coefficients invariant to the JPEG compression. Then, He *et al.* proposed inserting a mark by using coefficients extracted from ART (*Angular Radial Transform*) descriptors, which are invariant to geometric type distortions (He *et al*. 2003). ART coefficients and the message inserted are both encoded by a Hamming code, thus enabling robustness to the most common image modifications. Their works focus on resistance to these attacks and do not offer a study on detection and

localization performance. As for Zhou *et al.*, they proposed a semi-fragile block authentication algorithm in the wavelet domain, capable of detecting and locating modified parts of the image (Zhou *et al.* 2004). Their authentication mechanism is based on a signature extracted from the host image, which is encoded by a small BCH code. Then, Chan and Chang (2007) developed a method for inserting BCH codewords into least significant bits to locate and restore modified pixels. They later improved their works by increasing the precision of their high significance bit prediction method (Chan and Chang 2007). At the same time, Chang *et al.* (2011) showed how to overcome packet error problems occurring in high-order bits, as well as how to resist the vector quantization attack described by Wong and Memon (2000). They describe a fragile method using a small Hamming code and a chaotic function (similar to a pseudo-random number generator) capable of locating modified regions.

In the rest of this section, we offer a non-exhaustive state of the art of correcting codes applied to *robust* digital watermarking. The required message to be inserted into a mark is encoded by a generator matrix of a correction code, then the code word obtained is used as *payload* at the insertion stage. At the detection stage, an estimate of the code word is calculated and then decoded by the correction code. The integration of the correction codes for the digital watermark is therefore done ad hoc.

The first works appeared in the early 2000s. Kesal *et al.* (2000) proposed the binary codes produced by Reed Muller with an iterative decoding (Kschischang *et al.* 2001; Al-Askary 2003) to deal with the Gaussian channel. Darbon *et al.* (2001) studied combinations of BCH codes and repeating codes to optimize robustness performance against attacks such as JPEG compression and the median filter, while ensuring optimal capacity. For greater clarity, we use the code concatenation diagram in Figure 3.10. We then have a BCH code $\mathcal{C}_1$, and a repetition code $\mathcal{C}_2$. The $\text{Enc}_1()$ encoding function can represent the use of a repetition code or a BCH code and the $\text{Enc}_2()$ function represents a repetition code.

$$m = (m_1, \ldots, m_k) \longmapsto \underbrace{c = mG_1 = (c_1, \ldots, c_{n_1})}_{\substack{\text{Encoding of } m \text{ by } \mathcal{C}_1 \\ n_1 - k \text{ repetition bits}}} \longmapsto \underbrace{d = cG_2 = (d_1, \ldots, d_{n_2}).}_{\substack{\text{Encoding of } c \text{ by } \mathcal{C}_2 \\ n_2 - n_1 \text{ repetition bits}}}$$

Figure 3.10. *Concatenation diagram (or hybrid coding) of two correction codes $\mathcal{C}_1(n_1, k)$ (inner coding) and $\mathcal{C}_2(n_2, n_1)$ (outer coding) of respective generator matrices G_1 and G_2. The word m is a required message of k bits. The final code word d is inserted into the image as a mark*

The concept of form of error is not present in this work. Similarly for the work of Baudry *et al.*, a similar study of these codes was proposed for the Gaussian

channel (Baudry *et al*. 2001). Even if Zinger *et al.* proposed similar works (*hybrid* encoding), they are interested in different variants on BCH codes, such as BCH codes *by subtraction*, *extended* and *by bit* (Zinger *et al*. 2001). The aim of these variants is to gain flexibility (length limited to $n = 2^m - 1$) on the parameters of the BCH codes in order to optimize the capacity. Using BCH codes by bit involves using several shorter length code words to encode as many bits as possible against an allowed *payload* size, while also improving the maximum number of errors to correct. The idea is the same with BCH codes by subtraction and extended BCH codes: the code words can be shortened or lengthened by a few bits while also keeping the same generator polynomial (see MacWilliams and Sloane (1977)).

Later, many works on codes inspired by this work have been published on different types of watermarking support. For example, Hsieh and Wu and Chan *et al.* offered improvements to the robustness of an image watermark against JPEG compression, low-pass filter, to certain geometric attacks and even against the Stirmark attack (Hsieh and Wu 2001; Chan *et al*. 2005).

Regarding the video, Chan *et al.* developed a method based on wavelet coefficient decomposition which inserts the repetition bits of a code word into the audio channel and obtained better robustness against various video attacks, such as reduced video quality (*frame dropping*) (Chan and Lyu 2003). There are also works for sound processing: Liu and Lin (2006) have incorporated BCH codes to improve the robustness of their mark against attacks, such as amplitude changes or MP3 compression.

Other examples have also proposed integrating RS codes to correct burst errors (Abdul *et al*. 2013). The work developed also implements list decoding for RS codes. In addition, the correction codes have made it possible to improve watermarking methods to authenticate images (Zhang *et al*. 2005; Chan and Chang 2007). Works have also been developed for tracing data (Schaathun 2008). More recent works propose new methods integrating correction codes, but their use remains the same (encoding before insertion and decoding after detection of the mark).

Finally, a large number of publications have applied *convolutional* codes (e.g. (Viterbi 1967; Hernandez *et al*. 2000; Lancini *et al*. 2002)) and *Turbo codes* (e.g. (Berrou *et al*. 1993; Baldo *et al*. 2001; Rey *et al*. 2003; Doërr *et al*. 2005; Dugelay *et al*. 2006; Chemak *et al*. 2007)) by watermarking. In the case where the marked signal is transmitted in a noisy channel (Gaussian channel, JPEG compression, or low-pass filters), Turbo codes make it possible to obtain the best detection performance while approaching the capacity (second Shannon theory). In order to summarize this section, we present a summary of some works in the literature in Table 3.2.

Authors	Codes	Contributions
Kesal *et al*. (2000)	T, RM, H, DI	QV, CMP, CG
Hernandez *et al*. (2000)	CConv, DV, BCH	SS, CMP, image/video, CG, JPEG
Eggers *et al*. (2000b)	T, BCH	CS + T, *blind/non-blind*, image, filter, CG
Darbon *et al*. (2001)	Hybrid[2]	*SW*+ QV, *tradeoff*, image, JPEG, CG
Baudry *et al*. (2001)	Hybrid, CConv, DV	*SW*, AR, image, compr JPEG,CG
Zinger *et al*. (2001)	Hybrid, BCH[3]	Theoretical, image, CBS
Hsieh and Wu (2001)	Hybrid	CMP, image, JPEG, filters, AG
Baldo *et al*. (2001)	Turbo, DI	SS, CMP, image, CG
Lancini *et al*. (2002)	CC, Turbo	CMP, image, video, MPEG, AG
Chan and Lyu (2003)	RS, Turbo	Audio domain, video, video attacks
Chan *et al*. (2005)	RS, Turbo	Audio domain, video, video attacks
Rey *et al*. (2003)	Turbo block, BCH, REP	Eurecom, CMP, image, CG, JPEG
Cvejic *et al*. (2003)	Turbo	SS, Codes, audio, filters, MP3
Zhang *et al*. (2005)	CC	Authentication, codes, color, fragile
Doërr *et al*. (2005)	Turbo	Eurecom, resynchr, EGM, AG
Dugelay *et al*. (2006)	Turbo	Eurecom, OFR, image, AG, Stirmark
Liu and Lin (2006)	BCH	Domain *Ceptstrum*, audio, asynchronous attacks
Verma *et al*. (2006)	CC, DV	Blue channel, image color
Chan and Chang (2007)	Hamming	Malicious image changes, codes, detection and reconstruction
Chemak *et al*. (2007)	Turbo	SS, multiresolution, medical, compression, noises, filters
Schaathun (2008)	RS, DL	*Fingerprinting*, CG, *cut-and-paste*
Abdul *et al*. (2013)	Hybrid, RS, DL	QV, DWT, color, CG, JPEG, filters
Al Maeeni *et al*. (2015)	Turbo, BCH product	Logo, concatenation codes, JPEG, *cropping*, median filter

Table 3.2. *Summary table of different contributions in watermarking and correction codes*

Sigles	Notations
RM	Reed-Muller
H	*Hamming codes*
T	*Turbo codes*
CConv	Convolutional codes
DI	Iterative decoding
DV	Viterbi Decoding (Algorithm)
CMP	Code comparisons
QV	Vector quantization
SS	Spread Spectrum
CG	Gaussian channel
CS	*Costa Scheme*
SW	Substitution Watermarking (Burgett *et al*. 1998)
AR	Redundancy analysis
CBS	Binary symmetric channel
AG	Geometric attacks
REP	Repetition codes
OFR	*Optical Flow Regulation*
DL	List decoding
EGM	*Elastic Graph Matching* (Lades *et al*. 1993)

Table 3.3. *Table of acronyms in Table 3.2*

3.7. Illustration of the influence of the type of code, according to the attacks

In this section, we simply illustrate the influence of a correction code on the robustness of a watermarking algorithm, as well as the differences that can exist depending on the code. For this, we suggest using the standard burying strategy presented at the beginning of the chapter; in other words, by index modulation (QIM) that we adapt to the color. The burial takes place in the wavelet domain and we adjust the insertion force parameter so that invisibility is ensured. The size of the signature is fixed at 64 bits. Finally, the attacks tested are JPEG compression, white noise, and saturation modification. We do not increase the illustration of attacks because the results remain consistent. Note that desynchronizing attacks (such as translation or

rotation for example) are not described, since the robustness against this type of attack does not result primarily from the codes, but from the burying algorithm itself.

To analyze different codes, we watermark different images of the Kodack database, with a repetition of the test with different keys for the signature. The results curves present the performance of the different codes against the different attacks, by indicating the average of the measurements of the BER. In all of the figures, the x-axis represents the attack parameter, and the y-axis represents the average BER.

3.7.1. *JPEG compression*

The first attack studied is JPEG compression. Due to the compression, we get an attacked image that loses some of its high-frequency content. The insertion is done at a scale associated with a "medium" frequency band. As a result, at standard compression rates, the error randomly affects the content of the mark (depending on the content of the image). We test the compression with a quality parameter from 1 to 96 in steps of 5. On images of size 512 $\times$ 512, the results are presented in Figure 3.11. It should be noted that for a quality factor lower than 50, the image is degraded.

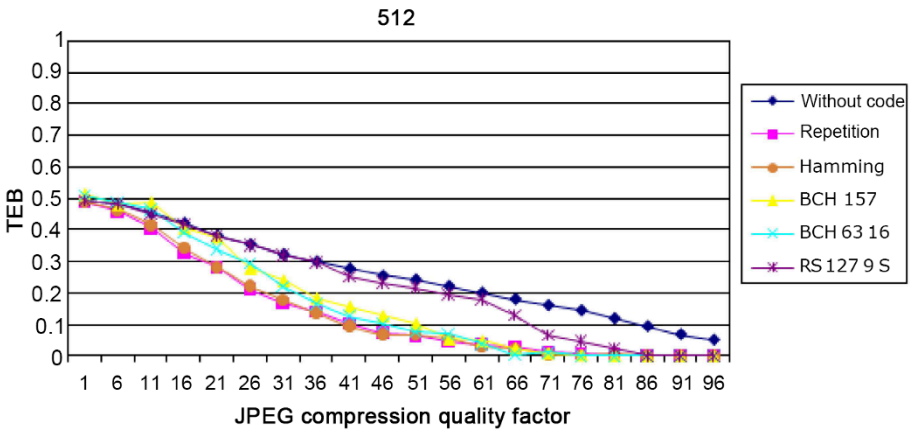


Figure 3.11. *Comparison of the robustness of different codes against the JPEG attack*

We note that from this quality level, all the codes have difficulties protecting the signature: the degradation is too strong and therefore the watermark channel is too noisy. Unsurprisingly, the absence of code is associated with the worst result. We can see that the codes intended to correct errors by erasure (error by blocking) are not suitable for this type of error or attack (RS code), unlike codes fighting against random errors.

The results will depend on the content of the image: the compression will adapt to this content and the errors will be greater when the images contain little information, for example textures, because the mark is the detail directly removed during the compression.

To illustrate this, we show the measurements for five different images with RS encoding. We apply a JPEG compression with a quality factor equal to 40. We can see Figure 3.12 that the results are very different depending on the content of the image. The frequential content of the image in Figure 3.13 is generally rich in high frequencies, while the image in Figure 3.14 is low in detail. We find that for the second image, it is more difficult to reconstruct the signature because of the content of the host image.

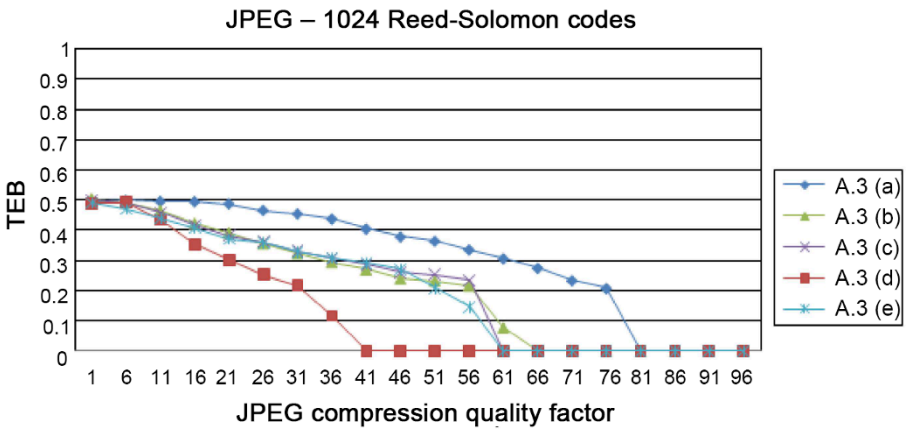


Figure 3.12. *Comparison for different images of the Kodack database of robustness with an RQ encoding and an attack JPEG Quality = 40*

This first simple experiment first of all shows the improvement, in terms of robustness, provided by the addition of a code. It also illustrates the importance of understanding and characterizing the attack in order to deduce therefrom the type of error that is associated, and therefore the most suitable correction code.

However, as we have seen, this result also depends on the element that we do not know *at first glance*, that is to say the content of the image.

Figure 3.13. *Image of a bear*

Figure 3.14. *Image of a plant*

3.7.2. *Additive Gaussian noise*

We now propose the study of the attack by adding Gaussian white noise. In this case, the parameter is the noise level or, more precisely, the signal-to-noise level. Since by definition, the associated error is random, we see in Figure 3.15 that the codes designed for errors by area are again, not suitable. It is evident that the larger the image, the more efficient the external repetition coding and, therefore, the decoded signature displays little or no error in the end.

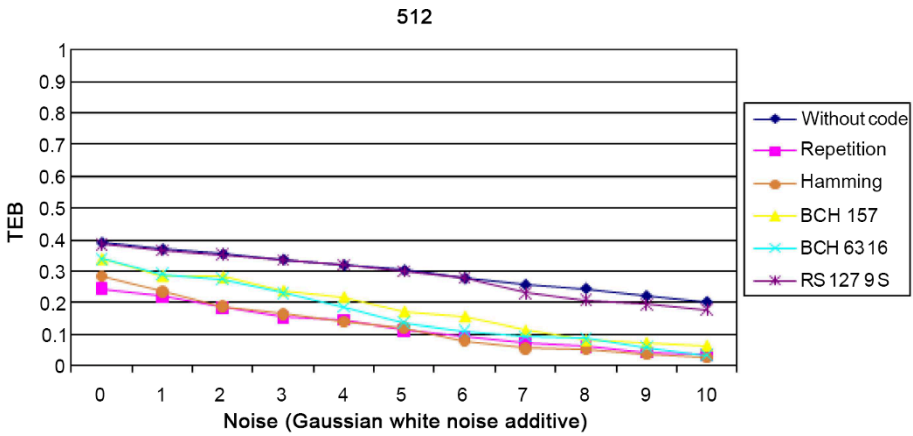


Figure 3.15. *Comparison of the different encodings against an attack by adding Gaussian white noise*

3.7.3. *Saturation*

Finally, we intend to study the impact of another type of attack: the modification of saturation. Saturation characterizes the colorful appearance of a pixel; in other words, the distance between the "color" pixel and the gray level axis. The modification of saturation will therefore involve bringing each pixel closer, or further from the gray axis. This is why the attack parameter takes both negative values (approximation of the gray axis) and positive values.

In order to fully understand the impact of this attack, we illustrate the result on the Lena image using an extreme setting, at –100 and +100. In the case of –100, the image becomes gray since each pixel is projected on the gray axis (Figure 3.16a). In the opposite direction, with a positive parameter at +100, there is an overflow phenomenon: the value of certain pixels comes out of the RGB cube, and therefore a phenomenon of "saturation" of the coordinates occurs on the different channels and they are suppressed to 255 (Figure 3.16b).

Figure 3.16. *Effects of changes to saturation on the Lena image. a) Minimal change to saturation: −100, b) maximal change to saturation: 100*

On reading the results of Figure 3.17, we see that the attack through weakening the saturation does not pose any problem, except when there is a projection on the gray axis, because in this case the image is no longer in color; however, the watermark used modifies the color vectors. We can see that when the saturation increases, above a factor of 30, apart from the RS code, the codes find it hard to reconstruct the signature. This is simply explained by the fact that the overflow phenomena will cause errors by area and therefore, this type of error will be better corrected by RS encoding.

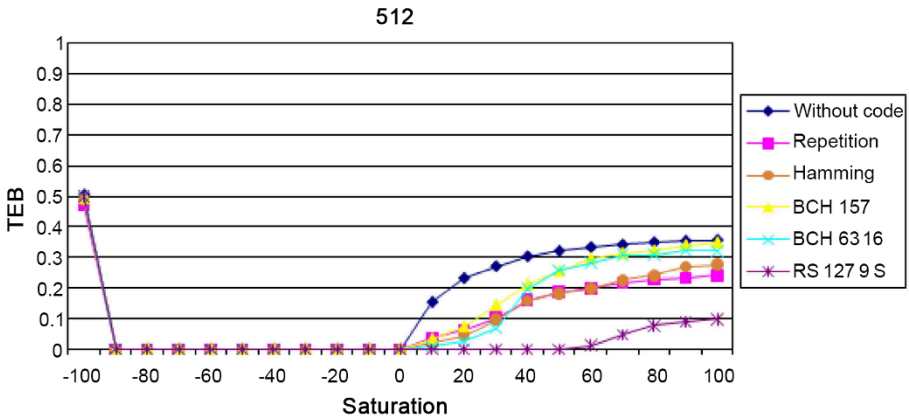


Figure 3.17. *Comparison of coding processes faced with saturation for an image of size 512 × 512*

In this section, we have seen that, faced with certain image modifications, the error produced can have a particular error structure. Taking the latter information into account, a suitable choice of correction codes makes it possible to optimize the detection performance. For example, when considering BCH and RS codes, BCH codes are more efficient than RS codes when the error is random. When the error has a per-packet structure, the RS codes are better. To continue this discussion between the right choice of code and the type of attack, we intend to introduce a particular code in section 3.8.

3.8. Using the rank metric

A useful tool for increasing the robustness of a mark is using error-correction codes. They make it possible to correct the errors produced by a given attack. Depending on the attack and the structure of the resulting errors, the type of code used is more or less effective.

Hamming distance-based codes have been around for over 40 years and have been the subject of many studies (MacWilliams and Sloane 1977) and applications in various fields, such as watermarking (Abdul *et al*. 2013). For some attacks, it happens that the errors are per packet, as we have seen. In this case, it is better to use more structured codes, that is, codes defined on a larger alphabet (e.g. $GF(2^m)$), such as RS codes where it is possible to decode errors by packet. The errors are then no longer decoded independently on each bit, but on packets of m bits so that several errors in the same binary packet only count for a single error (a symbol error) in a code word.

In this section, we consider the use of a new metric called *rank metric*. These codes are already widely used in telecommunications for coding networks (Silva and Kschischang 2009) and in cryptography (Gabidulin *et al*. 1991; Gaborit *et al*. 2014). They are able to correct errors with a specific structure. Consider a code on $GF(2^m)$ of size m. Each coordinate of a code word on $GF(2^m)$ is encoded on m bits and since the length of the code is m, each code word can be seen as a matrix of size $m \times m$.

As the metric used is the rank of a binary matrix, the decoding condition is shown with this metric; in other words, the code words received, whose errors have a low rank, can be corrected accurately. For example, an attack which reverses all the bits of a mark (in other words of a code word) cannot be corrected by a Hamming code. On the other hand, with rank metric codes, the error will be of rank 1 because the error is a matrix only made of binary symbols 1, and so, the code word that has been transmitted is found with ease.

We intend to introduce this type of correction in a watermarking process. Rank metric codes will first be defined, then a watermarking method combining the Lattice

QIM method and the rank metric codes will be described. Thanks to a block decomposition of the image, we show how the proposed method can be resistant to image cropping.

3.8.1. *Rank metric correcting codes*

First, we propose to define and analyze rank metric codes.

3.8.1.1. *Definitions and properties*

Consider linear code $\mathcal{C}$ of length n, defined in an alphabet $\mathcal{A} = GF(q^m)$. The code words $\mathcal{C}$ are line vectors of the vector space $GF(q^m)^n$. Each component of the code word of $\mathcal{C}$ can be expressed as a vector belonging to $GF(q)^m$. It is possible to write these components in the form of a column vector and therefore represent a code word by a matrix defined on $GF(q)_{m\times n}$.

Considering a basis $\mathcal{B}$ from $GF(q^m)$ over $GF(q)$ and a code word $x = (x_1, ..., x_n) \in GF(q^m)^n$, we get a matrix representation of x, denoted as $Mat(x) = (x_{ij})_{i,j}$, or X when there is no ambiguity defined, such that:

$$X = \begin{pmatrix} x_{11} & \dots & x_{1n} \\ \vdots & & \vdots \\ x_{m1} & \dots & x_{mn} \end{pmatrix}$$

as for everything $1 \leq j \leq n$:

$$x_j = \sum_{i=1}^{m} x_{ij}\beta_i$$

It is because of this matrix representation that it is possible to define a new metric on $GF(q^m)^n$ by using the rank of a matrix.

3.8.1.2. *Rank distance*

Let $x = (x_1, \dots, x_n) \in GF(q^m)^n$. Rank x, denoted as $w_R(x)$, equals:

$$w_R(x) = rk(X) \qquad [3.8]$$

Let $y = (y_1, \dots, y_n) \in GF(q^m)^n$. The rank distance of x to y, denoted as $d_R(x, y)$, equals:

$$d_R(x) = rk(X - Y) \qquad [3.9]$$

Of course, d_R does have the properties of a distance. Compared to Hamming distance, we have the following property:

$$w_R(x) \leq w_h(x) \quad [3.10]$$

with x as a code word, and w_h as the Hamming weight function.

We can also deduce that:

$$d_R(x) \leq d_h(x) \quad [3.11]$$

Briefly, this property is real because the number of linearly independent rows or columns (i.e. the rank) of a code word is always smaller than the number of different symbols in twos (Hamming weights). In other terms, since the rank of a vector is independent to the base used, the rank metric is less precise than the Hamming metric, because two vectors with the same weight can have the same rank.

3.8.1.3. *Principle of rank metric codes*

Delsarte was the first to study the rank metric (Delsarte 1978). Many properties of Hamming codes have adapted to rank metric codes. Linear code $\mathcal{C}$ defined on a finite field $GF(q^m)$ can be seen as a subspace of $GF(q^m)^n$, but also as a metric space that has rank distance.

In addition, as $\mathcal{C}$ is a linear code, $\mathcal{C}$ is a subspace of $GF(q^m)^n$. The linearity of a code is an interesting property since it makes it possible to manipulate code words more easily. As for Hamming codes, it is possible to define the minimal distance d_{min} from a rank metric code, such that:

$$d_{min} = \min_{x \neq y \in \mathcal{C}} d_R(x, y) \quad [3.12]$$

with the linearity property of the code, we have:

$$d_{min} = \min_{x \in \mathcal{C}^*} w_R(x). \quad [3.13]$$

We note a linear rank metric code $\mathcal{C}$ is of length n, dimension k and minimal distance d by its parameters $[n, k, d]_r$ or $[n, k, (d-1)/2]_r$ or even $[n, k]_r$, if it is not necessary to specify the minimum distance.

3.8.1.4. *Decoding Gabidulin codes*

The decoding bounds of rank metric codes (Singleton and Gilber–Varshanov bounds) are similar to those of Hamming codes. They are very useful for building decoding algorithms. Unlike classical Hamming codes, there is only a small number of families of codes for which a decoding algorithm is known.

Gabidulin codes are one of these families and have as parameters $[n, k, n-k+1]_r$ on $GF(q^n)$, with n being the length of the code, k is the size of the code and $d = n-k+1$ is the minimal distance (Gabidulin 1985). These codes are called *Maximum Rank Distance* (MRD), and so can correct up to $(n-k)/2$ errors. They can be seen as a rank metric family, similar to the famous RS codes (which are MDS). Since 1985, many decoding algorithms have been proposed (Gabidulin 1992; Loidreau 2006).

On closer inspection, the RS code decoding algorithm can be adapted for Gabidulin codes. By using a variant of the Welch–Berlekamp algorithm on linear polynomials, it is possible to obtain a quadratic complexity decoding.

3.8.1.5. *Introduction to rank metric in a watermarking strategy*

In practice, we use these codes in an extension $GF(q^m)$ of $GF(2)$ and we can unite a binary vector of size m to each coordinate of a code word in such a way that code word c can be represented by a binary matrix of size $m \times m$.

The integrated code word c is modified by an error e that can also be seen as a binary matrix of size $m \times m$.

To assess the case in which the rank metric is more effective than the classical Hamming metric, we compare their behaviors by observing some examples of errors. Let $y = c + e$ be a code word received after the extraction of a mark.

EXAMPLE.– $m = 4$ and a code word c:

$$c = \begin{pmatrix} 1\,0\,1\,0 \\ 1\,1\,0\,1 \\ 1\,0\,0\,1 \\ 1\,1\,0\,1 \end{pmatrix}$$

with y such that:

$$y = \begin{pmatrix} 1\,0\,1\,0 \\ 1\,0\,0\,0 \\ 1\,0\,0\,1 \\ 1\,1\,1\,0 \end{pmatrix}$$

We have the following error:

$$e = \begin{pmatrix} 0\,0\,0\,0 \\ 0\,1\,0\,1 \\ 0\,0\,0\,0 \\ 0\,0\,1\,1 \end{pmatrix}$$

We observe that the error matrix e is of rank 2. If a rank metric code is able to correct up to 2 errors, then the word y is uniquely decoded in c. With a Hamming code of length 16, the error is of weight equal to 4. In this case, it is possible to find efficient codes with the two metrics for a reasonable dimension k.

Regardless of y and c, if we choose an error e such that:

$$e = \begin{pmatrix} 1\,1\,0\,1 \\ 1\,1\,0\,1 \\ 1\,0\,0\,0 \\ 1\,0\,1\,1 \end{pmatrix}$$

the Hamming weight of e is 9, while the rank of e is 4: so it is impossible to find efficient codes with the two metrics.

If e is the identity matrix, the error is of full rank and only Hamming codes are useful.

From this example, we see that the rank metric codes become more interesting when the error has a particular structure. For example:

$$e = \begin{pmatrix} 1\,1\,1\,0 \\ 1\,1\,1\,0 \\ 1\,1\,1\,0 \\ 0\,0\,0\,0 \end{pmatrix}$$

with the Hamming metric, we have nine errors in 16 transmitted bits and there is no code capable of decoding correctly, whereas with rank metric, e is only of rank 1. Therefore, we can easily decode such an error (e.g. with a Gabidulin code of parameters $[4, 2, 3]$).

In the same way that some attacks will be "linked" to RS or BCH codes, this type of error is found in the context of specific attacks. Rank metric codes will, in this case, be an alternative application to digital watermarking which is more efficient than classical Hamming codes.

We now propose a watermarking method integrating rank metric codes in order to protect against image cropping.

3.8.2. *Code by rank metric: a robust watermarking method for image cropping*

Image cropping erases an area of an image, which seriously damages the image. The information that has been inserted locally is therefore destroyed. One of the oldest contributions on image cropping was proposed by Swanson *et al.* (1996a, 1996b). Their method of watermarking uses a modulation algorithm of the bits of low weights on the DCT coefficients to insert their mark. Image cropping can also be associated with a similar attack called *collage attack*, which involves replacing an area of an image by another. This attack is also known as a variation of the image forgery attack, which is studied by Holliman and Memon (2000) in the context of the authentification of a digital image, studied by Fridrich and Goljan (2001). This type of attack is mainly studied in issues of image authentication or detection of modifications (malicious or not).

Later on, other works (Fridrich and Goljan 2001) proposed a mechanism for inserting an image into itself (or *self-embedding*) to withstand several attacks, such as falsification, cropping (or reframing) or even the replacement of areas in an image. The idea involves inserting a compressed version of a host image into itself using least significant bit modulation on DCT coefficients.

More recent works were published on the same subject (Aggarwal and Singla 2011; Khalid *et al*. 2013; Saneie and Naghsh 2016; Goli and Naghsh 2017). For example, an original approach is inspired by the mathematics of Sudoku (Felgenhauer and Jarvis 2006; Russell and Jarvis 2006). Their insertion strategy is based on the same principle as the self-embedding mechanism (reduction of the size of the host image to obtain a smaller mark), previously described to resolve the problem of image cropping and reframing. In the insertion stage, the host image is divided into N cells, identified by the integers from 1 to N (as in a sudoku grid). A new image is then generated with each cell replaced in a solution grid, the size of which is reduced, then inserted into a host image because of the modulation of the low-order bits on the DCT coefficients. However, these contributions only address problems of image authentication and detection of modifications by fragile watermarking methods. Compared to robust watermarking, the objectives are completely different.

The problem of image cropping can become more complex when the size of the attacked image is smaller than that of the marked image. At the detection stage, the mark must be synchronized before it can be extracted. One of the classic approaches against this problem was proposed by Kutter (1999). To insert a robust mark, this is repeated in several places of the image, as for the coding by repetition studied in this chapter.

To illustrate the possible contribution of codes, we intend to use the distinct features of the rank metric code to fight against this type of attack.

3.8.2.1. *Description of the "rank metric method against cropping"*

It seems rather difficult to design a method resistant to a cropping type attack. Basically, all contributions are based on the use of information repetition, that is, on the use of some form of information redundancy. Therefore, each locally inserted detail must depend on other details inserted in other regions of the image to detect the mark without error. The only solution to reconstruct the original message without error is to distribute the mark throughout the image. If an area is cropped and small enough, then the message can be extracted correctly. In the book, the approaches used are equivalent to using a repeating code.

We propose a different approach to deal with the image cropping problem using rank metric codes. To do this, a host image is broken down in n^2 blocks with n as the length of a rank metric code (decomposition scheme illustrated in Figure 3.18). Each bit of the rank code word is then inserted into a block using $L = 2$ coefficients. We get an image that looks exactly like the matrix of a rank code word. Therefore, the distortions produced by the attacked image are reproduced directly on the error matrix e.

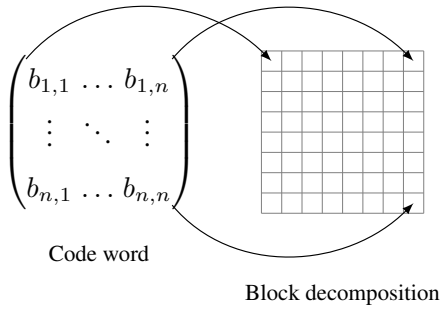


Figure 3.18. *Insertion strategy using rank metric code and image block decomposition. Each bit $b_{i,j}$ is associated with a block*

For example, a square-shaped area (of size l) is cropped. The blocks affected by this attack are directly transposed onto the error matrix e. Now consider a cropped row or column of width l. In the matrix e, a 1-bit line or a column appears. As we have seen previously, e has a specific error structure that is perfectly managed by rank metric codes. In fact, $rk(e) = r'$ with r' is the number of blocks (in width) affected by the image cropping.

The two examples previously described produce errors of the same rank. In the second case, the distortion is maximized compared to the first case (cropped square) for errors of the same rank. In Figure 3.19, we show examples of attacked images, which all have a matrix error of the same rank.

Another interesting example of the rank metric is as follows: swapping two rows or columns of the same width does not change the rank of the error matrix if the coefficients chosen in each block have the same positions and the cut bands overlap the entire deconstruction of the image exactly. This swapping operation can be repeated indefinitely without changing the rank of the error. This case study is not considered in this chapter for lack of realism.

In section 3.8.2.2, we describe and analyze the robustness of the method proposed against image cropping.

3.8.2.2. *Robustness of the rank metric against cropping*

In our study, we distinguish between two types of cropping: the first type groups together images whose vertical or horizontal bands are entirely cut (Figure 3.20a) and the second type groups together images where only rectangular shapes are cut

(Figure 3.20b). Also note that the cut bands do not necessarily overlap the block decomposition of the image. For errors of the same rank, the distortion is maximized with the first type compared to the second.

Figure 3.19. *Cropped images with errors from the same rank*

In our experimental measurements, we consider the first type of cropping for convenience and clarity. With the second type, the average rank of the error is exactly the same with a lower cropping percentage.

We measure distortions of images using the percentage of cropped image region denoted by cr. For the first type, we define cr such that:

$$cr = 100.\frac{l}{h} \qquad [3.14]$$

where l is the number of pixels, in width, of the cut column and h is the height of the image.

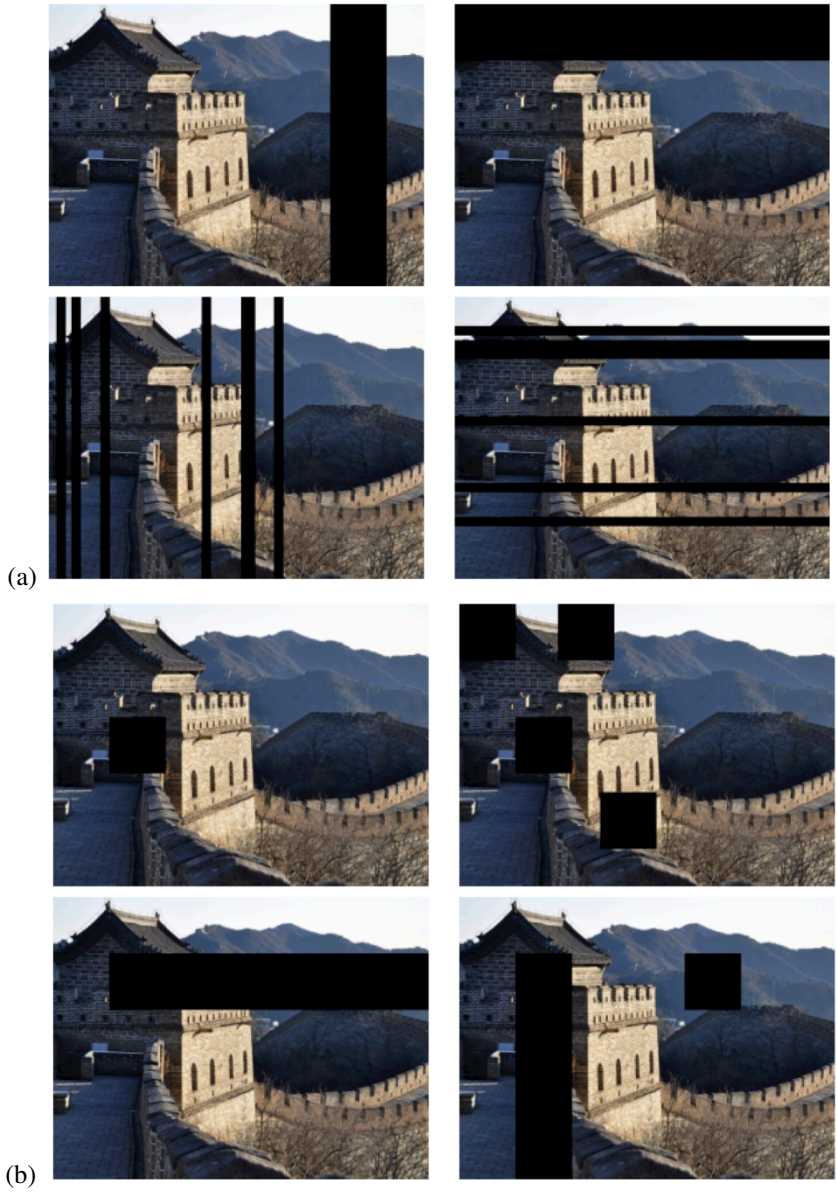


Figure 3.20. *Types of image cropping. The top two rows (a) represent type 1 errors and the two bottom rows (b) represent type 2 errors*

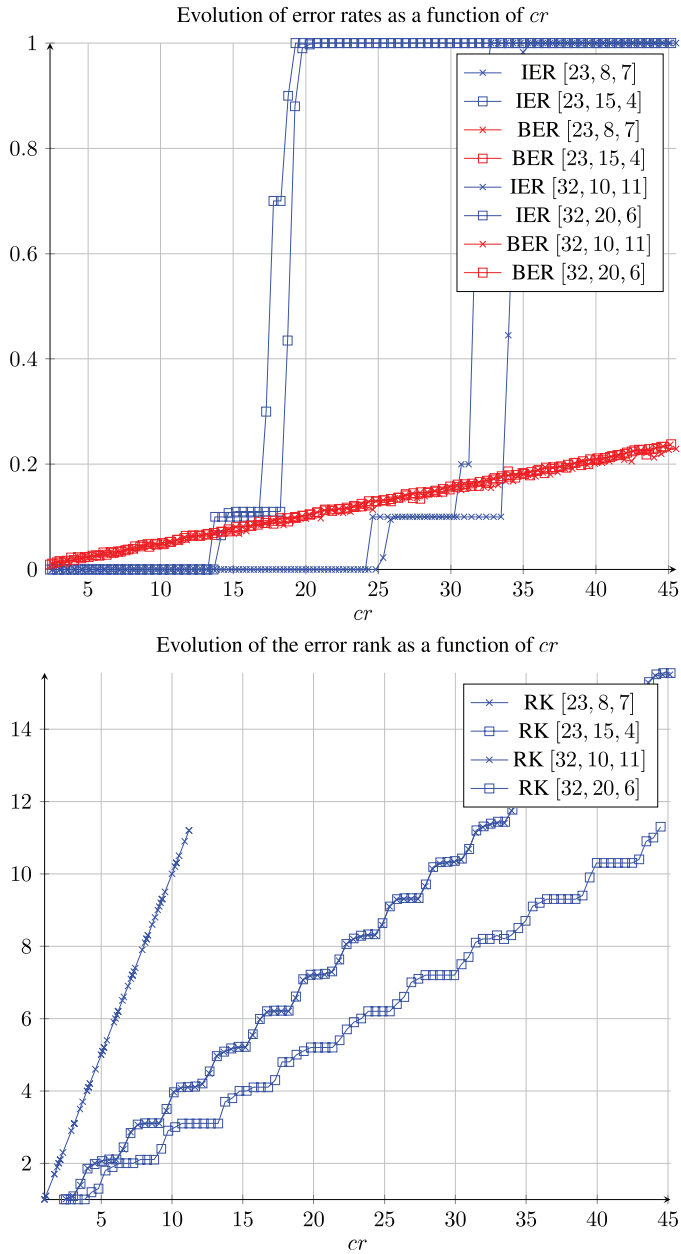


Figure 3.21. *Average error rate and rank as a function of the cropping percentage cr*

We have randomly chosen 1,000 images from the Corel image database whose size is 300×400 or 400×300 and calculated the respective averages of the BERs, the image error rates and error ranks. We have chosen different Gabidulin codes to assess the robustness of the proposed method against image cropping. In Figure 3.21, we see that the binary error rate and error rank curves are linearly increasing as cr increases for all values of n. When the average error rank ($\pm$ 1.6) becomes greater than the maximum number of errors, we can see that the IER curves vary quickly from 0 to 1. Therefore, it becomes impossible to detect the original message. We then denote by cr_{max} the highest percentage allowing an error-free detection.

This first observation shows, through experimentation, the applicability of rank metric codes in a digital watermarking strategy against image cropping. Of course, sometimes the detection performance is very poor. The main idea of the proposed method is to take advantage of the mathematical properties of the rank of a matrix. It is also possible to produce an example of a marked, and then cut image where detection is impossible (see Figure 3.22).

Figure 3.22. *Examples of attacked images with the poorest detection performance. The error rank is large ($\geq (n-k)/2$) compared with the percentage of cropped area (cr')*

Discussing the parameters in more detail, when the encoding rate k/n decreases, the correction power t increases, which results in a higher value of cr_{max}; in other words, the mark is more robust. Likewise, when n increases with fixed k, the encoding rate decreases and the correction power also increases, which also makes it possible to obtain a more robust mark.

However, the image quality should be carefully evaluated when inserting the mark because if n increases, it significantly degrades the host image. In fact, a greater value of n implies the quantization of $n^2 \times L$ coefficients (the blocks are smaller).

Compared to BCH codes, rank metric codes are more efficient when we are faced with the first type of image cropping. We can see that the values of cr_{max} reached by the rank metric codes correspond to higher BERs than the correction rates given by equivalent BCH codes. For example, with $(n, k) = (23, 8)$ and cr_{max} = 37%, we have a BER $= 0.12 > t'/n' = 0.09$ (see Table 3.4 for other examples).

Gabidulin			BCH	
$[n, k, t]$	cr_{max}	BER to cr_{max}	$[n', k', t']$	t'/n'
16, 5, 5	41 %	0,11	255, 87, 26	0,10
16, 8, 4	42 %	0,1	255, 131, 18	0,07
16, 11, 2	41 %	0,04	255, 171, 11	0,04
23, 8, 7	37 %	0,12	511, 175, 46	0,09
23, 12, 5	36 %	0,10	511, 250, 31	0,06
23, 15, 4	39 %	0,07	511, 340, 20	0,04
32, 10, 11	35 %	0,13	1023, 348, 87	0,09
32, 16, 8	35 %	0,10	1023, 513, 57	0,06
32, 21, 5	35 %	0,07	1023, 688, 36	0,04

Table 3.4. *Table of code correction parameters. For each row of the table, we have parameters of a rank metric code on the left and on the right are the parameters of an equivalent BCH code* $(t/n \simeq t'/n')$

However, rank metric codes are less effective than BCH codes when we are faced with the second type of image cropping. This is explained by the fact that the BER obtained at the value cr_{max} is smaller than the correction rate t'/n', associated with the equivalent BCH code. In addition, BCH codes can correct random errors and, therefore, are more robust against all types of image cropping if the BER is smaller than t'/n'. However, in this particular application, we can consider that the rank metric codes remain a better choice of correction codes, especially in the case where the Gabidulin codes are MRD. Even though BCH codes are the best for random errors, we are constrained by the choice of parameters. In fact, the length of a BCH code is an integer of the form $n = 2^m - 1$.

Combined with the LQIM method and a block decomposition strategy in the spatial domain, we have shown that the proposed method can be robust to different types of image cropping by taking into account certain constraints. Additionally, these newly-introduced codes in digital watermarking make it possible to obtain better detection performance in certain cases (first type of image cropping). However, an important problem remains associated with this type of image modification: the

synchronization of the mark when the size of the attacked image is different to that of the marked image. The aim of this section is to illustrate how a reflection on the nature of the error can lead to the use of more "specialized" codes.

3.9. Conclusion

In this chapter, in order to introduce the contribution of correction codes, we have chosen to explain the paradigm of robust watermarking, which aims to insert an invisible and robust mark to image modifications.

We then offered an introduction to correction codes by repetition, BCH and RS codes and recalled how they could be integrated very simply into robust watermarking. Through a simple example with a burying strategy by index modulation, we have shown how the use of codes improves robustness against attacks whose errors have a specific structure. We focused on the fact that the study of the error structure will guide us on the use of similar encoding. Simple experiments have illustrated this, which seems to be a line of thought that should probably be followed.

Finally, following this idea, we presented a type of error correction code, rarely in watermarking, using the rank metric instead of Hamming metric. Technically, Gabidulin codes were combined with the LQIM method to obtain a new type of resistance against specific attacks. For this, it was necessary to adapt the burying strategy (strategy of block decomposition of the image instead of inserting information at random positions). The image is then divided into blocks and each bit of a code word rank is associated with a block. This insertion strategy allows us to take advantage of the structure of the rank metric against different image cropping configurations. After studying the robustness of the proposed method with rank metric codes against this attack, we also showed that Gabidulin codes are more effective than BCH codes when the distortions are maximized for a fixed error rank. The last section showed how the design of the watermarking algorithm can, and should, take into account both the nature of the errors, the properties of the code and the burying process.

3.10. References

Abdul, W., Carré, P., Gaborit, P. (2013). Error correcting codes for robust color wavelet watermarking. *EURASIP Journal on Information Security*, 1, 1.

Aggarwal, A. and Singla, M. (2011). Robust watermarking of color images under noise and cropping attacks in spatial domain. *Image*, 6(9), 11.

Al-Askary, O. (2003). Iterative decoding of product codes. PhD Thesis, Royal Institute of Technology, Stockholm.

Al Maeeni, S., Kalbat, F., Al-Ahmad, H. (2015). Logo embedding in grayscale images using turbo product codes. *Proceedings of the International Conference on Information and Communication Technology Research (ICTRC)*. IEEE, Abu Dhabi, 96–99.

Baldo, F., González, F.P., Scalise, S. (2001). Turbo coding for sample-level watermarking in the DCT domain. In *Proceedings of the International Conference on Image Processing*. IEEE, 1003–1006.

Barni, M., Bartolini, F., Furon, T. (2003). A general framework for robust watermarking security. *Signal Processing*, 83(10), 2069–2084.

Bas, P., Le Bihan, N., Chassery, J.-M. (2003). Color image watermarking using quaternion Fourier transform. *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*. IEEE, Hong Kong, III–521.

Baudry, S., Delaigle, J.-F., Sankur, B., Macq, B., Maitre, H. (2001). Analyses of error correction strategies for typical communication channels in watermarking. *Signal Processing*, 81(6), 1239–1250.

Berrou, C., Glavieux, A., Thitimajshima, P. (1993). Near Shannon limit error-correcting coding and decoding: Turbo-codes 1. *Proceedings of the International Conference on Communications*. IEEE, Geneva, 1064–1070.

Bose, R.C. and Ray-Chaudhuri, D.K. (1960). On a class of error correcting binary group codes. *Information and Control*, 3(1), 68–79.

Bravo-Solorio, S., Calderon, F., Li, C.-T., Nandi, A.K. (2018). Fast fragile watermark embedding and iterative mechanism with high self-restoration performance. *Digital Signal Processing*, 73, 83–92.

Burgett, S., Koch, E., Zhao, J. (1998). Copyright labeling of digitized image data. *IEEE Communications Magazine*, 36(3), 94–100.

Chan, C.-S. and Chang, C.-C. (2007). An efficient image authentication method based on Hamming code. *Pattern Recognition*, 40(2), 681–690.

Chan, P.-W. and Lyu, M.R. (2003). A DWT-based digital video watermarking scheme with error correcting code. *Proceedings of the International Conference on Information and Communications Security*. ICICS, Hohhot, 202–213.

Chan, P.W., Lyu, M.R., Chin, R.T. (2005). A novel scheme for hybrid digital video watermarking approach, evaluation and experimentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(12), 1638–1649.

Chang, C.-C., Chen, K.-N., Lee, C.-F., Liu, L.-J. (2011). A secure fragile watermarking scheme based on chaos-and-hamming code. *Journal of Systems and Software*, 84(9), 1462–1470.

Chemak, C., Lapayre, J.C., Bouhlel, M.S. (2007). A new scheme of image watermarking based on 5/3 wavelet decomposition and turbo-code. *WSEAS Transactions on Biology and Biomedicine*, 4(4), 45–52.

Chen, B. and Wornell, G.W. (1999a). Dither modulation: A new approach to digital watermarking and information embedding. *Proceedings of SPIE: Security and Watermarking of Multimedia Contents*, San Jose, 3657, 342–353.

Chen, B. and Wornell, G.W. (1999b). Quantization index modulation: A class of provably good methods for digital watermarking and information embedding. *IEEE Transactions on Information Theory*, 47(4), 1423–1443.

Cox, I.J., Doërr, G., Furon, T. (2006). Watermarking is not cryptography. In *Digital Watermarking*, Shi, Y.Q., Jeon, B. (eds). Springer, Berlin/Heidelberg.

Cvejic, N., Tujkovic, D., Seppanen, T. (2003). Increasing robustness of an audio watermark using turbo codes. *Proceedings of the International Conference on Multimedia and Expo*. IEEE, Baltimore, I–217.

Darbon, J., Sankur, B., Maitre, H. (2001). Error correcting code performance for watermark protection. *SPIE Security and Watermarking of Multimedia Contents III*, 4314, 663–673.

Delsarte, P. (1978). Bilinear forms over a finite field, with applications to coding theory. *Journal of Combinatorial Theory, Series A*, 25(3), 226–241.

Doërr, G., Rey, C., Dugelay, J.-L. (2005). Watermark resynchronization based on elastic graph matching. *Proceedings of the International Conference: Sciences of Electronic, Technologies of Information and Telecommunications*. SITET, Sousse.

Dugelay, J.-L., Roche, S., Rey, C., Doërr, G. (2006). Still-image watermarking robust to local geometric distortions. *IEEE Transactions on Image Processing*, 15(9), 2831–2842.

Eggers, J.J., Su, J., Girod, B. (2000a). A blind watermarking scheme based on structured codebooks. *Proceedings of the IEE Seminar on Secure Images and Image Authentication*. IEEE, London.

Eggers, J.J., Su, J.K., Girod, B. (2000b). Robustness of a blind image watermarking scheme. *Proceedings of the International Conference on Image Processing*. IEEE, Thessaloniki, 17–20.

Felgenhauer, B. and Jarvis, F. (2006). Mathematics of Sudoku I. *Mathematical Spectrum*, 39(1), 15–22.

Fridrich, J. and Goljan, M. (2001). Protection of digital images using self embedding. *Proceedings of the Symposium on Content Security and Data Hiding in Digital Media*. Newark, 1–6.

Gabidulin, E.M. (1985). Theory of codes with maximum rank distance. *Problemy Peredachi Informatsii*, 21(1), 3–16.

Gabidulin, E.M. (1992). *A Fast Matrix Decoding Algorithm for Rank-Error-Correcting Codes*. Springer, Berlin/Heidelberg.

Gabidulin, E.M., Paramonov, A., Tretjakov, O. (1991). Ideals over a non-commutative ring and their application in cryptology. *Workshop on the Theory and Application of Cryptographic Techniques*. EUROCRYPT, Brighton, 482–489.

Gaborit, P., Ruatta, O., Schrek, J., Zémor, G. (2014). New results for rank-based cryptography. *Proceedings of the International Conference on Cryptology in Africa*. AFRICACRYPT, Marrakech, 1–12.

Goli, M.S. and Naghsh, A. (2017). Introducing a new method robust against crop attack in digital image watermarking using two-step Sudoku. *Proceedings of the International Conference on Pattern Recognition and Image Analysis (IPRIA)*. IEEE, Faro, 237–242.

Guruswami, V. and Sudan, M. (1998). Improved decoding of reed-solomon and algebraic-geometric codes. *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*. IEEE, Palo Alto.

Hamming, R.W. (1950). Error detecting and error correcting codes. *Bell Labs Technical Journal*, 29(2), 147–160.

He, D., Sun, Q., Tian, Q. (2003). A semi-fragile object based video authentication system. *Proceedings of the 2003 International Symposium on Circuits and Systems*. IEEE, Bangkok, III–III.

Hernandez, J.R., Delaigle, J.-F., Macq, B.M. (2000). Improving data hiding by using convolutional codes and soft-decision decoding. *Proceedings of SPIE: Security and Watermarking of Multimedia Contents II*, 3971, 24–48.

Ho, C.K. and Li, C.-T. (2004). Semi-fragile watermarking scheme for authentication of jpeg images. *Proceedings of the International Conference on Information Technology: Coding and Computing*. IEEE, Las Vegas, 7–11.

Holliman, M. and Memon, N. (2000). Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes. *IEEE Transactions on Image Processing*, 9(3), 432–441.

Hsieh, C.-T. and Wu, Y.-K. (2001). Digital image multiresolution watermark based on human visual system using error correcting code. *Journal of Applied Science and Engineering*, 4(3), 201–209.

Justesen, J. and Høholdt, T. (2004). *A Course in Error-correcting Codes*, Volume 1. European Mathematical Society, Helsinki.

Kesal, M., Mihcak, M.K., Koetter, R., Moulin, P. (2000). Iteratively decodable codes for watermarking applications. *Proceedings of the 2nd International Symposium on Turbo Codes and Related Topics*. ISTC, Brest.

Khalid, S.K.A., Deris, M.M., Mohamad, K.M. (2013). Anti-cropping digital image watermarking using Sudoku. *International Journal of Grid and Utility Computing*, 4(2–3), 169–177.

Khan, A., Siddiqa, A., Munib, S., Malik, S.A. (2014). A recent survey of reversible watermarking techniques, *Information Sciences*, 279, 251–272.

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

Kutter, M. (1999). Watermarking resistance to translation, rotation, and scaling. *Proceedings of SPIE: Multimedia Systems Applications*, 3528, 423–432.

Lades, M., Vorbruggen, J.C., Buhmann, J., Lange, J., von der Malsburg, C., Wurtz, R.P., Konen, W. (1993). Distortion invariant object recognition in the dynamic link architecture. *IEEE Transactions on Computers*, (3), 300–311.

Lancini, R., Mapelli, F., Tubaro, S. (2002). A robust video watermarking technique in the spatial domain. *Proceedings of the International Symposium on VIPromCom Video/Image Processing and Multimedia Communications*. IEEE, Zadar, 251–256.

Lee, J. and Won, C.S. (2000). A watermarking sequence using parities of error control coding for image authentication and correction. *IEEE Transactions on Consumer Electronics*, 46(2), 313–317.

Lefèvre, P., Carré, P., Gaborit, P. (2018). Watermarking and rank metric codes. *Proceedings of the 2018 International Conference on Acoustics, Speech and Signal Processing (ICASSP2018)*. IIEEE, Calgary.

Lefèvre, P., Carré, P., Gaborit, P. (2019). Application of rank metric codes in digital image watermarking. *Signal Processing: Image Communication*, 74, 119–128.

Li, Q. and Cox, I.J. (2007). Improved spread transform dither modulation using a perceptual model: Robustness to amplitude scaling and JPEG compression. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*. IEEE, Honolulu, II–185.

Liu, S.-C. and Lin, S.D. (2006). BCH code-based robust audio watermarking in the cepstrum domain. *Journal of Information Science and Engineering*, 22(3), 535–543.

Loidreau, P. (2006). *A Welch–Berlekamp Like Algorithm for Decoding Gabidulin Codes*. Springer, Berlin/Heidelberg.

Lu, H., Shen, R., Chung, F.-L. (2003). Fragile watermarking scheme for image authentication. *Electronics Letters*, 39(12), 898–900.

MacWilliams, F.J. and Sloane, N.J.A. (1977). *The Theory of Error-correcting Codes*. Elsevier, Amsterdam.

Maeno, K., Sun, Q., Chang, S.-F., Suto, M. (2006). New semi-fragile image authentication watermarking techniques using random bias and nonuniform quantization. *IEEE Transactions on Multimedia*, 8(1), 32–45.

Moulin, P. and Koetter, R. (2005). Data-hiding codes. *Proceedings of the IEEE*, 93(12), 2083–2126.

Oostveen, J.C., Kalker, T., Staring, M. (2004). Adaptive quantization watermarking. *Proceedings of SPIE: Security, Steganography, and Watermarking of Multimedia Contents VI*, 5306, 296–304.

Pérez-González, F., Mosquera, C., Barni, M., Abrardo, A. (2005). Rational dither modulation: A high-rate data-hiding method invariant to gain attacks. *IEEE Transactions on Signal Processing*, 53(10), 3960–3975.

Reed, I.S. and Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2), 300–304.

Rey, C., Amis, K., Dugelay, J.-L., Pyndiah, R., Picart, A. (2003). Enhanced robustness in image watermarking using turbo codes. *Proceedings of SPIE: Security and Watermarking of Multimedia Contents V*, 5020, 330–337.

Russell, E. and Jarvis, F. (2006). Mathematics of Sudoku II. *Mathematical Spectrum*, 39(2), 54–58.

Saneie, S.H. and Naghsh, A. (2016). Introducing a new method of robust digital image watermarking against cropping and salt and pepper noise using sudoku. *Majlesi Journal of Multimedia Processing*, 4(4), 1–4.

Schaathun, H.G. (2008). On error-correcting fingerprinting codes for use with watermarking. *Multimedia Systems*, 13(5–6), 331–344.

Shehab, A., Elhoseny, M., Muhammad, K., Sangaiah, A.K., Yang, P., Huang, H., Hou, G. (2018). Secure and robust fragile watermarking scheme for medical images. *IEEE Access*, 6, 10269–10278.

Silva, D. and Kschischang, F.R. (2009). On metrics for error correction in network coding. *IEEE Transactions on Information Theory*, 55(12), 5479–5490.

Sudan, M. (1997). Decoding of Reed Solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1), 180–193.

Sun, Q. and Chang, S.-F. (2002). Semi-fragile image authentication using generic wavelet domain features and ECC. *Proceedings of the International Conference on Image Processing*. IEEE, Rochester, II–II.

Swanson, M.D., Zhu, B., Tewfik, A.H. (1996a). Robust data hiding for images. *Proceedings of the Digital Signal Processing Workshop*. IEEE, Loen, 37–40.

Swanson, M.D., Zhu, B., Tewfik, A.H. (1996b). Transparent robust image watermarking. *Proceedings of the International Conference on Image Processing*. IEEE, Lausanne, 211–214.

Verma, B., Jain, S., Agarwal, D.P., Phadikar, A. (2006). A new color image watermarking scheme. *INFOCOMP*, 5(3), 37–42.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2), 260–269.

Wang, S.-H. and Lin, Y.-P. (2004). Wavelet tree quantization for copyright protection watermarking. *IEEE Transactions on Image Processing*, 13(2), 154–165.

Wolfgang, R.B. and Delp, E.J. (1999). Fragile watermarking using the VW2D watermark. *Security and Watermarking of Multimedia Contents*, 3657, 204–214.

Wong, P.W. and Memon, N.D. (2000). Secret and public key authentication watermarking schemes that resist vector quantization attack. *Proceedings of SPIE: Security and Watermarking of Multimedia Contents II*, 3971, 417–428.

Yeung, M.M. and Mintzer, F. (1997). An invisible watermarking technique for image verification. *Proceedings of the International Conference on Image Processing*. IEEE, Santa Barbara, 680–683.

Zamir, R. and Feder, M. (1994). On lattice quantization noise. *Proceedings of the Data Compression Conference*. IEEE, Snowbird, 380–389.

Zhang, F., Zhang, X., Chen, Z. (2005). Digital image authentication based on error-correction codes. *Proceedings of the International Conference on Computational and Information Science*. ICCS, Atlanta, 433–438.

Zhou, X., Duan, X., Wang, D. (2004). A semifragile watermark scheme for image authentication. *Proceedings of the 10th International Multimedia Modelling Conference*. IEEE, Brisbane, 374–377.

Zinger, S., Jin, Z., Maitre, H., Sankur, B. (2001). Optimization of watermarking performances using error correcting codes and repetition. In *Communications and Multimedia Security Issues of the New Century*, Steinmetz, R., Dittman, J., Steinebach, M. (eds). Springer, Boston.

4

Invisibility

Pascal LEFÈVRE[1], Philippe CARRÉ[1] and David ALLEYSSON[2]

[1] *XLIM, CNRS, University of Poitiers, France*

[2] *LPNC, Grenoble Alpes University, Savoie Mont Blanc University, CNRS, Grenoble, France*

In this chapter, we intend to discuss the different strategies to ensure the invisibility of a message embedded in a color image. The term "invisibility" here relates to the visual aspect. For grayscale images, many solutions have been developed, attempting to take into account the modeling of the human visual system (HVS) in order to limit degradation. However, the extension of these solutions to color images often proposes an adaptation of the strategies dedicated to grayscale images, configured by a wise choice of a color component. Even when a dedicated approach to color is introduced (such as through the use of quaternion formalism), the same challenge is present; in other words, the correct choice of the direction of the insertion color. In this chapter, we discuss these aspects and we conclude by introducing a method configuring this choice of color direction, depending on psycho-visual considerations.

4.1. Introduction

In this chapter, we focus on the watermarking of color images that involves the current framework of modern watermarking algorithms. In this book, different watermarking methods for color images are presented, however, many of them

For a color version of all figures in this chapter, see www.iste.co.uk/puech/multimedia1.zip.

Multimedia Security 1,

coordinated by William PUECH.

Multimedia Security 1: Authentication and Data Hiding,
First Edition. William Puech.

proceed in a similar way: selecting a chromatic or achromatic channel in order to have only one scalar detail to manipulate per pixel, and then a watermarking strategy is applied from the techniques that are defined for grayscale images. Keeping this in mind, the first color solutions (Kutter *et al*. 1997; Yu *et al*. 2001) suggested watermarking the blue component in order to minimize the visual impact in the modified image. In fact, it is well known that modifications of the blue component will be less visible. As is often the case, this stronger invisibility will go hand in hand with a greater fragility. Continuing this strategy, but looking for other compromises, different methods introduced the modification of the luminance component (Voyatzis and Pitas 1998) or the saturation component (Kim *et al*. 2001), which allowed them to be more robust, but generally had greater distortion during insertion. In fact, the modification of a component like luminance leads to visible damage (if not, the insertion force must be very small).

Since then, several propositions for color have been made, but the majority of them use strategies that we consider "grayscale", or scalar methods, as they only use one component. Therefore, the challenge is always to choose this component well. Through the choice of a single component, the risk is that the concept of color is not taken into account, or more precisely, the vector dimension of the information is not used. This generally leads, from the choice of the component, to favoring robustness or invisibility. In addition, poor choice of the component does not make it possible to obtain an optimal invisibility for the human visual system (HVS). In order to take into account this information that is translated by three values, certain methods tend to treat each color component independently of one another and again apply a grayscale method on each component (e.g. Dadkhah *et al*. (2014)). These methods, therefore, treat the color components independently. However it is now well-established that the color components are correlated, and so these marginal approaches do not allow us to obtain an optimal invisibility for the HVS.

This is why, in order to maximize the invisibility/robustness compromise, approaches have appeared that consider color information in a more global way. For example, Abadpour and Kasaei (2008) used the information obtained from the projections of a main component analysis. Other contributions suggest the use of transformed domains that are adapted to color, such as the quaternion discrete cosine transform (Li *et al*. 2018) or specific manipulations (Parisis *et al*. 2004). There are also vector methods on wavelet coefficients that are calculated from the three RGB planes or from another space, which will try to limit the visual degradation by setting the insertion force in the domain of the transform to using, for example, the concept of *Just Noticeable Difference* (JND) (Chou and Liu 2010) that we describe below. In this case, the algorithm does not take into account the perceptual dimension of the color during the decomposition, but in the configuration of the modification. To integrate the concept of HVS, the works proposed by Chareyron *et al*. (2004) allow the insertion of a 2D mark in the chromatic xy plane. Their approach is based on a histogram manipulation offered by Coltuc and Bolon (1999). To improve the

invisibility of the mark, the color distances corresponding to the distortions are calculated in the L*a*b* color space (Chareyron and Trémeau 2006). As we have mentioned, in order to introduce HVS modeling, the concept of JND has been used for a long time. JND *initially* makes it possible to assess the damage linked to a watermarking strategy, but this concept remains an open problem with regard to digital models, in particular without *preconception* about the content of the image. JND is used in grayscale approaches, particularly for insertion in the transform domain (DCT). For example, Watson's perceptual model (Watson 1993), proposed in 1993, makes it possible to visually optimize the quantization DCT matrices for a given image during its compression due to adaptations of contrast and luminance. These works were incorporated into digital watermarking by Li and Cox (2007) and also by Hu *et al*. (2010) in order to minimize insertion noise. However, as we can see, these works remain in the domain of grayscale images, and inclusion of color according to this strategy remains more unreliable (Chou and Liu 2010; Wan *et al.* 2020). Some of the literature dedicated to watermarking color images uses the concept of quaternions. However, as we will show in this chapter, the representation space remains configured by choices of chromatic direction controlling this question of invisibility. The quaternionic context for color image watermarking is discussed in section 4.3.

A more detailed understanding of the processing of color by the HVS would improve the invisibility of a mark. To illustrate this point, this chapter discusses a vector quantification method (section 4.2), associated with a biological model of HVS photoreceptors to be able to insert information into color images. The model studied is based on a psychovisual approach (section 4.4), making it possible to understand the HVS's perception of color differences (work introduced by Alleysson and Méary (2012)). We then adapt it to digital watermarking in order to minimize psychovisual distortions. We then detail the different steps of a robust watermarking scenario for color images, as well as the algorithms used to adapt the QIM Lattice (LQIM) method.

First of all, in order to illustrate the issue of color watermarking, we propose a general framework in section 4.2, making it possible to manipulate the color within the framework of a watermarking algorithm.

4.2. Color watermarking: an approach history?

As we have said, different approaches have been offered, based on the use of color space suitable for the insertion of a watermark, that is invisible to the naked eye. As a reminder, one of the first approaches involved inserting a mark in the blue component of the RGB space because the HVS is less sensitive to distortions in this channel (Kutter *et al*. 1997; Yu *et al*. 2001).

It is therefore a question of constructing a vector modification procedure, and to illustrate this, we choose a modification by quantization. Note that the discussion conducted in this framework by quantization can be generalized to other modification strategies and to simplify the understanding of the impact of modification, we intend to describe the process in the spatial domain.

4.2.1. *Vector quantization in the RGB space*

Vector quantization allows us to insert information about the values of a color pixel (three dimensions). It can be considered as a quantization on a line segment generated by a direction vector (Figure 4.1).

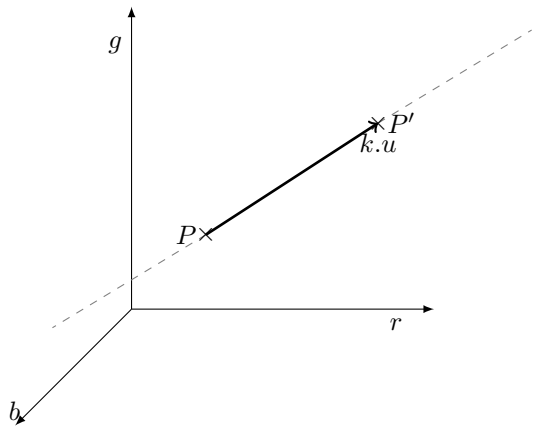


Figure 4.1. *Quantization in the RGB color space on an oriented line by a direction vector* u

Let P be a value of a color pixel. The result of quantization, denoted by P' is defined by:

$$P' = P + \frac{s' - s}{\|u_P\|^2}.u_P \qquad [4.1]$$

with u_P, a direction vector, and $s =< P, u_P >$, the canonical scalar product of P by u_P.

This equation modifies the color P according to s along the direction axis u_P. Depending on the color P on which the vector quantization is applied, the color distortion perceived by the HVS is different.

At the detection stage, the value to be estimated is s'. If the color is modified after the insertion of a watermark, we have:

$$s' =< P'', u_{P''} > \quad [4.2]$$

with P'', a color modified after inserting information, and $u_{P''}$, its associated direction vector. We then have a decoding condition on the direction vectors: it is necessary that the direction vectors u_P and $u_{P''}$ are close enough to have a good estimate of s'.

Assuming we are in a case where detection is possible, that is, the color difference $P'' - P'$ is small enough, we then have:

$$< P'', u_{P''} > \simeq < P', u_P >= s'$$

4.2.2. *Choosing a color direction*

The main differences between color specific watermarking methods are as follows:

– the insertion method (as for grayscale approaches);

– the choice of "color" information that will carry the message.

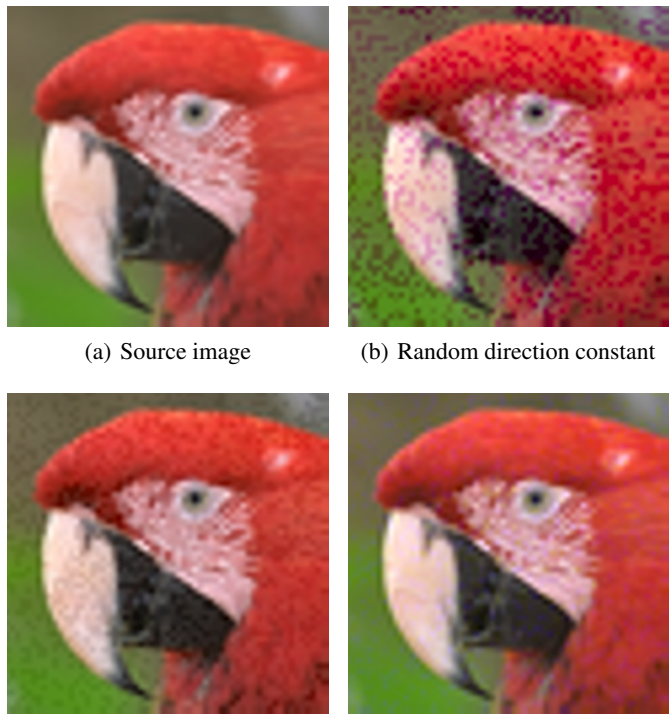
Whether the approaches are direct (use of a color channel), indirect (change of spaces) or even adapted (construction of a specific transform), in general, they can be modeled by the insertion of a message following a certain direction in the RGB cube. Therefore, the aim is to find a direction that minimizes the color quantization noise for the HVS.

To start with, we will study a basic approach that involves choosing a fixed insertion direction for the whole image. This approach illustrates the influence of this choice of direction when there is no local adaptation. If we randomly choose a direction vector constant for every color in the image, the HVS detects color distortions very easily (Figure 4.2(b)) since the choice of direction introduces colors that are not relevant to the content of the image.

We can reduce the detected quantization noise by using a direction that is judiciously fixed for the whole image; this is the case for a large number of algorithms in this book. A direction vector that limits color distortions is vector $u_g = (1, 1, 1)/\sqrt{3}$, which represents the gray level axis in the RGB space. An example is illustrated in Figure 4.2(c). In fact, compared to Figure 4.2(b), for the same level of modification, we no longer see false colors appearing.

However, these first results are not satisfactory in terms of watermarking invisibility. Depending on the color we modify, the detection of quantization noise is

different, we have to introduce a so-called *adaptive* approach. For each color P, we must determine which direction vector choice minimizes the quantization noise, in the sense of the HVS. Figure 4.2(d) shows an example of a watermarked image with an adaptive approach that we will describe in section 4.4.5. We can see that the quantization noise is less visible. However, on closer inspection, we observe a slight colored noise in the homogeneous (or non-textured) areas of the image, such as the green and brown background; the local context of the image is also important.



(a) Source image (b) Random direction constant

(c) u_g direction constant (d) Optimal direction suitable for each color

Figure 4.2. *Example of inserting a mark with different approaches and direction vectors with equivalent digital distortion (overall, the signal to noise ratio is equivalent for the three images). The image used is a cropped version of the image kodim23.png from the Kodak Image Database (available at: http://r0k.us/graphics/kodak/kodim23.html)*

As we suggested in the introduction, approaches using a more evolved representational space must address the same question: which is the best direction? To show this, we intend to recall the context of image watermarking according to quaternionic transforms.

4.3. Quaternionic context for watermarking color images

Quaternions can be a tool to manipulate color, which explains the different works offering quaternion-based watermarking (Bas *et al*. 2003; Tsui *et al*. 2008). In this section, we offer a few introductory elements on the tool.

4.3.1. *Quaternions and color images*

The quaternion algebra is algebra, $\mathbb{H}$, of dimension 4 over $\mathbb{R}$, made up of the elements of the form:

$$q = (q_0, q_1, q_2, q_3) = (S(q), V(q)) \qquad [4.3]$$

with the real number $S(p)$, the scalar part, and the vector $V(p)$, the vector part of the quaternion q.

All the usual geometric operations of $\mathbb{R}^3$ can be found when we limit the operations of quaternions to the set $\mathbb{H}_0$ of pure quaternions (if $q = (S(q), V(q))$ then $S(q) = 0$). Let q_1 and q_2 be two pure quaternions representing the vectors V_1 and V_2 of $\mathbb{R}^3$, respectively, so:

1) $q_1 + q_2 = V_1 + V_2$;

2) $q_1.q_2 = V_1.V_2$;

3) $q_1 q_2 = -V_1.V_2 + V_1 \wedge V_2$.

Note that the quaternionic product of two pure quaternions contains, in its real part, the scalar product of the two vectors represented, and, in its imaginary part, the cross product. This result is present in some uses of quaternions for color images.

The algebra is provided with the product:

$$pq = (S(p)S(q) - V(p).V(q), S(p)V(q) + S(q)V(p) + V(p) \times V(q))$$

where $V(p).V(q)$ (respectively, $V(p) \times V(q)$) denotes the scalar product (respectively, mixed or vectorial) of two vectors $V(p)$ and $V(q)$ of $\mathbb{R}^3$.

The algebra $\mathbb{H}$ is associative but not commutative.

We can find the so-called *Hamilton* form $q = q_0 + iq_1 + jq_2 + kq_3$, of quaternion q, by writing: $i = (0, 1, 0, 0)$, $j = (0, 0, 1, 0)$ and $k = (0, 0, 0, 1)$. The calculation rules then become:

$$i^2 = j^2 = k^2 = -1, \quad ij = -ji = k, \quad jk = -kj = i, \quad ki = -ik = j$$

We find a vocabulary that is similar to that of complex numbers. The conjugate $\overline{q}$ of a quaternion q is defined by: $\overline{q} = q_0 - iq_1 - jq_2 - kq_3$. Any quaternion q that is not zero accepts an inverse, called $q^{-1} = q/|q|^2$, where $|q|^2 = q\overline{q} = \overline{q}q = q_0^2 + q_1^2 + q_2^2 + q_3^2$. The positive real number $\sqrt{q\overline{q}} = \sqrt{|q|^2}$ is *the magnitude* of q, denoted $|q|$. As for complex numbers, it is possible to define combined quaternions (if $q = (S(q), V(q))$ then $S(q)^2 + \|V(q)\|^2 = 1$).

Since a color contains only three components in the RGB space, it has been suggested that we describe the color information on the imaginary part of the quaternions (Sangwine 2000):

$$\begin{aligned} f : [m, n] &\mapsto (f_r, f_g, f_b) \\ \mathbb{Z} \times \mathbb{Z} \quad &\to \mathbb{H}_0 \end{aligned}$$

The color of a pixel of an image f at spatial coordinates (m, n) is then coded in the following way:

$$f[m, n] = f_r[m, n]i + f_v[m, n]j + f_b[m, n]k \qquad [4.4]$$

with $f_r[m, n]$, $f_v[m, n]$ and $f_b[m, n]$ as the red, green and blue components of the pixel of coordinates (m, n), respectively. This is how, in the vast majority of quaternion based color watermarking algorithms, the information is modified.

Consider a pixel color $f[m, n] = (f_r[m, n], f_v[m, n], f_b[m, n])$ coded by a pure quaternion q:

$$q = (0, f_r[m, n], f_v[m, n], f_b[m, n]) \qquad [4.5]$$

We want to extract the colorimetric coordinates (*intensity, hue* and *saturation*). First we need to define a vector μ (pure unitary quaternion) representing the axis of *intensities*. The *intensity* I of q is the projection of this on μ:

$$I = q.\mu$$

This axis is generally defined as the grayscale axis, so $\mu = (0, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$, but in a watermarking strategy a choice for this direction μ can be more adaptive. Following this idea, we can generalize the operations for manipulating color information through the quaternion formalism (Carré *et al*. 2012):

$$\hat{q} = \left(\frac{\alpha}{2} + \frac{\beta}{2}\right) e^{\mu\phi} q e^{-\mu\phi} - \left(\frac{\alpha}{2} - \frac{\beta}{2}\right) \mu e^{\mu\phi} q e^{-\mu\phi} \mu$$

with the result:

$$\left\{ \begin{array}{l} \hat{T} = T + \phi \, ; \\ \hat{S} = \beta S \, ; \\ \hat{I} = \alpha I \end{array} \right\}$$

These few simple operations allow us to illustrate the first links between quaternions and color. We note that, again, the central element to all modifications is the definition of an axis μ, around which the different transforms will be configured. This is the case for quaternionic watermarking algorithms. However, most of them use a transformed domain, particularly a frequency domain.

4.3.2. *Quaternionic Fourier transforms*

Quaternionic Fourier transforms (QFT) were introduced in the context of color images by Sangwine and Ell (2001) in order to generalize the complex Fourier transform for color images. There are several versions of quaternionic Fourier transforms with different aims, particularly because the quaternionic product is not commutative. Sangwine offers a color version by adding a direction that is characterized by a pure unitary quaternion, to the expression of the exponential μ (Ell and Sangwine 2006). Most often, in order to avoid favoring any color for the spectral analysis, the neutral quaternion $\mu = \mu_{gris} = (i + j + k)/\sqrt{3}$ is chosen and corresponds to the achromatic axis of the RGB space. The definition is as follows:

$$F_{gauche}[o,p] = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} e^{-2\mu\pi\left(\frac{om}{M} + \frac{pn}{N}\right)} f[m,n] \qquad [4.6]$$

It is possible to place the exponential to the right of the function f. We then obtain the definition of another Fourier transform since the quaternionic product is not commutative.

Different authors have looked to give an interpretation to the information described by the spectrum coefficients. We can identify three main approaches:

– describe the quaternion spectrum with the exponential representation (Assefa *et al*. 2010);

– break down each spectral coefficient into two parts, one simplex and one perplex, in order to separate the luminosity data from the chrominance data (Sangwine and Ell 2001);

– carry out an analysis of elementary atoms (Denis *et al*. 2007).

This frequency information present in the quaternionic spectrum is especially used in digital image watermarking (Bas *et al*. 2003). To analyze watermarking in quaternion space, we introduce the Cayley–Dickson construction, which allows the separation of data into a luminosity part and a chromaticity part (Ell and Sangwine 2000). This rewriting has made it possible to analyze the transformations carried out with the quaternions through a "colormetric" view. We have seen that a quaternion can be considered as a vector of the base (e, i, j, k) with $e = 1$, the real unitary vector:

$$q = ae + bi + cj + dk$$

with a, b, c, d as any four real numbers of which a is the real part and (b, c, d) is the imaginary part.

We can choose another base, for example $(e, \mu, \nu, \mu\nu)$ with μ and ν, two pure unitary orthogonal quaternions. Their product $\mu\nu$ is also a pure and orthogonal quaternion to the other vectors of the base. We can write q in this base:

$$q = q_e + q_\mu\mu + q_\nu\nu + q_{\mu\nu}\mu\nu$$

with, for example, the real number $q_e = q.e$ defined by the scalar product and which therefore corresponds to the projection of q on e.

Likewise, we can calculate $q_\mu = q.\mu$, $q_\nu = q.\nu$ and $q_{\mu\nu} = q.\mu\nu$ of real numbers.

Notice that it is possible to regroup the quaternion q in two isomorphic subspaces in $\mathbb{C}$. In fact, the first block (q_e, q_μ) corresponds to the real part and the projection term on μ, the remainder being expressed in the plane $(\nu, \mu\nu)$, which is the plane perpendicular to μ. The quaternion can therefore be written as:

$$q = q_\parallel + q_\perp\nu$$

with $q_\parallel = q_e e + q_\mu\mu$ and $q_\perp = q_\nu e + q_{\mu\nu}\mu$, $q_\parallel$ being the parallel part to μ and $q_\perp$ the perpendicular part.

In the case of pixel color manipulation, two pure quaternions are used for this decomposition: the first vector μ is, for example, the gray axis in the cube RGB:

$$\mu = \mu_{grey} = \frac{i + j + k}{\sqrt{3}}$$

The second is the axis perpendicular to μ in the direction of the red color (pointing out axis i):

$$\nu = \sqrt{\frac{2}{3}}(i, -\frac{j}{2}, -\frac{k}{2})$$

since the red is used as a reference for a zero-value hue in the chromticity plane.

In this case, when a color image is divided into parts parallel and perpendicular to μ, it seems that this decomposition allows the separation of luminosity data on the parallel part, and chromatic data on the part perpendicular to μ. We find the classic representation associated with luminance–chrominance spaces.

When a watermarking algorithm suggests using the quaternion space to carry out the insertion of a message, especially in the transformed domain, in many cases the operation can be interpreted through the Cayley–Dickson representation. This results in a modification of the color pixels in one of the directions: the impact of the modification will therefore depend on the explicit or implicit choice of the μ axis. Although the approach uses a more complex model which seems to take the color into account, the algorithm remains configured by a wise choice of insertion color direction.

In section 4.4, we propose to discuss a psychovisual model that makes the choice of direction as adaptive as possible, that is, we show how to determine the optimal direction vectors for any RGB color.

4.4. Psychovisual approach to color watermarking

4.4.1. *Neurogeometry and perception*

Understanding how humans perceive color differences can allow us to minimize insertion color distortions perceptually; in other words, to improve watermark invisibility. To achieve this objective, we need a color discrimination model. This model would be three dimensional, due to the human trichromy, and should mimic the representation of light by the cones of the human retina for their spectral sensitivities as well as their intensity encoding dynamics.

This type of model, known as neurogeometry, was suggested in the context of the perception of shapes based on achromatic data. A detailed description of the construction of forms by the primary visual cortex is given in Petitot (2003, 2008). Few other contributions have considered neurogeometry in the context of color vision (Koenderink *et al*. 1970; Zrenner *et al*. 1999; Alleysson and Méary 2012), and even less in the application context of color watermarking.

In this section, we limit our study to the modeling part and its application for color watermarking: more precisely, we use a chromatic discrimination ellipsoid model based on the Naka–Rushton law of photoreceptor dynamics.

A simple and effective way to model color discrimination data involves considering color discrimination to be a nonlinear operation in a three-dimensional

Euclidean vector space $\varphi \cong \mathbb{R}^3$. The space has the standard scalar product $xy = x_1y_1 + x_2y_2 + x_3y_3$, where $x = [x_1, x_2, x_3]$ and $y = [y_1, y_2, y_3]$ are two vectors in $\mathbb{R}^3$ linked to two points, M and N, as $\overrightarrow{OM} = x$ and $\overrightarrow{ON} = y$ with O, the origin of vectorial space associated with φ. The magnitude of a vector x is given by $||x|| = \sqrt{x.x}$. The distance between x and y is given by $d(x, y) = ||x - y||$.

Assume that this space φ is a physical space in which the HVS maps another representation ψ, marked by a quadratic form $q(x) = x^tGx$ where G is a symmetrical positive definite matrix representing the metric brought about by the nonlinearity of the HVS. This quadratic form defines a new form for vectors in the visual space, such as $|x| = \sqrt{q(x)}$, and a scalar product balanced such that:

$$< x, y >= x^tGy \qquad [4.7]$$

In this model, we assume that the HVS maps the physical space of light φ and the perceptual space ψ. An equal discrimination contour is represented by a circle on an isoluminant surface and the physical space is a space in which this constant discrimination circle in ψ corresponds to an ellipse in φ. The mapping function between these spaces is given by the neuronal function. This model is illustrated in Figure 4.3. In this figure, we assume that the HVS is a nonlinear function $\mathbf{f}$ between the flat physical space of light φ and the curved space of perception ψ. In this section, we use a modification of the Naka–Rushton transduction equation to account for visual nonlinearity.

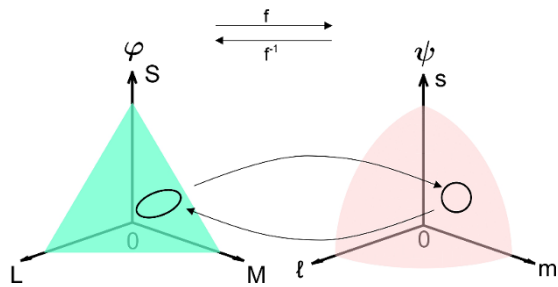


Figure 4.3. *Relationship between physical space φ and perceptual space ψ*

For the sake of generality, we must consider that G depends on the physical stimulus x and on the adaptation state of the observer x_0, so we write $G(x, x_0)$. The question to be solved for a color discrimination space is to find the relationship between the nonlinearity brought about by the HVS, and the shape of the metric $G(x, x_0)$. One way to clarify this relationship is to fix the adaptive nonlinearity

function. This cannot be done in general, but we assume that the nonlinearity of the HVS operates on each color channel separately and that it is parametric with an adaptation parameter, which modifies the curve of the nonlinearity.

4.4.2. *Photoreceptor model and trichromatic vision*

Based on this principle, we introduce a three-dimensional construction of an HVS trichomatic model (Alleysson and Méary 2012).

An HVS is a complex system that adapts according to environmental conditions, and it is no surprise that color vision is a nonlinear, adaptive phenomenon.

Assume for now that x is a physical one-dimensional variable corresponding to light entering the eye. $y = f(x, x_0)$ is the transformation of this variable by the HVS according to an adaptation factor x_0.

We define f as a simplification of the Naka–Rushton function, which checks the transduction of light by the photoreceptor (Alleysson and Hérault 2001):

$$f(x) = \alpha \frac{x}{x+1}$$
$$y = f(x/x_0) = \alpha \frac{x}{x+x_0} \qquad [4.8]$$

where y is the level of transduction. In other words, y represents the electric response of a cone as a function of x, the level of excitation of the cone produced by light and x_0, the state of adaptation.

x_0 is modified according to the average excitation level of the photoreceptor. Equation [4.8] represents the behavior of the photoreceptor according to two constants, α and x_0, set by the environmental conditions and the HVS state. In fact, the adaptation process of a photoreceptor is only partially understood and this model is probably a simplification.

We assume that this is the nonlinear, adaptive function of equation [4.8], which matches physical space to perceptual space. The function saturates α when $x \to \infty$ and x_0 gives the half-maximum of the curve. The function's curve changes according to x_0. A low x_0 gives a high curvature, while a high x_0 gives a low curvature. This nonlinear function is shown in Figure 4.4.

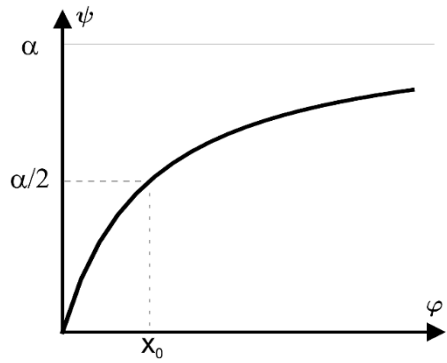


Figure 4.4. *Nonlinear function of perception*

Therefore, considering the three-dimensional space of the physical colors φ, we establish the link between the physical and perceptual space by:

$$y = \begin{bmatrix} \ell \\ m \\ s \end{bmatrix} = \begin{bmatrix} f(L/L_0) \\ f(M/M_0) \\ f(S/S_0) \end{bmatrix} = \mathbf{f}\left(\begin{bmatrix} L \\ M \\ S \end{bmatrix}, \begin{bmatrix} L_0 \\ M_0 \\ S_0 \end{bmatrix} \right) = \mathbf{f}(x, x0) \quad [4.9]$$

where $x = [L, M, S]$ and $x_0 = [L_0, M_0, S_0]$ are coordinates of the physical stimulus in the space φ, and the adaptation of the observer expressed as a parameter in the space φ, respectively.

Here, we implicitly assume, without losing generality, that the coordinates in the space φ correspond to the responses of cones L, M and S. L, M and S are then the encoding of the physical stimulus x by the HVS, suitable for x_0.

To summarize, the human retina possesses three types of photoreceptors, which are cones L, M and S. The responses of each of these cones are processed by a nonlinear parametric function that defines the perceptual variables. For each color channel, there are two parameters, α and x_0:

$$\begin{cases} l = \alpha_L \dfrac{L}{L + L_0} \\ m = \alpha_M \dfrac{M}{M + M_0} \\ s = \alpha_S \dfrac{S}{S + S_0} \end{cases} \quad [4.10]$$

where the parameters α_L, α_M and α_S are gains on the respective components L, M and S. The constants L_0, M_0 and S_0 are the adaptation states of the respective cones.

In the trichromatic model of color vision, it is possible to find a three-dimensional construction of color discrimination with the photoreceptor model described above. In the space of transduction lms, pairs of points separated by the same distance have the same level of perception of the color difference. When these pairs are converted in the excitation space LMS, Euclidean distances are no longer equal even though the same level of perception is maintained.

This distortion phenomenon can be better observed by constructing a sphere centered at $P_{lms} = (l_c, m_c, s_c)$ in the space lms defined as:

$$\begin{cases} l = r\cos(u)\cos(v) + l_c \\ m = r\sin(u)\cos(v) + m_c \\ s = r\sin(v) + s_c \end{cases} \quad [4.11]$$

with r, the radius of the sphere, $-\pi \leq u \leq \pi$ and $-\pi/2 \leq v \leq \pi/2$.

By converting the sphere in the space LMS, we get a volume $\mathcal{V}$ defined by:

$$\begin{cases} L = \dfrac{(r\cos(u)\cos(v) + l_c)L_0}{\alpha_L - r\cos(u)\cos(v) - l_c} \\ M = \dfrac{(r\sin(u)\cos(v) + m_c)M_0}{\alpha_M - r\sin(u)\cos(v) - m_c} \\ S = \dfrac{(r\sin(v) + s_c)S_0}{\alpha_S - r\sin(v) - s_c} \end{cases} \quad [4.12]$$

The distances between the center of the volume P_{lms} and the points on the surface of the volume are therefore constant:

$$\|P - P_{lms}\|_2 = r, \forall P \in \S \quad [4.13]$$

with a constant radius of r.

Note that r is no longer constant in the case of the volume $\mathcal{V}$ of center $P_{LMS} = \mathbf{f}^{-1}(P_{lms})$.

4.4.3. *Model approximation*

We then need to explain how to calculate $G(x, x_0)$, the metric from $\mathbf{f}$. To do this, we consider that in the perceptual space ψ, an equal discrimination contour around a point (ℓ_c, m_c, s_c) is a sphere S with radius 1. So we write:

$$S = \{y \in \psi | dy^t dy = 1, dy = [\ell - \ell_c, m - m_c, s - s_c]\} \quad [4.14]$$

As we have seen, in the physical space, the resulting surface is given by the linear approximation of the nonlinearity $\bar{f}(x_c, x_0)$ around x_c. As we can consider dy to be very small, it is given by the Jacobian of $\mathbf{f}$:

$$J(x_c, x_0) \quad [4.15]$$

$$= \begin{bmatrix} \frac{\partial f(L_c,L_0)}{\partial L_c} & \frac{\partial f(L_c,L_0)}{\partial M_c} & \frac{\partial f(L_c,L_0)}{\partial S_c} \\ \frac{\partial f(M_c,M_0)}{\partial L_c} & \frac{\partial f(M_c,M_0)}{\partial M_c} & \frac{\partial f(M_c,M_0)}{\partial S_c} \\ \frac{\partial f(S_c,S_0)}{\partial L_c} & \frac{\partial f(S_c,S_0)}{\partial M_c} & \frac{\partial f(S_c,S_0)}{\partial S_c} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\alpha_L L_0}{(L_c+L_0)^2} & 0 & 0 \\ 0 & \frac{\alpha_M M_0}{(M_c+M_0)^2} & 0 \\ 0 & 0 & \frac{\alpha_S S_0}{(S_c+S_0)^2} \end{bmatrix} \quad [4.16]$$

This approximation makes it possible to establish a direct correspondence between $dy = [\ell - \ell_c, m - m_c, s - s_c]$ and $dx = [L - L_c, M - M_c, S - S_c]$:

$$dy = J(x_c, x_0)dx \iff dx = J^{-1}(x_c, x_0)dy \quad [4.17]$$

As we have mentioned, a circle in perceptual space ψ corresponds to an ellipse in physical space φ expressed as:

$$E = \{x \in \mathcal{E} | dx^t G dx = 1, G = J(x_c, x_0)^{-t} J(x_c, x_0)^{-1}\} \quad [4.18]$$

G is equal to:

$$G = \begin{bmatrix} \frac{(L_c+L_0)^4}{\alpha_L^2 L_0^2} & 0 & 0 \\ 0 & \frac{(M_c+M_0)^4}{\alpha_M^2 M_0^2} & 0 \\ 0 & 0 & \frac{(S_c+S_0)^4}{\alpha_S^2 S_0^2} \end{bmatrix} \quad [4.19]$$

We see that, from a color defined by coordinates (L_c, M_c, S_c), we have two groups of parameters in this transformation, (L_0, M_0, S_0) and $(\alpha_L, \alpha_M, \alpha_S)$.

4.4.4. *Parameters of the model*

To use this model, the constants L_0, M_0 and S_0 and the gains α_L, α_M and α_S can be calculated using MacAdam ellipses (MacAdam 1942) (Figure 4.5).

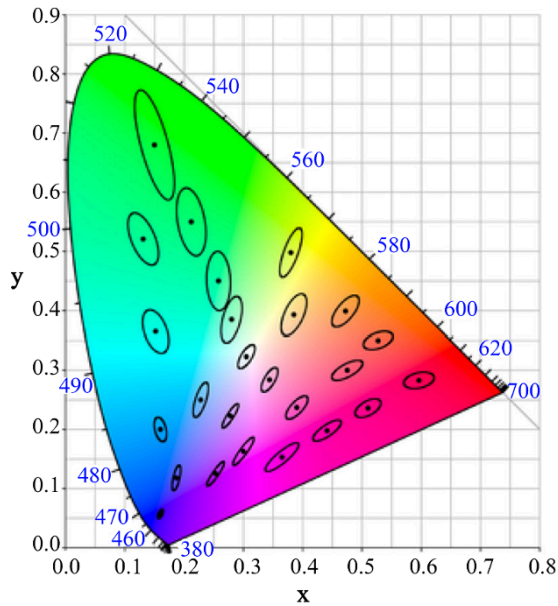


Figure 4.5. *MacAdam ellipses in the luminance plane of the color space xyY, 1931. Ellipses are 10 times larger than their original sizes*

If we want to calculate these parameters from the MacAdam ellipses, we have to first transform the MacAdam measurement made in the CIE-xyl space into space LMS. To do this, we use the following transformation:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = Y \begin{bmatrix} 0.15514 & 0.54312 & 0.03286 \\ -0.15514 & 0.45684 & 0.03286 \\ 0 & 0 & 0.00801 \end{bmatrix} \begin{bmatrix} x/y \\ 1 \\ \frac{1-x-y}{y} \end{bmatrix} \quad [4.20]$$

With this transformation of (x, y, l) into (L, M, S) and with the parameter $(\alpha_L = 1665, \alpha_M = 1665, \alpha_S = 226)$, $(L_0 = 66, M_0 = 33, S_0 = 0.16)$, it has been shown that this allows a good reconstruction of MacAdam ellipses (Figure 4.6) (Alleysson 1999).

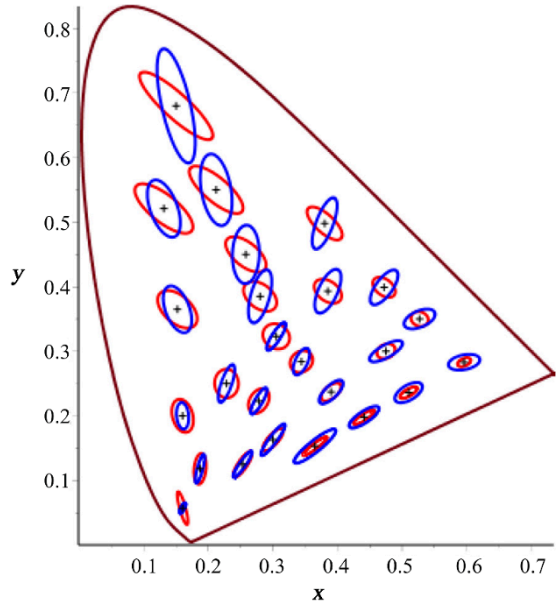


Figure 4.6. *The ellipses obtained from the psychovisual model almost correspond to the MacAdam ellipses after an optimal choice of constants (Alleysson 1999)*

4.4.5. *Application to watermarking color images*

In section 4.4.4, we introduced a color vision model to develop a less visible method of watermarking. In the color space LMS, we constructed ellipsoids that concretely illustrate how to control psychovisual distortion while ensuring satisfactory robustness.

In fact, the idea of psychovisual distorsion represents the perception of the color difference on insertion, as opposed to digital (Euclidean) distortions obtained with metrics such as the PSNR. As psychovisual distortions better describe the invisibility of the watermark for the HVS than digital distortion measurements, the theory is that it becomes possible to obtain a better compromise between invisibility and robustness: for the same level of psychovisual distortion, there are different levels of digital distortion, depending on the insert setting, and choosing maximum digital distortion improves robustness.

In this section, we intend to remove, for each color pixel P, the direction vector u_P which has the largest magnitude in its ellipsoid.

4.4.6. *Conversions*

Here we give the color space conversion matrices to work from XYZ and LMS psychovisual spaces to RGB space. The transformations are chosen according to the 1931 CIE standard[1]. The conversion between color spaces is represented by the following matrices:

$$M_1 = M_{RGB \to XYZ} = \begin{pmatrix} 0.488718 & 0.310680 & 0.200602 \\ 0.176204 & 0.812985 & 0.010811 \\ 0.000000 & 0.010205 & 0.989795 \end{pmatrix}$$

$$M_2 = M_{XYZ \to LMS} = \begin{pmatrix} 0.38971 & 0.68898 & -0.07868 \\ -0.22981 & 1.18340 & 0.04641 \\ 0.0 & 0.0 & 1.0 \end{pmatrix}$$

Let P_{RGB}, P_{XYZ} and P_{LMS} be the color pixels in their respective RGB spaces, XYZ and LMS. We have:

$$\begin{aligned} P_{LMS} &= M_2 P_{XYZ} \\ P_{XYZ} &= M_1 P_{RGB} \\ P_{LMS} &= M_2 M_1 P_{RGB} \end{aligned} \qquad [4.21]$$

In section 4.2, we have seen that the choice of a direction vector has an enormous impact on the invisibility of a watermark. When we choose a fixed direction for all the colors, we note strong psychovisual distortions in certain areas of the image. However, invisibility is greatly improved when the best direction vector is chosen for each color pixel. Since we assume that the psychovisual distortions are the same for all the elements of ψ, we can choose the point furthest away from P, indicated by P_f, to allow the maximum numerical distortions authorized during integration in order to reinforce robustness. The direction vector u_P is defined such that $u_P = \overrightarrow{PP_f}$.

Note that $\mathcal{E}$ is the set of points belonging to the volume of perception associated with P. We have the farthest point P_f from P:

$$P_f = \max_{P' \in \mathcal{E}} \|P - P'\|_2 \qquad [4.22]$$

1 White point of equal energy E.

To estimate P_f or u_P, we use the Jacobian: to extract an optimal direction, we simply use the fact that the main axes of the ellipses can be calculated from the eigenvectors of the metric. Note that, from equation [4.19], it is possible to directly calculate the expression of the direction u_p for each color.

4.4.7. *Psychovisual algorithm for color images*

The algorithm that we present in this section makes it possible to insert a mark in a color image whose quality is improved from a psychovisual point of view. Compared to a classical insertion scenario, we suggest adding a step to adapt insertion to the color space (illustrated in Figure 4.7).

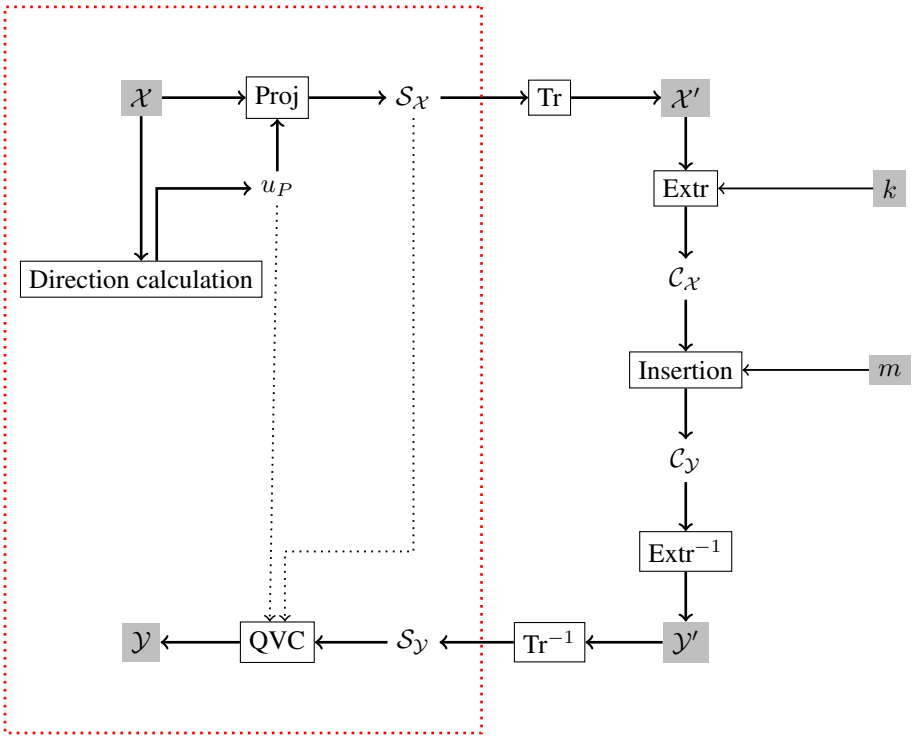


Figure 4.7. *Classical insertion diagram combined with color vector quantization (QVC) based on a psychovisual model. The elements within the red frame represent the steps of vector quantization. Tr() and Extr() are the space transformation and coefficient extraction functions, k is the secret key and m is the binary message*

Before transforming the host image in the chosen quantization space, we calculate a direction vector for each color pixel of the image $\mathcal{X}$. This direction vector will be

defined from the optimal axis presented in section 4.4.6. Then we introduce the scalar product, s (the "Proj" step in Figure 4.7).

$$s =< P, u_P > \quad [4.23]$$

We then obtain a scalar value image $\mathcal{S}_\mathcal{X}$. Considering the knowledge of the set of computed direction vectors, we have a one-to-one correspondence between $\mathcal{X}$ and $\mathcal{S}_\mathcal{X}$. In Figure 4.8, we can see that image $\mathcal{S}_\mathcal{X}$ is a dark grayscale version of the color image $\mathcal{X}$.

Figure 4.8. *Pairs of images (host image $\mathcal{X}$, associated scalar image $\mathcal{S}_\mathcal{X}$). Random images from the Corel database (available at: https://sites.google.com/site/dctresearch/Home/content-based-image-retrieval)*

The next step then involves changing the representation of $\mathcal{S}_\mathcal{X}$. A good first choice of representation is the spatial domain, but a transformed space can be used (transformed into discrete cosine [DCT] or transformed into wavelet coefficients [DWT]). These different representations (denoted Tr and Tr^{-1} for the inverse operation) of the image make it possible to obtain useful properties according to requirements, such as breaking the image down into independent frequency bands.

An extraction function Extr must then be chosen in order to select the coefficients to be changed. In this chapter, we have chosen a function which randomly determines insertion sites of the image $\mathcal{S}_{\mathcal{X}}$. Another method for selecting coefficients was presented in Chapter 3, which involves breaking an image down into blocks and then randomly selecting coefficients in each block. The result of this extraction takes place in the watermarking space, which is a secret space (known only to the creator of the mark and its recipient). Access to this space can be secured with a secret key k.

Once the coefficients $\mathcal{C}_{\mathcal{X}}$ are selected, they are modified by the insertion method (the LQIM method, but it is also possible to use other watermarking methods) according to the message m, then integrated into the representation of the image used for extraction and we get the marked image $\mathcal{Y}'$. So, for any modified color P' (QVC stage), we have:

$$P' = P + \frac{(s' - s)}{\|u_P\|^2} u_P \qquad [4.24]$$

where s' is the scalar modified by the chosen insertion method.

The set of s' forms the grayscale image $\mathcal{S}_{\mathcal{Y}}$. Of course, note that the use of different error correcting codes, such as those proposed in Chapter 3 (e.g. Hamming, BCH, Reed–Solomon and Gabidulin codes), is completely possible and straightforward. In this chapter, we focus only on the "invisibility control" aspect and its optimization with respect to robustness.

At the detection stage (Figure 4.9), we identify the grayscale image $\mathcal{S}_{\mathcal{Z}}$ by recalculating the direction vectors associated with each color of the image received $\mathcal{Z}$ (equation [4.23]), and then accessing the watermark channel using the Extr extraction function and the key k.

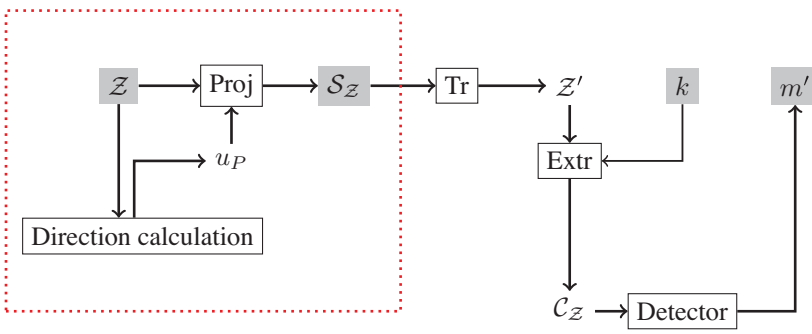


Figure 4.9. *Classical mark detection diagram. As with insertion, we find the extraction part of the scalar image $\mathcal{S}_{\mathcal{Z}}$ framed in red*

We then apply the detector associated with the chosen insertion method to the coefficients $\mathcal{C}_{\mathcal{Z}}$. If the power of the attack is reasonable and the mark robustness parameters are well chosen, the estimate m' should be the same as m.

As we said for the detection, the calculation of the direction vectors is an important step. Assuming that the colors are reasonably modified, the ability to correctly detect the message lies in the variation of the direction vector. In our experiments we will see that the variations of the direction vectors are acceptable and make it possible to ensure good detection.

We have detailed a classical watermarking algorithm for color images using a color vector quantization method based on a psychovisual model of the HVS. The LQIM method was chosen to be adapted to watermarking color images (introduced in section 4.2). To be able to use it, we adapt the classical insertion diagram with our psychovisual quantization model 4.7, then we integrate the quantifier Q_m into the insertion function of the LQIM method. The insertion sites are chosen randomly from the coefficients available for insertion. One of the advantages of this insertion strategy is improving the invisibility of the mark, especially in textured areas of the image. At the detection stage, we also combine the extraction stage of the received coefficients $\mathcal{C}_{\mathcal{Z}}$ (Figure 4.9) with the classical decoding scheme using the LQIM detector.

Section 4.4.8 gives an evaluation of the robustness performance of the CLQIM method against various image modifications.

4.4.8. *Experimental validation of the psychovisual approach for color watermarking*

The images used for the experiments belong to the Corel database (1,000 random images selected from the 10,000 available). The insertion sites are determined randomly. To evaluate the invisibility, parameters such as the size of the message or the dimension of the Euclidean lattice L of the Lattice QIM method are calculated in order to obtain an adequate image quality and an adequate insertion rate.

4.4.8.1. *Validation of invisibility*

In this section, we propose different groups of marked images in order to appreciate the improvement in invisibility, compared to marked images following a vector quantization of constant direction vector.

We then denote the following two approaches by GA and AA:

– the color version with a constant direction axis $u = (1, 1, 1)$ (GA). We have chosen this direction vector because choosing the luminance axis is, in our opinion, a satisfactory compromise in order to guarantee a good level of invisibility of a mark;

– the color version with an adaptive direction axis u_P (AA).

These two watermarking methods are the color adaptations of the LQIM method previously presented in this chapter. For each approach, the direction vectors are the same magnitude (arbitrarily set at 0.5). For the same level of digital distortion, it is a question of validating, through experimentation, that the AA approach inserts a less visible mark than the GA approach.

In these experiments, we evaluate the digital distortion thanks to the signal-to-noise ratio, characterizing the quantization noise of the LQIM method, denoted as DWR.

In Figure 4.10, we show examples of color images marked with the GA approach of the LQIM method. For each image in this figure, we can easily see that the colors saturate toward gray. It therefore becomes easier to perceive color differences by comparing them to neighboring pixels. The visual appearance of quantization noise is salt and pepper noise and it adds texture to the image.

Figure 4.10. *Cropped color images (Lena and Kodak base of size 60 $\times$ 60) marked with the LQIM method (GA approach), DWR $\simeq -$ 5.5 dB on average and ER $=$ 0.5*

With the AA approach (Figure 4.11), the quantization noise has the effect of saturating the colors toward blue and green. This adapts according to the modified color. It is therefore more difficult to perceive the noise. Compared to neighboring pixels in areas of the same color, we find that detecting color difference is difficult.

Figure 4.11. *Cropped color images (Lena and Kodak base of size 60 × 60) marked with the LQIM method (AA approach), DWR ≃ − 5.5 dB on average and ER = 0.5*

At equal digital distortion, images marked with the GA and AA approaches are not affected with the same quantization noise from a psychovisual point of view. The perception of quantization noise with the AA approach is much lower for the HVS compared to that of the AA approach. We can see this in the images of Figures 4.10 and 4.11, but these results were also confirmed in other tests.

To able to better observe the insertion noise, we have chosen images of size 60 × 60 and have chosen a strong insertion rate ($ER = 0.5$). In practice, the images are much larger in size (e.g. 1,080 × 1,920 for high definition) and the insertion rate is weaker, which makes the insertion noise less visible for the HVS.

We repeated the comparison experiment of the GA and AA approaches described above with different observers (15 in number) and with the Kodak image base, composed of 24 elements of size 768 × 512. For each image of this database, we offered a pair of images marked with the two color approaches GA and AA (still with the LQIM method). For the 24 pairs of images, each observer voted for the "least noisy" image.

The results of this experiment are presented in Table 4.1. Of these subjects, only 4% of images on average were described as "least noisy" with the GA approach. Of these results, we conclude that the use of a psychovisual model allows us to obtain a much better invisibility.

Approaches	GA approaches	AA approaches
Average votes	$4 \pm 3\%$	$96 \pm 3\%$

Table 4.1. *Psychovisual experiments of marked image comparisons*

COMMENT ON TABLE 4.1.– *Each person had to decide which of the images was more degraded (a watermarked image with the constant approach, and the other with the adaptive approach). The parameters are DWR = 20 dB, ER = 0.5 for each image. This table shows the percentage of images noted as less degraded for each approach.*

We now intend to compare the two approaches in terms of their robustness performance against several image modifications. We offer a simple experiment for this. We propose to insert a message of size $n = 128$ bits. In order to ensure satisfactory image quality and mark invisibility, we determined the average maximum quantization step before a mark was visible for each method. We measured average bit error rates (100 repetitions for the same quantization step) depending on the strength of the applied image modification. We now propose to analyze the robustness against classical attacks.

4.4.8.2. *Impact on robustness*

4.4.8.2.1. Modification of luminance

We model the modification of luminance by equation [4.25]:

$$y = x + \beta \times (1, \ldots, 1) \qquad [4.25]$$

where y is the modified version of x, the value of a pixel.

The robustness results are shown in Figure 4.12. The error curves oscillate like a step function between 0 and 1 periodically. This attack is special because it applies modifications that are not dependent on the image. This behavior is justified, both by the use of the LQIM method, and by the structure of the error produced by the modification of luminance (studied in detail in Chapter 3).

For the LQIM AA curve, the oscillation period is greater than that of the LQIM GA curve, which gives a larger error-free interval around 0.

4.4.8.2.2. Modification in the HSV space

We now consider examples of color image modifications. By representing an image in the HSV color space, we modify each color component h, s and v.

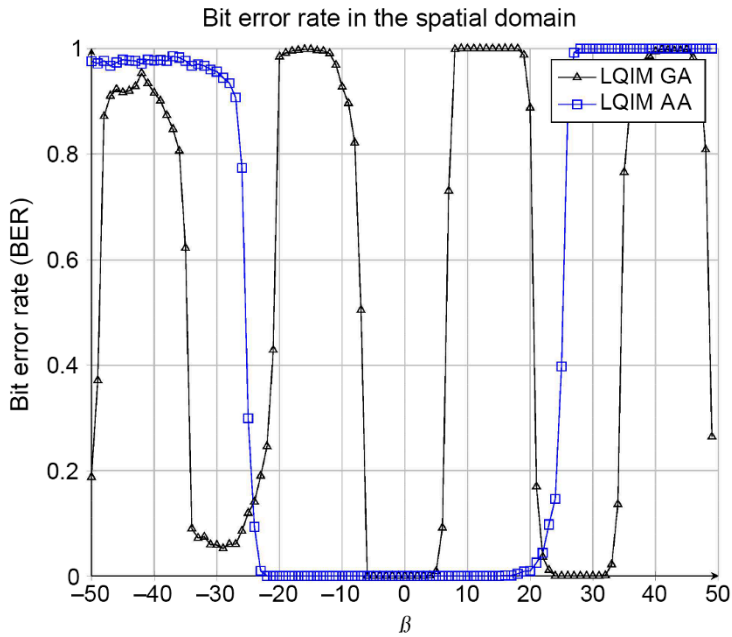
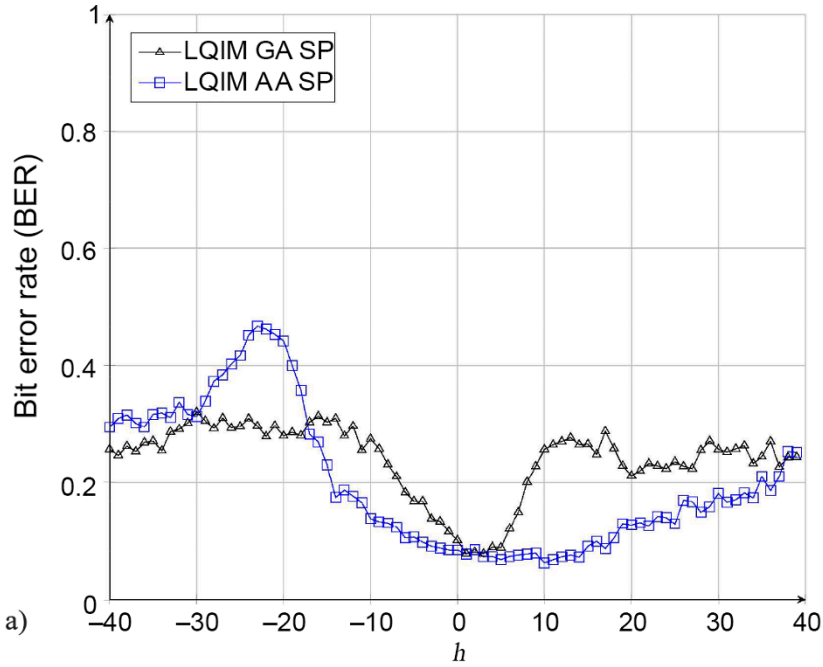


Figure 4.12. *Binary error for methods GA and AA depending on the parameter* β

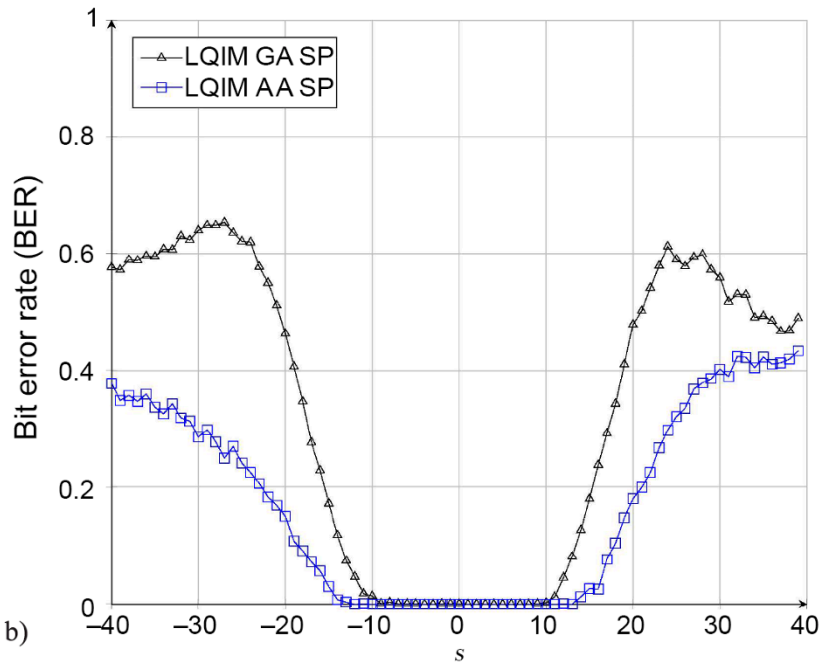
In Figure 4.13(b), we observe that the LQIM AA SP curves are closer to 0 than the LQIM GA SP curves on each component, therefore showing that taking an HVS model into account allows us to improve the robustness of a watermarking algorithm, not just its invisibility.

4.5. Conclusion

In this chapter, we have dealt with the challenge of the invisibility of hidden information in the context of color images. For this, the latest developments of digital watermarking for color images have been proposed. We find that the majority of the methods are an extension of the approaches for grayscale images. We then showed that, in many cases, the key is wisely choosing the color direction vector at insertion, and this is true even with dedicated spaces, such as quaternions. We then recalled the impact of this direction choice and discussed the importance of taking the HVS into account.


LQIM GA SP
LQIM AA SP
Bit error rate (BER)
1
0.8
0.6
0.4
0.2
0
–40
–30
–20
–10
0
10
20
30
40
h

a)


LQIM GA SP
LQIM AA SP
Bit error rate (BER)
1
0.8
0.6
0.4
0.2
0
–40
–30
–20
–10
0
10
20
30
40
s

b)

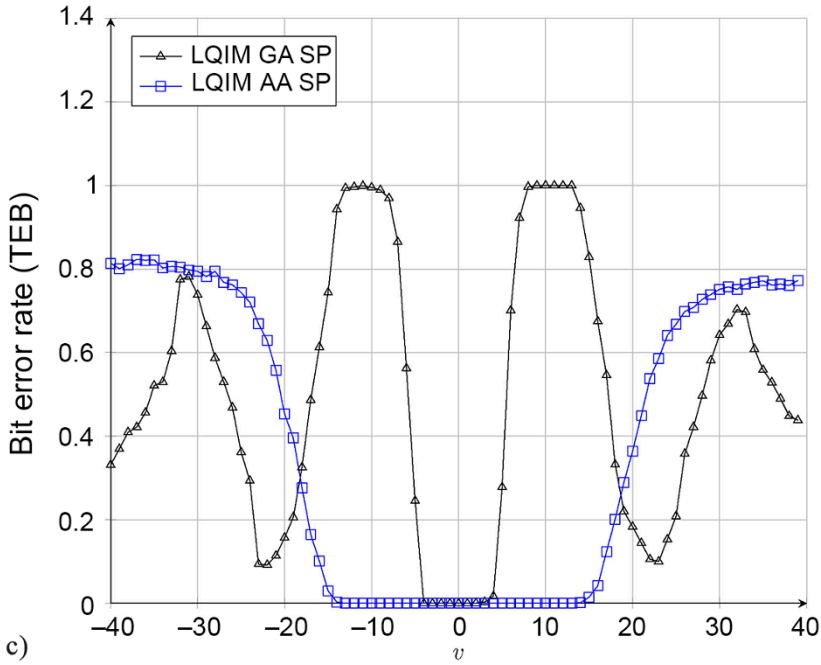


Figure 4.13. *Bit error rate for (a) hue; (b) saturation; and (c) value modification*

We then discussed a psychovisual approach, allowing us to model the behavior of photoreceptors in the human retina. Thanks to the calibration of the constants, based on the measurements of MacAdam ellipses, the photoreceptor model makes it possible to simulate the behavior of an average HVS in terms of the perception of color differences. We then applied it to digital watermarking of color images in order to extract the direction vectors needed for vector quantization and chose to adapt the LQIM method to make it able to watermark color images. In terms of invisibility, this method makes it possible to obtain much better invisibility than a non-adaptive method with equal digital distortion. We also see an improvement in robustness. The last section shows the importance of this parameter in an invisibility problem for color watermarking.

4.6. References

Abadpour, A. and Kasaei, S. (2008). Color PCA eigenimages and their application to compression and watermarking. *Image and Vision Computing*, 26(7), 878–890.

Alleysson, D. (1999). Le traitement du signal chromatique dans la rétine : un modèle de base pour la perception humaine des couleurs. PhD Thesis, Joseph Fourier University, Grenoble.

Alleysson, D. and Hérault, J. (2001). Variability in color discrimination data explained by a generic model with nonlinear and adaptive processing. *Color Research & Application*, 26(S1), S225–S229.

Alleysson, D. and Méary, D. (2012). Neurogeometry of color vision. *Journal of Physiology – Paris*, 106(5–6), 284–296.

Assefa, D., Mansinha, L., Tiampo, K.F., Rasmussen, H., Abdella, K. (2010). Local quaternion Fourier transform and color image texture analysis. *Signal Processing*, 90(6), 1825–1835.

Bas, P., Le Bihan, N., Chassery, J.-M. (2003). Color image watermarking using quaternion Fourier transform. In *International Conference on Acoustics, Speech, and Signal Processing*, Hong Kong, 6–10 April.

Carré, P., Denis, P., Fernandez-Maloigne, C. (2012). Spatial color image processing using Clifford algebras: Application to color active contour. *Signal, Image and Video Processing*, 8(7), 1357–1372.

Chareyron, G. and Trémeau, A. (2006). Color images watermarking based on minimization of color differences. In *International Workshop on Multimedia Content, Representation, Classification and Security*, Istanbul, Turkey, 11–13 September.

Chareyron, G., Macq, B., Tremeau, A. (2004). Watermarking of color images based on segmentation of the XYZ color space. In *Conference on Colour in Graphics, Imaging, and Vision*, 1, 178–182 Aachen, Germany, 5–8 April.

Chou, C. and Liu, K. (2010). A perceptually tuned watermarking scheme for color images. *IEEE Transactions on Image Processing*, 19(11), 2966–2982.

Coltuc, D. and Bolon, P. (1999). Robust watermarking by histogram specification. In *International Conference on Image Processing*, 2, 236–239, Kobe, Japan, 24–28 October.

Dadkhah, S., Manaf, A.A., Hori, Y., Hassanien, A.E., Sadeghi, S. (2014). An effective SVD-based image tampering detection and self-recovery using active watermarking. *Signal Processing: Image Communication*, 29(10), 1197–1210.

Denis, P., Carre, P., Fernandez-Maloigne, C. (2007). Spatial and spectral quaternionic approaches for colour images. *Computer Vision and Image Understanding*, 107(1), 74–87.

Ell, T.A. and Sangwine, S.J. (2000). Decomposition of 2D hypercomplex Fourier transforms into pairs of complex Fourier transforms. In *10th European Signal Processing Conference*, 1–4, Tampere, Finland, 4–8 September.

Ell, T.A. and Sangwine, S.J. (2006). Hypercomplex Fourier transforms of color images. *IEEE Transactions on Image Processing*, 1(16), 22–35.

Hu, R., Chen, F., Yu, H. (2010). Incorporating Watson's perceptual model into patchwork watermarking for digital images. In *International Conference on Image Processing*, 3705–3708, Hong Kong, 26–29 September.

Kim, H.-S., Lee, H.-K., Lee, H.-Y., Ha, Y.-H. (2001). Digital watermarking based on color differences. *Security and Watermarking of Multimedia Contents III*, 4314, 10–17.

Koenderink, J., van de Grind, W., Bouman, M. (1970). Models of retinal signal processing at high luminances. *Kybernetik*, 6(6), 227–237.

Kutter, M., Jordan, F.D., Bossen, F. (1997). Digital signature of color images using amplitude modulation. *SPIE*, 3022, 518–526.

Li, Q. and Cox, I.J. (2007). Using perceptual models to improve fidelity and provide resistance to valumetric scaling for quantization index modulation watermarking. *IEEE Transactions on Information Forensics and Security*, 2(2), 127–139.

Li, J., Lin, Q., Yu, C., Ren, X., Li, P. (2018). A QDCT- and SVD-based color image watermarking scheme using an optimized encrypted binary computer-generated hologram. *Soft Computing*, 22, 47–65.

MacAdam, D.L. (1942). Visual sensitivities to color differences in daylight. *Josa*, 32(5), 247–274.

Parisis, A., Carre, P., Fernandez-Maloigne, C., Laurent, N. (2004). Color image watermarking with adaptive strength of insertion. In *International Conference on Acoustics, Speech, and Signal Processing*, 81–85, Montreal, Canada, 17–21 May.

Petitot, J. (2003). The neurogeometry of pinwheels as a sub-Riemannian contact structure. *Journal of Physiology – Paris*, 97(2), 265–309.

Petitot, J. (2008). *Neurogéométrie de la vision : modèles mathématiques et physiques des architectures fonctionnelles*. Éditions de l'École Polytechnique, Palaiseau.

Sangwine, S. (2000). Colour in image processing. *Electronics and Communication Engineering Journal*, 12(5), 211–219.

Sangwine, S. and Ell, T. (2001). Hypercomplex Fourier transforms of color images. In *International Conference on Image Processing*, 137–140, Thessaloniki, Greece, 7–10 October.

Tsui, T.K., Zhang, X., Androutsos, D. (2008). Color image watermarking using multidimensional Fourier transforms. *IEEE Transactions on Information Forensics and Security*, 3(1), 16–28.

Voyatzis, G. and Pitas, I. (1998). Digital image watermarking using mixing systems. *Computers & Graphics*, 22, 405–416.

Wan, W., Zhou, K., Zhang, K., Zhan, Y., Li, J. (2020). JND-guided perceptually color image watermarking in spatial domain. In *IEEE Access*, 8, 164504–164520.

Watson, A.B. (1993). DCT quantization matrices visually optimized for individual images. In *IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology*, San Jose, 31 January–5 February.

Yu, P.-T., Tsai, H.-H., Lin, J.-S. (2001). Digital watermarking based on neural networks for color images. *Signal Processing*, 81(3), 663–671.

Zrenner, E., Stett, A., Weiss, S., Aramant, R.B., Guenther, E., Kohler, K., Miliczek, K.-D., Seiler, M.J., Haemmerle, H. (1999). Can subretinal microphotodiodes successfully replace degenerated photoreceptors? *Vision Research*, 39(15), 2555–2567.

5

Steganography: Embedding Data Into Multimedia Content

Patrick BAS[1], Rémi COGRANNE[2] and Marc CHAUMONT[3]

[1] *CRIStAL, CNRS, University of Lille, France*

[2] *LIST3N, University of Technology of Troyes, France*

[3] *LIRMM, Université de Montpellier, CNRS, University of Nîmes, France*

This chapter presents the basic concepts in steganography. These concepts most often use digital images as a medium, but many of them can be applied to other content coming from sensors, such as sounds or videos. After a reminder of the theoretical foundations linked to steganography, we will present the fundamental principles of the field and then detail the basic methods, using image in its spatial format[1] or JPEG[2].

The second part of this chapter will present more advanced concepts in steganography; these concepts either allow an increase in the security, or take other practical contexts into account (e.g. the steganography of a group of images, steganography of color images or the use of a high-resolution image during embedding).

For a color version of all figures in this chapter, see www.iste.co.uk/puech/multimedia1.zip.

1 In other words, where each pixel is coded by a gray level or by three color channels.

2 Here, the image is encoded as a set of DCT coefficients, see https://fr.wikipedia.org/wiki/JPEG.

Multimedia Security 1,

coordinated by William PUECH.

Multimedia Security 1: Authentication and Data Hiding,
First Edition. William Puech.

5.1. Introduction and theoretical foundations

Steganography seeks to modify the content of a document (most commonly called a "*cover*" document) in order to embed a message that is *undetectable*, by producing a steganographied document called a "*stego*" . The practical context of the use of steganography, presented in Simmons (1984) as "The Prisoners' Problem", is presented as follows: Alice seeks to transmit sensitive information to Bob over a benign communication channel (e.g. sending a vacation photo attached to an email). The adversary, usually called Eve[3], is able to detect the use of steganography methods by using steganalysis methods[4].

The rise of steganography and steganalysis came after September 11, 2001, when American newspapers reported that the Al Qaeda group could use steganography methods to communicate within their network. History has also shown that steganography was used in particular for various sensitive, and most often malicious activities, such as:

– terrorist networks[5];

– pedophile networks, to hide images within images[6];

– "botnet" networks, to send commands to slave computers without being blocked by firewalls (Pevnỳ *et al*. 2016).

More generally, steganography can be used as a solution for communication, without a hint of suspicion.

In the academic field, steganography and steganalysis have also benefited from the enthusiasm at the beginning of the 21st century for methods that make it possible to hide data ("data-hiding"). Such methods can also be used to address copyright issues in digital content by linking the embedded message to the owner of the image. For more than 20 years now, steganography and steganalysis have been well-established disciplines within the broader field of information security.

In this chapter, we show key concepts in steganography using digital images as media, however many of the concepts presented can be used for other media acquired by sensors, such as sound or videos. The different building blocks of steganography are illustrated in Figure 5.1.

3 For "eavesdropper", a secret listener to private conversations.

4 These methods are described in Chapter 6.

5 See: www.unodc.org/documents/frontpage/Use_of_Internet_for_Terrorist_Purposes.pdf.

6 See: http://news.bbc.co.uk/2/hi/science/nature/2082657.stm.

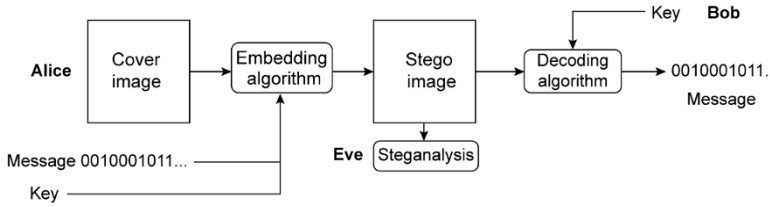


Figure 5.1. *General operation of steganography and roles of different actors (Alice, Bob and Eve)*

In general, a steganography algorithm must face two constraints:

1) the embedding of the message must be *undetectable*; in other words, Eve must not be able to detect either the use of the algorithm, or the embedding of a message;

2) the algorithm must also *maximize the embedding capacity*; in other words, the size of the embedded message.

In the remainder of this chapter, we will focus our attention on steganography of digital images, compressed in the JPEG format or not.

First, we will present the fundamental principles in steganography (section 5.2), we will then offer a concise overview of the basic methods in steganography (section 5.3), and finally conclude with a presentation of advanced methods (section 5.4).

5.2. Fundamental principles

The theoretical contributions that make it possible to satisfy the constraints presented above generally use information theory, coding theory and image analysis.

5.2.1. *Maximization of the size of the embedded message*

The embedding capacity, that is, the *maximum size of the message* that the steganography algorithm is likely to embed in the cover content, is calculated using the *Source Coding Theorem* by Shannon (1948). This theorem can be presented as follows.

Let:

– x_i, a sample i of cover image[7];

7 By sample, we mean the value of a pixel of an image for one of the three color channels if it is coded without loss, or the value of a DCT coefficient for the JPEG format.

– a steganographic signal associated with sample i is defined by $s_i \in \{K_{\min}, \ldots, K_{\max}\}$, with $\{p_i(K_{\min}), \ldots, p_i(K_{\max})\}$, the $(K_{\max} - K_{\min} + 1)$ modification probabilities associated with each of the values of s_i. Note here that for most relatively secure schemes, embedding only allows binary modifications (so $K_{\min} = -1, K_{\max} = 0$ or $K_{\min} = 0, K_{\max} = +1$) or ternary ($K_{\min} = -1$, $K_{\max} = +1$);

– an additive embedding scheme, where the sample content stego is given by $y_i = x_i + s_i$.

Therefore, the *Source Coding Theorem* states that the amount of maximum information supported by the sample y_i is given by the entropy of the random variable S_i, so:

$$H(S_i) = -\sum_{k=K_{\min}}^{K_{\max}} p_i(k) \log_2 p_i(k) \,\text{bits} \qquad [5.1]$$

The informed reader will note that this entropy does not depend on the host sample x_i. This is justified by the fact that the host content is known by Alice, who can therefore design an encoding system that does not have to take the signal into account, even if it remains unknown to Bob. This result is justified by the results on the consideration of side information, as presented in Costa (1983).

From a practical point of view, taking the entropy calculation into account is essential. Take an example where Alice tries to embed 100 bits in 1,000 pixels of a grayscale encoded image (each pixel has a value between 0 and 255). If Alice uses the basic least-significant-bit-based substitution method (also called LSB substitution[8], see section 5.3.1), on average, 50 pixels are modified like this and the probability of modification of a pixel is therefore 50/1,000 = 0.05.

The source coding theorem shows that for the same average number of modifications, the maximum message size that can be embedded using an encoding method, which is less naive than LSB substitution, is equal to:

$$1000\,(-0.05\log_2(0.05) - 0.95\log_2(0.95)) \approx 286\,\text{bits}$$

so by going from 100 to 286 bits, the length of the embedded message is almost three times greater than that of the embedded message by LSB substitution.

8 Least-significant-bit-based substitution methods (also see section 5.3.1) are basic methods where in order to embed a message of N bits, the least significant bits of N pixels selected with the secret key are directly replaced by the message to transmit.

5.2.2. *Message encoding*

Finding a coding method that could be used in steganography and that was close to Shannon's entropy was not an easy task.

The coding methods with linear codes proposed very early on by Crandall (1998) have made it possible to increase the embedding capacity (for a given average number of modifications) without really approaching the theoretical limit. Just like more advanced methods, these methods are based on the idea that there are several stego documents that are coding the same message, and that sensible coding involves selecting the stego content which is the closest to the cover content (see Figure 5.2).

Figure 5.2. *Coding for steganography: the coding system generates several code words associated with the same message, m, in order to select the one that is closest to the cover content, and so minimizes the distortion between the cover content, and the stego content*

Let us take the toy example of a cover content composed of 3 randomly drawn bits $[x_1, x_2, x_3]$, and its stego counterpart $[y_1, y_2, y_3]$, and assume that the coding system is looking to embed 2 bits by minimizing the average number of modifications. The LSB substitution method[9] involves selecting two of three bits with a secret key and replacing them with the message to embed. Therefore, on average, each bit has a probability of $\frac{1}{2} \times \frac{2}{3} = \frac{1}{3}$ of being modified. Now, if we use linear coding, which involves acknowledging that the first bit is coded[10] by $m_1 = y_1 \oplus y_2$, and the second bit is coded by $m_2 = y_2 \oplus y_3$; then x_2 is only modified if it allows both values of $x_1 \oplus x_2$ and $x_2 \oplus x_3$ to be changed, that is, with a probability of 0.25. Furthermore, x_1 (respectiely, x_3) is only modified if it allows the value of $x_1 \oplus x_2$ (respectively, $x_2 \oplus x_3$) to change, that is, with a probability of 0.25 for each, and also a probability of 0.25 that the 2 bits to be embedded are already present in the cover content. In the end, the probability of modification of each bit is only 0.25, compared to 0.33 for the

9 See, once again, section 5.3.1.

10 With $\oplus$ being the operator XOR.

LSB substitution scheme. Note once again that the source coding theorem says that for a modification probability of 0.25, the maximum size of the embedded message is $-0,25\log_2(0,25) - 0,75\log_2(0,75) \approx 0,81$ bit embedded per modified sample, instead of 0.75 bit per sample, as proposed by the method described.

However, Filler *et al*. (2011) suggested a convenient way to get closer to this maximum limit by designing codes using a coding system by syndromes, and using a trellis. These codes are called *Syndrome Trellis Codes* (STC), they combine the principle of syndrome coding and a sequential optimization algorithm, which allows us to embed the desired message while minimizing the number of modifications required (the Viterbi algorithm which follows the trellis). Note that this coding system can be used more generally with any additive distortion[11] between the cover image and stego image. We do not go into detail about the operation of the STC, but the reader can remember that in the case of an additive distortion, the performances of the STC are very close to the maximum bound, given by the Shannon entropy.

5.2.3. *Detectability minimization*

As specified in the Introduction, a steganography algorithm will only be secure if it is undetectable, which will happen if Eve does not suspect the presence of a hidden message when analyzing stego content.

It is unrealistic to consider that detectability is directly proportional to the number of modifications made to the cover image to embed the message. For example, a modified pixel in a noisy part of the image (for example a texture) will help to increase the detectability of the image much less than a pixel modified in a standard part of the image, or even in the worst case, a constant area. In fact, it is easy to see that Eve will have a much easier time finding a suspicious modification if it appears in an area with weak fluctuations (or even zero when the pixel levels are saturated) than in an area that difficult to model. The detectability measurement therefore cannot be a simple Euclidean distance between the cover image and the stego image, and we need to analyze the cover image beforehand in order to be able to associate each sample of the image with its own empirical detectability.

The theoretical formalization adopted in most cases to minimize detectability is the use of *additive embedding costs*. Each sample of the cover image x_i is linked to a cost ρ_i^k for each modification k (this comes down to considering that each sample s_i of the steganographic signal is drawn independently of the others). The costs are additive since the modification of a sample following embedding does not lead to a

11 See section 5.2.3 for a more precise definition of distorsion.

modification of the costs related to the other samples[12]. The mathematical expectation of the distortion D between the cover image and the stego image is therefore given by:

$$\mathrm{E}[D] = \sum_{i,k} p_i^k \rho_i^k \qquad [5.2]$$

Due to the lack of space, we cannot go into detail about the calculations that highlight the relation between the modification cost ρ_i^k and the probability of modifying each sample p_i^k. However, it is important to remember that a Lagrangian formulation[13] makes it possible to minimize the distortion given in equation [5.2] while respecting the constraint linked to entropy (equation [5.1]), in order to express a clear relationship between these two characteristics. As an example, in the case of binary embedding, and considering that the cost associated with no modification is zero, the relationship between the cost of a modification ρ_i and its probability p_i is given as:

$$\rho_i = \lambda \log \frac{1 - 2p_i}{p_i} \qquad [5.3]$$

where λ is a constant depending only on the size of the message to embed.

It is important to note that in practice, these costs ρ_i^k can be used directly by the STC in order to minimize a distortion which, when the number of samples is large, is very close to D.

If this protocol follows a rigorous theoretical framework, the calculation of costs ρ_i^k from the cover image is completely different. In fact, this calculation involves using heuristic functions based on the rational that a cost must be high if the value of the cover sample can be easily predicted from its surroundings; on the contrary, a cost must be low if this same prediction becomes difficult; in other words, if the sample is located in a noisy area.

The majority of implementations in steganography, therefore, seek to first calculate the cost ρ_i^k for each sample i relative to each modification k, then second, use STC coding to embed the desired message, while minimizing the sum of costs relating to embedding. The breakdown of these different stages is illustrated in Figure 5.3.

12 Non-additive costs can, however, be considered via synchronization methods (see section 7.3).

13 See: https://en.wikipedia.org/wiki/Lagrange_multiplier.

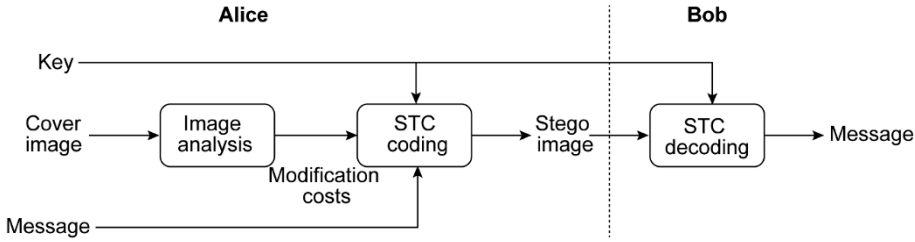


Figure 5.3. *General principle of embedding, encoding and decoding in steganography. The secret key can be used to encrypt the message, permute samples of cover content and/or configure the STC*

5.3. Digital image steganography: basic methods

In this section, we go into detail about the different implementation methods in steganography, which mainly differ by the function used to calculate the costs.

5.3.1. *LSB substitution and matching*

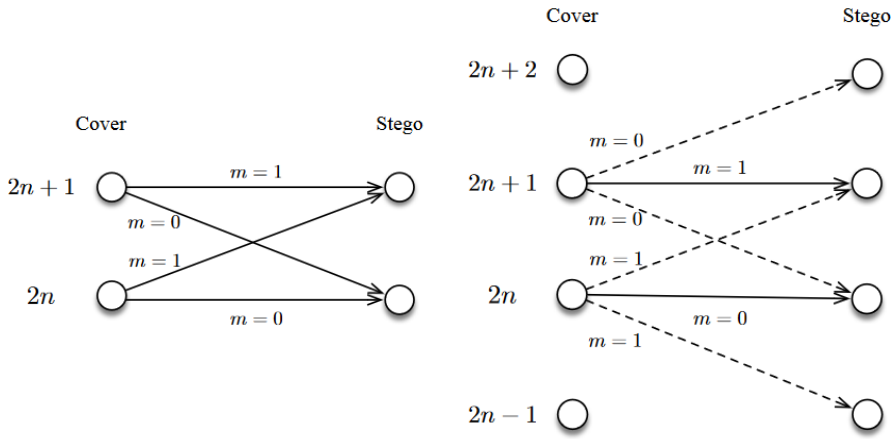
The substitution of the least significant bits (also called LSB substitution for *Least Significant Bits*) is an extremely naive method, which involves replacing the least significant bits of a pixel or DCT coefficient with the bit to be embedded. The least significant bit being the parity bit, this embedding therefore amounts to applying the rule specified in Figure 5.4(a). The decoding of the message embedded by Bob is done very simply by reading the parity bits of the samples that have the message[14].

Even though this steganography method is very simple, it has two major drawbacks:

1) as shown in section 5.2.1, coding by substitution is clearly suboptimal;

2) the naive selection of the samples carrying the message also makes this method very detectable, and there are even steganalysis methods that make it possible to easily estimate the size of the message embedded by the LSB substitution, as in the example presented in Dumitrescu *et al*. (2003) or in section 8.3.2 of Chapter 8. An alternative to this embedding strategy involves using a random number by performing an operation of type $+1$ and -1 in an equiprobable way to change the value of the least significant bit: this is the principle of LSB matching, which is illustrated in Figure 5.4(b), and its steganalysis is studied in section 8.3.3 of Chapter 8.

14 These samples can be chosen by Alice and Bob by using a same secret key.



(a) LSB substitution (b) LSB correspondence

Figure 5.4. *Principles of LSB substitution and LSB correspondence, the dotted arrows represent a modification performed with a probability of 0.5,* $n \in \mathbb{N}^+$

5.3.2. *Adaptive embedding methods*

We now present *adaptative* steganography methods, that is, methods that analyze the cover image beforehand in order to associate a detection cost with each sample. These methods all use the rational so that a sample (pixel or DCT coefficient), which is located in a complex or noisy area of the image, will be associated with a lower cost (since the sample will be difficult to predict) than a coefficient found in a simple or not very noisy area of the image.

The first two methods presented operate in the spatial domain, and the third operates in the JPEG domain.

5.3.2.1. *Example of costs in the pixel domain (HILL scheme)*

The HILL scheme is remarkable for its simplicity and efficiency. The creators who proposed this algorithm (Li *et al.* 2014) started from the principle that for an image encoded in the spatial domain, the pixels belonging to textured or noisy areas had to be associated with a low modification cost, while the pixels belonging to standard zones had to be associated with a high cost. For this scheme, the characterization of noise is simply done by using a high-pass filter $\mathbf{H}$ (differentiating filter) and two low-pass filters, $\mathbf{L}_1$ and $\mathbf{L}_2$, which are also used to take into account the variations

in the surroundings of the pixel considered. The calculation of the cost map ρ for an image $\mathbf{I}$, coded in grayscale, is then the result of three 2D convolutions:

$$\rho = \frac{1}{|\mathbf{I} * \mathbf{H}| * \mathbf{L}_1} * \mathbf{L}_2 \qquad [5.4]$$

with:

$$\mathbf{H} = \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}$$

and $\mathbf{L}_1$ and $\mathbf{L}_2$ are two averaging low-pass filters of sizes 3×3 and 15×15, respectively.

Figure 5.5 shows the associated cost image for a given grayscale image. We can see that the costs located in the textured areas are much lower than the costs located in the standard areas.

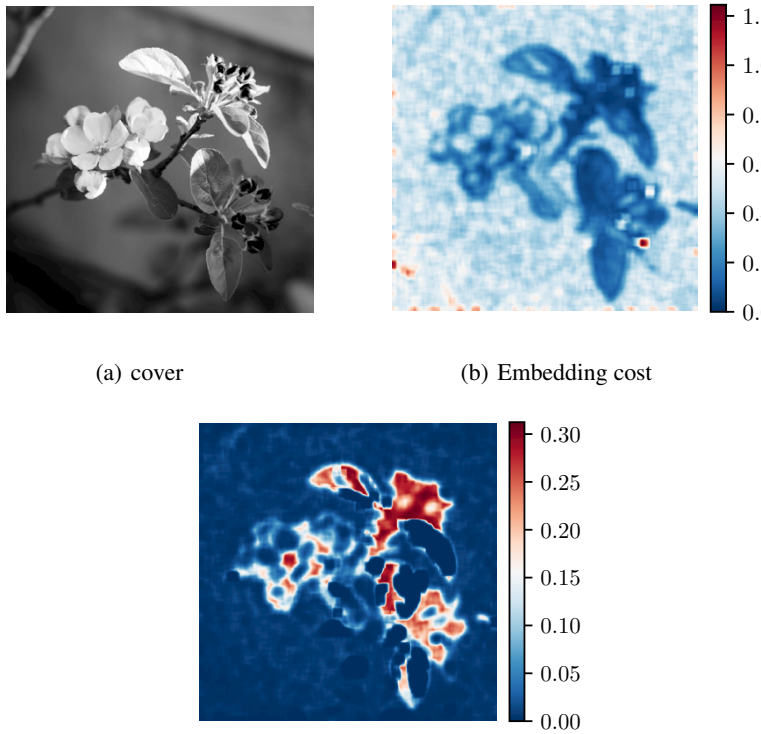
5.3.2.2. *Cost taking into account detectability (MiPod scheme)*

MiPod is a much less heuristic method that involves matching a statistical detectability to each cost (Sedighi *et al*. 2016). The statistical detectability of a modification can be associated with a quantity, called deflection, which is linked to the probability that the image or pixel considered belongs to the cover class or the stego class. The greater the deflection, the more important the evidence added to the stego hypothesis. Assuming that each pixel before quantization can be considered as a continuous random variable following a Gaussian law of variance σ_i^2, and that the probability of performing an operation ± 1 during embedding is equal to p_i, then the cost is given by:

$$\rho_i = \frac{p_i}{\sigma_i^4} \qquad [5.5]$$

with the square of the deflection being equal to p_i^2/σ_i^4 for each sample.

It is important to note that the cost does not only depend on the properties of the image, such as the variance σ_i^2 (directly estimated from the local variance of each pixel of the image), but also on the modification probability p_i; equation [5.3] is therefore not directly suitable for embedding, but the resolution of the two constraints presented in section 5.1 remains possible. We can also see that the cost decreases when σ_i^2 increases (the noise associated with each pixel becomes greater and greater), or when p_i decreases (the probability of modifying the pixel decreases). Figure 5.6 shows the modification probabilities of each pixel for two different embedding rates.



(a) cover

(b) Embedding cost

(c) Embedding probabilities (0.3 bit per pixel)

Figure 5.5. *a) Cover Image. b) Associated cost map for the HILL algorithm. Image textures are associated with the lowest costs. c) Modification probability map, the maximum probability being equal to 1/3 for a ternary embedding and the maximal embedding rate at* $\log_2(3) \simeq$ *1.6 bit per pixel*

5.3.2.3. *Example of costs in the JPEG domain (UERD scheme)*

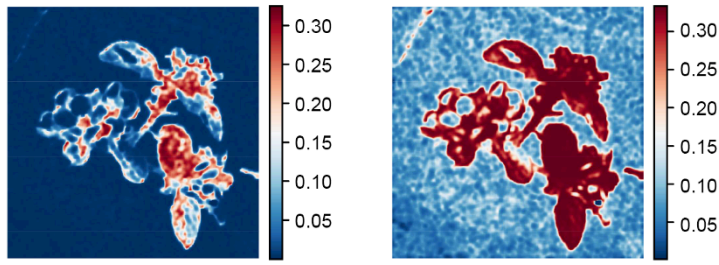
The cost calculation proposed by the UERD scheme (Guo *et al.* 2015) is carried out for cover images compressed in JPEG, so it is necessary to associate a cost with each quantified DCT coefficient c_i of the image (q_i being the associated quantization step). The authors of this scheme intend to ensure that the coefficients belonging to textured blocks[15] are associated with a low cost (high, respectively). Measuring "texture" is simply performed by calculating the energy $D_{\mathcal{B}}$ of the 8×8 $\mathcal{B}$ JPEG block containing c_i. This energy is defined by $D_{\mathcal{B}} = \sum_{i \in \mathcal{B}} q_i |c_i|$, and we can see that

15 That is, containing many non-zero DCT coefficients.

the constructed cost is proportional to the quantization step q_i, in order to show the fact that an operation of type $(+1)$ on a quantized coefficient results in a difference of q_i on a DCT transform of the image. Apart from the DC coefficients that have a slightly different cost, the cost ρ_i linked with each coefficient c_i is therefore given by:

$$\rho_i = \frac{q_i}{D_{\mathcal{B}} + 0.25\left(D_{\mathcal{B}}^{\leftarrow} + D_{\mathcal{B}}^{\rightarrow} + D_{\mathcal{B}}^{\downarrow} + D_{\mathcal{B}}^{\uparrow} + D_{\mathcal{B}}^{\swarrow} + D_{\mathcal{B}}^{\searrow} + D_{\mathcal{B}}^{\nwarrow} + D_{\mathcal{B}}^{\nearrow}\right)} \quad [5.6]$$

The second value of the denominator makes it possible to take into account the energies of the eight blocks related to the block considered.



(a) 0.3 bits per pixel (b) 1.0 bit per pixel

Figure 5.6. *Map of associated modification probabilities for the MiPod algorithm for two different embedding rates*

5.4. Advanced principles in steganography

After having presented the basic principles of steganography (coding and cost calculation), we now present different ways which allow further improvements of the security of an embedding scheme, or allow it to adapt to different contexts. We would like to point out to the reader that these recent advances often remain associated with areas where research will probably lead to further developments in the future. These advanced principles are listed as follows:

– the *synchronization of modifications*, which makes it possible to correlate the surrounding modifications of the image;

– *steganography of color images*, which applies the methods used on grayscale images to three-component images;

– *batch steganography*, which allows the distribution of the message within several images;

– the *use of side information*, which takes errors related to quantization into account, in order to improve the security of the scheme;

– *steganography mimicking a statistical model*, which reinforces security by making the embedding truthful to a statistical model of the image;

– *adversarial steganography*, which uses an embedding that directly seeks to bypass the steganalyst;

– finally, the conclusion highlights the need to develop *robust steganography* methods to transcoding.

5.4.1. *Synchronization of modifications*

In order to decrease the detectability of an embedding scheme, a possible strategy is to synchronize the modifications. If, for example, in the spatial domain, a pixel has been modified by an operation of type $(+1)$, it may be a good idea to favor an embedding of type $(+1)$ rather than $(+0)$ or (-1) on a neighboring pixel, in order to favor the appearance of a stego signal whose variations are similar to those of the image. Therefore, it is necessary to make the calculation of the cost of a coefficient dependent on the modifications already carried out in its surroundings. This is the principle of the synchronization of modifications.

A general way to carry out this synchronization is to break down the natural sampling grid of the coefficients that code the image[16] into a group of separated grids[17], such that for a given lattice, each coefficient can be modified independently from the others, that is, using an additive cost. This principle is illustrated in Figure 5.7 where four lattices $\{\Lambda_1, \Lambda_2, \Lambda_3, \Lambda_4\}$ are used.

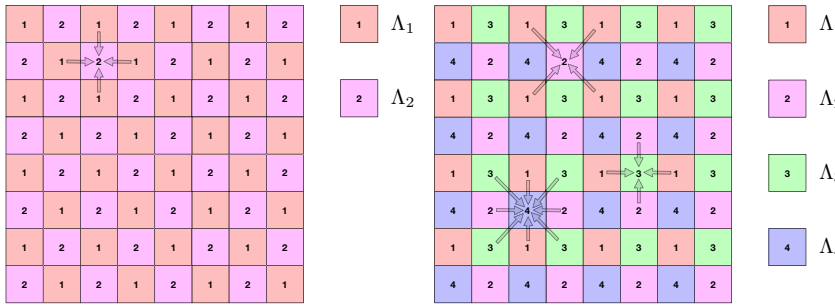
It is important to distinguish two main classes of synchronization methods:

1) heuristic methods, which increase or decrease the costs related to embedding. The vast majority of methods belong to this class. For example, if in the spatial domain a + 1 has been carried out on a pixel, the cost relative to the modification $(+1)$ on a related pixel will be reduced; on the other hand, a cost relating to a modification (-1) will be increased;

2) methods based on probabilistic models of the steganographic signal. Here, the costs come directly from the conditional probabilities specified above, and then apply the relationships between the costs and the modification probabilities.

16 For example, the natural grid of the pixels of the image in the spatial domain, or a matrix of DCT coefficients in the JPEG domain.

17 Called *lattices*.



(a) Embedding on 2 lattices (dependency on 4 connections)

(b) Embedding on 4 lattices (dependency on 8 connections)

Figure 5.7. *Principle of synchronized embedding using two or four lattices*

COMMENT ON FIGURE 5.7.– *The costs or the probabilities of embeddings are first calculated on lattice* Λ_1*, where the modifications are assumed to be independent. The embedding is then carried out on* Λ_1*, then the costs or the probabilities of embeddings are calculated on lattice* Λ_2*, taking into account the modifications already made on* Λ_1*. When four lattices are used, the algorithm then iterates up to the lattice* Λ_4*, where the costs are calculated from the modifications made to* Λ_1*,* Λ_2 *and* Λ_3*. The arrows represent the connectivity relationships used to account for dependencies and update costs.*

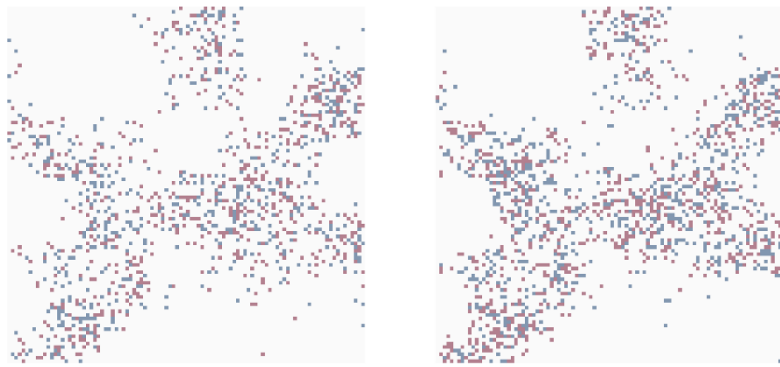
In order to illustrate an implementation belonging to the first class method, we briefly present the method proposed by Li *et al.* (2015) via the synchronization scheme *Clustering Modification Directions* (CMD). This scheme can be used, for example, with the cost proposed by HILL in section 5.3.2.1. Four lattices are used (see Figure 5.7(b)). Without loss of generality, here we consider a modification of type $(+\ 1)$, where μ_{ij} is the average of the modifications already made in the surroundings of the pixel (i, j) on previous lattices; the new cost ρ'_{ij} is given by :

$$\rho'_{ij}\left(+1\right) = \frac{1}{9}\rho_{ij}\left(+1\right), \text{ if } \mu_{ij} > 0 \qquad [5.7]$$

$$\rho'_{ij}\left(+1\right) = \rho_{ij}\left(+1\right), \text{ if } \mu_{ij} \leq 0 \qquad [5.8]$$

Note that the weight 1/9 is arbitrary and is used to maximize the performance of the scheme. The embedding stops once all four lattices have been searched. Figure 5.8 shows the effect of synchronization on the modifications $(-\ 1/+\ 1)$ made to the image, and shows how this strategy helps facilitate related modifications. The gain linked to synchronization, in terms of practical security, is significant, since the probability

of detection error for the steganalyst can increase by 5%, with relation to the HILL scheme for a embedding rate of 0.4 per pixel.



(a) HILL (b) CMD

Figure 5.8. *Visualization of modifications (− 1, + 1) made to the image without synchronization using a) the HILL method and b) the CMD strategy*

COMMENT ON FIGURE 5.8.– *We can see that changes of the same sign are more related when synchronization is used. Also note that even if the security of the schema is increased, the number of modifications still increases by 20% on average after synchronization.*

5.4.2. *Batch steganography*

Batch steganography, introduced by Ker (2007), considers a scenario where Alice does not have a single image to embed her message, but several images. Unlike a classical steganography scenario where Alice only transmits one image to Bob, here Alice will have to distribute the message in each of the images making up the group. The enemy, Eve, could design an analysis method using the group of images sent by Alice (which leads to group steganalysis, or pool steganalysis, see section 8.6.6 of this work).

The main part of the problem with batch steganography is therefore how Alice distributes her message in different images. From a practical point of view, this scenario is explained in at least two situations: when Alice transmits several images to Bob at once and when Alice sends images to Bob sequentially. Note that in the first case, the size of the message transmitted will be known, while in the second

case, Alice may have to adjust the total size of the message to be transmitted to suit her communication needs. Without going into detail in terms of steganalysis (this is not the objective of this chapter), it should be noted that the more content transmitted by Alice to Bob that Eve accumulates, the better she will be able to perform an accurate analysis to decide whether or not Alice is transmitting stego data to Bob.

The main message distribution methods studied in the literature (Pevnỳ and Nikolaev 2015; Cogranne *et al*. 2017; Zakaria *et al*. 2019) are described as follows:

– the transmission of the message in the smallest possible number of images, and the other images of the group therefore belonging to the cover class;

– the distribution of the message evenly within the group; this uniformity can result in:

- an identical message size for each image,
- an identical embedding rate for each image[18];

– but also distribution strategies taking into account the content of the images:

- the cost of modification (see equation [5.2]) is constant for each image,
- the statistical distortion (see equation [5.5]) is constant for each image,
- the overall cost minimization applied to all the images. Note that this strategy is only possible if Alice first has all of the cover images to transform into stego images, while this is not the case for the two previous strategies.

If the conclusions related to the safest strategy are not developed enough to be definitive yet (they depend on Eve's steganalysis strategy), some interesting results emerge:

– in Cogranne *et al*. (2017), the overall cost minimization provides the most secure strategy for an all-knowing steganalyst, who would know the distribution strategy;

– when this strategy needs to be estimated (Zakaria *et al*. 2019), constant statistical distortion would maximize practical security.

Note that in the cited references, neither transmitting the message in as few images as possible nor the strategy of evenly distributing the message across the group appears to be the best option.

18 In the case of images in the JPEG format, with the embedding rate often being expressed in bits, not non-zero AC coefficients; this is not equivalent to an identical size.

5.4.3. *Steganography of color images*

The methods of the literature propose to embed the message in a grayscale image in the majority of cases, and a really fast shortcut would involve developing a steganography method for color images[19], by inserting the same amount of data in each channel independently. This reasoning is flawed for several reasons:

1) the color components are not independent, this dependency is important for red/green/blue channels and much less important for the luminance/red chrominance/blue chrominance channels[20] used by JPEG coding;

2) the steganalyst has many more samples to carry out their analysis (at the most, three times more data), and the results from Ker *et al*. (2008) show that at equal detectability, the size of the embedded message must theoretically change with the square root of the number of samples for naive coding;

3) the content of each of the channels can be very different from one channel to another: while it is often very similar for RGB images, for JPEG images coded in luminance/red chrominance/blue chrominance, the perceptual content is much more significant on the luminance component than on the two chrominance components. So it is suitable to embed less data on these two components.

A slightly more thorough way of approaching the problem of color steganography is to understand it as a problem of batch steganography (see section 5.4.2), where Alice has to distribute the message within each of the three channels.

An approach like this was developed by Cogranne *et al*. (2020) in the case of JPEG images. The different strategies mentioned in section 5.4.2 are applied, and the authors find that the strategy minimizing the statistical distortion is most often the most effective. An alternative proposed by Taburet *et al*. (2018) consists of setting the proportions of the message to be distributed among the three components. Practical security is maximized when 80% of the message is embedded into the luminance component, while no subsampling is applied to the chrominance channels[21]. Finally, when the images are coded in the spatial domain, the work by Wang *et al*. (2019) has shown the importance of being able to synchronize the modifications, both between the neighboring pixels and the color components.

This scheme intends to first carry out embedding on the green channel of the image, and then use the CMD algorithm (see section 5.4.2) to update the costs on the red and blue channels, according to the modifications made on the green channel.

19 That is, coded on three components.

20 See: https://fr.wikipedia.org/wiki/Chrominance.

21 For more information on color subsampling, see: https://en.wikipedia.org/wiki/Chroma_subsampling.

The green channel is chosen first since it is the one that is the most correlated with the other two channels. The updating rules are identical to the CMD rules presented in equations [5.7] and [5.8], but the term $\mu_{i,j}$ now takes the modification carried out in the green channel into account. Compared to the naive HILL algorithm, the gain in terms of practical security is substantial (the error rate increases by more than 10%) when the embedding rates are about 0.4 bit per pixel.

5.4.4. *Use of side information*

Side information embedding methods use a reliable measure of undetectability based on the quantization error produced when transforming the image, such as JPEG compression, a geometric or colorimetric transformation followed by a recompression. The term "side information" comes under "the pre-cover image", that is, the image before it is transformed into a cover image, and which is encoded, either in the continuous domain or in a high-resolution domain (e.g. 16 bits for the pre-cover image versus 8 bits for the cover image).

In the case of binary embedding, intuitively it is quite easy to realize that if the coefficient before quantization is almost equidistant from two quantization cells (case (a) in Figure 5.9), then this coefficient can be modified equally toward either of the cells and in this case, the modification will be undetectable. However, if the coefficient happens to be very close to a quantization cell (case (c) in Figure 5.9), then modifying this coefficient toward a cell further away will be more detectable.

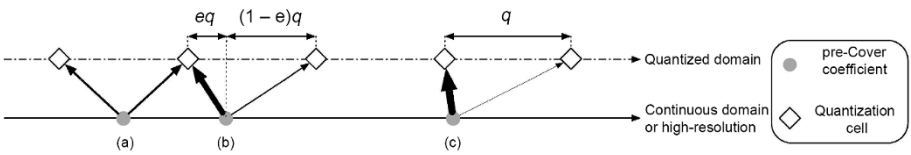


Figure 5.9. *Principle of weighting by the quantization error: situation (a), where the unquantized value is equidistant from two quantized values, will be more favorable to the modification of the value toward the surrounding quantized area than situation (c), where the unquantized value is very close to the quantized value. Situation (b) is intermediate*

We can cite several applications, in chronological order, which use the concept of side information.

In the case of double JPEG compression, the *Perturbed Quantization* (PQ) algorithm (Fridrich *et al*. 2004, 2005) suggested using *wet-paper codes* to modify only the DCT coefficients, such as $(\frac{1}{2} - e) = 0$, where e represents the quantization error associated with the quantization step q (see Figure 5.9). Filler *et al*. (2011)

revisits this statement by specifying that the cost must be equal to $\rho_{SI} = (\frac{1}{2} - e)$ for the non-zero coefficients, which do not change. This strategy is called S_1.

The initial article presenting STC offered by Filler *et al.* (2011) also proposes three other strategies called S_2, S_3 and S_4. In each case, the zero coefficients are not modified. Also note that the costs only depend on the pre-cover content (i.e. before quantization), unlike the adaptive schemes that were presented before.

– For S_2, $\rho_{SI} = (\frac{1}{2} - e)q$.

– For S_3, $\rho_{SI} = 1$ if the quantified value is equal to –1 or 1 and $\rho = 2\left(\frac{1}{2} - e\right)$ otherwise.

– For S_4, $\rho_{SI} = q$, if the quantified value is equal to –1 or 1 and $\rho = 2\left(\frac{1}{2} - e\right)q$ otherwise.

Among the four strategies, S_1, S_2, S_3 and S_4, Filler *et al.* (2011) show that it is strategy S_4 that is the least detectable.

The article presented by Denemark and Fridrich (2015) reviews the use of side information for modern adaptive embedding algorithms S-Uniward (Holub *et al.* 2014), HILL and J-Uniward (Holub *et al.* 2014). For each of the algorithms, binary and ternary embeddings are considered. For binary embedding, the relationship between adaptive cost ρ_{Ad} and the new cost is given as:

$$\rho_{SI} = 2\left(\frac{1}{2} - e\right)\rho_{Ad} \qquad [5.9]$$

We can see here that the quantization step q has been replaced by the adaptive cost ρ_{Ad}, which amounts to considering that this implementation seeks to minimize the difference between the sum of costs produced by the conversion from pre-cover content to cover content. Also note that factor 2 does not change the optimization process that involves minimizing the sum of costs.

For a ternary embedding, the authors of the same article propose two different variations. Without loss of generality, if the unquantified value is closer to the lower value than to the upper value (which is the case, for example, (b) in Figure 5.9), then the (–1) operation does not take side information into account (it is an arbitrary choice, perhaps justifiable by experimentation):

$$\rho_{SI}^{(-1)} = \rho_{Ad} \qquad [5.10]$$

while the operation (+1) considers:

$$\rho_{SI}^{(+1)} = 2\left(\frac{1}{2} - e\right)\rho_{Ad} \qquad [5.11]$$

It should be noted that most often the use of side information makes it possible to considerably increase the performance of the embedding scheme in terms of security. For example, the SI version of J-Uniward for a quality factor of 75% allows the increase of the error rate in steganalysis by more than 20% for an embedding rate around 0.4 bit per non-zero AC coefficient.

5.4.5. *Steganography mimicking a statistical model*

Another relatively intuitive strategy for hiding information is camouflaging (unlike methods that use a cost measuring statistical distortion); here the idea is to directly mimic the statistical properties of the cover image. As an illustrative example, if the cover content can be completely characterized by a given distribution $f()$, it is "enough" to produce stego content, also following the distribution $f()$.

Such an operation, however, is only conceivable when we can characterize $f()$ clearly. If not, the embedding inevitably becomes detectable. In this section, we present two examples based on the principle of mimicking, one relatively old and the other more recent.

5.4.5.1. *Distribution of DCT coefficients*

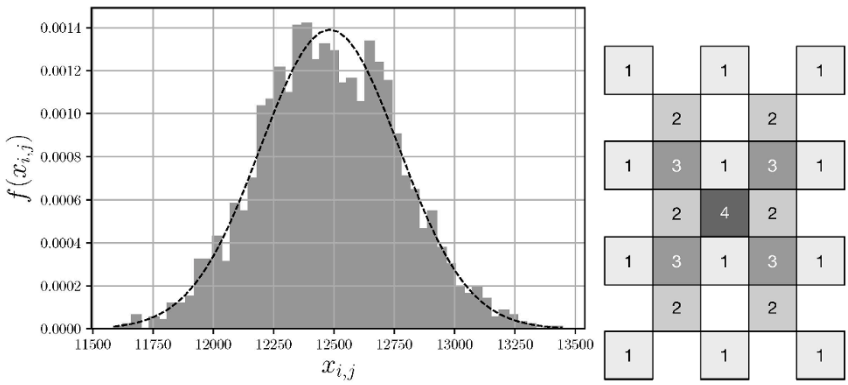
Sallee (2003) proposes to mimic the distribution of the DCT coefficients of a JPEG image. The chosen distribution $f()$ is a generalized Cauchy distribution of type $f(x) = \frac{p-1}{2s}(|x/s| + 1)^{-p}$, where p and s are two parameters to estimate. The author assumes that the coefficients are statistically independent of each other. The "mimicking" is carried out by ensuring that the distribution of the coefficients, after setting the least significant bits to zero, remains unchanged. The coding is based on an arithmetic decoder that makes it possible to embed a message while respecting, at best, the probabilities of appearance of each of the values of the coefficients. The only constraint linked to this type of coding is that the histogram of coefficients, with their least significant bits at zero, must remain unchanged to be able to build a decoder, which is possible for this implementation, but can be difficult in other circumstances.

This method was pioneering, while also offering advanced concepts in steganography at the same time (imitation of a model, advanced coding) and it has remained a reference in steganography for a long time. However, it has become relatively detectable by steganalysis methods that capture the dependencies between the DCT coefficients, since they are not preserved.

5.4.5.2. *Photonic noise*

Much more recently, Bas (2016) and Taburet *et al*. (2020) sought to imitate the statistical model of photonic noise, either regarding the pixels of the image or the DCT coefficients. Here, steganographic embedding (called "natural steganography") does not mimic the statistical model of the cover image, but instead mimics the statistical model of a cover image that would have been acquired with a higher ISO sensitivity than that of the cover image.

The statistical distribution of this photonic noise depends on the processing chain that makes it possible to switch from the RAW image saved by the sensor to the cover image (coded in the spatial domain or in the JPEG domain). In the RAW domain, the statistical model is simple since the photonic noise applied to each photo-site is independently distributed according to a normal distribution $\mathcal{N}(0, a\mu + b)$, where μ represents the noise-free value of the photo-site and (a, b) are the constants linked only to the sensor and parameter ISO_1 (also see Figure 5.10(a)). In the RAW domain, the steganographic signal added to the photo-sites is also distributed by normal distribution $\mathcal{N}(0, a'x + b')$, where x is the value of the observed photo-site and (a', b') are chosen to mimic a sensitivity $\mathrm{ISO}_2 > \mathrm{ISO}_1$. Assuming that $x \simeq \mu$, the image seems to have been corrupted by a photonic noise of distribution $\mathcal{N}(0, (a + a')\mu + b + b')$.



(a) Standardized histogram of photonic noise (Leica Monochrome, ISO 4 000) in the RAW domain and associated normal distribution (dotted lines)

(b) JPEG intervening blocks for an embedding carried out in the central block

Figure 5.10. *Natural steganography seeks to mimic the normal distribution of the photonic noise a); this distribution in the JPEG domain becomes multivariate, and it must use a lattice decomposition to take the correlations between the DCT coefficients b) into account*

Since an image developed and transmitted by Alice is rarely in the RAW domain, it is necessary to model the distribution of the photonic noise in the spatial domain or in the JPEG domain. In both cases, when the development is linear, the distribution of the steganographic signal is a multivariate Gaussian, which takes into account the relationships between neighboring pixels or neighboring DCT coefficients. Therefore, from a practical point of view, the modifications should be synchronized (see section 7.3) on a number of lattices that can be very large (4×64, see also Figure 5.10(b)). Furthermore, the conversion between the law of probability imitating the photonic noise and the costs used during the embedding is done using equation [5.3].

When it is possible to correctly model the image development chain, natural steganography allows a large amount of information to be embedded[22], while also ensuring good, practical security. These good performances are explained, on the one hand, by the methodology used (imitating a statistical model), and, on the other hand, by using synchronization schemes (see section 7.3) and the RAW image that exploits the rounding error (see section 5.4.4).

5.4.6. *Adversarial steganography*

Adversarial steganographic methods automatically take into account an enemy, the steganalyst, Eve, to produce embedding costs. We can consider two families of adversarial methods described below: attacking methods and methods using adversarial generators.

5.4.6.1. *Attacking steganography*

Adversarial steganographic methods are designed in such a way that Alice directly seeks to bypass the enemy, in this case, the steganalyst Eve. These schemes can become repetitive: Alice first uses an embedding scheme $\mathrm{Ins}_1()$ and Eve constructs a steganalysis method $\mathrm{An}_1()$ that seeks to detect $\mathrm{Ins}_1()$. Second, Alice tries to design an embedding scheme $\mathrm{Ins}_2()$, which will bypass $\mathrm{An}_1()$. Depending on the security scenario considered, that is, depending on Alice's knowledge of Eve's strategy and vice versa, this game between Alice and Eve can continue to develop. Eve can, for example, design a method $\mathrm{An}_2()$ seeking to detect $\mathrm{Ins}_2()$ or $\{\mathrm{Ins}_1(), \mathrm{Ins}_2()\}$. Alice can then try to bypass this new classifier via a $\mathrm{Ins}_3()$ scheme, and so forth.

The first adversarial embedding was proposed by Kouider *et al*. (2013), here the costs used by $\mathrm{Ins}_2()$ are built from a set of linear classifiers (see section 8.4.2) taught

22 In direct relation to the gap between the two ISO parameters.

to detect $\mathrm{Ins}_1()$. Therefore, if a modification of type $+1$ is made on a coefficient x_i, the cost ρ_i will be such that:

$$\rho_i = \sum_k [f_k(x_i + 1) - f_k(x_i)] \qquad [5.12]$$

with $f_k()$ being the function that returns the value before thresholding for the kth classifier.

In this way, the low costs are associated with undetectable modifications.

More recently, Tang *et al.* (2019) proposed an adversarial scheme where the enemy, Eve, trained a deep neural network $\mathrm{An}_1()$. These networks have two advantages; they have become the references in steganalysis (see section 8.5 of Chapter 8), and the function $\mathrm{An}_1()$ is a function which differentiates itself simply and quickly. In general, the costs of certain DCT coefficients of the image are modified for a modification $(+1)$ as follows:

$$\rho_i'(+1) = \begin{cases} \frac{1}{\alpha}\rho_i(+1) & \text{if} \frac{\partial \mathrm{An}_1}{\partial x_i}(\mathbf{x}) < 0 \\ \rho_i(+1) & \text{if} \frac{\partial \mathrm{An}_1}{\partial x_i}(\mathbf{x}) = 0 \\ \alpha\rho_i(+1) & \text{if} \frac{\partial \mathrm{An}_1}{\partial x_i}(\mathbf{x}) > 0 \end{cases} \qquad [5.13]$$

where α is usually 2.

Therefore, the costs decrease if the modification allows the image to move from the cover region to the stego region (negative derivative for a $(+1)$ operation), and they increase in the opposite scenario. This new embedding method $\mathrm{Ins}_2()$, called ADV-EMB (for *Adversarial Embedding*), most often allows the classifier $\mathrm{An}_1()$ to be bypassed, but does not give Alice a solution if Eve intends to train a new detector again $\mathrm{An}_2()$ targeted at $\mathrm{Ins}_2()$, because in the next step, Alice will have to face two enemies, $\mathrm{An}_1()$ and $\mathrm{An}_2()$.

To answer this problem, Bernard *et al.* (2019) offered a strategy, involving the iteration k in selecting the adversarial stego image generated during the k first iterations (each image bypassing the last classifier trained by Eve), which bypasses at "best" (i.e. the most considered as a cover image) the *best* k classifiers (see also Figure 5.11, which illustrates the strategy). In game theory, this strategy is called a min–max strategy. This strategy allows each iteration to increase the practical security of the embedding scheme. The gain can be significant, for example the

adversarial version for the UERD scheme (see section 5.3.2.3) allows, after eight iterations, the increase of the error rate in the detection of more than 10% for an embedding rate of 0.4 bits per non-zero AC coefficient.

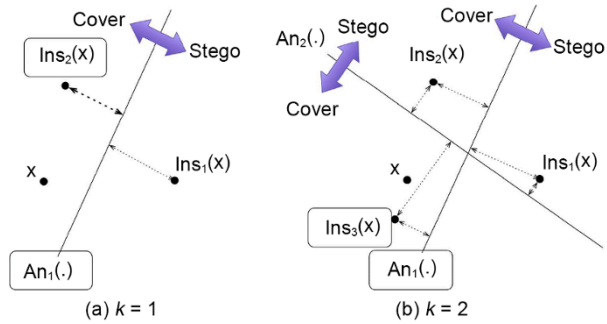


Figure 5.11. *The first two iterations of the strategy proposed by Bernard et al. (2019) on a toy example, where the enemies are linear classifiers and the marked distance at the decision boundary represents the output value of the functions* $\mathrm{An}_k()$

COMMENT ON FIGURE 5.11.– *(a) The first iteration, for a cover content* $\mathbf{x}$*, the original stego content* $\mathrm{Ins}_1(\mathbf{x})$ *is logically replaced by its adversarial version* $\mathrm{Ins}_2(\mathbf{x})$ *(boxed); at this stage, Eve only has one classifier* $\mathrm{An}_1()$*. (b) Eve constructs a new classifier* $\mathrm{An}_2()$*, which is again bypassed by Alice, creating the content* $\mathrm{Ins}_3(\mathbf{x})$*. In the end, the content chosen by Alice will be the content* $\mathrm{Ins}_3(\mathbf{x})$*, it is, in fact, the most difficult content to classify by the best classifier; in this case,* $\mathrm{An}_1()$*, since this maximizes its output for the best adversarial content.*

5.4.6.2. *Steganography through adversarial generator*

If attacking steganography can attack a specific embedding scheme by seeking to bypass it, there are adversarial methods that automatically learn to generate embeddings that try to be undetectable. These methods, based on the concept of generative adversarial networks[23], use two neural networks, one playing the role of the steganograph (Alice) and the other playing the role of the steganalyst (Eve). The optimization of the respective constraints of Alice and Eve is done together. The basic principle of this type of scheme first presented by Tang *et al.* (2017) and improved by Yang *et al.* (2019) is illustrated in Figure 5.12, where two neural networks are shown:

1) an embedding probability generator network, which analyzes the image in order to predict a probability of modification p_i for each pixel (i). The associated cost,

23 See Goodfellow *et al.* (2014).

ρ_i, is then calculated using equation [5.3]. This network is associated with a loss function, related to the size of the message to be embedded via equation [5.1]. This generator analyzes the cover image based on a network initially dedicated to image segmentation;

2) an adversarial network, playing the role of the steganalyst, using a steganalysis network as detailed in section 8.5 of Chapter 8, and which is associated with a loss function related to the detectability of the embedding used.

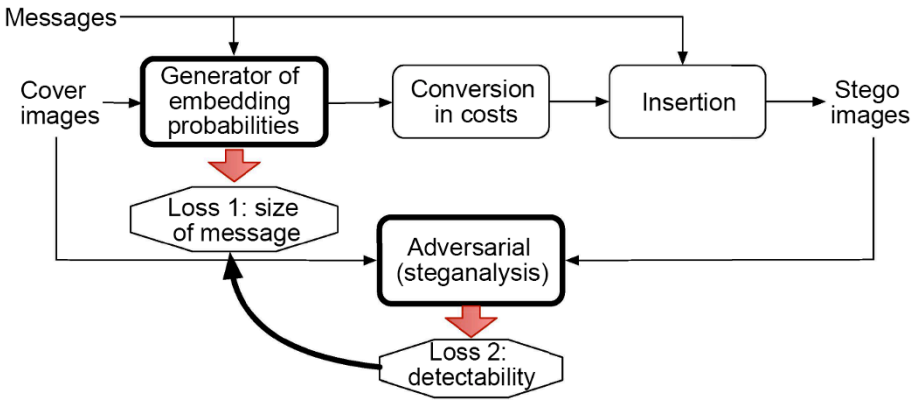


Figure 5.12. *Principle of steganography based on adversarial generators*

The training of the two networks takes place alternately:

1) the loss function of the generator is a weighted sum of the loss associated with the size of the embedded message and the loss associated with detectability. The aim here is to find an image generator capable of embedding a message of the desired size, while minimizing detectability;

2) the enemy looks for it, only to maximize detectability, as traditionally in steganalysis, its loss function uses a group of cover images and stego images for training.

So that the training of these two networks is possible, it is necessary that all of the operations performed are differentiable: the function generating a stego image must be able to be differentiated with respect to its parameters. The embedding function is therefore modified so that it takes this constraint into account.

In terms of performance, the embedding scheme obtained by Yang *et al*. (2019) after the convergence of the system makes it possible to have performance, in terms

of detectability, which is slightly superior to the HILL scheme (see section 5.3.2.1), which is remarkable given the totally automated nature of this approach.

5.5. Conclusion

This chapter concludes by clarifying that the field of steganography, which saw its first formalizations at the end of the 20 century, remains in constant development. These developments are linked to the fact that the steganalysis domain is progressing, but also that new paths (see section 5.4) have emerged.

If other areas of research were presented by Ker *et al*. (2013) (non-additive embedding, scaling laws, aspects linked to security), it should also be noted that the future methods of steganography will increasingly have to take possible transcoding[24] of stego content into account. In fact, this operation is used more and more by social networks or videoconferencing platforms to have a transmission rate that can adapt to the quality of service. Steganography will then have to take into account the constraints linked to the watermarking of documents (see Bas *et al*. (2016)) in order to allow indetectable transmission, but also be robust to this processing that has become common.

5.6. References

Bas, P. (2016). Steganography via cover-source switching. In *Workshop on Information Forensics and Security (WIFS)*. IEEE, Abu Dhabi.

Bas, P., Furon, T., Cayre, F., Doërr, G., Mathon, B. (2016). *Watermarking Security: Fundamentals, Secure Designs and Attacks*. Springer, Singapore.

Bernard, S., Pevny, T., Bas, P., Klein, J. (2019). Exploiting adversarial embeddings for better steganography. In *IH-MMSEC*. ACM, Paris.

Cogranne, R., Sedighi, V., Fridrich, J. (2017). Practical strategies for content-adaptive batch steganography and pooled steganalysis. In *International Conference on Acoustics, Speech and Signal Processing*. IEEE, New Orleans, 2122–2126.

Cogranne, R., Giboulot, Q., Bas, P. (2020). Steganography by minimizing statistical detectability: The cases of JPEG and color images. In *Information Hiding and MultiMedia Security (IH&MMSec)*. ACM, Denver.

Costa, M. (1983). Writing on dirty paper. *IEEE Transactions on Information Theory*, 29(3), 439–441.

Crandall, R. (1998). Some notes on steganography. Document, Steganography mailing list, 1–6.

24 Transcoding involves switching from one format to another, and it most often involves lossy recompression.

Denemark, T. and Fridrich, J. (2015). Side-informed steganography with additive distortion. In *International Workshop on Information Forensics and Security (WIFS)*. IEEE, Rome, 1–6.

Dumitrescu, S., Wu, X., Wang, Z. (2003). Detection of LSB steganography via sample pair analysis. In *IEEE Transactions on Signal Processing*, 1995–2007.

Filler, T., Judas, J., Fridrich, J. (2011). Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6(3), 920–935.

Fridrich, J., Goljan, M., Soukal, D. (2004). Perturbed quantization steganography with wet paper codes. In *Workshop on Multimedia and Security*. ACM, Magdeburg, 4–15.

Fridrich, J., Goljan, M., Soukal, D. (2005). Perturbed quantization steganography. *Multimedia Systems*, 11(2), 98–107.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, 2672–2680.

Guo, L., Ni, J., Su, W., Tang, C., Shi, Y.-Q. (2015). Using statistical image model for JPEG steganography: Uniform embedding revisited. *IEEE Transactions on Information Forensics and Security*, 10(12), 2669–2680.

Holub, V., Fridrich, J., Denemark, T. (2014). Universal distortion function for steganography in an arbitrary domain. *EURASIP Journal on Information Security*, 1, 1–13.

Ker, A.D. (2007). Batch steganography and pooled steganalysis. In *Information Hiding*, Camenisch, J.K., Collberg, C.S., Johnson, N.F., Sallee, P. (eds). Springer, Berlin/Heidelberg.

Ker, A.D., Pevnỳ, T., Kodovskỳ, J., Fridrich, J. (2008). The square root law of steganographic capacity. In *10th ACM Workshop on Multimedia and Security*. ACM, Oxford, 107–116.

Ker, A.D., Bas, P., Böhme, R., Cogranne, R., Craver, S., Filler, T., Fridrich, J., Pevnỳ, T. (2013). Moving steganography and steganalysis from the laboratory into the real world. In *First ACM Workshop on Information Hiding and Multimedia Security*. ACM, Montpellier, 45–58.

Kouider, S., Chaumont, M., Puech, W. (2013). Adaptive steganography by oracle (ASO). In *International Conference on Multimedia and Expo (ICME)*. IEEE, San José, 1–6.

Li, B., Wang, M., Huang, J., Li, X. (2014). A new cost function for spatial image steganography. In *International Conference on Image Processing (ICIP)*. IEEE, Paris, 4206–4210.

Li, B., Wang, M., Li, X., Tan, S., Huang, J. (2015). A strategy of clustering modification directions in spatial image steganography. *IEEE Transactions on Information Forensics and Security*, 10(9), 1905–1917.

Pevnỳ, T. and Nikolaev, I. (2015). Optimizing pooling function for pooled steganalysis. In *International Workshop on Information Forensics and Security (WIFS)*. IEEE, Rome, 1–6.

Pevnỳ, T., Kopp, M., Křoustek, J., Ker, A.D. (2016). Malicons: Detecting payload in favicons. *Electronic Imaging*, 2016(8), 1–9.

Sallee, P. (2003). Model-based steganography. In *International Workshop on Digital Watermarking (IWDW)*. LNCS, Seoul, 2.

Sedighi, V., Cogranne, R., Fridrich, J. (2016). Content-adaptive steganography by minimizing statistical detectability. IEEE *Transactions on Information Forensics and Security*, 11(2), 221–234.

Shannon, C. (1948). A mathematical theory of communication. *Bell Labs Technical Journal*, 27, 379–423, 623–656.

Simmons, G. (1984). The prisoners' problem and the subliminal channel. In *Proceedings of CRYPTO '83*, Chaum, D. (ed.). Plenum Press, New York.

Taburet, T., Filstroff, L., Bas, P., Sawaya, W. (2018). An empirical study of steganography and steganalysis of color images in the JPEG domain. In *International Workshop on Digital Forensics and Watermarking*. IWDW, Jeju.

Taburet, T., Bas, P., Sawaya, W., Fridrich, J. (2020). Natural steganography in JPEG domain with a linear development pipeline. *IEEE Transactions on Information Forensics and Security*, 16, 173–186.

Tang, W., Tan, S., Li, B., Huang, J. (2017). Automatic steganographic distortion learning using a generative adversarial network. *IEEE Signal Processing Letters*, 24(10), 1547–1551.

Tang, W., Li, B., Tan, S., Barni, M., Huang, J. (2019). CNN-based adversarial embedding for image steganography. *IEEE Transactions on Information Forensics and Security*, 14(8), 2074–2087.

Wang, Y., Zhang, W., Li, W., Yu, X., Yu, N. (2019). Non-additive cost functions for color image steganography based on inter-channel correlations and differences. *IEEE Transactions on Information Forensics and Security*, 15, 2081–2095.

Yang, J., Ruan, D., Huang, J., Kang, X., Shi, Y.-Q. (2019). An embedding cost learning framework using GAN. *IEEE Transactions on Information Forensics and Security*, 15, 839–851.

Zakaria, A., Chaumont, M., Subsol, G. (2019). Pooled steganalysis in JPEG: How to deal with the spreading strategy? [Online]. Available at: http://arxiv.org/abs/1906.11525.

6

Traitor Tracing

Teddy FURON
IRISA, University of Rennes, Inria, CNRS, France

This chapter presents the problem of traitor tracing and the most efficient solution known, Tardos codes. Tracing codes form an application overlay to the transmission layer by watermarking: a codeword is generated for each user, then inserted, by watermarking, into a confidential document to be shared. First and foremost, the chapter emphasizes the modeling of collusion when several traitors combine their copies. Thanks to this model, mathematics (statistics, information theory) give us the basic limits of this tool.

6.1. Introduction

A valuable document is distributed to a group $\mathcal{U}$ of n users, each labeled by an integer: $\mathcal{U} = [n]$, where $[n] := \{1, \ldots, n\}$. These users are not to be trusted. Some of them may "leak" this document. Nothing can stop them, apart from a deterrent weapon: traitor tracing. Each user receives a unique personalized copy of the content. We assume that it is possible to do this personalization by inserting a unique codeword into the content using a watermarking technique. If a leak occurs, decoding the watermark of the illegal copy will reveal the identity of the traitor, exposing them to severe prosecution. Traitor tracing explores the case where several traitors secretly

For a color version of all figures in this chapter, see www.iste.co.uk/puech/multimedia1.zip.

Multimedia Security 1,
coordinated by William PUECH. © ISTE Ltd 2022.

Multimedia Security 1: Authentication and Data Hiding,
First Edition. William Puech.

cooperate to cover their tracks and prevent identification. This group of dishonest users is called a *collusion*.

This chapter is organized as follows. Section 6.1 introduces the main components: code construction, collusion strategy, accusation with a simple decoder, based on a score function and thresholding. Section 6.2 provides a simplified proof of Tardos' original scheme. Section 6.3 presents the many extensions found in the literature, two of which are detailed in sections 6.4 (the search for better score functions) and 6.5 (the search for better thresholding).

6.1.1. *The contribution of the cryptography community*

Cryptologists were the first to study this problem. They came up with a model that explains what content and collusion attack are. Content is a series of discrete symbols (e.g. bits), some of which can be modified without damaging the use of the content. Only the defender (the official source of the document) knows the location of these modifiable symbols, which they can use to insert a user's codeword into the content. On receiving the document, the traitors can reveal certain so-called *detectable* positions by comparing their versions. This model, known by the name *Marking Assumption*, was invented by Boneh and Shaw (1998).

The model restricts what traitors can do to detectable positions. The group of c traitors is denoted by $\mathcal{C} = \{j_1, \ldots, j_c\} \subset \mathcal{U}$. The *narrow-sense version* (or *restricted digit model*) requires traitors to assemble the pirated copy of the document symbol by symbol. If they have the same symbol in a certain location, the position is not detectable and this symbol is found in the pirated copy. Otherwise, at a detectable position, they choose one of their symbols at this location, according to a certain collusion strategy. This model is summarized by:

$$y_i \in \{x_{j_1,i}, \cdots, x_{j_c,i}\}, \forall i \in [m] \qquad [6.1]$$

where $\mathbf{y}$ is the hidden sequence in pirated content, y_i is its symbol at the ith position and $x_{j,i}$ is the hidden symbol in the copy of the jth user at position i.

Note that there are as many codewords $(\mathbf{x}_j)_{j=1}^n$ as there are users, and their length is m.

There are other less restrictive models for collusion: the *wide-sense version* or *arbitrary digit model* (Boneh and Shaw 1998), the *unreadable digit model* (Boneh and Shaw 1998), the *general digit model* or the *weak marking assumption* (Safavi-Naini and Wang 2001; Fodor *et al*. 2018).

The second contribution of cryptologists was the definition of desirable properties of a code (group of identifiers) (Stinson and Wei 1998) and the proposal of solutions

that were mainly based on error correction codes. These will not be covered since we now know that Tardos codes are better.

6.1.2. *Multimedia content*

In multimedia applications, content is modeled as a series of blocks (a few seconds of audio, a scene from a movie, a block of pixels from an image). A watermarking technique (see Chapter 3) hides a symbol of an identifier in each block. The idea of detectable position is useless here: everyone knows that each block hides a symbol. However, watermarking is a secret codeword primitive. Without this codeword, we assume that we cannot read or modify the hidden symbols. Therefore, an attack involves constructing the pirated copy by copying and pasting blocks from the personalized versions. Thus, the *Marking Assumption* model described in equation [6.1] becomes valid.

Figure 6.1 shows a standard solution in the film industry (called "DNA watermarking"). The watermarking of the video blocks is done in advance. For a binary code, there are two versions of each block: watermarked with a "1" or a "0". The personalization during the distribution is fast because it is enough to sequentially take the blocks that hide the codeword of the user (Shahid *et al*. 2013). When the traitors have a certain block in only one version (watermarked with "1" for example), this symbol ends up in the pirated copy.

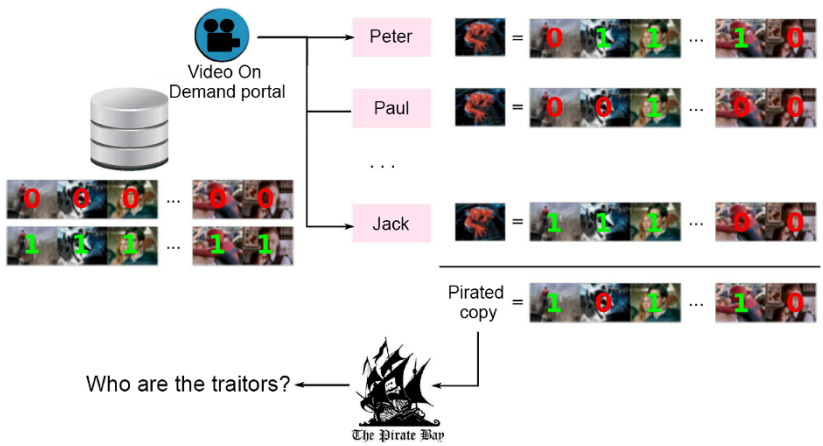


Figure 6.1. *Sequential watermarking of a movie. A thumbnail image represents a video block, watermarked to hide the symbol "1" or "0". The traitors sequentially form a pirated movie by selecting one of their blocks*

6.1.3. *Error probabilities*

The most important criterion in the specifications is to avoid accusing an innocent user. The probability of accusing at least one innocent user is noted by $\mathbb{P}_{\mathsf{FP}}$. This probability must be low and we must make sure that it is smaller than a level η_R stated. This is what we call proof of *robustness*.

The second criterion is the identification of traitors. We must set a goal:

– "catch one": the aim is to identify at least one traitor;

– "catch all": the aim is to identify all of the traitors. You must assume that all traitors participate in the collusion fairly. It is impossible to identify a traitor if they have little involvement in the counterfeiting of the pirated copy;

– "catch a colluder": the aim is to identify one traitor in particular.

The *completeness* of a scheme comes down to proving that the probability $\mathbb{P}_{\mathsf{FN}}$ of not achieving the set goal is smaller than a certain stated level η_C. The first two aims require investigating the events of accusation of the c traitors. This is difficult because these events are not independent. The last aim is simpler because it only considers one traitor.

In practice, η_C should be low enough so that traitor tracing is deterrent enough. $\mathbb{P}_{\mathsf{FN}}$ is sometimes used as a metric to compare accusation schemes that meet all of the first criterion on η_R. Other comparisons are based on the number of identified traitors ("catch many") still under the constraint $\mathbb{P}_{\mathsf{FP}} \leq \eta_R$.

6.1.4. *Collusion strategy*

The collusion strategy, or attack, describes the process that traitors use to create pirated content, and more precisely its impact on the series of symbols that will be extracted by decoding the watermark of the pirated content. This series of symbols is called the pirated series and is denoted by $\mathbf{y}$.

The descent set is the set of all pirated series that a collusion can construct from the codewords of the traitors. According to the *Marking Assumption* [6.1], the descent set can contain 2^{m_d} series for a binary code, where $m_d < m$ is the number of detectable positions. A working hypothesis is that traitors share the risk evenly (unless there are traitors among traitors; seec Zhao and Liu (2006)), which reduces the descent set. How the accusation behaves for each pirated series of this set should then be judged, and this should be done for any possible collusion of a certain size. This is impossible. It results in more assumptions about the collusion strategy.

Four hypotheses are commonly used (Furon *et al*. 2008):

– *memoryless (Moulin 2008, definition 2.4)*: as we will see later, in Tardos codes, the symbols of the identifiers are statistically independent. We make the same assumption about the symbols of the pirated series. So the value of y_i depends on the traitors' symbols at this position only $\{x_{j_1,i}, \cdots, x_{j_c,i}\}$;

– *permutation invariant (Moulin 2008, definition 2.5)*: there is no concept of order among traitors. So the way of deciding the value of y_i is invariant to any permutation of $\{x_{j_1,i}, \cdots, x_{j_c,i}\}$. This means that this strategy only considers the empirical distribution (the histogram) of these symbols. In the binary case, this distribution is completely defined by the number σ_i of symbols "1" to position i: $\sigma_i := \sum_{j \in \mathcal{C}} x_{j,i}$ (so the number "0" is $c - \sigma_i$);

– *stationary*: we assume that collusion has a unique strategy that it employs across all positions. So we can leave out the index $i \in [m]$ later;

– *random*: this strategy is not necessarily deterministic.

The following model of a collusion attack of size c comes from these four hypotheses:

$$\boldsymbol{\theta}_c := (\theta_{c,0}, \ldots, \theta_{c,c})^\top \in [0,1]^{(c+1)}, \text{ with } \theta_{c,\sigma} := \mathbb{P}\left(Y = 1 | \Sigma = \sigma\right) \qquad [6.2]$$

The attack is then described by the probability that the traitors choose a symbol "1" in the pirated series, when they have σ "1" of c symbols in their codeword to a certain position. In other words, traitors have $c+1$ biased coins. At a certain position i, they have σ_i symbols "1", they then flip the coin σ_i to choose the value of the symbol y_i. The *Marking Assumption* (see equation [6.1]) requires that $\theta_{c,0} = 0$ and $\theta_{c,c} = 1$. Amiri and Tardos (2009) mention an *eligible channel*. This model defines the set $\boldsymbol{\Theta}_c$ of size c collusion strategies of size:

$$\boldsymbol{\Theta}_c := \{\boldsymbol{\theta}_c \in [0,1]^{(c+1)} | \theta_{c,0} = 1 - \theta_{c,c} = 0\} \qquad [6.3]$$

Here is a list of classic strategies at each position $i \in [m]$:

– *interleaving attack* (or uniform attack): traitors randomly pick which of them will place their content block into pirated content. If σ of them have block "1", then the probability of this block ending up in the pirated copy is:

$$\theta_{c,\sigma} = \sigma/c, \quad \forall \sigma \in \{0, \ldots, c\} \qquad [6.4]$$

– *majority vote*: traitors choose the majority symbol:

$$\theta_{c,\sigma} = \begin{cases} 1, & \sigma > c/2 \\ 1/2, & \sigma = c/2 \quad (\text{if } c \text{ is even}) \\ 0, & \text{if not} \end{cases} \qquad [6.5]$$

– *minority vote*: traitors choose the minority symbol when possible:

$$\theta_{c,\sigma} = \begin{cases} 1, & 0 < \sigma < c/2 \text{ where } \sigma = c \\ 1/2, & \sigma = c/2 \quad (\text{if } c \text{ is even}) \\ 0, & \text{if not} \end{cases} \qquad [6.6]$$

– *coin flip*: the traitors toss an unbiased coin to choose when given a choice:

$$\boldsymbol{\theta}_c = (0, 1/2, \ldots, 1/2, 1)^\top \qquad [6.7]$$

– *all 1 attack*: the traitors always choose the symbol "1" when they can, that is, if $\sigma > 0$:

$$\boldsymbol{\theta}_c = (0, 1, \ldots, 1)^\top \qquad [6.8]$$

– *all 0 attack*: the traitors always choose the symbol "0" when they can, that is, if $\sigma < c$:

$$\boldsymbol{\theta}_c = (0, 0, \ldots, 0, 1)^\top \qquad [6.9]$$

These are only a few examples; the set of available strategies $\boldsymbol{\Theta}_c$ is actually infinite.

6.2. The original Tardos code

In this chapter, Tardos codes denote a family of codes that share the same original structure (Tardos 2003, 2008). In other words, any modification to the parameters of the code, or to the way of identifying, compared to the original version by Tardos is not a reason to change the name: these are all Tardos codes. This section offers the original version.

6.2.1. *Constructing the code*

The construction is done in three stages:

– a r.v. (random variable) $P \in (0,1)$ is defined. This can be a discrete r.v. with a set of possible values $\mathcal{P} = \{\omega_i\}_{i=1}^K \subset (0,1)$ and associated probabilities $\{f_i := \Pr(P = \omega_i)\}_{i=1}^K$ or an absolutely continuous r.v. with a probability density $f : (0,1) \to \mathbb{R}^+$;

– the encoder takes m realizations of this r.v. independently according to its law. These realizations are stored in the vector $\mathbf{p} := (p_1, \ldots, p_m)$, called secret series;

– the encoder generates the codewords of the users by independently drawing $n \times m$ Bernoulli r.v. such that $X_{j,i} \sim \mathcal{B}(p_i)$. The code $\mathbf{X}$ is a binary matrix of size $n \times m$, and the codeword of the user j is $\mathbf{x}_j = (x_{j,1}, \ldots, x_{j,m})$. This means that, for all users, $\Pr(X_{j,i} = 1) = p_i$ depends on the position in the series.

The law of the r.v. P is public while the vector $\mathbf{p}$ is a secret shared with the accusation algorithm. The code is made up of n private codewords, in that a user only knows, at most, their own identifier, unless they are a traitor. By forming a collusion, traitors can share their knowledge of their c codewords.

6.2.2. *The collusion strategy and its impact on the pirated series*

The hypotheses on the collusion strategy in section 6.1.4 are very useful in deriving a statistical model of r.v. $\{Y_i\}_{i=1}^m$ decoded symbols of the pirated series. We define $\Pi(p_i) := \Pr(Y_i = 1|p_i)$ (the dependency in $\boldsymbol{\theta}_c$ is omitted, unless there is ambiguity). At the index i, the symbols of the codewords of the traitors are distributed as Bernoulli r.v. $\mathcal{B}(p_i)$, given the construction of the code. So Σ_i, the number of "1" for c traitors in position i, following a binomial distribution $\mathcal{B}(c, p_i)$:

$$\mathbb{P}(\Sigma_i = \sigma) = \binom{c}{\sigma} p_i^\sigma (1-p_i)^{c-\sigma}, \quad \forall \sigma \in \{0, 1, \ldots, c\} \qquad [6.10]$$

For a given position i, Y_i is a Bernoulli r.v.: $Y_i \sim \mathcal{B}(\Pi(p_i))$ of parameter:

$$\Pi(p_i) := \mathbb{P}(Y_i = 1|p_i) = \sum_{\sigma=0}^{c} \mathbb{P}(Y_i = 1|\sigma)\mathbb{P}(\Sigma_i = \sigma|p_i) \qquad [6.11]$$

$$= \sum_{\sigma=0}^{c} \theta_{c,\sigma} \binom{c}{\sigma} p_i^\sigma (1-p_i)^{c-\sigma} \qquad [6.12]$$

We see that Y_i follows a different law from X_{ji} and that this law depends on the collusion strategy. Note the following properties of this function illustrated in Figure 6.2:

1) $\Pi(p)$ is a polynomial of degree at most c, where $\Pi(0) = \theta_{c,0} = 0$ and $\Pi(1) = \theta_{c,c} = 1$;

2) two collusion strategies of the same size c produce two different distributions:

$$\boldsymbol{\theta}_c \neq \boldsymbol{\theta}'_c \rightarrow \exists p \in (0,1),\ \Pi(p;\boldsymbol{\theta}_c) \neq \Pi(p;\boldsymbol{\theta}'_c); \quad [6.13]$$

3) a collusion strategy produces an attack "between 'All-0' and 'All-1"', in that:

$$\forall p \in [0,1], \quad p^c \leq \Pi(p) \leq 1-(1-p)^c \quad [6.14]$$

4) the so-called interleaving attack has the remarkable property that $\Pi(p)$ is independent from c: $\forall p \in [0,1],\ \Pi(p) = p$.

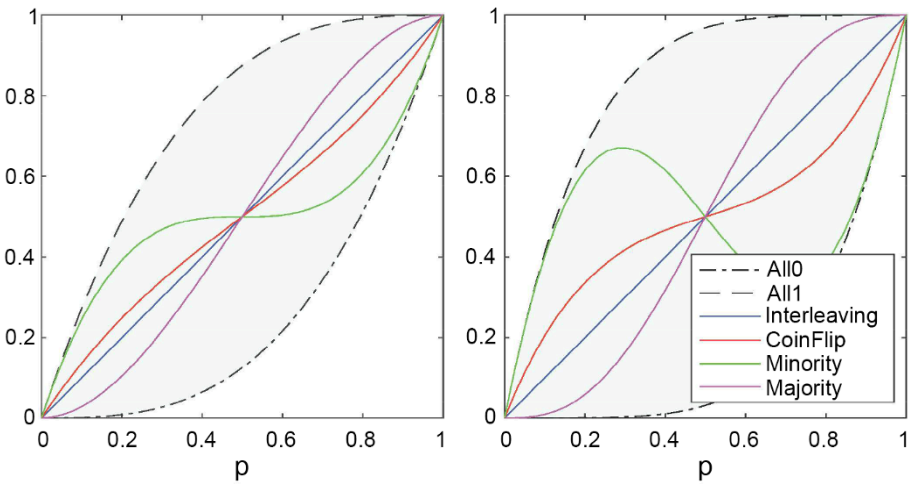


Figure 6.2. $\Pi(p) := \mathbb{P}(Y=1|p)$ *for* $c = 3$ *a) and* $c = 5$ *b). All 1 attack:* $\Pi(p) = 1-(1-p)^c$, *all 0 attack:* $\Pi(p) = p^c$, *interleaving:* $\Pi(p) = p$, *coin flip:* $\Pi(p) = (1-(1-p)^c + p^c)/2$, *minority and majority do not have a simple formula. This function remains in the blue zone, marked by the "All-1" and "All-0" strategies*

The second property proves the identifiability of the model when the size of the collusion is known. This means that we can learn the $\boldsymbol{\theta}_c$ strategy if we see a large number of (Y_i, P_i) occurrences. However, the fourth property shows that the identifiability of the model no longer remains when we do not know the size of the collusion: it is sometimes impossible to distinguish attacks of different sizes (since they generate the same distribution of symbols Y_i).

In the same way, we can also calculate the distribution of Y_i, knowing that one of the traitors has a "1" symbol in the i position (say $x_{j_1 i} = 1$). Together, the collusion has $\sigma > 0$ symbols "1", if the remaining traitors $(c-1)$ have $\sigma - 1$ "1":

$$\mathbb{P}(\Sigma_i = \sigma | p_i, x_{j_1 i} = 1) = \binom{c-1}{\sigma-1} p_i^{\sigma-1} (1-p_i)^{c-\sigma}$$

If $x_{j_1 i} = 0$, the collusion has $\sigma < c$ symbols "1", if the remaining traitors $(c-1)$ have σ "1":

$$\mathbb{P}(\Sigma_i = \sigma | p_i, x_{j_1 i} = 0) = \binom{c-1}{\sigma} p_i^{\sigma} (1-p_i)^{c-\sigma-1}$$

Generally, we note $\Pi_x(p_i) := \mathbb{P}(Y_i = 1 | p_i, x_{j_1 i} = x)$ for $x \in \{0, 1\}$:

$$\Pi_x(p_i) = \sum_{\sigma=x}^{c-1+x} \theta_{c,\sigma} \binom{c-1}{\sigma-x} p_i^{\sigma-x} (1-p_i)^{(c-1-\sigma+x)} \quad [6.15]$$

The following properties are given:

1) $\Pi(p)$ is the barycenter of $\Pi_1(p)$ and $\Pi_0(p)$ with the weights p and $1-p$:

$$\Pi(p) = p\Pi_1(p) + (1-p)\Pi_0(p) \quad [6.16]$$

2) at the extreme values of p, we have:

$$\lim_{p\to 0} \Pi_0(p) = 0, \quad \lim_{p\to 0} \Pi_1(p) = \theta_{c,1}$$

$$\lim_{p\to 1} \Pi_0(p) = \theta_{c,c-1}, \quad \lim_{p\to 1} \Pi_1(p) = 1$$

3) by a simple calculation: $\forall p \in (0, 1)$:

$$\Pi_1(p) = \Pi(p) + c^{-1}(1-p)\Pi'(p) \quad [6.17]$$

$$\Pi_0(p) = \Pi(p) - c^{-1} p \Pi'(p) \quad [6.18]$$

where $\Pi'(\cdot)$ is the derivative of $\Pi(\cdot)$ (see equation [6.11]), with relation to p. This last equation does not have a simple interpretation, but it is essential for proving the completeness of the Tardos code (see section 6.2.3.2).

6.2.3. *Accusation with a simple decoder*

Once a pirated copy is found, decoding the watermark of content block by block results in the pirated series $\mathbf{y}$, which is no longer random. To decide if user j is a

traitor, a simple decoder calculates a score s_j from the codeword $\mathbf{x}_j$, the pirated series $\mathbf{y}$ and the secret $\mathbf{p}$:

$$s_j = \sum_{i=1}^{m} U(x_{ji}, y_i, p_i) \qquad [6.19]$$

where $U(\cdot)\colon \{0,1\} \times \{0,1\} \times (0,1) \to \mathbb{R}$ is called the *score function*.

The user is accused if $s_j > \tau$, where τ is a threshold. In other words, the accusation tests two hypotheses for each user: $\mathcal{H}_0$, the user j is innocent versus $\mathcal{H}_1$, the user j is guilty. Let $\mathbb{P}_{\mathsf{fp}}$ and $\mathbb{P}_{\mathsf{fn}}$ be false positive (wrongly accusing someone of being innocent) and false negative probabilities (exonerating someone that is guilty) of this test *for each user*. The r.v. modeling the score of an innocent user (a traitor) is denoted S_{inn} (respectively S_{tra}). So $\mathbb{P}_{\mathsf{fp}} = \mathbb{P}(S_{\mathsf{inn}} > \tau)$ and $\mathbb{P}_{\mathsf{fn}} = \mathbb{P}(S_{\mathsf{tra}} < \tau)$.

6.2.3.1. *Robustness*

The robustness requires the probability $\mathbb{P}_{\mathsf{FP}}$ of accusing one innocent user in n tests to be smaller than η_R. As the codewords of the innocent users are independent (knowing $\mathbf{p}$ and $\mathbf{y}$), the scores are also independent:

$$\mathbb{P}_{\mathsf{FP}} = \mathbb{P}\left(\cup_{j\notin\mathcal{C}}\{S_j > \tau\}\right) = 1 - \mathbb{P}\left(\cap_{j\notin\mathcal{C}}\{S_j < \tau\}\right) \qquad [6.20]$$

$$= 1 - (1 - \mathbb{P}_{\mathsf{fp}})^{n-c} < \eta_R \qquad [6.21]$$

where $n - c$ is the number of innocent users.

It is easy to see that if $\mathbb{P}_{\mathsf{fp}} < \eta_R/n$, the constraint of equation [6.21] is respected. Robustness is not so easy since we are faced with a rare event: a classic requirement is $\eta_R \approx 10^{-3}$ and $n \approx 10^6$, so much so that η_R/n is of the order 10^{-9}.

6.2.3.2. *Completeness*

Completeness considers the scores of c traitors. The aim is to relate the fact that $\mathbb{P}(S_j < \tau) = \mathbb{P}_{\mathsf{fn}}$, $\forall j \in \mathcal{C}$ to the total probability, $\mathbb{P}_{\mathsf{FN}}$, for different objectives (section 6.1.3):

– "catch one": failing to identify at least one traitor means that none have been accused:

$$\mathbb{P}_{\mathsf{FN}} = \mathbb{P}\left(\cap_{j\in\mathcal{C}}\{S_j < \tau\}\right) \leq \min_{j\in\mathcal{C}} \mathbb{P}(S_j < \tau) \leq \mathbb{P}_{\mathsf{fn}} \qquad [6.22]$$

This aim is therefore achieved if $\mathbb{P}_{\mathsf{fn}} < \eta_C$;

– "catch all": failing to accuse all traitors means that at least one has not been accused. According to the union bound, we have:

$$\mathbb{P}_{\mathsf{FN}} = \mathbb{P}\left(\cup_{j\in\mathcal{C}}\{S_j < \tau\}\right) \leq \sum_{j\in\mathcal{C}} \mathbb{P}(S_j < \tau) = c\mathbb{P}_{\mathsf{fn}} \quad [6.23]$$

which shows that the objective is achieved if $\mathbb{P}_{\mathsf{fn}} < \eta_C/c$;

– "catch a colluder": the probability of failing to catch a particular traitor is $\mathbb{P}_{\mathsf{FN}} = \mathbb{P}_{\mathsf{fn}}$.

For a given level η_C, the most difficult objectives to achieve, in that $\mathbb{P}_{\mathsf{fn}}$ is more constrained, are in the following order: "catch all", "catch a colluder" and "catch one".

6.2.4. *Study of the Tardos code-Škorić original*

This section explains the choices made in Tardos (2003, 2008) and the improvements offered in Škorić *et al.* (2008a). The proofs of robustness and completeness are simplified. We assume that $\eta_C = 1/2$ for a "catch one" or "catch a colluder" objective. Therefore, it is enough to show that $\mathbb{P}_{\mathsf{fn}} < 1/2$.

The main idea is to choose a score function, such that the statistics of S_{inn} do not depend on the collusion strategy. So the scores of the users are compared to a universal threshold τ. Here, universal means that the threshold is set, regardless of the collusion strategy and $(\mathbf{y}, \mathbf{p})$. The constraint $\mathbb{P}_{\mathsf{fp}} = \mathbb{P}(S_{\mathsf{inn}} > \tau)$ must be guaranteed, whatever the size and collusion strategy. In statistical terms, the size c and the collusion strategy are *nuisance parameters* since they are unknown to the accusation algorithm. Ideally, the score of someone innocent must be a *pivotal quantity*, that is, whose distribution does not depend on nuisance parameters. In practice, only moments of orders 1 and 2 will be invariant.

For an innocent user, insisting that $\forall p \in (0,1)$, $\forall y \in \{0,1\}$, $\mathbb{E}(U(X_{\mathsf{inn}}, y, p)) = 0$ and $\mathbb{V}(U(X_{\mathsf{inn}}, y, p)) = 1$ is equivalent to choosing the following score function (Furon *et al.* 2008) called Tardos-Škorić:

$$\begin{aligned} U(1,1,p) &= \sqrt{\frac{1-p}{p}}, \quad U(0,0,p) = \sqrt{\frac{p}{1-p}} \\ U(0,1,p) &= -\sqrt{\frac{p}{1-p}}, \quad U(1,0,p) = -\sqrt{\frac{1-p}{p}} \end{aligned} \quad [6.24]$$

We see that the score function takes positive values when the user has the same symbol as the one found in the pirated series. This quantity is especially large as

this symbol is rare at this position in the code. This gives us a clue that this user is potentially a traitor. In the same way, the score function takes negative values if the symbols are different. This gives us a clue that the user is potentially innocent. Their final score reflects this evidence by adding up these quantities over all of the positions of the code. Let us define the random variables:

$$S_{\mathsf{inn}} = \sum_{i=1}^{m} U(X_{\mathsf{inn},i}, y_i, p_i) \qquad [6.25]$$

$$S_{\mathsf{tra}} = \sum_{i=1}^{m} U(X_{\mathsf{tra},i}, y_i, p_i) \qquad [6.26]$$

We see that this score function has good sense, making $\mathbb{E}(S_{\mathsf{inn}}) = 0$ and $\mathbb{V}(S_{\mathsf{inn}}) = m$, whatever the secret series, $\mathbf{p}$, and the decoded pirated series $\mathbf{y}$ may be.

Bernstein's inequality gives an upper bound to the probability $\mathbb{P}_{\mathsf{fp}}$, which decreases exponentially with τ (Furon and Desoubeaux 2014). Other inequalities are possible (e.g. Hoeffding, Kearns and Saul). This makes it possible to find the necessary code length given in Tardos (2003, 2008). The r.v. $U(X_{\mathsf{inn},i}, y_i, p_i)$ are independent (according to the construction of the code and the collusion strategy model) of zero-expectation and unit variance. Furthermore, $\forall i \in [m]$, $|U(X_{\mathsf{inn},i}, y_i, p_i)| < M$, where $M = \max(\{\sqrt{p_i/(1-p_i)}\} \cup \{\sqrt{(1-p_i)/p_i}\})$. So, $\forall \tau > 0$:

$$\mathbb{P}_{\mathsf{fp}} \leq e^{-\frac{3\tau^2}{6m+2M\tau}} \qquad [6.27]$$

This Bernstein bound is true for a given secret series since M depends on $\mathbf{p}$. It is very useful in practice to find the value of the threshold τ, but this bound is not universal. One solution is to introduce a limit parameter $0 < t < 1/2$, which is used to bound the score function[1]. The score function becomes:

$$\bar{U}(x, y, p) = \begin{cases} U(x, y, p) & \text{if } t < p < 1 - t \\ 0 & \text{otherwise} \end{cases} \qquad [6.28]$$

One consequence is that we can now set M to $1/\sqrt{t}$. Equation [6.27] is now valid, whatever the couple $(\mathbf{p}, \mathbf{y})$.

1 Tardos and his successors use t to limit the probability density medium f of the r.v. P in encoding. The modification proposed here simplifies the calculations.

As for the score of a traitor, using the properties of section 6.2.2, we can show that:

$$\mathbb{E}(\bar{U}(X_{\mathsf{tra}}, Y, p)) = \sum_{(y,x)\in\{0,1\}} \bar{U}(x, y, p) \Pr(Y = y | X_{\mathsf{tra}} = x, p) \Pr(X_{\mathsf{tra}} = x | p)$$

$$= \begin{cases} 2\sqrt{p(1-p)}(\Pi_1(p) - \Pi_0(p)), & \text{if } t < p < 1 - t \\ 0 & \text{otherwise} \end{cases}$$

$$= \begin{cases} \frac{2}{c}\sqrt{p(1-p)}\Pi'(p), & \text{if } t < p < 1 - t \\ 0 & \text{otherwise} \end{cases} \quad [6.29]$$

where $\Pi'(\cdot)$ is derivative of $\Pi(\cdot)$ (see equation [6.11]) in relation to p.

The choice made in the original article (Tardos 2003) by an absolutely continuous r.v. P of density:

$$f(p) = \frac{1}{\pi\sqrt{p(1-p)}}, \quad \forall 0 < p < 1 \quad [6.30]$$

gives:

$$\mathbb{E}\left(S_{\mathsf{tra}}\right) = m\mathbb{E}\left(\bar{U}(X_{\mathsf{tra}}, Y, P)\right) = \int_0^1 \mathbb{E}\left(\bar{U}(X_{\mathsf{tra}}, Y, p)\right) f(p)dp \quad [6.31]$$

$$= \frac{2m}{\pi c}\int_t^{1-t} \Pi'(p)dp = \frac{2m}{\pi c}\left(\Pi(1-t) - \Pi(t)\right) \quad [6.32]$$

For any strategy following the model explained in section 6.1.4, $\mathbb{E}\left(S_{\mathsf{tra}}\right) \to {}^{2m}/_{\pi c}$ when $t \to 0$ (in fact, we can show that $2(1-t)^c - 1 \leq \Pi(1-t) - \Pi(t) \leq 1$). The more traitors there are, the lower $\mathbb{E}\left(S_{\mathsf{tra}}\right)$ is, and, at the first order, the distribution of S_{tra} converges with that of S_{inn}. To combat this effect, we need to increase the length of the m code.

The score S_{tra} is a sum of m finite variance independent r.v. The law of S_{tra} tends toward a Gaussian distribution when $m \to \infty$, according to the central limit theorem (assuming Lindeberg's condition to be true). So, mean and median converge asymptotically: $\mathbb{P}(S_{\mathsf{tra}} < \mathbb{E}(S_{\mathsf{tra}})) \approx {}^1/_2$ (a more precise proof of completeness uses a probability bound that S_{tra} deviates from its expectation, like a Chebychev bound, for example). This makes $\mathbb{P}_{\mathsf{fn}}$ smaller than ${}^1/_2$ if τ is smaller than $\mathbb{E}(S_{\mathsf{tra}})$.

All of these elements placed end-to-end give a constraint on the length of the code m. With a probability greater than ½ (assuming this is sufficiently deterrent), a

particular traitor will be identified (whatever the collusion strategy) if $2m/\pi c(2(1-t)^c - 1) = \tau$. At the same time, the probability of accusing at least one innocent user will be smaller than η_R, if $e^{-\frac{3\tau^2}{6m+2M\tau}} \leq \eta_R/n$ by equation [6.27]. These two parts limit the length of the code:

$$m \geq \frac{\pi^2 c^2}{2} \log\left(\frac{n}{\eta_R}\right) G(t,c) \qquad [6.33]$$

where:

$$G(t,c) := \frac{1}{(2(1-t)^c - 1)^2}\left(1 + \frac{2}{3\pi}\frac{2(1-t)^c - 1}{c\sqrt{t}}\right) \qquad [6.34]$$

By making t dependent on c, we can limit $G(t,c)$. For example, let $t_c := 1 - (^{(h+1)}/_2)^{1/c}$ (s.t. $2(1-t_c)^c - 1 = h$ where $0 < h < 1$) to get $G(t_c, c) \leq 2$ for $h = 0,9$. In the end, robustness and completeness are proved if $m > \underline{m}(n, c, \eta_R, 1/2)$ where:

$$\underline{m}(n, c, \eta_R, 1/2) = \pi^2 c^2 \log\left(\frac{n}{\eta_R}\right) \qquad [6.35]$$

This is an asymptotic result (using the central limit theorem for completeness). In other words, $m = \Omega(c^2 \log {}^n/_{\eta_R})$.

6.2.5. *Advantages*

This demonstration has the following advantages.

6.2.5.1. *Pedagogy*

Equation [6.29] is very important for understanding the construction of the code. Suppose the set secret code, that is, $f(p) = \delta(p - p_0)$. In other words, P is no longer a r.v. but a constant $p_0 \in (t, 1-t)$. In this case, $\mathbb{E}(S_{\mathsf{tra}}) = m\mathbb{E}(\bar{U}(X_{\mathsf{tra}}, Y, p_0))$. The code will be even shorter, as the expectation $\mathbb{E}(S_{\mathsf{tra}})$ will be larger than $\mathbb{E}(S_{\mathsf{inn}}) = 0$.

Figure 6.3 plots the function $\mathbb{E}(\bar{U}(X_{\mathsf{tra}}, Y, p))$ according to p for classical attacks. We can see that $p_0 = 1/2$ is a great choice to fight against the collusion strategy "interleaving", since function y is maximal. On the other hand, this choice is catastrophic for the collusion strategy "minority vote", since the function takes a zero value in $1/2$ for $c = 3$. It then becomes impossible to distinguish the score of someone that is innocent from the score of a traitor since they have the same expectation. For $c > 3$, the minority strategy even gives negative expectations. To fight against the

"minority vote", it is better to choose a value of p_0 close to 0 or 1, but this is a very bad choice against interleaving, for example. Figure 6.3 shows that it is impossible to choose a satisfactory value of p_0 for all of the strategies.

Since the value of p_0 must be decided at the code creation before the traitors choose their attack, the game is unfair for the defender. Introducing a random variable P reverses the situation: not only can traitors no longer adapt their strategy to the p_i values (since $\mathbf{p}$ is a secret sequence), but they must also choose a unique strategy to "fight" against all cases, i.e. all of the values stored in $\mathbf{p}$. Not knowing this strategy in advance, the defender is advised to choose values across the $(0, 1)$ support, so that on average it gives $\mathbb{E}(S_{\mathsf{tra}})$. This is shown in equation [6.31].

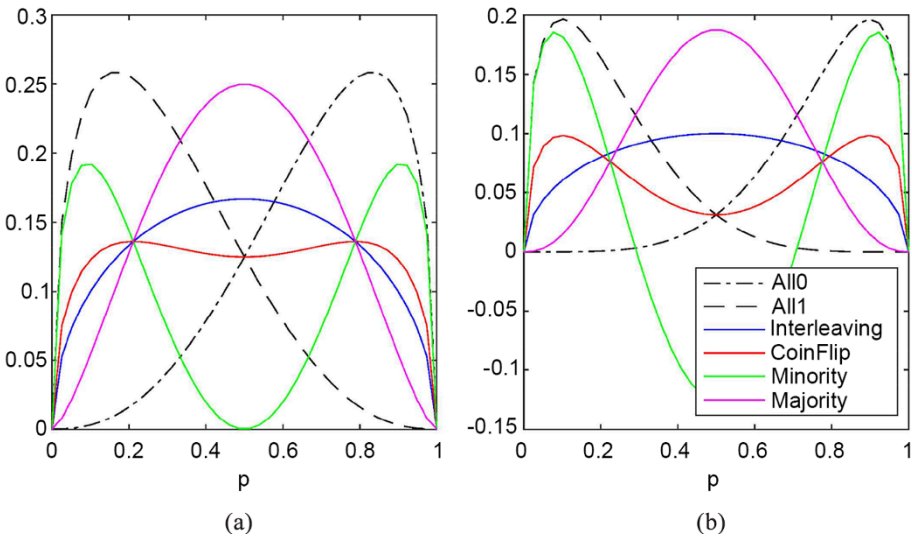


Figure 6.3. *The function $p \to \mathbb{E}(\bar{U}(X_{\mathsf{tra}}, Y, p))$ for c = 3 (a) and c = 5 (b)*

6.2.5.2. *Latest developments*

Robustness and completeness were proved by Tardos (2003) for code length $m = 100c^2 \log {}^{n}/_{\eta_R}$. This constant 100 is unusual, but understandable: a code length proportional to $c^2 \log {}^{n}/_{\eta_R}$ was the Holy Grail at the time. Our very simplified proof finds a much smaller multiplicative constant π^2, which is only twice as large as the best result ${}^{\pi^2}/_{2}$ (valid asymptotically when $c \to \infty$ (Škorić *et al.* 2008a)).

6.2.5.3. *Worst case attack*

The strength of a collusion strategy is measured by its ability to decrease $\mathbb{E}\,(S_{\mathsf{tra}})$, which for equation [6.29] is proportional to the quantity $\Pi(1 - t_c) - \Pi(t_c)$. For a given collusion size, it is the "minority vote" attack that is the most dangerous for this

score function and density f (see equation [6.30]). The article by Škorić *et al.* (2008a) provides the same conclusion.

6.2.6. *The problems*

This demonstration has three problems.

6.2.6.1. *Too restrictive*

This educational study is simple, but only works for $\eta_C = 1/2$ and asymptotically for $n \to \infty$. A more useful proof uses an upper bound of $\mathbb{P}_{\mathsf{fn}}$, which generally depends on $\mathbb{V}(S_{\mathsf{tra}})$. This variance is bounded by m since $\mathbb{E}(\bar{U}(X_{\mathsf{tra}}, Y, p)^2) = 1, \forall p \in [t, 1-t]$. A very loose bound, such as the Chebyshev inequality, gives a good result. Suppose that $\mathbb{P}_{\mathsf{fn}}$ must be less than ϵ, then for all values of n:

$$\underline{m}(n, c, \eta_R, \epsilon) = \pi^2 c^2 \left(\log \frac{n}{\eta_R} + \frac{1}{3\epsilon} + O\left(\sqrt{\frac{1}{\epsilon} \log \frac{n}{\eta_R}} \right) \right) \qquad [6.36]$$

In fact, the most critical part is the proof of the robustness, where the finest possible bound is required, while the proof of completeness is less challenging.

6.2.6.2. *Ill-posed problem*

This problem is common in any theoretical study, starting from the specifications set out in section 6.1. The size c of the collusion is an integral part of the problem, so much so that the code depends on c. This is true for the parameters (m, t, τ). However, the true size of the collusion is not known, neither to the codeword generator nor the accusation algorithm. One way to work around the problem is to assume that $c \leq c_{\max}$. This blinds the code to the true value of c. The robustness is always guaranteed, but the completeness is proven under the assumption that the true collusion size is less than, or equal to, $c_{\max}$.

6.2.6.3. *Function score that is too constrained*

Here is a problem with the score function $U(\cdot)$ (see equation [6.24]). Its choice is too constrained. It must provide separability (the score of a traitor is statistically larger than the score of someone innocent) and, at the same time, the independence regarding the collusion strategy (the distributions of S_{inn} and S_{tra} are almost set, at least for the first and second statistical moments – and using an approximation for S_{tra}). This second property is crucial in order to find a universal threshold, τ, because of equation [6.27]. Disconnecting the separability and independence from the nuisance parameters provides more freedom to find more discriminating score functions. However, finding a threshold becomes more difficult.

6.3. Tardos and his successors

Many papers have been published on Tardos codes. This section provides a review by summarizing the known results on the length of code needed, as part of section 6.2.4, other criteria and extensions.

6.3.1. *Length of the code*

The most known improvement on code length was provided by Škorić *et al.* (2008a), which mirrored the initial score function by dividing the length by two. This score function (see equation [6.24]) is now the norm.

The following works have kept the same definition of P and $U(\cdot)$, as in Blayer and Tassa (2008), Škorić *et al.* (2008a), and Laarhoven and de Weger (2014), but have developed a finer analysis to reveal smaller constants. The latest developments on the constant $\kappa := {}^{m}/_{(c^2 \log {}^{n}/_{\eta_R})}$ are as follows:

– asymptotically when $c \to \infty$, $\kappa \to {}^{\pi^2}/_{2} \approx 4,93$ (Škorić *et al.* 2008a);

– for a large collusion, $\kappa = {}^{\pi^2}/_{2} + O(c^{-1/3})$ (Laarhoven and de Weger 2014);

– for all $c > 2$, $\kappa = 10.89$.

6.3.2. *Other criteria*

6.3.2.1. *Code length*

This is the ultimate criterion. The authors of an article on Tardos codes propose other choices of P and $U(\cdot)$, and/or other bounds for error probabilities to deduce a necessary code length, depending on (c, n, η_R, η_C), smaller than latest developments. This is what has been performed in the previous articles (Tardos 2003, 2008; Blayer and Tassa 2008; Laarhoven and de Weger 2014).

6.3.2.2. *Signal-to-noise ratio*

Replacing the threshold τ by $\mathbb{E}(S_{\mathsf{tra}})$, we note the following condition: $\mathbb{P}(S_{\mathsf{inn}} > \mathbb{E}(S_{\mathsf{tra}})) = {}^{\eta_R}/_{n}$, which means that someone innocent should not have a score as high as the typical score of a traitor. A rough approximation is to say that S_{inn} follows a Gaussian law, since it is the sum of a large number of independent r.v. (but not of the same law, hence Lindeberg's condition). So $m \approx \rho^{-1} \left(\Phi^{-1}(1 - {}^{\eta_R}/_{n})\right)^2$, where ρ is similar to a signal-to-noise ratio for a symbol coming out of the score function:

$$\rho := \frac{(\mathbb{E}(U(X_{\mathsf{tra}}, Y, P)) - \mathbb{E}(U(X_{\mathsf{inn}}, Y, P)))^2}{\mathbb{V}(U(X_{\mathsf{inn}}, Y, P))} \qquad [6.37]$$

This criterion is used in the articles (Furon *et al*. 2008; Škorić *et al*. 2008a, 2008b). Special attention is given to the attack that minimizes ρ to define the code length needed to fight against a worst-case collusion. Note that the central limit theorem explains that ρ is a citerion (the larger it is, the better), but this theorem cannot be used for the proof of robustness.

6.3.2.3. *Theoretical rates*

A third criterion is the attainable rate of a simple decoder $R^{(S)}(P;\boldsymbol{\theta}_c)$, which is defined as mutual information $I(Y;X_{\mathsf{tra}}|P,\boldsymbol{\theta}_c)$ between the symbols of the pirated series and symbols of the traitor's codeword. Section 6.4.2 provides more detail on this quantity from information theory, and defined by the equation [6.41]. As the decoder performs a hypothesis test for each user (is the user j innocent or guilty?), this quantity is the maximum among all of the simple decoders of the error exponent E_{fp} of the probability of wrongly accusing an innocent person (for a given false negative probability η_C):

$$E_{\mathsf{fp}} := -\lim_{m\to\infty}\frac{1}{m}\log \mathbb{P}_{\mathsf{fp}} \qquad [6.38]$$

If it is not zero, $\mathbb{P}_{\mathsf{fp}}$ can converge to 0, as m approaches infinity as fast as $e^{-mR^{(S)}(P;\boldsymbol{\theta}_c)}$. In fact, a Chernoff type upper bound decreases toward 0 for the best simple decoder. Therefore, we guarantee $\mathbb{P}_{\mathsf{fp}} \leq \eta_R/n$ if:

$$m \geq \frac{1}{R^{(S)}(P;\boldsymbol{\theta}_c)}\log\frac{n}{\eta_R} \qquad [6.39]$$

This makes the quantity $R^{(S)}(P;\boldsymbol{\theta}_c)$ a criteria (the larger it is, the better it is). Note that this is a theoretical result: the best score function achieving this error exponent is given by the Neyman–Pearson lemma (see section 6.4.1). However, this score function is the log-likelihood (see equation [6.40]), which requires the expression of $\mathbb{P}(Y=y, X_{\mathsf{tra}}=x|p,\boldsymbol{\theta}_c)$ and $\mathbb{P}(Y=y|p,\boldsymbol{\theta}_c)$. This is a problem since these probabilities depend on the collusion strategy $\boldsymbol{\theta}_c$, which is unknown by the decoder initially. On the other hand, any simple decoder has a lower error exponent than $R^{(S)}(P;\boldsymbol{\theta}_c)$. Therefore, the collusion strategy minimizing this quantity will be a powerful attack for any simple decoder.

This criterion has been generalized to joint decoders. These decoders calculate a score per user group of size ℓ. These decoders are theoretically more powerful, but their complexity proportional to the number of groups of size ℓ among n is much bigger. In fact, for $\ell = c$, this leads to the concept of capacity (Moulin 2008; Huang and Moulin 2012, 2014).

6.3.3. *Extensions*

Literature on Tardos codes proposes three families of the following extensions.

6.3.3.1. *Binary codes*

The idea is to keep Tardos codes binary, but the pirated series $\mathbf{y}$ is not necessarily binary. This is an assumption that makes sense in multimedia applications, where the codeword symbols are hidden in the content using a watermark technique. This watermark layer is not perfect when traitors degrade the quality of pirated content (by lossy compression in video, for example). Deletions (noted by $\times$ symbols) or decoding errors (decoding a "1" while "0" was hidden in the content block) produce *Marking Assumption* violations.

The collusion strategy itself can be more complex than just copying and pasting blocks. Traitors can also merge "1" and "0" blocks (Schaathun 2014). The simplest is to merge an average pixel-to-pixel of images to video blocks. This will disrupt the decoding of the watermark, which can result in a "1" or "0" symbol, in $\times$ deletion (as in the *unreadable digit model* (Boneh and Shaw 1998; Huang and Moulin 2014) or the *general digit model* (Safavi-Naini and Wang 2001)), or a double detection (denoted by d): the watermarking decoder detects the merging of a block containing a "1" and a block containing "0". In other words, the pirated series is no longer binary: $\mathbf{y} \in \{0, 1, \times, \mathsf{d}\}^m$.

Another extension is to work with a watermark decoder that gives soft outputs (Kuribayashi 2010, pp. 103–117; Meerwald and Furon 2012, section V.B), like the likelihoods $(l_{1,i}, l_{0,i})$ that the ith block contains the symbols "1" and "0".

From a theoretical point of view, the difficulty is extending the *Marking Assumption*: we must limit the power of traitors by constraining the set of possible collusion strategies. For example, if $\theta_{c,\sigma}(\times) = 1, \forall \sigma \in \{0, \ldots, c\}$, then the pirated series is just a series of deletions that cancels any chance of identifying the traitor. From a practical point of view, the difficulty is creating score functions for these new symbols, $\times$ and d, (Pérez-Freire and Furon 2009, section 4) or these soft outputs (Meerwald and Furon 2012, section V.B).

6.3.3.2. *q-ary codes*

Another family of extensions offers the construction of code on a q-ary alphabet: $\mathbf{x}_j \in \{0, 1, \ldots, q-1\}^m$. The *Marking Assumption* stays the same: when the traitors all have the same symbol, it is found in the pirated series. The difficulty is modeling the collusion strategy for detectable positions. The models *wide-sense version* (or *arbitrary digit model*) (Barg *et al.* 2003) and *narrow-sense version* are two examples.

The generalization of the probabilistic model $\boldsymbol{\theta}_c$ in the case of q-ary was never considered since its size is too large: $\boldsymbol{\theta}_c$ stores all of the probabilities $\mathbb{P}(y|\mathbf{t})$ for $y \in$

$\{0, 1, \ldots, q-1\}$ and the type $\mathbf{t}$ (or empirical distribution, histogram of frequencies) of the traitors' symbols. It turns out that the number of types on c symbols of an alphabet of size q can grow as $\binom{c+q-1}{q-1}$, i.e. $O(c^{q-1})$. Simpler models are used instead (Boesten and Škorić 2011, 2013; Huang and Moulin 2014; Škorić and Oosterwijk 2015).

6.3.3.3. *Joint decoders*

The final extension is the concept of joint decoders. As mentioned before, these decoders calculate a score for a group of ℓ users. Information theory shows that score functions can be more powerful: the ℓ-uplets made up of only traitors will statistically have a higher score than ℓ-uplets of innocent people. However, the number of scores to calculate is of the order $O(n^\ell)$, which quickly becomes prohibitive in calculation time.

Proceeding iteratively gives a tractable complexity. At the first iteration, simple scores are calculated, that is, one score per user. The users who have the greatest scores are suspected, and the others are innocent. The second iteration calculates the scores for all pairs among the suspected users, and so forth. The idea is to reduce the number of suspects so that each iteration calculates a more or less constant number of scores, although ℓ gradually increases.

Another idea is that an identified traitor is a snitch. Knowing the identity of certain traitors and their codewords, makes it possible to design more discriminating score functions. So, an iterative decoder can accuse someone if their score is high enough, and integrate their codeword into the score function, which will allow more separate, new scores to be calculated to find a second traitor more easily, and so on.

6.4. Research of better score functions

Among all of the further developments of the Tardos code, this section presents new, more powerful score functions for binary codes with the model described previously.

These score functions are more powerful since they are less constrained than the function of equation [6.24]. In fact, they give more separate expected scores of innocent people and traitors, but these do not have independent distributions of the collusion strategy. So, it is not possible to find a universal threshold like before. We return to this problem in section 6.5.

6.4.1. *The optimal score function*

The simple decoder corresponds to a series of n hypotheses tests to decide if the user $j \in [n]$ is innocent or a traitor. Decision theory gives us the optimal test, that is

to say the one that minimizes $\mathbb{P}_{\text{fn}}$ for a set $\mathbb{P}_{\text{fp}}$. This is the Neyman–Pearson test. For this, we need to have a statistical model of the hypotheses to be tested.

– If the user is innocent, then the symbols of their codeword are independent of the decoded symbols in the pirated content: $\Pr(Y = y, X = x|p, \boldsymbol{\theta}_c) = \Pr(Y = y|p, \boldsymbol{\theta}_c)p^x(1-p)^{(1-x)}$.

– If the user is a traitor, then the symbols correlate with those of the pirated copy: $\Pr(Y = y, X = x|p, \boldsymbol{\theta}_c) = \Pr(Y = y|x, p, \boldsymbol{\theta}_c)p^x(1-p)^{(1-x)}$.

The optimal score function is simply the log-likelihood of the ratio of these two probabilities (we also call this accusation a maximum likelihood decoder):

$$\begin{aligned} U(x, y, p) &= \log \frac{\Pr(Y = y|x, p, \boldsymbol{\theta}_c)}{\Pr(Y = y|p, \boldsymbol{\theta}_c)} \\ &= y \log \frac{\Pi_x(p; \boldsymbol{\theta}_c)}{\Pi(p; \boldsymbol{\theta}_c)} + (1-y) \log \frac{1 - \Pi_x(p; \boldsymbol{\theta}_c)}{1 - \Pi(p; \boldsymbol{\theta}_c)} \end{aligned} \quad [6.40]$$

The expressions of $\Pi(\cdot)$ and $\Pi_x(\cdot)$ are given by equations [6.11] and [6.15]. Here, we have added the notation $\boldsymbol{\theta}_c$ to make it clear that these functions are calculated for a given collusion strategy. In fact, this score function is only optimal if the collusion strategy is definitely $\boldsymbol{\theta}_c$. However, when accusing, this is never known. So this optimal score function is elusive in practice. Its interest is theoretical to measure the difference between a heuristic score function and optimality if we know the collusion strategy. Applying this test blindly proves to be dangerous: it will be optimal if the traitors really have used this strategy, it will possibly be disastrous for another strategy.

6.4.2. *The theory of the compound communication channel*

Let us pick out one specific traitor. Collusion can be seen as the transmission of its codeword through a binary communication channel. This channel is characterized by the transition probabilities $\Pi_x(p; \boldsymbol{\theta}_c)$ that depend on a state of the channel $\boldsymbol{\theta}_c$.

This is exactly the scenario of the compound communication channel, where a codeword is transmitted through a channel which is not totally unknown. The decoder knows that this channel belongs to a certain family, but it does not know which member of this family transmitted the codeword. Under certain conditions, it has recently been shown that a good strategy is to expect the worst (Abbe and Zheng 2010). Some channels produce more transmission errors than others. We can measure their quality by mutual information $I(Y; X)$. The worst channel of the family is the one with the lowest measurement. If this is strictly positive, then a Neyman–Pearson decoder designed for this channel is theoretically justified. If the channel that was

used is actually the worst, then the decoder is optimal. If the channel that was used is not the worst, then the decoder will at least be as powerful as the worst case.

In Tardos codes, the conditions of this compound communication channel theorem are met (Meerwald and Furon 2012). To apply this result, we need to find the worst collusion strategy, that is, the one that minimizes mutual information, as mentioned in section 6.3.2, $R^{(S)}(P;\boldsymbol{\theta}_c) = I(Y;X_{\mathsf{tra}}|P,\boldsymbol{\theta}_c)$.

Knowing that the more traitors there are, potentially the more powerful they are (also many of us are often stupid), the decoder will assume that $c \leq c_{\max}$. Prosaically, the formula for the length of the necessary code (see equation [6.36]) says that it is pointless to want to accuse if there are more traitors. So the worst collusion strategy $\check{\boldsymbol{\theta}}_{c_{\max}}$ is one that minimizes $R^{(S)}(P;\boldsymbol{\theta}_c)$ for $c = c_{\max}$. We will find this collusion strategy numerically, then we will put it into equation [6.40] to get a score function. We must therefore minimize the function:

$$R^{(S)}(P;\boldsymbol{\theta}_c) := I(Y;X_{\mathsf{tra}}|P,\boldsymbol{\theta}_c) = \mathbb{E}_P\left(I(Y;X_{\mathsf{tra}}|P=p,\boldsymbol{\theta}_c)\right) \qquad [6.41]$$

with:

$$I(Y;X_{\mathsf{tra}}|P=p,\boldsymbol{\theta}_c) = \sum_{x\in\{0,1\}} p^x(1-p)^{1-x}$$
$$\times\left(\Pi_x(p;\boldsymbol{\theta}_c)\log\frac{\Pi_x(p;\boldsymbol{\theta}_c)}{\Pi(p;\boldsymbol{\theta}_c)} + (1-\Pi_x(p;\boldsymbol{\theta}_c))\log\frac{1-\Pi_x(p;\boldsymbol{\theta}_c)}{(1-\Pi(p;\boldsymbol{\theta}_c))}\right)$$

Minimizing this function is not so easy when c is large. A useful simplification observes that the "interleaving" strategy quickly becomes (as c increases) one of the worst strategies in the $\boldsymbol{\Theta}_c$ set. So the idea is to inject this model into equation [6.40] to get a new function score. This is what Laarhoven suggests (2014, equation (2)). The optimal score function for the strategy $\boldsymbol{\theta}_{c_{\max}} = (0, {}^1\!/_{c_{\max}}, \ldots, {}^{(c_{\max}-1)}\!/_{c_{\max}}, 1)$ simplifies to:

$$U(x,y,p) = \begin{cases} \log\left(1+\frac{1}{c_{\max}}\left(\frac{1-p}{p}\right)^{(2y-1)}\right) & \text{if } x=y \\ \log\left(1-\frac{1}{c_{\max}}\right) & \text{if } x\neq y \end{cases} \qquad [6.42]$$

Oosterwijk *et al.*'s score function (Oosterwijk *et al.* 2013, equation (43)) is in fact a first-order approximation when $c_{\max}$ is large:

$$U(x,y,p) \propto \begin{cases} \left(\frac{1-p}{p}\right)^{2y-1} & \text{if } x=y \\ -1 & \text{if } x\neq y \end{cases} \qquad [6.43]$$

Just like for the Tardos–kori decoder, the parameter $t > 0$ is needed to limit its amplitude. This score function is optimal, in that it produces a saddle point equilibrium for the signal-to-noise ratio criterion (see equation [6.37]) (Oosterwijk *et al*. 2013, th. 2).

6.4.3. *Adaptive score functions*

Adaptive score functions proceed in two steps. They analyze the pirated series $\mathbf{y}$, knowing the secret $\mathbf{p}$, to deduce an inference as to the collusion strategy. Then, they calculate a score using a score function adapted to what it has learned. This approach is sometimes called "learn and match".

6.4.3.1. *Estimation of the collusion strategy knowing c*

If the decoder knows the size c but not the strategy, it can estimate $\boldsymbol{\theta}_c$ from the observations $(\mathbf{y}, \mathbf{p})$. This is possible, according to property 2 of the function $\Pi(\cdot)$ (see equation [6.13]). The maximum likelihood estimator is given by the following formulation, which is solved numerically:

$$\hat{\boldsymbol{\theta}}_c = \arg\max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_c} \sum_{i=1}^{m} y_i \log(\Pi(p_i;\boldsymbol{\theta})) + (1-y_i)\log(1-\Pi(p_i;\boldsymbol{\theta})) \qquad [6.44]$$

Another possibility is the expectation–maximization algorithm, since the distribution of Y_i knowing p_i is a mix of Bernoulli distributions. Let $\Sigma_i = \sum_{j\in\mathcal{C}} X_{j,i}$, that is, the number of "1" symbols that the traitors have at the index i. This variable will be a latent variable. The expectation–maximization algorithm starts by choosing a random collusion strategy, $\hat{\boldsymbol{\theta}}_c^{(0)} \in \boldsymbol{\Theta}_c$, and then iterates through the following steps.

6.4.3.1.1. Step E

At the iteration t, this step evaluates the distribution of $\Sigma_i \in \{0,\ldots,c\}$ for an estimated strategy $\hat{\boldsymbol{\theta}}_c^{(t-1)}$, because of the Bayes' rule:

$$\begin{aligned} T_{i,\sigma}^{(t)} &:= \Pr\left(\Sigma_i = \sigma | y_i, p_i, \hat{\boldsymbol{\theta}}_c^{(t-1)}\right) \\ &= \Pr\left(Y_i = y_i | \Sigma_i = \sigma, p_i, \hat{\boldsymbol{\theta}}_c^{(t-1)}\right) \frac{\Pr(\Sigma_i = \sigma | p_i)}{\Pr\left(Y_i = y_i | p_i, \hat{\boldsymbol{\theta}}_c^{(t-1)}\right)} \\ &= (\hat{\theta}_{c,\sigma}^{(t-1)})^{y_i}(1-\hat{\theta}_{c,\sigma}^{(t-1)})^{1-y_i} \\ &\times \frac{\Pr(\Sigma_i = \sigma | p_i)}{\sum_{\sigma'=0}^{c}(\hat{\theta}_{c,\sigma'}^{(t-1)})^{y_i}(1-\hat{\theta}_{c,\sigma'}^{(t-1)})^{1-y_i}\Pr(\Sigma_i = \sigma' | p_i)} \end{aligned} \qquad [6.45]$$

where $\Pr(\Sigma_i = \sigma | p_i)$ is given by equation [6.10].

6.4.3.1.2. Step M

At the t iteration, this step gives a new estimate $\hat{\boldsymbol{\theta}}_c^{(t)}$ of the collusion strategy knowing the probabilities $\{T_{i,\sigma}^{(t)}\}$ of the latent variables. This is done by maximizing the expectation of the log-likelihood: $\hat{\boldsymbol{\theta}}_c^{(t+1)} = \arg\max Q(\boldsymbol{\theta}_c)$ with:

$$Q(\boldsymbol{\theta}_c) = \sum_{i=1}^{m} \sum_{\sigma=0}^{c} T_{i,\sigma}^{(t)} \log(\Pr(Y_i = y_i | \Sigma_i = \sigma, \boldsymbol{\theta}_c)) \quad [6.46]$$

The $(c-1)$ parameters $\hat{\theta}_{c,1}, \ldots, \hat{\theta}_{c,c-1}$ are estimated by maximizing this quantity, while $\hat{\theta}_{c,0}^{(t)} = 1 - \hat{\theta}_{c,c}^{(t)} = 0$, according to the *Marking Assumption* (see equation [6.3]). With the second derivative always being negative, maximizing means canceling the gradient given by:

$$\frac{\partial Q(\boldsymbol{\theta}_c)}{\partial \theta_{c,\sigma}} = \sum_{i=1}^{m} T_{i,\sigma}^{(t)} \left(\frac{y_i}{\theta_{c,\sigma}} - \frac{1 - y_i}{1 - \theta_{c,\sigma}} \right) \quad [6.47]$$

So:

$$\hat{\theta}_{c,\sigma}^{(t)} = \frac{\sum_{i=1}^{m} y_i T_{i,\sigma}^{(t)}}{\sum_{i=1}^{m} T_{i,\sigma}^{(t)}} \quad [6.48]$$

The algorithm iterates these two steps until the maximum of the function $Q(\cdot)$ no longer increases. This expectation–maximization algorithm can be generalized to take deletions or double detections into account (see section 6.3.3.1; (Furon *et al.* 2012, section V)).

6.4.3.2. *Size of the unknown collusion*

If the collusion strategy was correctly estimated, the score function used would be the likelihood ratio calculated for $\boldsymbol{\theta}_c$, that is, the optimal score function. However, the collusion size is unknown when accusing, which prevents c and $\boldsymbol{\theta}_c$ (see section 6.2.2) being identified. Compared with the theory of the compound communication channel of section 6.4.2, it reverses the situation.

In fact, assume that the collusion chooses the strategy $\boldsymbol{\theta}_c$, and consider the set of strategies which give the same distribution of the symbols in the pirate series. Note this set, $\mathcal{E}(\boldsymbol{\theta}_c) = \{\boldsymbol{\theta} | \Pi(p; \boldsymbol{\theta}) = \Pi(p; \boldsymbol{\theta}_c), \forall p \in (0,1)\}$. This set is not restricted to the singleton $\{\boldsymbol{\theta}_c\}$ (hence the error of identifiability). In fact, we can show that for any integer $c' > c$, there is a strategy $\tilde{\boldsymbol{\theta}}_{c'}$ of size c', which "mimics" $\boldsymbol{\theta}_c$: $\mathcal{E}(\boldsymbol{\theta}_c) \cap \boldsymbol{\Theta}_{c'} = \{\tilde{\boldsymbol{\theta}}_{c'}\}$.

Suppose that $c < c_{\max}$, which means $\mathcal{E}(\boldsymbol{\theta}_c) \cap (\cup_{c=1}^{c_{\max}} \boldsymbol{\Theta}_c)$ is not empty. It turns out that the compound communication channel theory can be applied to this family of strategies (Meerwald and Furon 2012, annex), and that the worst strategy in this family is $\tilde{\boldsymbol{\theta}}_{c_{\max}}$. This theoretically justifies the score function, set by the following $c_{\max}$: (1) estimate $\tilde{\boldsymbol{\theta}}_{c_{\max}}$ by maximum likelihood (see equation [6.44]) or by expectation–maximization algorithm (see equation [6.48]). By saying that the collusion size is $c_{\max}$, we denote this estimate; (2) use the score function equation [6.40] where $\boldsymbol{\theta}_c = \hat{\boldsymbol{\theta}}_{c_{\max}}$.

6.4.4. *Comparison*

We have seen several alternative score functions to the Tardos-Škorić one (see equation [6.24]). Some are fixed, and others adapt to the pirated series received. Here is an experimental protocol to compare them in the spirit of the proof in section 6.2.4, where $\mathbb{P}_{\mathsf{fn}} = 1/2$. We choose a certain size and a certain collusion strategy. We generate a secret sequence, $\mathbf{p}$, and many codewords knowing $\mathbf{p}$, we choose c of them, form a pirated series, and calculate the traitors' and innocent people's scores. We choose a threshold τ as the median of the traitors' scores to have $\mathbb{P}_{\mathsf{fn}} \approx 1/2$ (half of the traitors are accused). Finally, we estimate $\mathbb{P}_{\mathsf{fp}} = \mathbb{P}(S_{\mathsf{inn}} > \tau)$.

Figure 6.4 gives a comparison of these score functions for the classical collusion strategies from section 6.1.4. Here, the quantity indicated is $|\log_{10}(\hat{\mathbb{P}}_{\mathsf{fp}})|$, the bigger it is, the more the scores of the innocent and traitors are statistically separated, and the better the score function. The Tardos-Škorić function has almost the same performance from one strategy to another. This is the sense of his invention: the distributions of the scores S_{inn} and S_{tra} are independent of $\boldsymbol{\theta}_c$. Its worst attack is minority voting (see section 6.2.4). The performances of the "Optimal" score function reveal the difficulty of the attack. So, "interleaving" and "coin flip" are the most dangerous collusion strategies for $c = 5$ (among those tested). The "Compound" and "Laarhoven" score functions are equal, with the major weakness of the majority vote for the Compound, and the minority vote for the Laarhoven score function. Finally, the adaptive score function has the performance closest to the optimal. However, it is more complex due to the estimation of $\hat{\boldsymbol{\theta}}_{c_{\max}}$.

6.5. How to find a better threshold

Finding a threshold for any score function is not easy because the distribution of an innocent person's score, S_{inn}, depends on the collusion strategy (except for the Tardos-Škorić strategy; equation [6.24]), which once again is unknown on accusation. This is illustrated by Figure 6.5.

One approach involves refusing to find a universal threshold which would guarantee a false positive probability, whatever the size and strategy of the collusion,

and whatever the secret $\mathbf{p}$. In practice, on the other hand, it is enough to find a threshold for a given realization $\mathbf{p}$, and for a decoded pirated series $\mathbf{y}$. Even in these conditions, the score S_{inn} stays a random variable, since the codewords of the innocent people were created randomly by nature: in equation [6.25], the symbol $x_{j,i}$ has been replaced by the r.v. $X_{\text{inn},i} \sim \mathcal{B}(p_i)$.

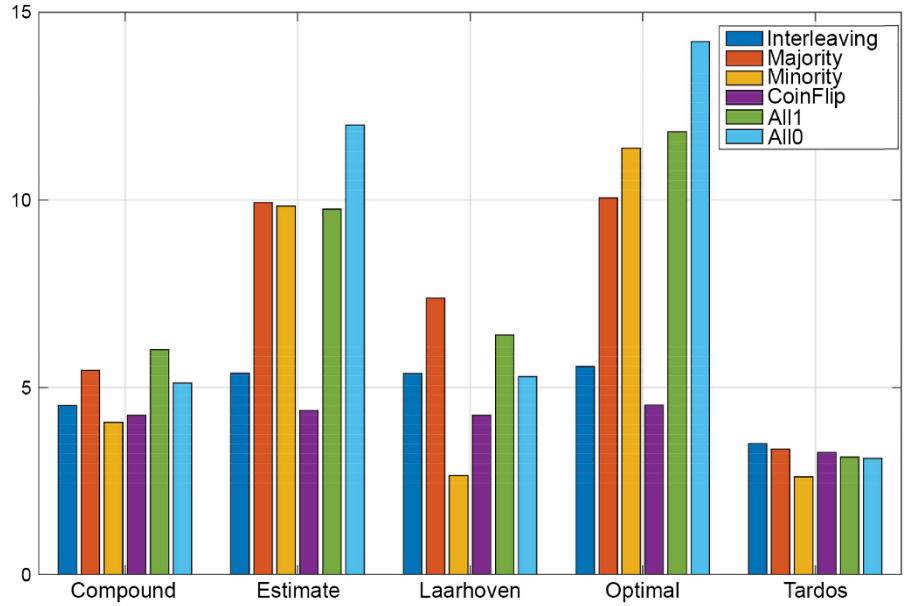


Figure 6.4. *The quantity* $|\log_{10}(\mathbb{P}_{\text{fp}})|$ *for* $\mathbb{P}_{\text{fn}} \approx 1/2$ *for the six collusion strategies in section 6.1.4 and the five following score functions: "Compound" [6.41], "Estimate" (section 6.4.3.2), "Laarhoven" [6.42], "Optimal" [6.40], and "Tardos" [6.24]. With m = 768, c = 5,* $c_{\max} = 10$

So we seek a specific threshold $\tau(\mathbf{p}, \mathbf{y})$, such as $\mathbb{P}_{\text{fp}} = \Pr(S_{\text{inn}} > \tau(\mathbf{p}, \mathbf{y}))$. Upper bounds give a threshold value, guaranteeing $\mathbb{P}_{\text{fp}} < \mathbb{P}_{\text{FP}}/n$, for example Kearns–Saul and Cramer–Chernoff inequalities (Furon and Desoubeaux 2014). However, it is difficult to know if these bounds are tight. If this is not the case, the threshold value will be very high and we risk not identifying any traitor.

Another possibility is estimating this threshold by a random Monte Carlo simulation. We generate a large number, n', of new codewords. As these are generated after receiving $\mathbf{y}$, their associated scores (for this pair $(\mathbf{y}, \mathbf{p})$) are likely to be scores of innocent people. An estimate of $\tau(\mathbf{y}, \mathbf{p})$ is the highest score $\lfloor n' \times \mathbb{P}_{\text{FP}}/n \rfloor$. So, a fraction $\mathbb{P}_{\text{FP}}/n$ of n' scores are above this threshold. The difficulty is that n' must be greater than $n/\mathbb{P}_{\text{FP}}$ (by several orders of magnitude for

good precision). If $n \approx 10^6$ and $\mathbb{P}_{\mathsf{FP}} \approx 10^{-3}$, we must generate a hundred billion codewords. There are stochastic simulations suitable for estimating such small probabilities. This area of numerical probabilities is called the study of rare events. So, we can estimate the threshold $\tau(\mathbf{y}, \mathbf{p})$ by *importance sampling* or by *importance splitting*, which has been implemented in Cérou *et al.* (2008) and Furon and Desoubeaux (2014).

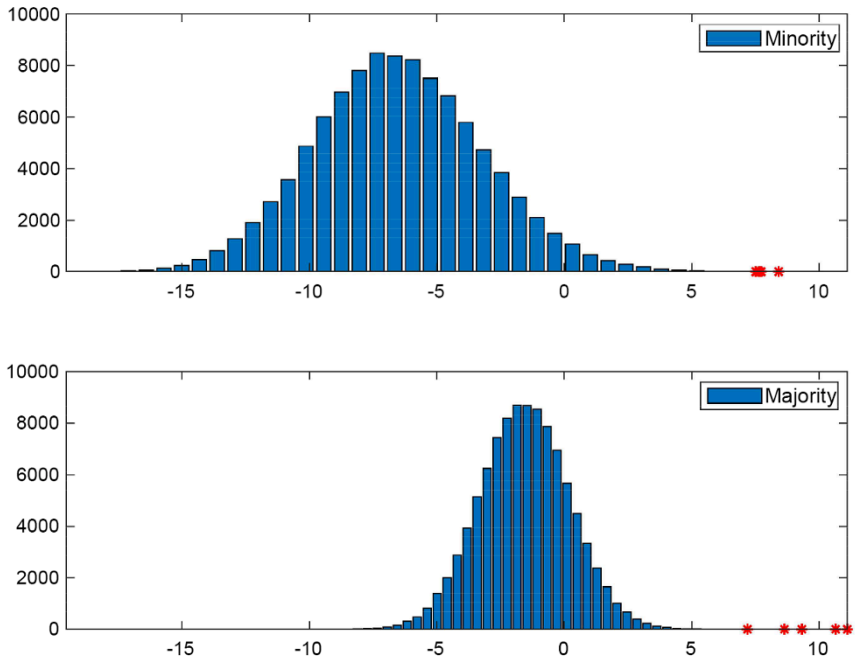


Figure 6.5. *Histograms of innocent people's scores for the "Laarhoven score function" [6.42] and two collusion strategies, "minority vote" and "majority vote", m = 768, c = 5, c_{max} = 10, n = 10^5. Traitor scores are displayed with red asterisks*

6.6. Conclusion

This chapter shows that binary Tardos codes are now well understood. It is a fully developed technology, which is now beginning to be transferred to industry. Latest developments are not so clear, regarding q-ary codes (see section 6.3.3.2). The required code length stays in $O(c^2 \log n)$, but the multiplicative constant is theoretically smaller. On the other hand, creating score functions that can achieve these theoretical performances remains to be done.

Finally, let us not underestimate the great weakness of Tardos codes: accusation is an exhaustive decoding that calculates the score of all of the users. This can be a problem when n is too big and there is a time limit for accusation.

6.7. References

Abbe, E. and Zheng, L. (2010). Linear universal decoding for compound channels. *IEEE Transactions on Information Theory*, 56(12), 5999–6013.

Amiri, E. and Tardos, G. (2009). High rate fingerprinting codes and the fingerprinting capacity. In *20th Annual ACM-SIAM Symposium on Discrete Algorithms*. ACM, New York.

Barg, A., Blakley, G.R., Kabatiansky, G.A. (2003). Digital fingerprinting codes: Problem statements, constructions, identification of traitors. *IEEE Transactions on Information Theory*, 49(4), 852–865.

Blayer, O. and Tassa, T. (2008). Improved versions of Tardos' fingerprinting scheme. *Designs Codes Cryptography*, 48(1), 79–103.

Boesten, D. and Škorić, B. (2011). Asymptotic fingerprinting capacity for non-binary alphabets. In *Information Hiding*, Tomáš, F., Tomáš, P., Scott, C., Andrew, K. (eds). Springer, Berlin/Heidelberg.

Boesten, D. and Škorić, B. (2013). Asymptotic fingerprinting capacity in the combined digit model. In *Proceedings of the 14th International Conference on Information Hiding*, Springer-Verlag, Berkeley, 255–268.

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

Cérou, F., Furon, T., Guyader, A. (2008). Experimental assessment of the reliability for watermarking and fingerprinting schemes. *EURASIP Journal on Information Security*, 414962.

Fodor, G., Schelkens, P., Dooms, A. (2018). Fingerprinting codes under the weak marking assumption. *IEEE Transactions on Information Forensics and Security*, 13(6), 1495–1508.

Furon, T. and Desoubeaux, M. (2014). Tardos codes for real. In *Workshop on Information Forensics and Security*. IEEE, Atlanta, 7.

Furon, T., Guyader, A., Cérou, F. (2008). On the design and optimization of Tardos probabilistic fingerprinting codes. In *10th Information Hiding Workshop*. LNCS, Santa Barbara.

Furon, T., Guyader, A., Cérou, F. (2012). Decoding fingerprinting using the Markov chain Monte Carlo method. In *Workshop on Information Forensics and Security*. IEEE, Tenerife.

Huang, Y.-W. and Moulin, P. (2012). On the saddle-point solution and the large-coalition asymptotics of fingerprinting games. *IEEE Transactions on Information Forensics and Security*, 7(1), 160–175.

Huang, Y.W. and Moulin, P. (2014). On the fingerprinting capacity games for arbitrary alphabets and their asymptotics. *IEEE Transactions on Information Forensics and Security*, 9(9), 1477–1490.

Kuribayashi, M. (2010). *Tardos's Fingerprinting Code Over AWGN Channel*. Springer, Berlin/Heidelberg.

Laarhoven, T. (2014). Capacities and capacity-achieving decoders for various fingerprinting games. In *Workshop on Information Hiding and Multimedia Security*. ACM, Salzburg.

Laarhoven, T. and de Weger, B. (2014). Optimal symmetric Tardos traitor tracing schemes. *Designs, Codes and Cryptography*, 71(1), 83–103.

Meerwald, P. and Furon, T. (2012). Towards practical joint decoding of binary Tardos fingerprinting codes. *IEEE Transactions on Information Forensics and Security*, 99, 1.

Moulin, P. (2008). Universal fingerprinting: Capacity and random-coding exponents [Online]. Available at: http://arxiv.org/abs/0801.3837.

Oosterwijk, J.-J., Škorić, B., Doumen, J. (2013). A capacity-achieving simple decoder for bias-based traitor tracing schemes. Report 2013/389, Cryptology ePrint Archive.

Pérez-Freire, L. and Furon, T. (2009). Blind decoder for binary probabilistic traitor tracing codes. In *International Workshop on Information Forensics and Security*. IEEE, London, 56–60.

Safavi-Naini, R. and Wang, Y. (2001). Collusion-secure q-ary fingerprinting for perceptual content. In *Security and Privacy in Digital Rights Management*, Sander, T. (ed.). Springer, Berlin/Heidelberg.

Schaathun, H.G. (2014). Attacks on Kuribayashi's fingerprinting scheme. *IEEE Transactions on Information Forensics and Security*, 9(4), 607–609.

Shahid, Z., Chaumont, M., Puech, W. (2013). H.264/AVC video watermarking for active fingerprinting based on Tardos code. *Signal, Image and Video Processing*, 7(4), 679–694.

Škorić, B. and Oosterwijk, J.-J. (2015). Binary and Q-Ary Tardos codes, revisited. *Design Codes Cryptography*, 74(1), 75–111.

Škorić, B., Katzenbeisser, S., Celik, M. (2008a). Symmetric Tardos fingerprinting codes for arbitrary alphabet sizes. *Designs, Codes and Cryptography*, 46(2), 137–166.

Škorić, B., Vladimirova, T., Celik, M., Talstra, J. (2008b). Tardos fingerprinting is better than we thought. *IEEE Transactions on Information Theory*, 54(8), 3663–3676.

Stinson, D.R. and Wei, R. (1998). Combinatorial properties and construction of traceability schemes and frameproof code. *SIAM Journal on Discrete Mathematics*, 11, 41–53.

Tardos, G. (2003). Optimal probabilistic fingerprint codes. In *35th Annual ACM Symposium on Theory of Computing*. ACM, San Diego, 116–125.

Tardos, G. (2008). Optimal probabilistic fingerprint codes. *Journal of the ACM*, 55(2).

Zhao, H.V. and Liu, K.J.R. (2006). Traitor-within-traitor behavior forensics: Strategy and risk minimization. *IEEE Trans. Information Forensics and Security*, 1(4), 440–456.

7

3D Watermarking

Sébastien BEUGNON[1], Vincent ITIER[2] and William PUECH[1]

[1] *LIRMM, Université de Montpellier, CNRS, France*

[2] *CRIStAL, University of Lille, CNRS, IMT Lille Douai, France*

Three-dimensional (3D) meshes are now widely used in industry across a range of fields, for example, video games, medical diagnosis, computer-aided design (CAD) or, more recently, 3D printing. In this context, data hiding techniques are attractive for hiding and preserving information associated with a mesh or a 3D point cloud. For example, it is possible to add the property rights of the model in an industrial context. In a medical setting, patient information can be added for connecting the patient with their data and the preservation of anonymity. The main advantage of data hiding methods, compared to encryption, is that meshes can be used and visualized. 3D mesh encryption is preferred if visual masking is required. In addition, data hiding methods prevent the sharing of multiple files. In this chapter, we present the challenges of these technologies, as well as the constraints linked to the structure of 3D meshes, in particular its impact on the synchronization step. We draw up a recent development, focused on a variety of different approaches. We then present in detail so-called high-capacity methods that allow the embedding of a significant payload, such as the history of meshes, their metadata or their textures. Finally, we discuss the current situation and future areas of research in the field of 3D watermarking.

For a color version of all figures in this chapter, see www.iste.co.uk/puech/multimedia1.zip.

Multimedia Security 1,

coordinated by William PUECH.

Multimedia Security 1: Authentication and Data Hiding,
First Edition. William Puech.

7.1. Introduction

3D objects can be represented digitally in several ways using meshes, implicit surfaces, NURBS or voxels, for example. A 3D mesh is an approximation of the surface of a 3D object, defined using geometric and connectivity information. 3D meshes have become a norm for 3D representation due to their ease of use. On the other hand, 3D scanner systems are more and more common, and 3D printing is now becoming easily accessible, which increases the presence of digital 3D representations in the form of meshes. In this context, data hiding makes it possible to embed additional information into a digital medium in an unnoticeable way, while respecting the initial format. This information can be a secret message, metadata, owner ID or a mark, for example. Data hiding involves modifying a digital medium to add information. In fact, a medium in which information is embedded must be able to be viewed or manipulated using standard software in a standard file format.

The work carried out in the fields of digital images or videos shows that data hiding is an interesting solution to these various problems. Data hiding allows us to embed a message into a 3D mesh in an unnoticeable way. There are several forms of data hiding. Robust watermarking allows owner ID to be embedded for copyright, and this owner ID must be protected, even after modifications. Fragile watermarking makes it possible to check the integrity of a mesh, and it is designed to be altered after certain modifications. There is also high capacity data hiding which makes it possible to hide a large amount of information in a 3D mesh, such as information on creation, semantic content or textures. Data hiding in 3D meshs is a recent area of study. The representation of surfaces in $\mathbb{R}^3$, in the form of a mesh, makes it possible to use these 3D meshes as a medium for a secret message. Most studied meshes are two-manifold triangle meshes. However, specific methods can be designed for other types of meshes or point clouds (points in $\mathbb{R}^3$ with no relationship between them). Therefore, 3D mesh media create new challenges for data hiding. For example, synchronization in 3D meshes, that is, the definition of an order of traversal of the elements of the mesh, is not trivial like in 2D, where the pixels are associated with a regular grid. In addition, in the context of image watermarking, the embedding medium is the pixel, i.e. the pixels are modified to hold additional information. On 3D objects, the embedding medium is larger, and it is possible to modify the position of the points in space or the connectivity of the mesh, for example. Another challenge is generating few distortions on the surface, while being potentially undetectable or robust against attacks.

In this chapter, section 7.2 defines the preliminary notions concerning the aspects of watermarking in general, and the digital representation of 3D objects. Section 7.3 poses the problem of synchronization. In section 7.4, we describe the principle of 3D data hiding. In section 7.5, we present a state-of-the-art method offering interdisciplinary approaches. In section 7.6, we explain the possible improvements to be made in a 3D data hiding process. In section 7.7, we present the results and

compare the different methods. Finally, in section 7.8, we analyze future areas of research and current trends.

7.2. Preliminaries

7.2.1. *Digital watermarking*

Data hiding involves hiding data in a host medium in an unnoticeable way. The different data hiding classes are shown in Figure 7.1. They are organized from the most general to the most specific; each is defined according to the data hiding usage scenario. Cox *et al*. (2007) clearly set the context for data hiding; the methods have two main modules, the embedding and extraction of the message. A synchronization stage is necessary to define the same order between the embedding and extraction steps. Data hiding methods must also be based on Kerckhoffs's principle, which states that the secret of a method cannot depend on the secret of the algorithm, but must only be based on the secrecy of a key (Kerckhoffs 1883).

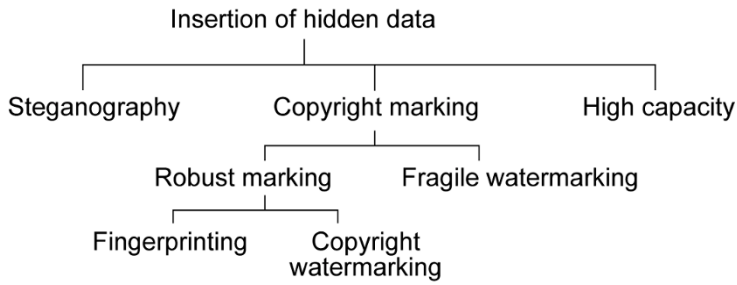


Figure 7.1. *Classification of data hiding methods, based on the work of Petitcolas et al. (1999) and Cheng and Wang (2007)*

First, a secret key is used to secure the synchronization step and define an order on the areas chosen for embedding. The embedding stage makes it possible to hide a message in the media format. For example, the most conventional method consists of modifying the low order bits encoding the pixels in the case of an image medium. Note that, generally, the message is encrypted using the secret key. The extraction module makes it possible to extract the hidden message thanks to the key used during the embedding phase. The message can be reconstructed in two steps that involve finding the embedding areas, and then extracting the bits in order. There are properties inherent to data hiding systems, which face a compromise between imperceptibility, robustness, capacity and security. In general, improving one of these

properties decreases the others. So we must compromise, as illustrated in Figure 7.2, which is dictated by the scenario considered (Figure 7.1). In the majority of cases, the mark must be imperceptible so as not to modify the shape, then the compromise is made between capacity, robustness and security.

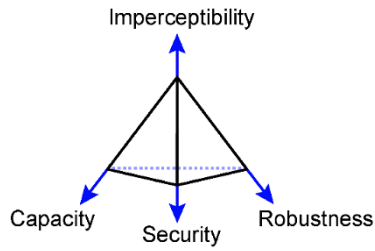


Figure 7.2. *Data hiding trade-off*

7.2.2. *3D objects*

A 3D mesh is an approximation of a continuous surface. A good approximation is possible if the density of vertices is suitable (Botsch *et al.* 2010). A 3D mesh, $M = (V, K)$, is therefore defined by its geometry, V, and its topological connectivity, K. A point cloud represents the geometry of a 3D object. This point cloud is the group of n points or vertices, denoted as V such that:

$$V = \{v_1, \ldots, v_n\}, v_i \in \mathbb{R}^3, 1 \leq i \leq n \quad [7.1]$$

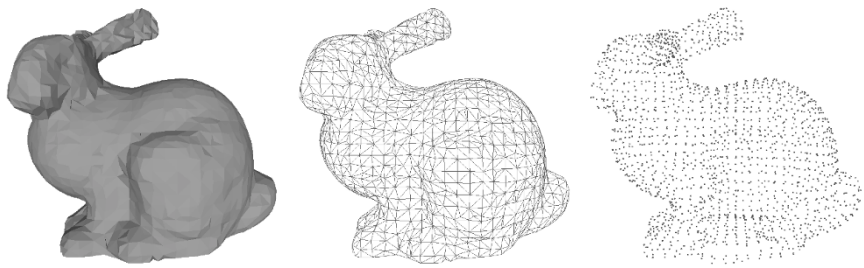
An example of a 3D object, the *Stanford Bunny*[1], is present in low resolution in Figure 7.3. The topological realization is a complex of cells, defining the decomposition of space into cells. An n-cell in a space is defined as a subset, homeomorphic to $E^n = \{x \in \mathbb{R}^n \mid |x| < 1\}$ (Dieck 2008). A mesh is then the geometric realization of a topological representation, which is independent of the space in which the mesh is embedded. We denote the mesh, $M = (V, F, E)$, and all its primitives: all the vertices, V, all the faces F, and all the edges, E, of the mesh. Most processing algorithms require triangle meshes $f \in V \times V \times V$, since they allow a representation to rely on more stringent topological properties. More generally, a surface is defined as a "two-dimensional manifold (2-manifold), dense, connected, orientable and possibly with an edge, embedded in $\mathbb{R}^3$" (O'Neill 2006). A two-dimensional connected surface is 2-manifolds, if at any point x, the nearby

1 Available at: https://graphics.stanford.edu/data/3Dscanrep/.

surroundings can be continuously deformed into a disk (Hoffmann 1989). More simply, we consider the case where an edge is shared by exactly two faces, and where the surface does not intersect itself. This definition of the surface assumes either a mesh created to respect these rules, or a step of preprocessing a point cloud or a polygon soup, to obtain a well-defined surface. 3D meshes are often represented as triangular faces, connected by their edges. Formally, a triangle mesh, M, is a simplicial complex (Dieck 2008), denoted as:

$$M = (V, S) \qquad [7.2]$$

where S is a finite, non-empty set of subset of V, and refers to connectivity.



(a) (b) (c)

Figure 7.3. *The Stanford Bunny in low-resolution, 1,889 vertices and 3,851 faces, a) making the mesh of the 3D object, b) faces and c) point cloud*

Finally, a mesh, M, is a two-dimensional simplicial complex, made up of simplexes:

– a set of vertices, the 0-simplexes of S:

$$S_0 = \{\{v_0\}; \{v_1\}; \dots\}$$

– a set of edges, the 1-simplexes of S:

$$S_1 = \{\{v_0, v_1\}; \{v_0, v_2\}; \dots\}$$

– a set of triangular faces, the 2-simplexes of S:

$$S_2 = \{\{v_0, v_1, v_2\}; \{v_0, v_1, v_3\}; \dots\}$$

A vertex is a single point, which is associated with a single point in the representation space. It is this association that allows the geometric realization of a topological shape.

7.3. Synchronization

3D meshes do not have structures allowing trivial ordering of primitives. A step of ordering the elements to be traversed is necessary. 3D mesh file formats are generally used to store the geometry and topology of a mesh. Since 3D meshes are unstructured, the position of these elements in a file does not affect the result. It is therefore necessary to have an inorder traversal of the elements of a mesh that does not depend on the storage format. The "traversal" of a mesh is used in different processes, such as compression, visualization or data hiding. Depending on the use, the authors choose to define this order from the topology of the mesh or from the geometry. 3D scheduling is a challenge since, unlike image processing or voxel representations, the mesh is not embedded in a regular 2D or 3D grid.

7.3.1. *Traversal scheduling*

Connectivity-based scheduling was first developed for triangle mesh compression systems. The first methods define an inorder traversal of the triangles, such as the "Edgebreaker" method, proposed by Rossignac (1999). This algorithm performs the same traversal of the mesh by passing from triangle to triangle, by their adjacency connection. These adjacency relationships are denoted by symbols, which show entropy coding. The use of a manifold and orientable triangle mesh makes it possible to traverse the sides. This traversal was initially suggested by Ohbuchi *et al*. (1997) and is called the TSPS (*Triangle Strip Peeling Sequence*). This type of sequencing was used in watermarking by Mao *et al*. (2001), Cayre and Macq (2003), and Bajaj *et al*. (1999), which unwind the sides in a spiral. This method is very sensitive to changes to mesh connectivity, such as edge collapse for triangle meshes. To traverse the sides of a mesh from a given side, Lin *et al*. (2013) uses a Breadth-First Search, the order of selection of the sides and vertices is given by a secret key. Huang and Tsai offer another traversal based on the Breadth-First Search but using principal component analysis to select the first side and inorder traversal (Huang and Tsai 2015).

7.3.2. *Patch scheduling*

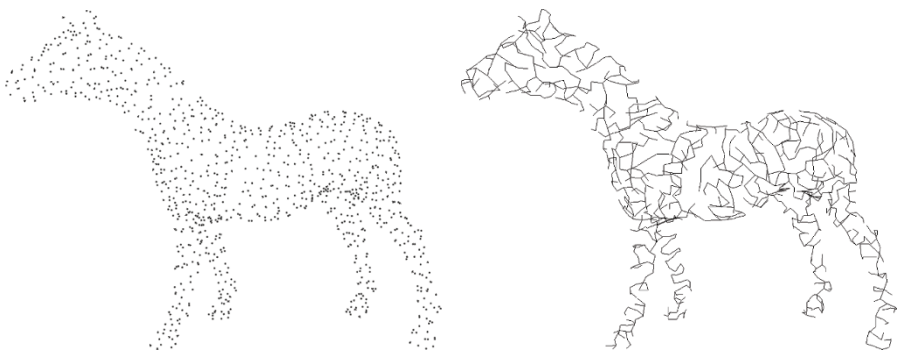
Luo and Bors proposed a watermarking method based on the partitioning of the mesh in regions with equal geodesic distance (Luo and Bors 2011). From a given vertex, the geodesic distances to the other vertices are calculated. Each strip of vertices is used as a medium to embed a bit. The order between the patches is immediately given by the distance to the input vertex. The watermarking method by Wang *et al*. (2011) generates cylindrical patches on the mesh, which are ordered by their location in space.

7.3.3. *Scheduling based on graphs*

Some authors consider meshes as graphs for the scheduling step. In fact, to a complex mesh $M = (V, F, E)$, a graph, $G = (V, E)$, is associated. The mesh is a geometric realization of this topological connectivity, the weighted graph, $G = (V, E, \omega)$, and its associated graph, where $\omega : E \to \mathbb{R}^+$. More generally, if we consider a complete graph $G' = (V, E')$ on the vertices of the mesh, then $\forall i, j$, $i \neq j$, $e_{v_i,v_j} \in E'$, and in particular $E \subseteq E'$. Graph traversal is a complex problem, and we present two types of traversal: spanning trees and Hamiltonian paths.

7.3.3.1. *Spanning trees*

A spanning tree on a graph is a connected and acyclic subgraph included in this graph, which connects all of its vertices. It is possible to define an order starting from a vertex and traversing the tree in width or depth, for example. Spanning trees are used in point cloud compression, since they make it possible to define a structure on the vertices without having to perform a mesh step, then a compression step which, in the end, will be more time consuming. Authors have proposed similar methods for iterative construction of a spanning tree on a point cloud (Gumhold *et al*. 2005; Merry *et al*. 2006). The edge to be added to the sub-tree is chosen to produce the smallest prediction residual for the compression method. Minimum spanning tree (MST) is a well-known problem in graph theory, in particular through the Kruskal algorithm (Kruskal 1956) and Prim's algorithm (Prim 1957). Figure 7.4a presents a point cloud of a 3D object, "horse", and the unique MST built on it, as shown in Figure 7.4b.



(a) (b)

Figure 7.4. *a) 3006 3D point cloud, and b) MST built on the point cloud*

Amat *et al*. (2010) proposed a fragile watermarking method. Scheduling is done by calculating the MST of a point cloud, then thanks to an entry point and scanning scheme, they define an order of traversal of the tree. Tournier *et al*. (2011) proposed

to build a robust Euclidean minimum spanning tree (EMST) in a point cloud in order to define a robust order to the displacement of vertices. The analysis of the stability of the EMSTs is done according to Prim's algorithm, since it makes it possible to build an ordered path from a starting vertex. For each stage of the algorithm, we want to calculate the area in which the vertex, $v_i \in V$, can be moved without changing the EMST connectivity. This area depends on the choice of the first vertex, v_0, and the set of vertices selected before v_i. In Figure 7.5, we show an example of a vertex moved so much that it causes a change in the construction of the EMST. The problem of analyzing the stability of the EMST is a complex problem that is easier to study in the form of a sub-problem:

– at the $i > 0$ stage of Prim's algorithm, we change the position of the vertex, v_i, and we note its new position, v^*;

– the geometric distortion is limited to the ray, $]f(v_i), v_i)$, where $f(v_i)$ is the father of v_i in Prim's algorithm.

In order to keep the same connections in the EMST, the following two conditions must be met:

– $v^* = v_i^*$, v^* is selected at the same stage, i, of Prim's algorithm;

– $f(v^*) = f(v_i^*)$, the father of v^*, is always the father of v_i.

We can calculate the two displacement radii of each vertex, v_i, on the ray, $]f(v_i), v_i)$. The first corresponds to a displacement in the direction of the sub-tree, r_i^-, the second in the opposite direction, r_i^+. The radius of movement is therefore given by $r_i = \min\{r_i^-, r_i^+\}$. Let us denote by $x = \frac{1}{||f(v_i) \cdot v_i||}(v_i - f(v_i))$ the normalized direction vector of the ray, $]f(v_i), v_i)$, if $v^* \in]v_i - r \cdot x; v_i + r \cdot x[$, so the EMST will not be changed at the i stage of Prim's algorithm (Itier *et al.* 2015b).

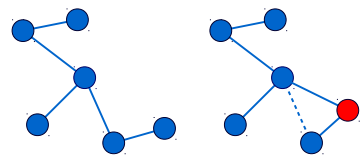


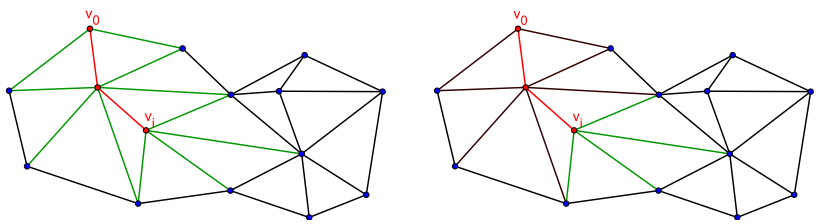
Figure 7.5. *The problem of sensitivity of EMSTs*

7.3.3.2. *Hamiltonian path*

Hamiltonian paths are paths which pass once, and only once, through any point of the graph (Hamilton 1853). A Hamiltonian graph is a graph with a Hamiltonian path. There is no necessary and sufficient condition to find a Hamiltonian path, but there

are many sufficient conditions, given for example by Dirac's theorem (1952), which states that a simple graph with $n \geq 3$ vertices is Hamiltonian if the degree of each vertex is greater than or equal to $n/2$. More generally, the Koenig–Redei theorem states that a complete graph is Hamiltonian. The first observation is to notice that the construction of a Hamiltonian path is less complex than that of an EMST. Furthermore, a Hamiltonian path is more stable in construction, with respect to changes in geometry, than an EMST. In fact, finding the closest neighbor from a single vertex offers less choice than the closest neighbor from a set of vertices given by Prim's algorithm. Figure 7.6 compares the stages of construction of an EMST (Figure 7.6a) and a Hamiltonian path (Figure 7.6b), from a starting vertex. At stage i, the edges in red belong to either the tree or the path, and the edges in green are compared to find v_{i+1}. We find that the number of edge weights to compare at each step is greater in the case of EMSTs.

Finding a minimal Hamiltonian path in a complete graph is also a complex problem that has been proven NP-complete: the traveling salesman's problem. However, finding any Hamiltonian path in a complete graph is a simple problem, it makes it possible to obtain an order in an efficient way in a point cloud, according to the heuristic of the choice of the vertex to be connected. In compression, Gurung *et al.* (2011) proposed a method of ordering triangles by following a Hamiltonian path, like "Edgebreaker". Zhang *et al.* (2013) improved this approach with a better compression rate. Hamiltonian paths depend on the cost given to each edge. For a graph in the $\mathbb{R}^3$ space, the Euclidean distance is often the option chosen. It is then possible to connect a vertex to another unvisited one, such that the cost is minimal. Note that the choice of the minimum cost is arbitrary, and it could be maximum.



(a) (b)

Figure 7.6. *Structure at the i stage. The vertices and edges in red are already covered, the edges in green are compared in order to add the next vertex, for a) the EMST built with Prim's algorithm, b) a Hamiltonian path*

Let us denote $G_n = (V_n, E_m, w)$ as the weighted graph, defined as the complete graph on the vertices of the mesh, where w: $E \rightarrow \mathbb{R}^+$. E_m represents the edges of the graph (different from the mesh connectivity) with $m = n(n-1)/2$. Each edge, e_i, has a weight defined as its length in Euclidean distance. The path built at the i

stage is a Hamiltonian sub-path, $\mathbf{P}_n$, on the set $V_i = \{v_0, \ldots, v_i\}$, which is made up of the edges chosen previously, $\{e_0, \ldots e_{i-1}\}$. All of the edges are either in V_i, or in $V_n \backslash V_i = \{v_{i+1}, \ldots, v_n\}$, the set of unvisited vertices, as shown in Figure 7.7.

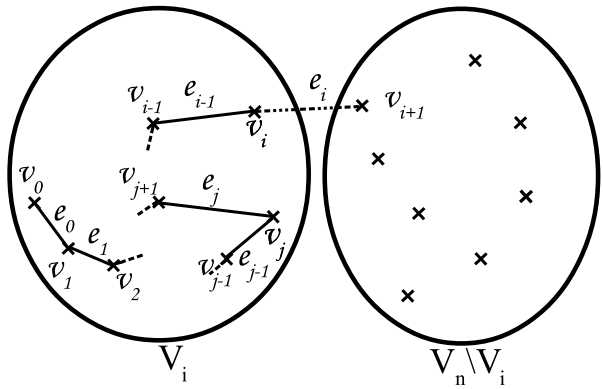


Figure 7.7. *State of the sets at step i: v_i the current vertex*

From a secret key, it is possible to get a starting vertex, v_0. Vertices are added to the path recursively by searching for the vertex, v_{i+1}, closest to the current vertex denoted by v_i, $i \in [0, n-1]$. The search for the nearest neighbor involves finding the edge of minimum weight, e_i between $v_i \in V_i$, and one of the unvisited vertices, $v_{i+1} \in V_n \backslash V_i$. The whole of the search is reduced to E'_i, which is the set of edges of v_i, such that $e_k = e\{v_i, v_k\}$, $k \in V_n \backslash V_i$ and $|E'_i| = n - i - 1$. The edge e_i is chosen as:

$$\omega(e_i) < \omega(e_k), i \neq k, e_i, e_k \in E'_i \qquad [7.3]$$

Finally, at the end of the construction, $\mathbf{P_n}$ is a Hamiltonian path on $G = (V_n, E_m)$.

To simplify the problem, we consider the movement of a vertex at each step of the path construction. The path built in the previous steps $\mathbf{P_{i+1}}$ must then be kept after the movement of the vertex, v_{i+1}. At iteration i, the vertex $v_j, j \in [0, |V|-1]$ belongs to $\mathbf{P_i}$ if $j \in [0, i]$. So two cases are possible for each vertex v_j, $j \neq i+1$:

1) $j \in [0, i] \Leftrightarrow v_j, \in V_i$;

2) $j \in [i+2, n] \Leftrightarrow v_j, \in V_n \backslash V_i$.

In the first case, the vertex v_{i+1} may be moved too close to the sub-path, $\mathbf{P_i}$, and a vertex of the set of vertices already added to the path, $v_j \in V_i, j \neq i$, could become its

predecessor, as illustrated in Figure 7.8a. The vertex v_{i+1} is moved to a new position, v'_{i+1}, close to v_j, so the weight $d_i = \|v_i, v'_{i+1}\|_2$ of the edge e_i becomes larger than the weight $d_j = \|v_j, v'_{i+1}\|_2$ of the edge e_j. Therefore, at the decoding step, the edge e_j between v_j and v'_{i+1} would be chosen and the Hamiltonian path would be modified. The constraint is defined as:

$$\|v_j, v_{j+1}\|_2 < \|v_j, v'_{i+1}\|_2, \ \forall v_j \in V_i, \ j \neq i \qquad [7.4]$$

In the second case, the vertex v_{i+1} is moved to a new position, v'_{i+1}, which is too far from its predecessor in the path. In this situation, another vertex $v_j \in V_n \backslash V_i$ becomes the closest neighbor to v_i. This case is shown in Figure 7.8b, which shows that another vertex is chosen as the decoding stage. So it is vital that:

$$\|v_i, v'_{i+1}\|_2 < \|v_i, v_j\|_2, \ \forall v_j \in V_n \backslash V_i, \ j \neq i{+}1 \qquad [7.5]$$

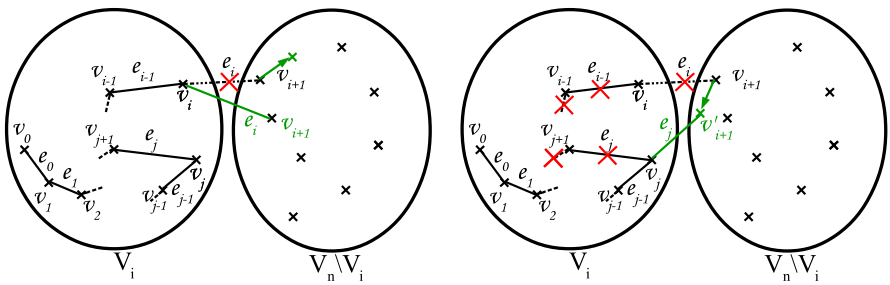


Figure 7.8. *a)* v_{i+1} *is moved too close to the sub-path, and b)* v_{i+1} *is placed too far from its predecessor*

It is therefore possible to define a displacement radius for each vertex. Let r^- be the maximum approach distance. It is defined as the limit distance between the current vertex and the sub-path; if the current vertex is found closer then it will be chosen before in the construction step. And vice-versa, let r^+ be the maximum distance away from the current vertex of the sub-path. This is the point from which the vertex will not be chosen at the same stage of the construction of the path. The displacement radius is then defined as:

$$r_{i+1} = min(r^+_{i+1}, r^-_{i+1}) \qquad [7.6]$$

where r^- and r^+ are calculated by:

$$r_{i+1}^{-} = min(\|v_j, v_{i+1}\|_2 - \|v_j, v_{j+1}\|_2),\ v_j \in V_i,\ j \neq i \quad [7.7]$$

$$r_{i+1}^{+} = min(\|v_i, v_j\|_2) - \|v_i, v_{i+1}\|_2,\ v_j \in V_n \backslash V_i,\ j \neq i{+}1 \quad [7.8]$$

This radius is useful for defining the most sensitive or the most robust vertices. In addition, this analysis highlights the constraints from which a watermarking method can be developed (Itier and Puech 2017).

7.4. 3D data hiding

3D data hiding is used in different ways depending on the desired objective, as presented in section 7.2. The embedding can be done by:

– *Injection*: the message is embedded directly into the media, which causes the media size to increase. This behavior is a security flaw, with respect to a potential attacker.

– *Substitution*: the message is embedded in such a way as to replace the redundant information of the medium, or to substitute a part of the information which alters the medium the least. This technique is the most used.

– *Distortion*: extraction is done by analyzing the difference between the media objects and the marked objects.

The distortion methods mainly require the support object (non-blind methods), which is rarely possible in practical cases. Substitution techniques were popularized by Cox *et al*. (1997) with spread spectrum methods, which embed a message from a linear combination of the host signal with a noise signal, modulated by the signal to be embedded. One approach to information theory is to think of the data hiding as communication with adjacent information. Theoretically, this point of view is described by the "Dirty Paper" codes, a practical implementation which was provided by Chen and Wornell with the *Quantization Index Modulation* (QIM) method (Chen and Wornell 2001). More generally, the data hiding methods are separated by the authors into two categories according to their embedding domain: the spatial domain and the transformed domain. So, to embed a message in the spatial domain, the methods directly change the geometry of a mesh; in other words, they modify the position of the vertices in the embedding space of the topology. In contrast, in the transformed domains, the embedding of the message takes place after a reversible transformation in the new domain. The properties of the data hiding methods according to the embedding domain are summarized in Table 7.1.

Properties	Spatial domain	Transformed domains
Capacity	High	Weak
Robustness	Weak	High
Perceptual quality	Controllable	Not very controllable
Complexity	Weak	High

Table 7.1. *Properties of data hiding methods, according to their embedding domain*

7.4.1. *Transformed domains*

Robust watermarking in a transformed domain requires a reversible preprocessing step that converts the meshes into coefficients in frequency space. Several transformed domains of 3D meshes have been used for data hiding, for example the harmonic transform (Liu *et al*. 2008; Wang *et al*. 2009), or the wavelet transform (Uccheddu *et al*. 2004; Wang *et al*. 2008). The wavelet transform allows embedding in low resolutions, which involves great robustness under the constraint that the same low resolution can be calculated after modifications on the mesh. The transformation step generally requires structured meshes, for example 2-manifold meshes, and is often not usable without preprocessing the mesh or the point cloud. These methods are especially used for robust 3D mesh watermarking applications. In fact, these applications only require a low capacity, for example 64 bits, while preserving the data embedded after modification of the mesh.

7.4.2. *Spatial domain*

In order to ensure robustness, methods in the spatial domain are generally based on the change of statistical distributions specific to the mesh (Cho *et al*. 2007; Luo and Bors 2011; Bors and Luo 2013). The strength of these methods is based on the modification of histograms, which removes the dependence of these methods on the synchronization of primitives of the mesh. However, these methods are less robust than embedding methods in transformed domains and offer a low embedding capacity compared to embedding methods in the spatial domain. Nevertheless, methods in the spatial domain generally allow a higher capacity but have lower robustness. Most high capacity or steganography methods fall into this category. These embedding schemes are often based on the concept of QIM, which allows a high capacity with few distortions of the support vector. One of the first methods that was defined as high capacity was proposed by Cayre and Macq, for embedding in triangle meshes (Cayre and Macq 2003). The synchronization step involves first organizing the triangles that serve as an information medium, thanks to the TSPS

method, and then inserting the information using a QIM-based technique, illustrated in Figure 7.9. The main idea of the method involves inserting a bit by considering the projection of a vertex on the opposite edge as support. Figure 7.1 shows the embedding of the bit, "1", in two different configurations, the vertex is moved so that its projection is in an interval which encodes the value "1". The authors suggest choosing the first triangle, which is the start of the synchronization method with two different approaches. The first approach involves choosing a triangle with the minimum area, and the second involves performing a principal component analysis (PCA) and use the triangles that intersect the three main axes. This method allows a certain robustness against affine transformations. In addition, the capacity of the method is approximately 1 bit per vertex (bpv). Nevertheless, this method requires triangle meshes for synchronization but also as support. The extension of the method to polygon meshes is not trivial. The capacity of Cayre and Macq's method (Cayre and Macq 2003) was improved by Wang and Cheng (2005) to achieve a minimum of 3 bpv. The authors use a multilevel embedding procedure, which first inserts in the same way as the reference method, then in the height of the triangle, thanks to a threshold, and lastly, in the rotation of the angle between the base and the height of the triangle. This method is limited by the numerical precision, depending on the number of divisions chosen. The authors propose keeping a capacity between 3 and 6 bpv, while taking visual distortions into account in their new method (Cheng and Wang 2007). This method extends the type of input meshes to polygon meshes, and defines a new synchronization based on contagious diffusion, as if the input polygon and one of its edges were contaminated. The transmission is done through a common edge, the cost in calculation time to traverse all the mesh is $\mathcal{O}(n)$. To increase the capacity, Chao *et al*. (2009) use a multilevel embedding method on each coordinate of a vertex. By using the three coordinates of a vertex, the method allows us to embed 3 bits per vertex in each layer, except for the three vertices used for synchronization, which gives a capacity of $3(|V| - 3)n_{layers}$ bpv. This capacity is limited by the precision of the floating-point number used to store the position of the vertices. The simple IEEE 754 standard offers 23 bits for the mantissa, which allows the authors to confirm that their method with this standard has a theoretical maximum capacity of 69 bpv.

7.4.3. *Other domains*

Other domains have been proposed, for example the data representation domain. It involves using redundancies in the representation of the mesh as an information medium. Meshes are generally represented by a list of vertices, their coordinates in $\mathbb{R}^3$ and a list of polygons. The order in which these items appear in the lists does not affect the 3D result. Bogomjakov *et al*. (2008) reorders the edges and sides. Therefore, the message is embedded in the primitive permutations, with respect to their order of traversal in the mesh (*Edgebreaker*, Rossignac (1999)).

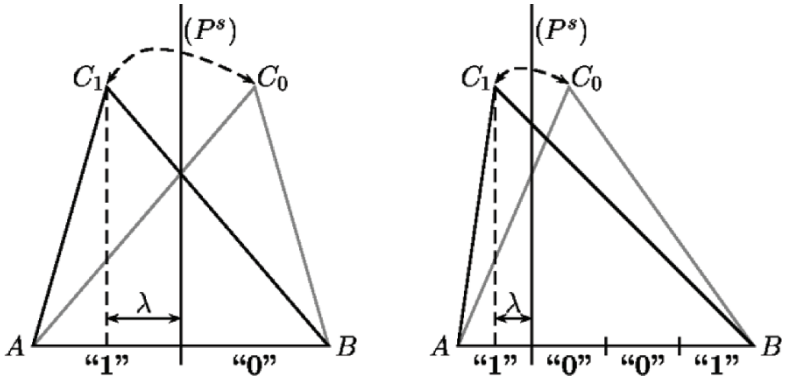


Figure 7.9. *Embedding method of the algorithm of Cayre and Macq (2003), the bit "1" is embedded by moving the vertex C_0 to the position C_1, so that its projection on the opposite edge corresponds to an interval that encodes the correct value. The opposite edge can be divided into a) two intervals and b) four intervals*

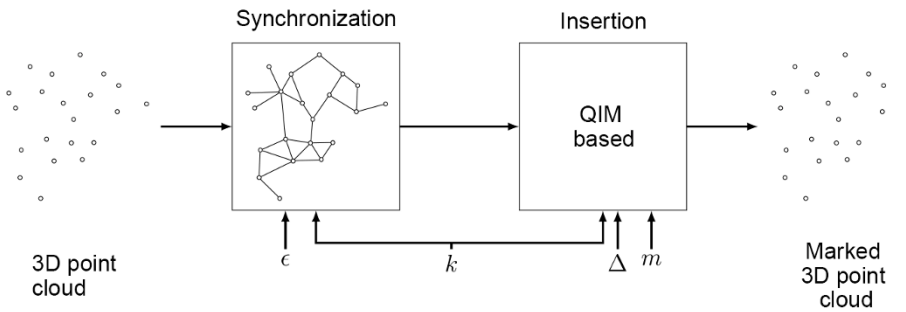


Figure 7.10. *Overview of a data hiding method in a point cloud in the spatial domain, where ϵ is the synchronization parameter, k is the secret key, Σ is the quantization step and m is the message to be embedded*

7.5. Presentation of a high-capacity data hiding method

In this section, we study a data hiding method in the spatial domain based on the steps presented in section 7.4. These steps are summarized in Figure 7.10. We illustrate each of these steps in the high-capacity data hiding method proposed by Itier *et al*. (2015a). Synchronization is done by following the construction order of the Hamiltonian path in the complete graph of the point cloud from a first vertex given by the secret key. To respect the constraints on the stability of the Hamiltonian path,

presented in section 7.3.3.2, embedding and synchronization are done together. The message is embedded on each pair of vertices, making up an edge of the Hamiltonian path, by quantization of the polar coordinates so as to embed part of the message and keep the traversal order of the vertices of the Hamiltonian path.

7.5.1. *Embedding of the message*

In this section, we present a practical implementation of the QIM method on the position of the vertices of the mesh. We want to embed the message $\mathbf{M}$ of size $|\mathbf{M}| < |V| - 1$, defined on an alphabet, $\mathcal{S} = \{s_0, ..., s_q\}$, in a mesh of $|V|$ vertices. For each iteration, $i, 0 \leq i < |V|$, the vertex v_{i+1} is moved, with respect to its predecessor, v_i, in the Hamiltonian path by changing its coordinate system, from the Cartesian coordinates at local spherical coordinates, with respect to the predecessor vertex. Embedding is then carried out according to a coordinate of the new coordinate system, ρ, θ and ϕ. The vector p_i between the vertices $v_i = (x_i, y_i, z_i)$ and $v_{i+1} = (x_{i+1}, y_{i+1}, z_{i+1})$ is defined as:

$$p_i = \begin{bmatrix} x_{i+1} - x_i \\ y_{i+1} - y_i \\ z_{i+1} - z_i \end{bmatrix} \qquad [7.9]$$

The conversion into spherical coordinates is written as $p_i = (\rho_i, \theta_i, \phi_i)$. Following the embedding of the message and the reverse transform in the initial coordinate system, the new position of the vertex v_{i+1} is denoted as v'_{i+1}. The main idea of the method is to define a displacement interval, Δ, which limits the quantization of the components of the vector. Let c be the value of the element, so the lower bound of the interval, Δ, is given as:

$$b_l = \left\lfloor \frac{c}{|\Delta|} \right\rfloor \times |\Delta| \qquad [7.10]$$

The upper bound is simply $b_u = b_l + |\Delta|$. The interval Δ is then divided into q subintervals, encoding the words of $\mathcal{S}$. Their lower bounds are denoted by: $\delta_j, j \in [0, q-1]$. Finally, to embed a word, s_j, it is enough to move the vertex into the corresponding interval:

$$c' = \begin{cases} b_l + \delta_j & \text{if } c < b_l + \frac{\delta_j + \delta_{j+1}}{2} \\ b_l + \delta_{j+1} - \gamma & \text{otherwise} \end{cases} \qquad [7.11]$$

where $\gamma = \frac{1}{k} \times |\delta|,\ k \in \mathbb{N}, k > 1$, is a fraction of the size of the subinterval.

Let x be the number of subintervals, and the theoretical capacity of this method for a mesh of $|V|$ vertices is given as:

$$cp = 3 \times log_2(x) \times (|V| - 1) \quad [7.12]$$

If we set $x = 2$, we get a binary system similar to the one proposed in Itier *et al.* (2015a). For a binary alphabet, $\mathcal{S} = \{0, 1\}$, the subintervals, δ_0 and δ_1, correspond to one bit of the message. From a practical point of view, the value $x = 256$ is interesting, since $\mathcal{S}$ defines a set of coded words on one byte. The capacity is then 24 bits per edge of the Hamiltonian path. This capacity is consistent with the precision of the floating point that is used in standard binary 3D mesh formats, like STL[2] or PLY[3]. However, a larger capacity may reach the precision limits of some formats.

7.5.2. *Causality issue*

Causality issues can occur when a vertex is moved, through modification of its coordinates, during an iteration of the embedding algorithm. In order to extract the message, the traversal of the mesh must follow the exact same route on the vertices as in the embedding step. This traversal is noted $\mathbf{P_n}$, where n is the number of vertices. Finally, the movement of a vertex can lead to loss of synchronization if the embedding path is different to the extraction path. At the i step, if the embedding on the vertex v causes a violation of the stability rules of the Hamiltonian path, then the traversal of the graph following the construction of a Hamiltonian path from the same vertex v_0 will no longer be the same. In fact, v will not be chosen at step i. To avoid this phenomenon, we propose a verification step, which authorizes or forbids a movement. The sub-paths of $\mathbf{P_n}$, denoted as $\mathbf{P_{0,i}}$ and $\mathbf{P_{i+1,n-1}}$, are Hamiltonian paths built from step 0 to step i, and from step $i + 1$ to step $n - 1$, respectively. The stability of the traversal $\mathbf{P_n}$ depends on the consideration of two conditions after the calculation of the new position of the vertex, v_i:

– preserve the path $\mathbf{P_{0,i}}$;

– preserve the path $\mathbf{P_{i+1,n-1}}$.

An obvious solution to the second case is to combine the synchronization and embedding steps. In fact, it is not necessary to keep the path after a vertex movement, it is enough to confirm at each step that the path previously constructed is maintained. The first case is more tricky. The constraints given by equations [7.4] and [7.5] must be confirmed for each movement of the vertex.

2 Developed by 3D Systems.

3 Developed by Stanford University.

Finally, if the movement produces a desynchronization, we can choose a simple solution that involves not moving this vertex and therefore, not using it as a medium for inserting information. In a blind scenario, it is not possible to know if a vertex carries information or not. Therefore, at the extraction stage, information will be read for this vertex and will potentially be false. These confirmation steps, in addition to the construction of the Hamiltonian path, add distance calculations to search for the closest neighbors at each iteration, which, from a naive implementation, is quite time restrictive.

7.6. Improvements

From the method presented in section 7.5, it is possible to construct new methods which respond to certain needs, for example to improve their security, robustness or execution time.

7.6.1. *Error-correcting codes*

One of the first improvements involves dedicating part of the embedding capacity to set up mechanisms that make it possible to secure the hidden message. In fact, due to the strong embedding conditions on the Hamiltonian path, some vertices can be found in the case where it is impossible to embed one or more bit(s). So in order to allow the correct Hamiltonian path to be read on extraction, the vertices have their position remain unchanged, which causes errors to appear in the extracted message. Furthermore, in some cases, using too small a quantization interval, Δ, may result in certain numerical precision errors. One of the solutions is to use error-correcting codes, such as Reed–Solomon codes (Reed and Solomon 1960) or Hamming codes (Hamming 1950). The use of these codes reduces the payload corresponding to the secret message but makes it possible to ensure better recovery of the secret message after decoding.

7.6.2. *Statistical arithmetic coding*

In some cases, the secret message does not have a uniform distribution, that is, if it is not encrypted or encrypted with a permutation method (or *Scrambling*). The point is to be able to preprocess the secret message by using its statistical properties to compress it or to make its embedding easier, for example. In this case, instead of dividing the quantization interval, Δ, depending on the number of bits, B, to be embedded per vertex uniformly, a more suitable type of encoding can be used in order to divide the quantization interval to better represent the message. Statistical arithmetic coding is one solution, as presented by Itier and Puech (2017). This coding makes it possible to find better positions for the vertices to embed the message with fewer

errors (of unmoved vertices). This approach also allows the reduction of geometric distortions induced in the 3D mesh. In fact, the subintervals coding the most frequent words are wider, which implies that the movement of the vertices in the interval Δ is smaller, on average. This approach also makes the tasks of analyzing and detecting hidden messages (or steganalysis) more difficult.

7.6.3. *Partitioning and acceleration structures*

One of the main jobs of the method proposed by Itier *et al.* involves, at each stage of the construction of the Hamiltonian path, finding the closest point in the set $V_n \backslash V_i$ of the last point, v_i, added in the path (Itier *et al.* 2015a). The method used to look for the closest vertex (or *Nearest Neighbor Search*) must have a very low execution time, but must also be able to take into account the modifications made to the vertex after inserting v'_{i+1}. So a naive approach like linear search, that is, a comparison of all the distances between v_i and the vertices of the set $V_n \backslash V_i$, becomes very difficult to sustain in this context because of its complexity in $\mathcal{O}(n)$. In order to be able to reduce the time for this task, implementing an acceleration structure is necessary. In the 3D domain, acceleration structures, such as Octree (Drost and Ilic 2018) and Kd-Tree can be put in place to accelerate the nearest neighbor search. However, depending on the implementation of these structures, it is sometimes difficult to update the structure. In fact, because of the modifications that can be made to the position of the vertices after embedding of the message, we need to partially update the structure without having to completely recalculate it. This is why we recommend the use of a K-d-B-Tree (Robinson 1981), as illustrated in Figure 7.11.

This k dimensional binary tree is made up of two types of nodes. First, region nodes offering a binary choice along a specific axis and a cut value (often the average of the bounding box on that same axis), where its left-child node contains the region aligned to the bounding box with vertices less than the cut value, and its right-child node contains the region with the vertices greater than the cut value. Finally, leaf nodes contain an array of vertex indices, located at the level of the region defined by the node. One of the advantages of this k dimensional binary tree is that it uses the properties of a search in a Kd-Tree, and that it is easily updated. To update the tree, the algorithm must test whether the vertex v'_{i+1} is in the same leaf of the tree as v_{i+1}. If it is, then the tree is not changed; otherwise, the leaf found is the new node, receiving the index of the vertex v'_{i+1}. It will then be necessary to remove the index of the vertex v_{i+1} from its starting node.

Thanks to this partitioning of the space and its update at each iteration of embedding in the Hamiltonian path, it is then possible to find the nearest vertex v_{i+1} of v_i very quickly. In addition, the use of an acceleration structure makes it possible to respond more quickly to one of the conditions of the causality problem and, in particular, the condition defined by equation [7.5] in section 7.3.3.2 by looking for the second closest point (or the first closest point by ignoring v_{i+1}).

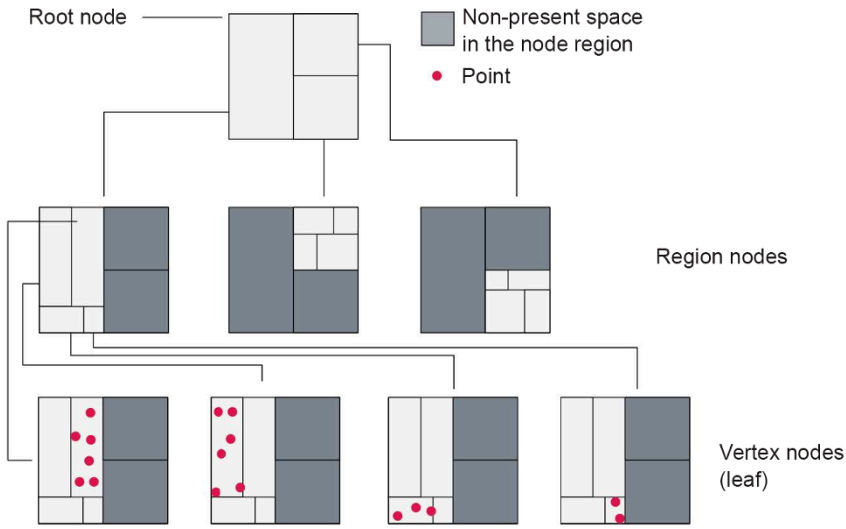


Figure 7.11. *Representation of a K-d-B-Tree*

7.7. Experimental results

The evaluation of data hiding approaches is based on the compromise presented in Figure 7.2 in section 7.5. We know that increasing the capacity of a data hiding method comes at the expense of its robustness. Furthermore, in order to increase the visual imperceptibility of the embedding, the shape and connectivity of the mesh must be modified as little as possible. We evaluate the capacity of the method and the distortions introduced into the 3D object following the embedding. The capacity of a method can be defined in bits per vertex or by its payload, that is, the space dedicated solely to the secret message. Note that additional data to the secret message, such as error-correcting codes, are not part of the payload of a method. There are many benchmark quality assessment metrics in 3D, and they allow us to objectively quantify the quality of one 3D object relative to another. These metrics are classed in two categories, those in correlation with the human visual system, and those that are not. The latter is very widely used because it is generally well integrated, for example the Hausdorff distance, the root mean square error (RMSE), $PSNR_1$ (or *peak signal-to-noise ratio*) (Chao *et al*. 2009) or even the geometric Laplacian (Karni and Gotsman 2000). The metrics correlated to the human visual system are more recent and are accompanied by subjective evaluations. For example, Corsini *et al*. (2007) proposed a metric called the "3D Watermarking Perception Metric" (3DWPM), with the aim of evaluating their 3D watermarking method. One of the metrics most correlated with the human visual system is the "Mesh Structural Distortion Measure 2" (MSDM2), proposed by Lavoué (2011).

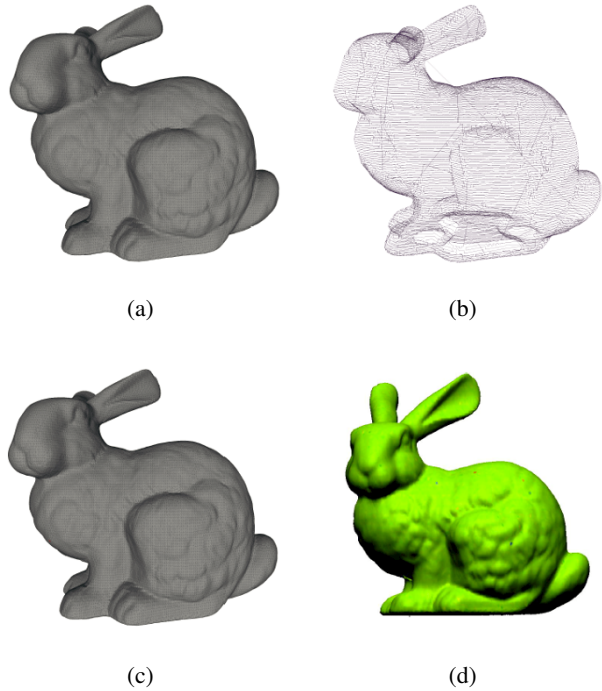


Figure 7.12. *Example of data hiding in a 3D Bunny object with $\Delta = 10^{-4}$: a) original 3D object, b) Hamiltonian path, c) marked 3D object and d) geometric distortions between the original 3D object and the marked one*

In this section, we present the experimental results of the method proposed by Itier and Puech (2017). This method is an extension of the method presented in section 7.1, using statistical arithmetic coding, as explained in section 7.6.2. Figure 7.12 presents the 3D "Bunny" object, containing 34,834 vertices, in which a secret message is embedded with the parameter $\Delta = 10^{-4}$. First, we can visually compare the original 3D object (see Figure 7.12a) and its marked version (see Figure 7.12c) and note that the embedding can be considered as imperceptible, given the very weak geometric distortions illustrated in Figure 7.12d. This low impact on the geometry is mainly due to the fact that the Hamiltonian path built, illustrated in Figure 7.12b, although based on the point cloud, passes through edges mainly from the 3D mesh.

The method studied is compared with other state of the art high-capacity methods, and Table 7.2 presents the results of the methods in terms of capacity, given in bits per vertex, Hausdorff distance (Cignoni *et al*. 1996) and $PSNR_1$ (Chao *et al*. 2009).

	Capacity (bpv)	Hausdorff distance $\times 10^{-3}$	$PSNR_1$ (dB)
Chao *et al.* (2009)	≤ 27	–	100.57
Gao *et al.* (2012)	≤ 1.5	0.548	70.02
(Itier *et al.* 2015b) $\Delta = 1{,}10^{-4}$	≤ 24	0.27	127.236

Table 7.2. *Comparison with previous methods on the 3D Bunny object*

To generalize on a larger number of objects, databases of 3D objects[4] are used. These bases contain a wide variety of shapes and sizes (in number of vertices). The number of vertices of these 3D objects varies between 1,000 and 200,000. Table 7.3 presents different 3D objects with their number of vertices and their theoretical maximum capacity, given in bit per vertex. Each object is marked by inserting a message using the maximum capacity of the object. Geometric distortions of marked objects are evaluated using $PSNR_1$ and MSDM2 metrics. Since the method does not always allow inserting into every vertex, the quality of the extracted message is measured using the bit error rate (BER) and the peak signal-to-noise ratio (PSNR).

As Table 7.3 shows, embedding in meshes causes geometric distortions to have a very low impact on 3D objects, as we notice with $PSNR_1$ greater than 110 dB and MSDM2 results lower than 10^{-2}. Also note that embedding in full capacity of each 3D object reveals a very low presence of errors in the message extracted from the 3D objects. The BER observed in the tested 3D objects is very low, close to 10^{-3}, or even equal to 0 in the case of the 3D "eagle" object. High BER values, as in the 3D object *chair1*, correspond to 3D objects that have vertices whose distances between them are close to or less than Δ, which prevents finding positions that do not break the causality of the Hamiltonian path when inserting the message. Thus, errors are mainly due to vertices that could not be moved. The use of error-correcting codes, proposed in section 7.6, make it possible to correct the majority of these errors. However, then the amount of information embedded (payload) is reduced.

7.8. Trends in high-capacity 3D data hiding

7.8.1. *Steganalysis*

Data hiding can be used as a secret communication tool. We shall therefore discuss steganography, as explained in detail in Chapter 5. The secret data are shared using an information medium, which may be a 3D object. Steganalysis (see Chapter 8) is concerned with detecting the presence of hidden messages within

4 MADRAS project, Stanford University, LGMA.

media. Recently, steganalysis approaches have been applied to 3D objects (Yang *et al*. 2014; Li and Bors 2016a, 2016b). To cope with these advances, interest in new embedding methods, which are visually and statistically imperceptible, has increased (Yang *et al*. 2017). For example, Itier and Puech's (2017) method introduces distortions that are detectable by the steganalysis system used by Li (2018). The proposed embedding strongly influences the dihedral angles of the 3D object. However, if the embedding is only done on the radial coordinate, the modification of the dihedral angles between the triangles is reduced.

3D object	Number of vertices	Theoretical capacity	3D metrics		Message metrics	
			$PSNR_{1\,(dB)}$	MSDM2($\times 10^{-3}$)	PSNR$_{(dB)}$	BER($\times 10^{-3}$)
Armadillo	172,974	518,922	134.44	1.27	23.49	2.632
Bitorus	3,000	9,000	116.862	0.55	31.34	4.024
Blade2	24,738	74,214	123.375	3.52	48.6	0.067
Bunny	34,834	104,502	127.236	0.41	52.61	0.042
Casting	5,096	15,288	118.308	3.48	32.77	2.411
Chair1	12,326	36,978	125.168	4.54	5.49	457.288
Dragon	50,000	150,000	128.618	1.84	55.07	0.031
Eagle	1,000	3,000	112.824	1.32	$+,\infty$	0
Hand	36,619	109,857	125.463	1.72	36.11	0.807
Horse	112,642	337,926	134.965	2.09	48.68	0.047
Rabbit	70,658	211,974	131.27	0.41	46.34	0.103
Venus	100,759	302,277	130.454	1.56	44.02	0.189

Table 7.3. *Evaluation of the quality of the approach studied (Itier and Puech 2017)*

7.8.2. *Security analysis*

Security in the data hiding field is defined as the impossibility that an unauthorized user can access the hidden channel (Kalker 2001). Perez-Freire and Perez-Gonzalez (2009) extended this definition: security also relies on the difficulty of estimating the secret parameters of the embedding function based on the observation of marked data. In general, thorough safety analysis of methods is not considered by the authors. Some authors assume that the method does not need to be secure. In other cases, the problem has an obvious high complexity but many biases. For example, the robust watermarking method proposed by Bors and Luo (2013) shows security vulnerabilities (Itier *et al*. 2014). Some methods are considered secure

since the number of difficulties makes the problem too complex for an attacker. In this case, it is much easier to remove the payload than to access it (Cayre and Macq 2003).

7.8.3. *3D printing*

3D data hiding was initially linked to digital applications such as cinema, games or modeling. One of its main purposes is the protection of shapes. With the rise of 3D printing technologies, for prototyping for example, the protection of shapes also takes place in the real world. These protections are based on the local variations of thickness of the object to embed the message. Extraction is then done using a scanner. One of the first methods proposed by Yamazaki *et al*. (2014) is non-blind and requires the realignment of the scanned shape with the reference model. Recently, the first methods of blind watermarking of 3D-printed objects have been proposed. Extraction can be done using a camera (Zhang *et al*. 2018) or a paper scanner (Delmotte *et al*. 2019).

7.9. Conclusion

In this chapter, we have presented essential concepts, as well as latest developments in 3D data hiding methods. We have shown, using an example, the different steps to be implemented in a data hiding scheme, and the problems they generate.

The security of 3D data hiding methods is still an unresolved question. Each application context requires an appropriate scenario. An interesting area of work is to propose efficient steganography methods, validated by rigorous steganalysis. A major unresolved problem involves developing a data hiding method that is not designed for a specific application. This method must take into account the compromise between the data and automatically adapt the parameters for each use. More generally, protection of 3D shapes should not consist only of the protection of a mesh associated with the shape, since it is possible to re-mesh an object. The emergence of 3D printing poses new challenges in this context. In fact, the sharpness of the resolution of printing and precise acquisition methods can produce totally different meshes while preserving shape. Thus, hiding 3D data becomes inevitable for designers and manufacturers, or for privacy. Industry makes profits by designing and exploiting its own shapes, and they need tools that help them preserve their properties. 3D acquisition and 3D printing are already available and used daily for entertainment, medical imaging and increasingly on social media. At the same time, governments, industry and hackers are collecting, exploiting or monetizing proprietary and personal data. It is therefore necessary to deepen the work in the field of 3D data hiding, in particular on the security of methods, and to develop tools for common use.

7.10. References

Amat, P., Puech, W., Druon, S., Pedeboy, J. (2010). Lossless 3D steganography based on MST and connectivity modification. *Signal Processing: Image Communication*, 25(6), 400–412.

Bajaj, C., Pascucci, V., Zhuang, G. (1999). Progressive compression and transmission of arbitrary triangular meshes. *Proceedings of the IEEE Conference on Visualization 1999*. IEEE, San Francisco, 307–537.

Bogomjakov, A., Gotsman, C., Isenburg, M. (2008). Distortion-free steganography for polygonal meshes. *Computer Graphics Forum*, 27, 637–642.

Bors, A.G. and Luo, M. (2013). Optimized 3D watermarking for minimal surface distortion. *IEEE Transactions on Image Processing*, 22(5), 1822–1835.

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Levy, B. (2010). *Polygon Mesh Processing*. Taylor & Francis, Abingdon-on-Thames.

Cayre, F. and Macq, B. (2003). Data hiding on 3-D triangle meshes. *IEEE Transactions on Signal Processing*, 51(4), 939–949.

Chao, M.-W., Lin, C.-H., Yu, C.-W., Lee, T.-Y. (2009). A high capacity 3D steganography algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 15(2), 274–284.

Chen, B. and Wornell, G.W. (2001). Quantization index modulation methods for digital watermarking and information embedding of multimedia. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 27(1/2), 7–33.

Cheng, Y.-M. and Wang, C.-M. (2007). An adaptive steganographic algorithm for 3D polygonal meshes. *The Visual Computer*, 23(9/11), 721–732.

Cho, J.-W., Prost, R., Jung, H.-Y. (2007). An oblivious watermarking for 3-D polygonal meshes using distribution of vertex norms. *IEEE Transactions on Signal Processing*, 55(1), 142–155.

Cignoni, P., Rocchini, C., Scopigno, R. (1996). Metro: Measuring error on simplified surfaces. Technical report, CNRS, Paris.

Corsini, M., Gelasca, E., Ebrahimi, T., Barni, M. (2007). Watermarked 3-D mesh quality assessment. *IEEE Transactions on Multimedia*, 9(2), 247–256.

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

Cox, I.J., Miller, M., Bloom, J., Fridrich, J., Kalker, T. (2007). *Digital Watermarking and Steganography*. Morgan Kaufmann, Burlington.

Delmotte, A., Tanaka, K., Kubo, H., Funatomi, T., Mukaigawa, Y. (2019). Blind watermarking for 3D printed objects by locally modifying layer thickness. *IEEE Transactions on Multimedia*, 22(11), 2780–2791.

Dieck, T. (2008). *Algebraic Topology*. European Mathematical Society, Helsinki.

Drost, B. and Ilic, S. (2018). Almost constant-time 3D nearest-neighbor lookup using implicit octrees. *Mach. Vis. Appl.*, 29(2), 299–311.

Gao, X., Zhang, C., Huang, Y., Deng, Z. (2012). A robust high-capacity affine-transformation-invariant scheme for watermarking 3D geometric models. *ACM Transaction on Multimedia Computing, Communications and Applications*, 8(2S), 34:1–34:21.

Gumhold, S., Kami, Z., Isenburg, M., Seidel, H.-P. (2005). Predictive point-cloud compression. *Proceedings of the ACM Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH)*, Los Angeles, 137.

Gurung, T., Luffel, M., Lindstrom, P., Rossignac, J. (2011). LR: Compact connectivity representation for triangle meshes. *ACM Transactions on Graphics*, 30(4), 1–8.

Hamilton, W.R. (1853). On the icosian calculus. *Proceedings of the Royal Irish Academy*, 6, 462.

Hamming, R.W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2), 147–160.

Hoffmann, C.M. (1989). *Geometric and Solid Modeling: An Introduction*. Morgan Kaufmann, Burlington.

Huang, Y.-H. and Tsai, Y.-Y. (2015). A reversible data hiding scheme for 3D polygonal models based on histogram shifting with high embedding capacity. *3D Research*, 6(2), 20.

Itier, V. and Puech, W. (2017). High capacity data hiding for 3D point clouds based on static arithmetic coding. *Multimedia Tools and Applications*, 76(24), 26421–26445.

Itier, V., Puech, W., Bors, A. (2014). Cryptanalysis aspects in 3-D watermarking. In *International Conference on Image Processing (ICIP)*. IEEE, Paris, 4772–4776.

Itier, V., Puech, W., Gesquière, G., Pedeboy, J.-P. (2015a). Joint synchronization and high capacity data hiding for 3D meshes. In *Proceedings of SPIE: Three-Dimensional Image Processing, Measurement*. SPIE, San Francisco, 939305.

Itier, V., Tournier, N., Puech, W., Subsol, G., Pedeboy, J.-P. (2015b). Analysis of an EMST-based path for 3D meshes. *Computer-Aided Design*, 64, 22–32.

Kalker, T. (2001). Considerations on watermarking security. *Proceedings of the Workshop on Multimedia Signal Processing*. IEEE, 201–206.

Karni, Z. and Gotsman, C. (2000). Spectral compression of mesh geometry. *Proceedings of the ACM Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH)*, New Orleans, 279–286.

Kerckhoffs, A. (1883). La cryptographie militaire. *Journal des sciences militaires*, IX, 5–38.

Kruskal Jr., J. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1), 48–50.

Lavoué, G. (2011). A multiscale metric for 3D mesh visual quality assessment. *Computer Graphics Forum*, 30(5), 1427–1437.

Li, Z. (2018). Steganalytic methods for 3D objects. PhD Thesis, University of York, York.

Li, Z. and Bors, A.G. (2016a). 3D mesh steganalysis using local shape features. *Proceedings of the International Conference on Acoustics, Speech and Signal Processing, (ICASSP)*. IEEE, Lujiazui, 2144–2148.

Li, Z. and Bors, A.G. (2016b). Selection of robust features for the cover source mismatch problem in 3D steganalysis. *Proceedings of the International Conference on Pattern Recognition*. IEEE, Cancun, 4256–4261.

Lin, C.-H., Chao, M.-W., Chen, J.-Y., Yu, C.-W., Hsu, W.-Y.A. (2013). A high-capacity distortion-free information hiding algorithm for 3D polygon models. *International Journal of Innovative Computing, Information and Control*, 20(3), 1321–1335.

Liu, Y., Prabhakaran, B., Guo, X. (2008). A robust spectral approach for blind watermarking of manifold surfaces. *Proceedings of the Workshop on Multimedia and Security*. ACM, Oxford, 43–52.

Luo, M. and Bors, A. (2011). Surface-preserving robust watermarking of 3-D shapes. *IEEE Transactions on Image Processing*, 20(10), 2813–2826.

Mao, X., Shiba, M., Imamiya, A. (2001). Watermarking 3D geometric models through triangle subdivision. *Proceedings of SPIE*, 4314, 253–260.

Merry, B., Marais, P., Gain, J. (2006). Compression of dense and regular point clouds. *Proceedings of the International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*. ACM, Cape Town, 15–20.

O'Neill, B. (2006). *Elementary Differential Geometry*, 2nd edition. Elsevier Science, New York.

Ohbuchi, R., Masuda, H., Aono, M. (1997). Watermaking three-dimensional polygonal models. *Proceedings of the International Conference on Multimedia*, 16, 261–272.

Perez-Freire, L. and Perez-Gonzalez, F. (2009). Spread spectrum watermark security. *IEEE Transactions on Information Forensics and Security*, 4(1), 2–24.

Petitcolas, F., Anderson, R., Kuhn, M. (1999). Information hiding – A survey. *Proceedings of the IEEE*, 87(7), 1062–1078.

Prim, R. (1957). Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36, 1389–1401.

Reed, I.S. and Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2), 300–304.

Robinson, J.T. (1981). The KDB-tree: A search structure for large multidimensional dynamic indexes. *Proceedings of the International Conference on Management of Data*. ACM, Ann Harbor, 10–18.

Rossignac, J. (1999). Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transactions on Visualization and Computer Graphics*, 5(1), 47–61.

Tournier, N., Puech, W., Subsol, G., Pedeboy, J.-P. (2011). Finding robust vertices for 3D synchronization based on Euclidean minimum spanning tree. *Proceedings of the IS&T/SPIE Electronic Imaging Symposium*, San Francisco, 78640W.

Uccheddu, F., Corsini, M., Barni, M. (2004). Wavelet-based blind watermarking of 3D models. *Proceedings of the Workshop on Multimedia and Security*. ACM, Madgebourg, 143–154.

Wang, C.-M. and Cheng, Y.-M. (2005). An efficient information hiding algorithm for polygon models. *Computer Graphics Forum*, 24(3), 591–600.

Wang, K., Lavoué, G., Denis, F., Baskurt, A. (2008). Hierarchical watermarking of semi-regular meshes based on wavelet transform. *IEEE Transactions on Information Forensics and Security*, 3(4), 620–634.

Wang, K., Luo, M., Bors, A., Denis, F. (2009). Blind and robust mesh watermarking using manifold harmonics. *Proceedings of the International Conference on Image Processing*. IEEE, Cairo, 3657–3660.

Wang, K., Lavoué, G., Denis, F., Baskurt, A. (2011). Robust and blind mesh watermarking based on volume moments. *Computer & Graphics*, 35(1), 1–19.

Yamazaki, S., Kagami, S., Mochimaru, M. (2014). Extracting watermark from 3D prints. *Proceedings of the 2014 International Conference on Pattern Recognition*. Stockholm, 4576–4581.

Yang, Y., Pintus, R., Rushmeier, H., Ivrissimtzis, I. (2014). A steganalytic algorithm for 3D polygonal meshes. *Proceedings of the International Conference on Image Processing*. IEEE, Paris, 4782–4786.

Yang, Y., Pintus, R., Rushmeier, H., Ivrissimtzis, I. (2017). A 3D steganalytic algorithm and steganalysis-resistant watermarking. *IEEE Transactions on Visualization and Computer Graphics*, 23(2), 1002–1013.

Zhang, J., Zheng, C., Hu, X. (2013). Triangle mesh compression along the hamiltonian cycle. *Visual Computer*, 29(6/8), 717–727.

Zhang, X., Wang, Q., Ivrissimtzis, I. (2018). Single image watermark retrieval from 3D printed surfaces via convolutional neural networks. *Proceedings of the Conference on Computer Graphics & Visual Computing*. Eurographics Association, Swansea, 117–120.

8

Steganalysis: Detection of Hidden Data in Multimedia Content

Rémi COGRANNE[1], Marc CHAUMONT[2] and Patrick BAS[3]

[1] *LIST3N, University of Technology of Troyes, France*

[2] *LIRMM, Université de Montpellier, CNRS, University of Nîmes, France*

[3] *CRIStAL, CNRS, University of Lille, France*

In Chapter 5, we presented the concept of steganography, that is, methods of hiding information in digital images. In particular, we concentrated on the fact that steganography methods are constructed in a context where an enemy seeks to identify images used to spread hidden information (Stego images) from a list of images. In this chapter, we will see how to perform an analysis of a digital image to obtain information about the data that may have been hidden there.

8.1. Introduction, challenges and constraints

Before getting to the heart of the matter, we will briefly remind you that the purpose of steganography is to hide secret information in digital media so that the latter remain visually and statistically "as close as possible" to the original media. The example of the prisoner problem[1] (Simmons 1984) helps to illustrate this

For a color version of all figures in this chapter, see www.iste.co.uk/puech/multimedia1.zip.

1 Also see section 5.1.

Multimedia Security 1,

coordinated by William PUECH.

Multimedia Security 1: Authentication and Data Hiding,
First Edition. William Puech.

situation. The aim of steganalysis can seem quite simple at first glance: it is a matter of "only" detecting media containing hidden information to prevent their transmission. In practice, the subject matter is much broader than that. In reality, as very often in the field of security, the goal of steganalysis largely depends on the *scenario* considered and, in particular, on the knowledge that Eve has "initially"[2].

In this chapter, we will see that steganalysis has been widely studied as a tool for evaluating steganography methods. This approach sets down a very specific situation in which Eve is assumed to be *omniscient*, in that she can access (almost) all the information about the hidden data. In this scenario, we generally assume that Eve is only unaware of (1) what the hidden message is, although its statistical distribution is known; (2) what insertion key is used and (3) whether a message is actually inserted. This scenario is very practical for the steganographer, Alice, because, on the one hand, it allows her to focus on the problem of interest without taking into account the incidental "technical difficulties". Furthermore, in this scenario, Alice considers the worst case in terms of steganalysis, so she can be sure that her assessment is pessimistic and that, in practice, a more realistic enemy will hardly be able to come up with such a precise steganalysis because they often do not know the size of the inserted message or the algorithm used. In section 8.6 we will see that the experimental conditions are also significantly different from the operational conditions under which steganalysis could be carried out.

8.1.1. *The different aims of steganalysis*

Many works have been proposed in order to implement different types of steganalysis; we will briefly describe them in the rest of this section:

– on the one hand, different detection methods can be considered, depending on the Cover media[3]: images, sounds and videos being different media and compressed differently, they should be treated differently to find hidden information. More broadly, steganography in texts or in computer networks is so different by nature that it seems difficult to analyze them in a similar way;

– other works, in a way which is more similar to what is done in cryptanalysis, focus on searching for the insertion key from images containing hidden information. These works often require the steganalyst, Eve, to have significant knowledge;

– in *quantitative* steganalysis, the aim is to estimate the size of a possible hidden message. We can easily understand that estimating the size of the message can be compared to a binary detection problem. Quite a naive first view involves estimating

2 See section 5.1 for the definition of the roles of Alice, Bob and Eve.

3 See section 5.1 for the definition of the terms Cover and Stego.

the size of the hidden message and deciding that the medium is Stego when this estimate exceeds a given threshold;

– finally, we can mention active steganalysis, in which Eve slightly modifies the transmitted medium in order to preserve its visual appearance while making extraction of the hidden message impossible.

Because of lack of space, it is difficult to tackle all the problems that make up steganalysis in only one chapter. In addition, even though these issues are interesting, we will stick to only the classical framework that concerns the detection of information hidden in digital images.

As we have explained previously, steganalysis was mainly developed to make the evaluation of steganography methods possible. Most works in this field refer to Kerckhoffs' principle (Kerckhoffs 1883), reworked by Claude E. Shannon, like the fact that "the enemy knows the system used". In accordance with this principle, steganalysis generally relies on the fact that Eve is only unaware of the inserted message and the insertion key used. We will return to this in greater detail in section 8.3 on the scenario of *targeted* steganalysis, but this assumes that Eve knows the type of media – generally an image, the insertion algorithm, the size of the message and its statistical properties.

8.1.2. *Different methods to carry out steganalysis*

The most effective methods for hidden information detection are undoubtedly those based on signatures. In this case, it is possible to detect steganography indirectly, by identifying a particular property that is found systematically when the insertion of a message has been carried out with a certain tool and only in this case. It is therefore a question of detecting a specific feature linked to the use of a precise tool, rather than the presence of hidden information itself. This approach can be compared to what is done for computer security (detection of network attacks or viruses, for example) and is discussed in section 8.2 through some examples.

A second family of more general steganalysis methods, based on modeling and statistical detection, is discussed in section 8.3. In practice, this sort of detection method is very complex to implement, since it requires a very precise statistical model of a Cover image and a Stego image in order to be able to *statistically* evaluate if a given image is more likely to come from one of these two models. In this section, we precisely define the framework of *targeted* steganalysis, which is necessary for building statistical models like this.

Section 8.4 deals with the use of statistical learning methods. The most effective steganalysis methods rely on this approach, and so naturally it is those which have been the most widely studied. Here, steganalysis can be subdivided into two distinct

problems: the first relates to the extraction of relevant *characteristics*, and the second relates to learning to automatically classify, based on a vast set of examples, *characteristics* from Cover and Stego images.

Recently, we have witnessed a spectacular development of deep learning methods, or *Deep Learning*, which, on the other hand, is carried out in a single phase which combines characterization and classification. The use of these methods for steganalysis is discussed in section 8.5.

This chapter concludes, in section 8.6, with a short, non-exhaustive list of the topics that seem the most interesting and which remain largely open.

8.2. Incompatible signature detection

In the field of computer security, signature detection is generally defined as a detection method based on looking for specific patterns or characteristic traces indicating the use of a particular software or algorithm. Here, the software aspect is important, because a signature does not depend on the insertion method considered, but is quite specific to a given implementation. As we will see in this section, this signature can appear in the metadata of the media, or in the properties of the signal that form the media.

A simple example is illustrated by the F5 steganography algorithm, proposed by A. Westfeld in 2001 (Westfeld 2001). It is combined with a demonstrator in order to allow its use within the scientific community. Westfeld did not want to fully program the JPEG compression part and used an available encoder, but this encoder inserted the following comment systematically in the file header "JPEG Encoder Copyright 1998, James R. Weeks and BioElectroMech". As this encoder is used very little, it was possible to detect the images containing hidden information with F5, not by directly detecting the hidden data, but by detecting this very specific comment. Here, it is a matter of detecting an "implementation error" and steganography software should generally not leave " signatures" that allow it to be identified. In this sense, a signature is always a flaw linked to a very specific implementation.

A second example of signature detection, this time using mathematical properties, was presented in Goljan and Fridrich (2015) and relates to color images. Remember that the sensor of a photographic camera is insensitive to color by nature. To show the color, a red, green or blue microfilter is placed in front of each photo-site of the sensor, which then records the color information corresponding to the color of the filter (Sharma and Bala 2017). You must then "reconstruct the two missing colors", which is performed from neighboring pixels. It was found in Goljan and Fridrich (2015) that steganography can go against the fundamental rules of reconstruction of the value of the missing colors. For example, in the linear interpolation framework, if the green

component of a pixel estimated from its neighbors becomes more important than all its neighboring pixels because of steganography, this value will become incompatible with its neighbor and the steganography will be shown.

Goljan and Fridrich (2015), therefore, proposed using seven features, which count the number of pixels whose value does not agree with the rules for reconstructing colors from neighboring pixels; these values are always assumed to be zero for a natural image and make it possible to detect steganography in color images by signature.

Another example of incompatible signature is that of Cover images encoded in the spatial domain, but which have been previously compressed in the JPEG format. This strategy may seem interesting at first, because an uncompressed image can theoretically accommodate a greater amount of hidden information than a compressed version. However, if an image encoded in the spatial domain was first compressed using the JPEG standard, the 8×8 block of pixels can be broken down as a sum of the components of the *DCT* representation, weighted by integer coefficients. More precisely, let us denote by $\mathbf{X}$ an 8×8 block of pixels which can be written as:

$$\mathbf{X} = \text{round}\left(\sum_{k=0}^{7}\sum_{l=0}^{7} c_{k,l} q_{k,l} \mathbf{D}_{k,l}\right) \quad [8.1]$$

where $q_{k,l}$ is the elements of the quantization matrix, $c_{k,l}$ is the (unknown) coefficients of the components of the DCT base, noted $\mathbf{D}_{k,l}$, and round$(\cdot)$ is the rounding function.

If some pixels of this block, $\mathbf{X}$, are modified after JPEG compression, it becomes impossible to find integer coefficients, $c_{k,l}$, that make it possible to show the $\mathbf{X}$ block. The steganalyst, Alice, can then conclude that this image seems to have been compressed in the JPEG format, but that some pixels conflict with what should have been obtained during the decompression. Once again, this mathematical incompatibility can be used to detect a Stego image.

More recently, a fairly similar signature, also based on what has been called "compatibility with JPEG compression", is proposed in Butora and Fridrich (2020) and Cogranne (2020). In this case, it is a question of detecting steganography in the images compressed in JPEG format by using the fact that modifying the DCT coefficients can lead to the production of pixel values that are not possible for a natural image. This detection method uses a statistical steganalysis method and is described in more detail at the end of section 8.3.

Finally, note that while these signature detection methods are generally very reliable in the sense that errors are uncommon, each software or algorithm must

nevertheless be carefully analyzed in order to find characteristic traces which reveal that it has been used. This type of analysis is very time consuming while also being difficult to generalize. The following steganalysis strategies are intended to be more common.

8.3. Detection using statistical methods

The detection methods presented in the rest of this chapter are often less reliable than signature detection methods, but have the advantage of being much more general, in the sense that they aim to detect changes related to hiding information in the very content of a medium.

We first introduce simple statistical methods. Consider an image $\mathbf{X}$ of size $M \times N$, described as a matrix of pixels encoded on 8 bits, $x_{m,n} \in \{0; \ldots; 255\}$, so the steganalysis involves choosing between the two following hypotheses:

1) $\mathcal{H}_0$: pixels $x_{m,n}$ come from a Cover image;

2) $\mathcal{H}_0$: pixels $x_{m,n}$ come from a Stego image.

The main difficulty is to define precisely what statistically characterizes a Cover image, and what differentiates it from a Stego image.

8.3.1. *Statistical test of* χ^2

Historically, the first approach that addressed that this problem was proposed in Westfeld and Pfitzmann (1999). Unable to statistically describe what a natural image is, it has been proposed that we model the pixels after using steganography. To explain how this test works, we need to remember how the insertion of a message by least significant bits (LSB) works[4]. In order to insert the ith bit of the message $m_i \in \{0; 1\}$ into the pixel $x_{m,n}$, steganography by substitution of the least significant bits modifies the pixel in the following way:

$$z_{m,n} = x_{m,n} - \text{LSB}(x_{m,n}) + m_i \qquad [8.2]$$

where $z_{m,n}$ is the pixel of the Stego image and $\text{LSB}(x_{m,n})$ is the least significant bit of $x_{m,n}$.

It is generally accepted in steganalysis that the message inserted, $\mathbf{m} = (m_1, \ldots, m_I)$, is a sequence of all independent and identically distributed bits (i.i.d) according to a uniform law: $\mathbb{P}[m_i = 0] = \mathbb{P}[m_i = 1] = 1/2$.

4 See also section 5.3.1, in Chapter 5.

The result of the insertion of the bit m_i in the pixel $x_{m,n}$ is that the Stego pixel can be modeled by the following probability distribution:

$$\mathbb{P}\left[z_{m,n} = x_{m,n}\right] = \frac{1}{2} = \mathbb{P}\left[z_{m,n} = \bar{x}_{m,n}\right] \quad [8.3]$$

with $\bar{x} = x + (-1)^x$ the integer value x whose least significant bit has been inverted.

Furthermore, assuming that all of the pixels are used to hide a (very) large secret message, the distribution of Stego pixels can be modeled as follows:

$$\forall k \in \{0; \ldots; 255\}, \forall (m,n), \mathbb{P}\left[z_{m,n} = k\right] = \mathbb{P}\left[z_{m,n} = \bar{k}\right] \quad [8.4]$$

It then becomes possible to statistically test whether an image is a Stego image by measuring the difference between the theoretical distribution of equation [8.4] and that observed on an analyzed image. This is precisely the purpose of the χ^2 test that measures the difference between a theoretical distribution and those from observations as follows:

$$\chi^2 = \sum_{k=0}^{255} \frac{(N_k - N_k^\star)^2}{N_k^\star} \quad \text{with} \quad N_k^\star = \frac{N_k + N_{\bar{k}}}{2} \quad [8.5]$$

where N_k represents the number of pixels whose value is k and $N_k^\star$ represents the expected number of pixels with the value k.

In fact, equation [8.5] represents a measure of the difference between theoretical distribution and empirical distribution through the term: $(N_k - N_k^\star)^2$. Figure 8.1 illustrates the distribution models of equations [8.4] and [8.5] for a Stego image; in fact, we see a number of pixels k and $\bar{k}$ which are "equalized" on the Stego image.

When the value of χ^2 is beyond a given threshold, τ, the image is considered to be a Cover image, or statistically too different to the theoretical distribution to be considered a Stego image [8.4]. How should the threshold τ then be set? The authors in Westfeld and Pfitzmann (1999) propose choosing a threshold to make sure that, theoretically, a Stego image is considered as natural with a probability[5] p_0. For this, the authors use the distribution of χ^2, which makes it possible to calculate this probability with the following relationship:

$$p_{\chi^2}(\tau) = \mathbb{P}\left[\chi^2 > \tau\right] = 1 - \int_0^\tau \frac{t^{(\nu-2)/2} e^{-t/2}}{2^{\nu-2}\Gamma(\nu)} dt \quad [8.6]$$

with $\nu = 128 - 1$, the "number of degrees of freedom" defined by the number of "pairs of values", which can be swapped between them.

5 Probability known as "missed detection" or "false-negative".

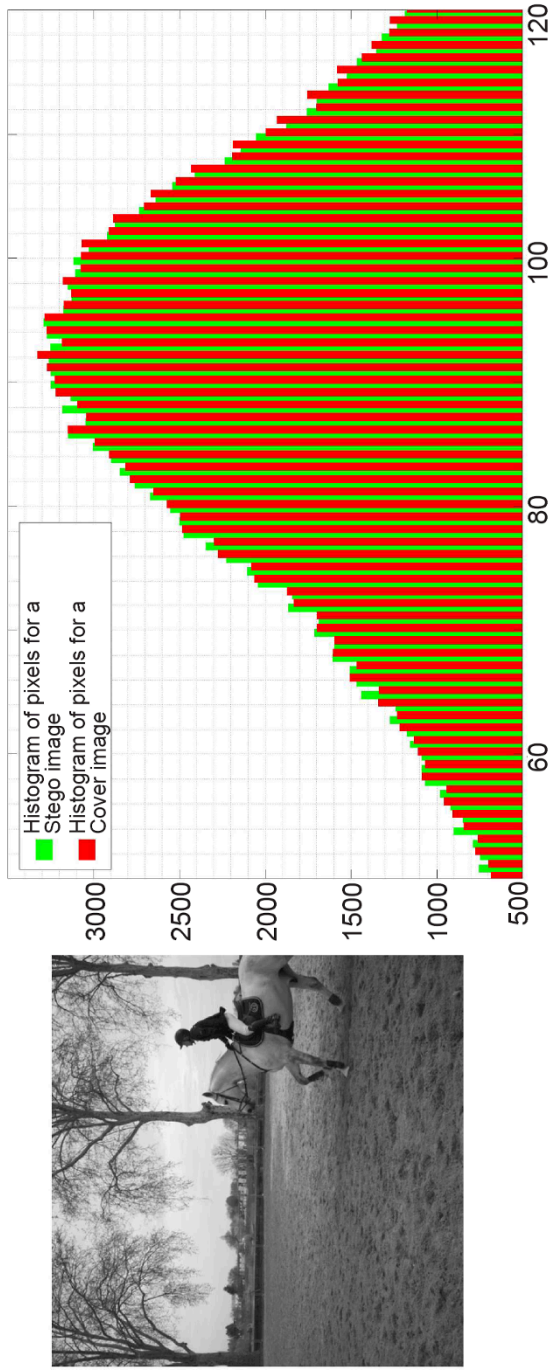


Figure 8.1. *Illustration of the impact of steganography by LSBR and its detection by the χ^2 test*

Figure 8.2 illustrates this way of setting the decision threshold depending on a "false-negative" probability, p_0, set previously. It also allows us to compare the theoretical distribution of the statistic, χ^2, from equation [8.5] with the theoretical distribution of χ^2. The difference between the observations and the theoretical model is mainly due to the fact that the vast majority of the images do not have pixels with all the values between 0 and 255, and therefore have a number of degrees of freedom less than $\nu = 128 - 1$.

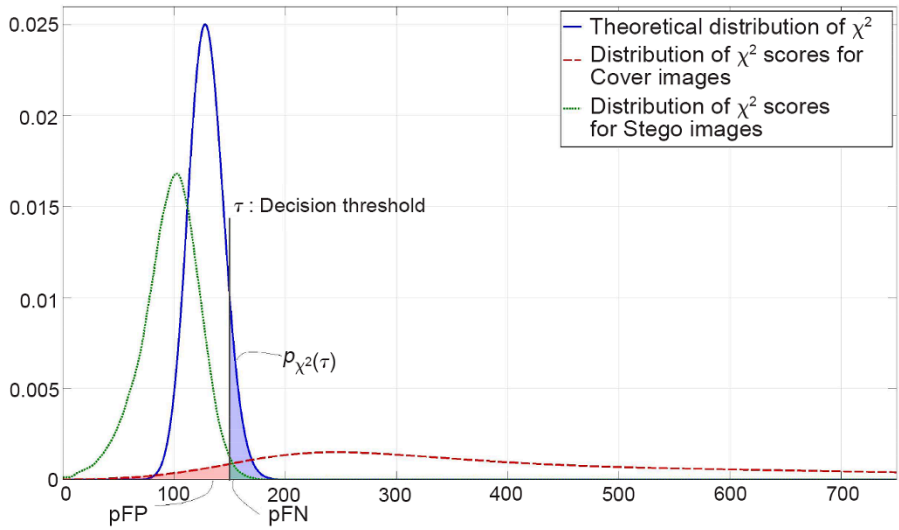


Figure 8.2. *Illustration of probability distributions (empirical and theoretical) for the result of the χ^2 test and the resulting error probabilities*

The strength of the χ^2 test is that it offers a statistical test without having to solve the (very tricky) problem of modeling the statistical distribution of pixels of a Cover image. In fact, the test essentially proposes checking whether the pixels of an analyzed image correspond to the model of a Stego image; otherwise the image is considered to be a Cover image by default. Another use of this test is to try to set the detection threshold, τ, based on a missed detection probability, $p_{\chi^2}(\tau)$, defined in equation [8.6].

Unfortunately, this test is not very efficient, especially since it does not model the statistical distribution of a Cover image, but only the Stego image. In statistical detection, when only one of the two hypotheses can be characterized, we generally carry out a "goodness-of-fit" test, measuring the adequacy of the observations to this model, and this is generally less reliable than when you can exactly characterize the

two competing hypotheses, as proposed by the likelihood-ratio test presented in section 8.3.2.

8.3.2. *Likelihood-ratio test*

In order to fully understand the approach presented in this section, it is necessary to formalize steganalysis and statistical detection.

A statistical test is a function δ, which, from a group of observations, $\mathbf{X}$, returns a binary decision: $\delta : \mathbf{X} \rightarrow \{0, 1\}$, so that the hypothesis, $\mathcal{H}_0$, is accepted if $\delta(\mathbf{X}) = 0$. Let us briefly recall that a test is never perfect and that there are two possible types of errors: false-positive and false-negative (or false alarm and no detection). In general, the hypothesis $\mathcal{H}_0$ is said to be "null", and corresponds to a "normal" case, while the alternative hypothesis, $\mathcal{H}_1$, corresponds to the difficult situation that we want to detect. The false-positive, therefore, corresponds to the case where the test decides to accept the hypothesis $\mathcal{H}_1$ when, in reality, the observations come from the hypothesis $\mathcal{H}_0$.

In the case that we are interested in, the analyzed image is a Cover image which is classified as a Stego image. On the other hand, a false-negative corresponds to the case where the test accepts the hypothesis $\mathcal{H}_0$, when the observations really come from the alternative hypothesis $\mathcal{H}_1$; for steganalysis, this is the case where the guardian, Eve, misses the detection of a Stego image.

In order to show how weak the χ^2 test is, we need a statistical model of the Cover images. This is built by assuming that the pixels are statistically independent and all come from a Gausian distribution:

$$x_{m,n} \sim \mathcal{N}\left(\mu_{m,n}, \sigma^2_{m,n}\right) \qquad [8.7]$$

where $\mu_{m,n}$ represents the mathematical expectation of the pixel (i.e. its "average" or theoretical value), and $\sigma^2_{m,n}$ represents the noise variance.

This is a model commonly used in image processing, which assumes that an image can be decomposed into a theoretical "content" and noise, linked to the camera defects. To be more precise, the pixels are represented by integer values and, for the sake of simplicity, we assume that this has no impact on the probability distribution of the pixels:

$$\mathbf{p}_0 = \mathbb{P}\left[x_{m,n}{=}k\right] = \left(p_0(k)\right)_{k\in\mathcal{Z}} \propto \left(\frac{1}{\sigma_{m,n}\sqrt{2\pi}} \exp\left(\frac{(k-\mu_{m,n})^2}{2\sigma^2_{m,n}}\right)\right)_{k\in\mathbb{Z}} \qquad [8.8]$$

First, we study the case where Alice uses steganography by LSB substitution[6]. Each pixel can be modified with the same probability of $\frac{\beta}{2} = \frac{I}{2MN}$, which relates to the ratio of the number of bits of the inserted message (I), per pixel (MN), and the distribution of a Stego pixel then becomes:

$$p_1(k) = \mathbb{P}\left[x_{m,n} = k | \mathcal{H}_1\right] = \left(1 - \frac{\beta}{2}\right) p_0(k) + \frac{\beta}{2} p_0(\bar{k}) \qquad [8.9]$$

Result \ Truth	Hypothesis 0: (Cover image)	Hypothesis 1: (Stego image)
Accept Hypothesis 0	Correct decision	False-negative (missed detection) (pFN: $1-\varsigma$)
Accept Hypothesis 1	False-positive (false alarm) (pFP: α=$\mathbb{P}\left[\delta(\mathbf{X})=\mathcal{H}_1 \middle| \mathcal{H}_0\right]$)	Correct decision (ς=$\mathbb{P}\left[\delta(\mathbf{X})=\mathcal{H}_1 \middle| \mathcal{H}_1\right]$)

Table 8.1. *The different possibilities of good and bad detection. For a color version of this table, see www.iste.co.uk/puech/multimedia1.zip*

How can we use these statistical models to decide whether an examined image, $\mathbf{X}$, comes from the hypothesis model instead $\mathcal{H}_0$[7], or from the hypothesis $\mathcal{H}_1$[8]?

There are several solutions that have one central element in common, the likelihood ratio (LR), which is expressed as:

$$\Lambda(\mathbf{X}) = \frac{p_1(\mathbf{X})}{p_0(\mathbf{X})} = \prod_{m,n} \frac{p_1(x_{m,n})}{p_0(x_{m,n})} \qquad [8.10]$$

with p_0 and p_1, which are the statistical distribution models for Cover and Stego images, respectively, defined by equations [8.8] and [8.9], respectively.

The second part of the equality of equation [8.10] results directly from the statistical independence model between the pixels which, although not totally exact, is very widely used because it significantly simplifies things.

Clearly, the likelihood ratio (LR) is greater if the probability of observing the data to be analyzed is greater under $\mathcal{H}_1$ than under $\mathcal{H}_0$; conversely, if the probability of

6 See also section 5.3.1.

7 Defined in equation [8.8].

8 Defined in equation [8.9].

observing these data is much greater under $\mathcal{H}_0$ than under $\mathcal{H}_1$, the LR will be low. Based on this observation, the likelihood ratio test simply involves thresholding this likelihood ratio: $\delta(\mathbf{X}) = \mathcal{H}_0$ if $\Lambda(\mathbf{X}) < \tau$ and $\delta(\mathbf{X}) = \mathcal{H}_1$ and $\Lambda(\mathbf{X}) \geq \tau$.

The use of the LR test is theoretically justified because the Neyman–Pearson lemma states that it helps achieve the greatest power[9] ς for a false-positive probability, α, set at $\alpha = \mathbb{P}\left[\Lambda(\mathbf{X}) > \tau\right]$. This last relationship also makes it possible to set the threshold in order to respect a previously established false-positive rate.

Using the Cover image model of equation [8.8] and the Stego image model of equation [8.9], the LR is written as:

$$\Lambda(\mathbf{X}) = \prod_{m,n} \left[(1-\beta) + \beta \frac{p_0(x_{m,n}) + p_0(\bar{x}_{m,n})}{2p_0(x_{m,n})} \right] \qquad [8.11]$$

For the sake of simplicity, the logarithm of LR is generally used. This is to replace the product in equation [8.11] by the sum. Moreover, by using the definition of $p_0(x_{m,n})$ equation [8.8], a (slightly tedious) calculation makes it possible to simplify the previous relationship as follows (Cogranne 2011):

$$\log\left(\Lambda(\mathbf{X})\right) = \sum_{m,n} \beta \log\left(\Lambda(x_{m,n})\right) = \sum_{m,n} \beta \frac{(x_{m,n} - \mu_{m,n})(x_{m,n} - \bar{x}_{m,n})}{2\sigma^2_{m,n}} \qquad [8.12]$$

In this approach, the most interesting thing is not so much having a simple relationship, making the calculation of the likelihood ratio possible (although some simplifications had to be made), but more being able to characterize the statistical distribution of the latter and therefore, *in fine*, controlling the probabilities of false alarms and non-detection.

In particular, it is possible to show the mathematical expectation and variance of the term $\log\left(\Lambda(\mathbf{X})\right)$ for a Cover image given by:

$$\mathbb{E}\left[\log\left(\Lambda(\mathbf{X})\right)|\mathcal{H}_0\right] = 0 \quad \text{and} \quad \mathbb{V}ar\left[\log\left(\Lambda(\mathbf{X})\right)|\mathcal{H}_0\right] = \frac{\beta^2}{4\sigma^2_{m,n}} \qquad [8.13]$$

It is then possible to standardize the LR as follows:

$$\log\left(\Lambda(\mathbf{X})\right) = \frac{1}{\varrho} \sum_{m,n} \log\left(\Lambda(x_{m,n})\right) \quad \text{with} \quad \varrho = \left(\sum_{m,n} \frac{\beta^2}{4\sigma^2_{m,n}} \right)^{1/2} \qquad [8.14]$$

9 See definition in Table 8.1.

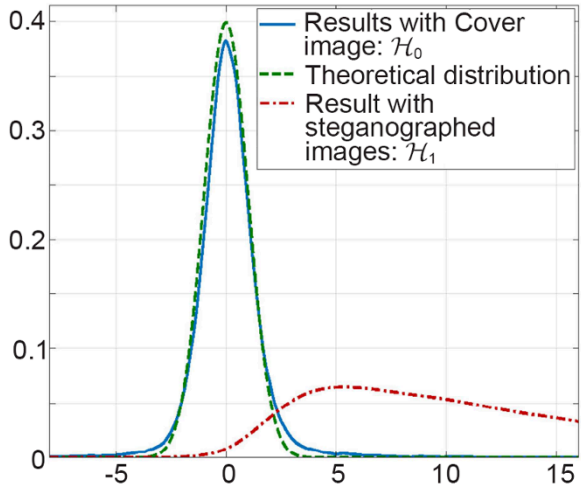
so that, in accordance with the central limit theorem (Lehmann and Romano 2005, theorem 11.2.5), it is possible to calculate the statistical distribution of the LR-log for a given image:

$$\begin{cases} \text{if } \mathcal{H}_0: & \log\left(\Lambda(\mathbf{X})\right) \sim \mathcal{N}\left(0,1\right) \\ \text{if } \mathcal{H}_1: & \log\left(\Lambda(\mathbf{X})\right) \sim \mathcal{N}\left(\varrho 1\right) \end{cases} \quad [8.15]$$

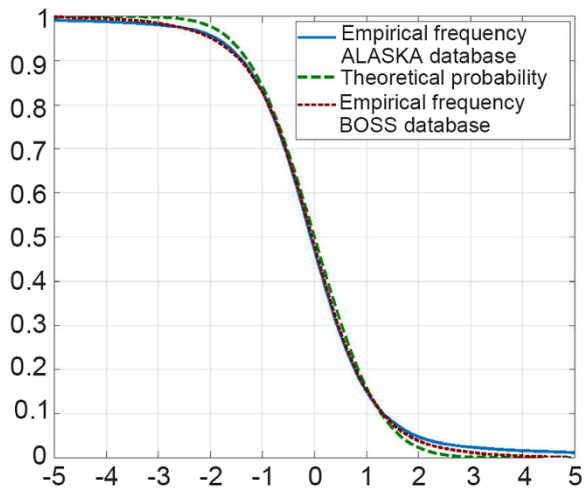
With the Gaussian distribution being fairly easy to use, we can easily calculate the decision threshold, τ, to ensure a fixed false-positive probability, α_0: $\tau(\alpha_0) = \Phi^{-1}\left(1-\alpha_0\right)$ so that $\mathbb{P}\left[\log\left(\Lambda(\mathbf{X})\right) > \tau(\alpha_0)|\mathcal{H}_0\right] = \alpha_0$. Similarly, the power of the test ς can be calculated as the probability, $\mathbb{P}\left[\log\left(\Lambda(\mathbf{X})\right) > \tau|\mathcal{H}_0\right] = 1 - \Phi\left(\tau - \varrho\right)$.

Many observations are necessary to understand these different results. On the one hand, let us note that it is very "typical" for the mathematical expectation of the LR for a Stego image to be equal to its variance. It is for this reason that the normalization factor, ϱ, in equation [8.14] also corresponds to the expectation in equation [8.15].

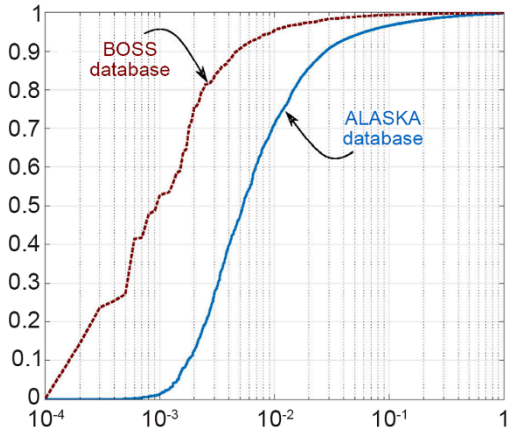
We also note that all the previous calculations require knowledge of the mathematical expectation, $\mu_{m,n}$, and the variance, $\sigma^2_{m,n}$, of each of the pixels, but, in practice, these variables are not known. It is therefore suggested that we replace them with estimates, which are quite hard to obtain "accurately", and this is where the application of this approach becomes much more complicated and the guarantee of the error probabilities in particular becomes very difficult. Some results are presented in Figure 8.3, using the ALASKA image database (Cogranne *et al.* 2019). Figure 8.3(a) shows that the distribution of the LR-log obtained for Cover images is fairly consistent with the theory. For Stego images, it depends on the ϱ factor, which varies for each image, creating this "spread". Figure 8.3(b) compares the false-positive probability depending on the decision threshold, τ, in theory and in practice, using two different image bases. We can also see that this property is valid for "fairly high" probabilities. However, note that, in the case where we want to obtain very low false-positive rates, the estimates are not precise enough to offer relevant guarantees. Finally, Figure 8.3(c) shows the performances obtained through a ROC (Receiver Operating Characteristic) curve, which presents a false-positive probability, $\alpha_0(\tau)$, depending on the detection power $\varsigma(\tau)$. We can see that for an insertion rate $\beta \approx 0.09$ (24 kilobits inserted into images of 512 x 512 pixels), the performance is very good on the two bases used for high error probabilities.



(a) Distribution of LR-log for Cover and Stego images



(b) Probability of false positive (PFA), α, depending on the decision threshold ; theory and empirical results



(c) ROC curve representing the power, ς depending on PFA, α

Figure 8.3. *Results of the application of the LRG test, equation [8.14]*

8.3.3. *LSB match detection*

Essentially, the detectors presented in section 8.3.2 show that steganography by substitution of the LSB clearly introduces a bias by increasing even values and decreasing odd values. This explains why this insertion method is to be avoided, to benefit the correspondence (LSBM or LSB$\pm$1), which modifies the least significant bits, according to the insertion rule defined in section 5.3.1.

After having presented the application of a statistical test for steganalysis in detail, we can address the detection of LSB$\pm$1, which has been studied in the literature through "simple" tests much less. We will also consider the case of an adaptive steganography scheme, that is, the probability of using the pixel $x_{m,n}$ is potentially different for each pixel[10] and will therefore be denoted $\beta_{m,n}$. The reader will note that it is very easy to replace the "average insertion rate", β with an insertion rate for each pixel, $\beta_{m,n}$, without any additional modification, in equations [8.9] and [8.11]–[8.14].

10 And this via the use of an insertion cost, see sections 5.2.3 and 5.3.2.

We will not explain all the details of the calculations, which can be found in Cogranne (2011) and Sedighi *et al.* (2016a), but the LR-log calculation for the least significant bits correspondence leads, after a few simplifications, to:

$$\Lambda^{\pm}(\mathbf{X}) = \sum_{m,n} \beta_{m,n} \Lambda^{\pm}(x_{m,n}) = \sum_{m,n} \beta_{m,n} \left(\frac{(x_{m,n} - \mu_{m,n})^2 - 1/12}{\sigma^4_{m,n}} - \frac{1}{\sigma^2_{m,n}} \right) \quad [8.16]$$

Again, the most interesting thing here is calculating the error probabilities of this test and, using the central limit theorem like before, this requires knowing the first two moments, which are given by:

$$\begin{aligned} &\mathbb{E}\left[\Lambda^{\pm}(x_{m,n}) | \mathcal{H}_0\right] = 0 \ , \ \mathbb{E}\left[\Lambda^{\pm}(x_{m,n}) | \mathcal{H}_1\right] = \frac{2\beta^2_{m,n}}{\sigma^4_{m,n}} \\ &\mathbb{V}ar\left[\Lambda^{\pm}(x_{m,n})\right] = \frac{2\beta^2_{m,n}}{\sigma^4_{m,n}} \end{aligned} \quad [8.17]$$

The comparison between the LR for LSB substitution detection, given in equation [8.12], and the LR for the detection of LSB±1, given in equation [8.16], shows that the detection of LSBR (substitution) is essentially based on a deviation from the expectation through the term $(x_{m,n}\mu_{m,n})$. Conversely, the detection of LSB±1 is essentially based on a difference between the theoretical variance and that observed through the term $(x_{m,n} - \mu_{m,n})^2$. However, if estimating the mathematical expectation of pixels is widely studied[11], the precise estimate of the variance of pixels is much harder and has been studied much less. We also want to detect a deviation from a variance that is not known and needs to be estimated, regardless of the presence of hidden information. All of this explains why the statistical steganalysis methods tackling adaptive insertion methods are not as efficient as the LSB substitution steganalysis seen in section 8.3.2.

The application of statistical detection explains why the steganalysis of LSB±1 is more difficult; it should be noted that the performance study also shows that, for a given pixel, the "detectablility" essentially depends on the "insertion-on-noise relationship", defined by $\beta^2_{m,n}/\sigma^4_{m,n}$ in the relationships of equation [8.17].

An interesting application of the theory of hypothesis testing has been to use this result concerning the "statistical detectability" of each pixel to design an insertion algorithm (Sedighi *et al.* 2016a) which, instead of minimizing the heuristic distortion, minimizes the theoretical detectability. Although this requires estimating the mathematical expectation and the variance of each pixel (which remains an unresolved problem), it has shown its effectiveness. More details on this application of the theory of hypothesis testing in steganography are given in section 5.3.2.2.

11 It is a matter of "denoising", which is of major interest for the improvement of images.

To conclude this section on statistical steganalysis, let us mention that a method of statistical steganalysis on JPEG images has recently been proposed that amazes with how efficient it is. This method, presented by Butora and Fridrich (2020) and Cogranne (2020), exploits the fact that the pixels are quantized before using the discrete cosine transform (DCT). For a compressed image, it is possible to decompress it[12] and to measure the variance of the quantization noise by:

$$\frac{1}{MN}\left\|\mathbf{X}-\text{round}\left(\mathbf{X}\right)\right\|_2^2=\frac{1}{MN}\sum_{m,n}\left(x_{m,n}-\text{round}\left(x_{m,n}\right)\right)^2 \qquad [8.18]$$

where M and N are the number of lines and columns of image $\mathbf{X}$, and round$\left(\cdot\right)$ is the rounding function to the closest integer value.

If information has been hidden in the DCT coefficients, the rounding error in the spatial domain makes it possible to obtain the statistic defined in equation [8.18], which will tend to increase. This test is really effective (it was slightly improved in Cogranne *et al*. (2020b) with a "near-perfect" detection of a few hundred bits). This can be explained by the fact that it is based on a very precise model and does not depend on the parameters to be estimated, but only on the quantization noise which is the same regardless of the image analyzed. However, this is a very specific case that can be equivalent to incompatible signature detection (see section 8.2).

8.4. Supervised learning detection

We will now describe steganalysis approaches that are radically different from what has been previously described. These methods are neither based on the presence of an incompatible signature, nor on the prior knowledge of a statistical model of the Cover and/or Stego image, but are based on supervised statistical learning methods, whose foundations can be broken down into two phases. First, it is a matter of extracting the "features" from objects in the database considered. These features must reveal the presence of hidden information (or what we want to detect in general) and must make it possible to reduce a complex and variable object, such as an image, of any size, to a real-valued vector of p. Based on these features, a learning method is used to determine a decision rule, which will provide a binary result, which will be the result of steganalysis. Thus learning is supervised, in that each image is associated, during this phase of learning, with a label that indicates if the image is a Cover or Stego. Determining a decision rule actually corresponds to the resolution of

12 That is, to recalculate the value of the pixels from the quantized DCT coefficients.

an optimization problem that seeks to associate a value that is as close as possible to the label to be predicted with each feature vector.

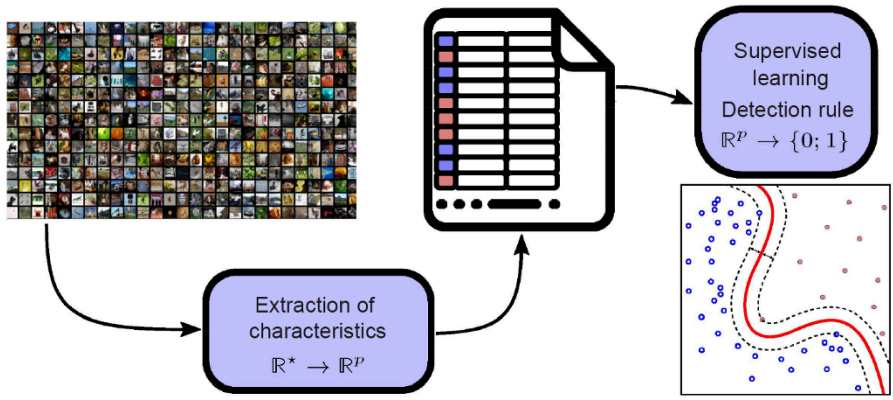


Figure 8.4. *Illustration of the supervised learning principal, which aims to determine a detection rule from a labeled database (right image licensed under CC BY-SA 4.0, produced by Zirguezi)*

In this section, we will now briefly describe how these two phases, (1) features extraction and (2) supervised learning, are usually implemented in steganography.

8.4.1. *Extraction of characteristics in the spatial domain*

8.4.1.1. *SPAM characteristics*

It would obviously take too long to describe the evolution of steganalysis up to modern techniques in detail, but note that, generally, a keystone of current methods originated in Pevný *et al*. (2010), which uses the differences between adjacent pixels that we will denote $\mathbf{D}^{\rightarrow}$:

$$d^{\rightarrow}_{m,n} = x_{m,n} - x_{m,n+1} \qquad [8.19]$$

with $\rightarrow$, which here represents the horizontal direction in which the differences are calculated, among the eight possible used $\{\leftarrow, \rightarrow, \uparrow, \downarrow, \nwarrow, \searrow, \nearrow, \swarrow\}$.

On the basis of these differences, it is suggested that we estimate the frequency with which successive differences are found in an image. Representing an empirical

frequency corresponds to a count that is generally, simply yet inaccurately, known as "histogram" in steganalysis. These counts are arranged in vector:

$$f^{\rightarrow}_{k,l} = \sum_{m,n} \mathbb{1}\left[d^{\rightarrow}_{m,n} == k, d^{\rightarrow}_{m,n+1} == l\right] \ , \ (k,l) \in \{-T, \ldots, T\} \qquad [8.20]$$

with $\mathbb{1}[\cdot]$, the indicator function, $\mathbb{1}[e] = 1$ if the event e is true – and $\mathbb{1}[c] = 0$ otherwise – and T, the maximum difference threshold considered.

Given that the values of adjacent differences together on neighboring positions are counted, these vectors are called "co-occurrence" in steganalysis. This concept can be generalized by using more than two disjoint values at the cost of increasing the number of possible co-occurrence values. In fact, by using co-occurrences of c adjacent differences between $-T$ and T, it gives $(2T+1)^c$ different values of possible co-occurrences. Finally, the last step proposed in Pevný *et al.* (2010) aims to limit the number of characteristics by gathering together the values calculated for opposite directions, for example $\leftarrow$ and $\rightarrow$, or even $\nwarrow$ and $\searrow$. The authors propose grouping together "diagonal" directions and horizontal and vertical directions:

$$f_{k,l} = \frac{1}{4}\left(f^{\rightarrow}_{k,l} + f^{\leftarrow}_{k,l} + f^{\uparrow}_{k,l} + f^{\downarrow}_{k,l}\right) \qquad [8.21]$$

This step, called "symmetrization", aims to reduce the number of characteristics.

The original SPAM, *Subtractive Pixel Adjacency Matrix* (Pevný *et al.* 2010) characteristics, use triplets of three adjacent differences, $c = 3$, $f_{k,l,m}$ (kown as second order), and three distinct values for each $T = 3$. This means counting a distinct number of "co-occurrences" of $(3 \times 2 + 1)^3 = 7^3 = 343$. Adding the fact that diagonal co-occurrences are distinguished from horizontal and vertical co-occurrences during the symmetrization phase, this gives a total of 686 characteristics.

Clearly this is quite a high number of characteristics (also know as dimensions) compared to the problems generally studied in the field of statistical learning.

8.4.1.2. *RM characteristics*

However, it has been empirically observed that, for the particular case of steganalysis, which essentially involves a very weak signal detection within the Cover media, generalizing an approach like this makes it possible to improve the detection performance. The approach that we have briefly described was largely developed afterward in a methodology whose operating principle is illustrated in Figure 8.5. This figure illustrates the "Spatial Rich Model" operating as proposed in Fridrich and Kodovskỳ (2012). This is essentially an improvement of the method

proposed in Pevný *et al*. (2010). Furthermore, these models are fairly standard, and also illustrate the fact that the current methods of steganalysis are divided into four main stages, which are (1) calculation of residuals; (2) quantization and thresholding; (3) counting co-occurrences and finally (4) redundancy reduction by symmetrization.

Calculating residuals is generally done by using a linear filter. From an image, $\mathbf{X}$, whose pixels are given by $x_{m,n}$, we apply the same relationship between adjacent pixels on the whole image:

$$r_{m,n}^{(n)} = \sum_{i,j} x_{m+i,n+j} k_{i,j}^{(n)} \qquad [8.22]$$

This weighted sum of neighboring pixels is a convolution operation, $\mathbf{R}^{(n)} = \mathbf{X} \star \mathbf{K}^{(n)}$, whose kernel $\mathbf{K}^{(n)}$ specifies the value of these weighting factors. We generally speak of a low-pass filter when $\sum_{i,j} k_{i,j} = 1$; it is typically a "smoothing", aiming to decrease the "noise". On the other hand, in steganalysis, the term "residuals" is used to characterize high-pass filtering, characterized by $\sum_{i,j} k_{i,j} = 0$, $k_{0,0} = -1$. In other words, it is about "estimating" the value of a pixel, $x_{m,n}$, from its neighbors, $\widehat{x}_{m,n}^{(n)} = \sum_{(i,j)\neq(0,0)} x_{m+i,n+j} k_{i,j}^{(n)}$, and then calculating the difference, $r_{m,n}^{(n)} = \widehat{x}_{m,n}^{(n)} - x_{m,n}$, which corresponds to the "estimation error", due to the filter, $\mathbf{K}^{(n)}$.

To be able to catch all the possible traces due to steganography, the number of filters is important; 78 residuals are used to construct the RM characteristics presented in Fridrich and Kodovskỳ (2012).

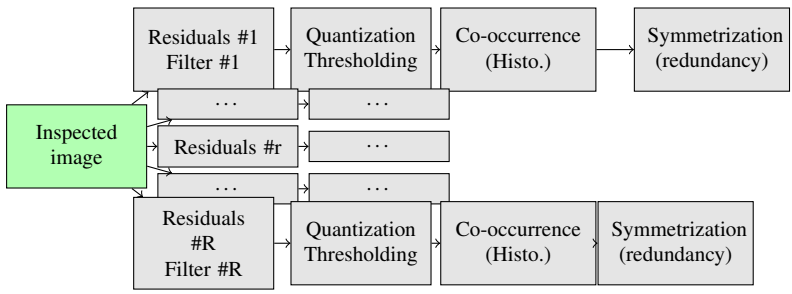


Figure 8.5. *Illustration of the principle of extraction of RM characteristics (Spatial Rich Model)*

The second step is quantization and thresholding. Generally, weighting factors used in the kernels $\mathbf{K}^{(n)}$ are not integers, and the residuals $\mathbf{R}^{(n)}$ must be grouped together into "close" values to be able to work out "histograms". The general idea is

dividing the residuals, $\mathbf{R}^{(n)}$, by a quantization step, q, and then rounding the results. The quantization step, q, then determines the "granularity" of the histogram. A very large step gives a rough histogram: for example, a step of $q = 10$ leads to rounding to the nearest 10; on the other hand, a step of $q = 1$ leads to rounding to the nearest integer. Thresholding simply involves limiting the range of possible values (despite quantization) in order to limit the extent of resulting "co-occurrences". It is simply suggested that we only count the values of the residuals below a certain threshold, T, and ignore the values beyond.

Finally, this second step can be represented by:

$$\widetilde{r}_{m,n}^{(n)} = \text{Threshold}_T \left(\text{Round} \left(\frac{r_{m,n}^{(n)}}{q} \right) \right) \quad [8.23]$$

with Threshold_T, the thresholding operation with the threshold T, and Round, the rounding operation to the nearest integer.

In Spatial Rich Models (Fridrich and Kodovskỳ 2012), three distinct quantization steps are used, ($q = \{1, 2, 3\}$), but the threshold is always set at $T = 3$, in order to be able to represent smaller or greater residue values, depending on the quantization step.

The third step aims to represent the "residuals" through histograms with several dimensions or co-occurrences. This mainly allows a better representation of the global statistical properties of the residuals, regardless of their positions in an image, but also to reduce the amount of data and, finally, to have an identical representation, whatever the size of the image analyzed. The co-occurrence calculation for a given type of residual, $\mathbf{R}^{(n)}$, is done, as explained in equation [8.20], by counting the number of adjacent values, whose values are all equal to a certain pattern. It is therefore a generalization of equation [8.20] to a tuplet of c adjacent residuals. It is also worth noting that by using c adjacent residuals, we can also multiply the directions, which can be much more general than just the eight directions considered, in the case of pairs. The calculation of co-occurrences is generally limited to vertical and horizontal directions, since all possible paths would become too large with co-occurrences of dimension $c = 4$, like the ones used in Fridrich and Kodovskỳ (2012).

Finally, the last step involves merging, similar co-occurrences, typically vertical to the left or to the right, generally by calculating an average. The underlying principle is that there is no objective reason to assume that traces of steganography are represented differently in one direction than they are in another. Symmetrization is a harder step to generally define, since it greatly depends on the filters used to calculate the residuals, on the one, and the co-occurrences used, on the other.

Generally, we agglomerate the different directions (vertical and horizontal co-occurrences), as well as the filters that can be deduced from one another by symmetry. This stage makes it possible to greatly reduce the number of characteristics used. In fact, consider the classic case of characteristics from rich spatial models (Fridrich and Kodovskỳ 2012). We count 78 distinct filters with three quantification steps, a thresholding with the threshold $T = 2$, and co-occurrences calculated horizontally and vertically. Eventually, an image characterized by RMs has (after symmetrization) 34,761 features.

8.4.1.3. *Extraction of characteristics in the JPEG domain*

With regard to images compressed in the JPEG format, extraction of characteristics introduces a first problem: should the DCT coefficients be used directly, which can be modified, or is it better to "decompress" the image in order to analyze the pixels?

The two approaches have been studied and, today, detection in the decompressed images shows more interesting performances. This is specifically due to the fact that the modeling and analysis of contiguous pixels are much simpler than the analysis of the DCT coefficients, which are the result of different filters and have properties that do not allow an easy analysis.

A compromise between these two areas of analysis has been proposed by Holub and Fridrich (2015); we will briefly describe it since it performs well and has led to many more recent works, in particular (Song *et al*. 2015). This approach, called DCTR (for DCT residuals), analyzes an image by first decompressing it, then applying a DCT transformation to the pixels, similar to that used in JPEG compression. More precisely, we have briefly explained that JPEG compression applies a series of 64 (8×8) orthogonal filters to disjoint blocks of 64 (8×8) pixels. The principle of DCTR is based on the use of these 64 filters on non-disjoint blocks. More precisely, these filters are applied by convolution with the decompressed image, which amounts to calculating DCT coefficients in addition to those used in the JPEG compression on parts of adjacent 8×8 blocks. The result of this operation is 64 distinct images that are the same size as the original image, each one corresponding to the application of one of the DCT transformation filters. This stage is equivalent to calculating 64 residuals.

These images are then quantified, modified in absolute values (negative values are reported as positive values), and then thresholded. The quantization is all the greater as the quality factor is small (in a similar way to JPEG quantization matrices) and the threshold is set at 4. For each of these 64 "sub-images", several histograms are calculated, depending on the position of the residuals. The idea is clearly to give a "reference" that is comparable to the DCT coefficients, which are used in JPEG compression (and therefore possibly modified by steganography) with those which are not used, but which correspond to images that are statistically very similar, as they are only shifted by a few pixels.

In the case of DCTR, no symmetrization is used; however, some positions of "DCT residuals" are gathered, and we end up with 25 histograms by DCT filters and $4 + 1$ values in the thresholded histogram, that is, a total of $64 \times 25 \times 5 = 8{,}000$ features.

8.4.2. *Learning how to detect with features*

Once the features have been extracted, a classification phase remains to be implemented. More precisely, it is necessary to extract these features for a large amount of Cover images in order to then hide information in these images using a steganography method to generate Stego images, and finally to extract characteristics from these Stego images. Learning a classification rule in such a framework is part of supervised statistical learning: we have seen two feature sets, the first for images without Cover information and the second for Stego images. There are many supervised statistical learning methods; in general, this corresponds mathematically to an optimization problem to find the parameters of a function, making it possible to maximize the detection performance. Since in steganalysis the main adapted classifiers are linear, we will focus on this type of method.

Consider two bases of L features calculated on I images, that we denote in matrix form $\mathbf{C} = (\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(I)})$ for Cover images (of class -1) and $\mathbf{S} = (\mathbf{s}^{(1)}, \ldots, \mathbf{s}^{(I)})$ for Stego images (of class 1). A linear classifier is based on a projection of the features on a discrimination vector, $\mathbf{w}^\top \mathbf{c}^{(i)} = \sum_{l=1}^{L} \mathbf{w}_l \mathbf{c}_l^{(i)}$. This operation is a weighted sum of features. The goal is therefore to find a vector of the weighting coefficients, $\mathbf{w}$, which makes it possible to differentiate the Cover and Stego data after projection. A simple criterion for this involves seeking weighting coefficients, $\mathbf{w}$, so that the vectors $\mathbf{c}_i$ are close to the value -1, while the vectors $\mathbf{s}_i$ are close the value 1.

If large vectors, $\mathbf{y}_0$ and $\mathbf{y}_1$, are created, with the first vector containing I times the value of -1 and the second vector containing I times the value of 1, it is then a question of finding the weighting factors, $\mathbf{w}$, that minimize the difference between the values $\mathbf{w}^\top \mathbf{C}$ and $\mathbf{y}_0$ (and vice versa between $\mathbf{w}^\top \times \mathbf{S}$ and $\mathbf{y}_1$), let $\left\|\mathbf{w}^\top \mathbf{C} - \mathbf{y}_0\right\|_2^2 + \left\|\mathbf{w}^\top \mathbf{S} - \mathbf{y}_1\right\|_2^2$.

This method actually comes to a minimization problem, in the sense of least squares. This approach is particularly interesting, since an analytical solution is given by:

$$\mathbf{w} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{y} \qquad [8.24]$$

where $\mathbf{X}$ is a matrix that groups together features $\mathbf{X} = (\mathbf{C}, \mathbf{S})$ and, similarly, the vector $\mathbf{y}$ contains all label values $\mathbf{y} = (\mathbf{y}_0; \mathbf{y}_1)$.

Therefore, this method functions correctly, and it is generally useful to add a regularization factor, which makes it possible to find a compromise between a

"simple" solution and a solution that is adapted to the data used for learning. It is this method which was proposed in Cogranne *et al*. (2015), and which allows performance that is very close to the state of the art in just a few seconds (for 10,000 images and 40,000 characteristics).

An interesting alternative approach that is based on another linear classifier is the Fisher approach, which is very similar to the previous one, but aims to maximize the separability criterion:

$$\frac{\mathbf{w}^\top (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)}{\mathbf{w}^\top (\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1)\, \mathbf{w}} \qquad [8.25]$$

where $\boldsymbol{\mu}_i$ is the mean of the features for the images of the class i, and $\boldsymbol{\Sigma}_i$ is the covariance matrix of the features of the class i.

This method also presents the advantage of having a directly calculable solution:

$$\mathbf{w} = (\boldsymbol{\Sigma}_0 + \boldsymbol{\Sigma}_1)^{-1} (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \qquad [8.26]$$

The main attraction of the method proposed in Kodovskỳ *et al*. (2012) is the use of a variety of classifiers. For this, many classifiers are trained on "sub-sets" of images and/or on "sub-sets" of characteristics. The attraction is mainly based on the fact that this specific training and this procedure for multiplying classifiers makes it possible to eventually build a "robust", nonlinear and efficient classifier. This method has been improved in Cogranne and Fridrich (2015), but, in both cases, the results are slightly better than those obtained with a simple linear classifier for a much greater computational complexity (of the order of 20–100 times more).

8.5. Detection by deep neural networks

We now present the latest developments in learning-based steganalysis methods, namely, the use of deep neural networks. Neural networks have been studied since the 1950s. Initially, they were proposed to model brain behavior. In computer science, especially artificial intelligence, they have been used for 30 years for learning purposes. In the early 2000s (Hinton and Salakhutdinov 2006), deep neural networks were considered to have a learning time that was too long, and to be less efficient than classifiers like SVMs or random forests.

Thanks to recent advances in the field of neural networks (Bengio *et al*. 2013), computing power provided by graphics cards (GPU) and, finally, the profusion of data, deep learning approaches have been proposed as a natural extension of neural networks. Since 2012, these deep networks have profoundly marked the fields of signal processing and artificial intelligence, because their performance has made it

possible to surpass the most efficient methods of the time, but also to solve problems that scientists could not solve until now (LeCun *et al*. 2015).

In steganalysis, over these last 10 years, detection of a hidden message in an image has mainly been carried out by the calculation of a rich model (RM)[13] (Fridrich and Kodovskỳ 2012) followed by a classification by an ensemble classifier (EC) (Kodovskỳ *et al*. 2012). In 2015, the first study using a convolutional neural network (CNN) obtained the first results of steganalysis by "deep-learning", coming close to the results of two-step approaches (EC + RM)[14] (Qian *et al*. 2015). Since then, over the period 2015 to 2019, many publications have shown that it is possible to obtain better performances, for spatial images (uncompressed/ steganalysis) and JPEG steganalysis, but also for informed steganalysis, quantitative steganalysis and color steganalysis.

In section 8.5.1, we present the structure of a deep neural network in a generic way. Reading this section can be supplemented by reading about artificial learning, and, in particular, about the definition of *perceptron*, *stochastic gradient descent*, *backpropagation* and the extension to the *multi-classes* case.

8.5.1. *Foundation of a deep neural network*

In the following sections, we recall major concepts of a CNN; we will recall the basic building blocks of a network based on the Yedroudj-Net network, which was published in 2018 (Yedroudj *et al.* 2018b) and which uses ideas that are present in Alex-Net (Krizhevsky *et al*. 2012), as well as those present in networks developed for steganalysis, including the very first network by Qian *et al.*, the Xu-Net network (Xu *et al*. 2016) and the Ye-Net network (Ye *et al*. 2017).

8.5.1.1. *Global view of a CNN*

Before describing the structure of a neural network, as well as its foundations, it is useful to remember that a neural network belongs to the family of machine learning. In the case of supervised learning, which is what interests us, it is necessary to have an image database, with a label for each image, that is, its class.

Deep learning networks are large neural networks that can directly take raw data as input.

In image processing, the network is directly supplied by the pixels that make up the image. A deep learning network then learns to extract the intrinsic characteristics

13 See section 8.4.1.2.

14 EC + RM will generally refer to two-step approaches based on the calculation of a rich model (RM) then the use of an ensemble classifier (EC).

(generally called a *feature map* or *latent space*), and to draw the separation boundary of the different classes (*separator planes*).

The learning protocol is similar to classic machine learning methods. Each image is given at the input of the network. Each pixel value is transmitted to one or more neurons. The network is made up of a given number of *blocks*. One block is made up of neurons that take real values as input, perform calculations and then transmit the calculated real values to the next block. A neural network can then be represented by a directed graph where each node represents a computation unit. The learning is then carried out by providing the network with examples composed of an image and its label, and the network modifies the parameters of these computation units by learning, thanks to the backpropagation and stochastic gradient descent algorithms.

The CNNs used for steganalysis are mainly built in three parts called *modules*: the preprocessing module, the convolution module and the classification module. As an example, Figure 8.6 outlines the network proposed by Yedroudj *et al.* in 2018 (Yedroudj *et al.* 2018b). This network processes grayscale images of pixel size 256×256.

8.5.2. *The preprocessing module*

In Figure 8.6, we can see that, in the preprocessing module, the image is filtered by 30 high-pass filters. The use of one or many high-pass filters as preprocessing is present in the majority of networks used for steganalysis during the period between 2015 and 2019. As an example, the kernel "square S5a" (Fridrich and Kodovskỳ 2012) is given by:

$$F^{(0)} = \frac{1}{12} \begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix} \quad [8.27]$$

This initial filtering step allows the network to converge faster and is probably necessary for good performance when the training base is too small (Yedroudj *et al.* 2018a) (only 4,000 Cover/Stego pairs of pixel size 256×256). The filtered images are then transmitted to the first convolution block of the network. The Yedroudj-Net network has five convolution blocks (Yedroudj *et al.* 2018b), like the Qian *et al.* network (Qian *et al*. 2015) and the Xu *et al.* network (Xu *et al*. 2016).

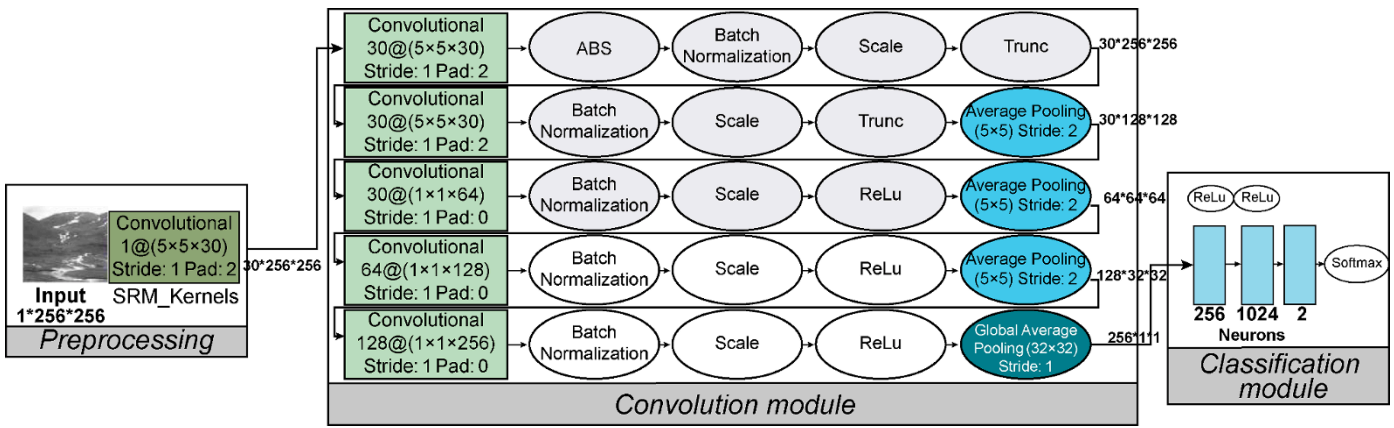


Figure 8.6. *The Yedroudj-Net (Yedroudj* et al. *2018b) network*

Note that the recent SRNet network (Boroumand *et al*. 2019) does not use fixed pre-filters, but learns the values of filters. This requires a much larger database (more than 15,000 Cover/Stego pairs of 256 × 256 pixels), and possibly using a preliminary training to start from a good initialization. There is also a debate in the community as to whether we should use fixed filter values, initialize the filter values with preselected values and then refine these values by training or have a random initialization and leave the network to learn the values of the filters. At the time of writing this chapter, we think that the best choice is related to the architecture of the network, the size of the training database used and the possibility of using pre-training/transfer learning.

8.5.2.1. *The convolution module*

To avoid any confusion on the terms, we will avoid using the term *layer*; the term *operations* will be preferred for an elementary function (convolution or activation, for example), and the term *block* for a set of these operations that can follow one another. A *block* is made up of computation units which take real values as input, perform calculations, then return real values, which are transmitted to the next block. In practical terms, a *block* takes a set of *feature maps*[15] on input and return a set of *feature maps* at output. Inside a block, we find a certain number of operations, including the following four: *convolution*, *activation*, *pooling* and finally *normalization*.

Note that the concept of a neuron as defined in the literature before the appearance of convolutional networks is still present, but it has disappeared from the source codes of data structures. In convolution modules, we have to imagine a neuron as a computation unit which, for a position in the *feature map* taken by the convolution kernel during the convolution operation, performs the weighted sum between the kernel and the group of pixels considered. The concept of a neuron corresponds to the scalar product between the input data, the pixels and data specific to the neuron[16] followed by the application of a function of $\mathbb{R}$ in $\mathbb{R}$ called the activation function. Then, by extension, we can consider that *pooling* and normalization are operations specific to neurons.

Not counting the preprocessing block, the Yedroudj-Net network (Yedroudj *et al.* 2018b) has a convolution module made up of five convolution blocks, like the Qian *et al*. network and the Xu *et al*. network (Xu *et al*. 2016). The Ye-Net network (Ye *et al*. 2017) has a convolution module made up of eight convolution blocks. The SRNet network (Boroumand *et al*. 2019) contains a convolution module made up of 11 convolution blocks.

In Figure 8.6, representing the Yedroudj-Net network, the first block of the convolution module generates 30 *feature maps*, each of pixel size 256 × 256. Note

15 Which can be seen as a group of images.

16 That is, the convolution kernel.

that this means 30 filters, and so 30 convolutions that are trained on the group of images given as input (30 filtered images) of pixel size 256×256. In each of the five blocks, there is a convolution operation, a normalization operation ("Batch" + "Scale"), an activation operation (ABS, trunc, ReLU) and a pooling operation ("average" or "max pooling").

8.5.2.2. *The classification module*

The last block of the convolution module (see section 8.5.2.1) is connected to the *classification module,* which is generally a *fully connected* neural network made up of one to three blocks. This *classification module* is often a traditional neural network where each neuron is completely connected to the next *layer* of neurons and to the previous *layer* of neurons.

After this completely connected neural network, we often find a "softmax" function, which makes it possible to standardize the two outputs provided by the network, so that the produced values belong to $[0, 1]$ and that their sum is equal to 1. The softmax function then returns a class membership score, that is, a score per output neuron. These scores are simply called probabilities. We will keep this name. In the typical scenario of binary steganalysis, the network therefore delivers two output values: one giving the probability of classification in the first class (e.g. the Cover class), and the other giving the probability of classification in the second class (e.g. the Stego class). The classification decision is then obtained by returning the class with the highest probability. The Yedroudj-Net network (see Figure 8.6) effectively provides two output values.

Note that before this *classification module*, we can find a particular *pooling* operation, such as a *Global Average Pooling*, a *Spatial Pyramid Pooling* (SPP) (He *et al*. 2014) or a *Statistical Moment Extractor* (Fuji-Tsang and Fridrich 2018). A pooling operation like this returns a vector of fixed size, that is, a feature map that is a fixed size, and this is whatever the dimension of the image at the input of the network. The block coming after this pooling operation is always connected to a vector of a fixed size, so it has a fixed number of input parameters. Therefore, it is possible to present images of all sizes to the network, without having to modify the topology of the network. This property is available in the Yedroudj-Net network (Yedroudj *et al.* 2018b), the Zhu-Net network (Zhang *et al*. 2020) or the (Fuji-Tsang and Fridrich 2018) network.

Note that Fuji-Tsang and Fridrich (2018) is the only article (up to 2019 and the ALASKA steganalysis competition (Cogranne *et al*. 2019; Yousfi *et al*. 2019)), which seriously considered the viability of a network invariant to the dimension of images. However, the problem remains unresolved. The solution proposed in Fuji-Tsang and Fridrich (2018) is a variation of the concept of "average pooling". Up to 2019, the small number of studies is not enough to determine which network topology to choose,

or how to build the training database, or to what extent the number of inserted bits influences learning.

We end here with the quick presentation of CNNs, seen through the major publications in steganography/steganalysis[17].

8.5.2.3. *Using the modification probability map (Selection Channel-Aware) (SCA)*

The modification probability map, when known to Eve, can significantly improve performance in steganalysis. At the end of 2018, two approaches (*Selection Channel-Aware* [SCA]) combine this knowledge with deep neural networks: SCA-Ye-Net (which is the SCA version of Ye-Net) (Ye *et al*. 2017) and SCA-SRNet (which is the SCA version of SRNet) (Boroumand *et al*. 2019). The idea is to use a network used for uninformed steganalysis and to inject, not only an image to be steganalyzed, but also a modification probability map. Therefore, we assume that Eve knows, or can obtain, a good estimate (Sedighi and Fridrich 2015) from this map, that is, that Eve has access to information about the selection channel (SCA).

This modification probability map is given to the SCA-Ye-Net (Ye *et al*. 2017) preprocessing block, and equally to the first convolution block for SCA-SRNet (Boroumand *et al*. 2019), but the kernel values are replaced by their absolute value. After convolution, each feature map is added point-by-point to the corresponding filtered modification probability map. Note that the activation function of this first convolution (preprocessing block for SCA-Ye-Net or first block for SCA-SRNet) is (if this is not already the case) replaced by a "ReLU" activation. In SCA-Ye-Net, the truncation activation function is, in fact, replaced by a ReLU. This makes it possible to "virtually" spread the information relating to the image and the information relating to the probability map throughout the network.

Note that this procedure for transforming a classical network into an SCA network is inspired by the propagation of the modification probability map proposed by Denemark *et al*. (2016). The two articles cited in the previous paragraph are an improvement in comparison to the previous maxRM *Rich Models* (Denemark *et al*. 2014). In maxRM, instead of accumulating the number of occurrences in the co-occurrence matrix, we use an accumulation of the maximum of a local probability. In Denemark *et al*. (2016), the idea is to transform the modification probability map in the same way as filtering the image, then update the co-occurrence matrix using the modified version of the modification probability map rather than the initial modification probability map.

17 For more details on the description of convolution, activation, pooling, normalization, effectiveness and complexity in terms of time and memory, or the link between *Deep Learning* and past approaches, or even integration approaches by *Deep Learning*, the reader can visit Chaumont (2020); available at: https://arxiv.org/abs/ 1904.01444.

8.5.2.4. *JPEG steganalysis*

The best CNN for JPEG steganalysis at the end of 2018 was SRNet (Boroumand *et al*. 2019). At that time, it was the only network for JPEG steganalysis to obtain an SCA version. It is interesting to list and briefly discuss previous CNNs used for JPEG steganalysis. The first network, published in February 2017, was the Zeng *et al.* network. It was assessed with 1 million images (Zeng *et al*. 2017, 2018). Then, in June 2017, at *IH&MMSec'2017*, two networks were proposed: PNet (Chen *et al*. 2017) and Xu-Net-Jpeg (Xu 2017). Finally, SRNet (Boroumand *et al*. 2019) was put online in September 2018.

In the Zeng *et al*. network (Zeng *et al*. 2017, 2018), the preprocessing block takes as an input a "de-quantized" image (real values), convolves it with 25 DCT bases and then quantizes and truncates the 25 filtered images. This preprocessing block uses handcrafted filter kernels (DCT bases), the kernel values are fixed, and these filters are inspired by the rich DCTR models (Holub and Fridrich 2015). It is based on three different quantizations, and the preprocessing block produces 3×25 residual images. The CNN is then made up of three sub-networks that each produce a feature vector of dimension 512. The sub-networks are inspired by Xu-Net (Xu *et al*. 2016). The three characteristic vectors, returned by the three sub-networks, are then given to a fully connected structure, and the network ends with a softmax layer.

Like what was done for spatial steganalysis, this network uses a preprocessing block proposed by Holub and Fridrich (2015). Note that the most effective rich models are the *Gabor Filter Rich Models* (GFR) (Song *et al*. 2015). Also note that this network takes advantage of the concept of an ensemble of features, which comes from the three sub-networks. The Zeng *et al.* network is less effective than the Xu-Net-Jpeg network (Xu 2017), but gives an interesting first approach, guided by *Rich Models*.

The main idea of PNet (and also VNet which is less efficient, but takes up less memory) (Chen *et al*. 2017) is to mimic Phase-Aware Rich Models, such as DCTRs (Holub and Fridrich 2015) or GFRs (Song *et al*. 2015), and to obtain an input image, broken down into 64 feature maps, representing the 64 phases of a JPEG image. The preprocessing block takes a de-quantized image (real values) as an input, performs convolutions with four filters, the "SQUARE5 $\times$ 5" coming from the *Spatial Rich Model* (Fridrich and Kodovskỳ 2012), a high-pass filter, "point", (called "catalyst kernel"), which completes the "SQUARE5 $\times$ 5", and two directional Gabor filters (angles 0 and $\pi/2$).

Straight after the second convolution block, a *PhaseSplit Module* divides the residual image into 64 feature maps (a map = a stage), in the same way as rich models. Some interesting procedures have been used, such as (1) linking convolutions with fixed values in the preprocessing block, with a second convolution whose weights are learned; (2) a smart update of the parameters of batch

normalization; (3) the use of "Filter Group Option", which potentially builds sub-networks; (4) *bagging* on five cross-validations; (5) using the last five evaluations to give the average error of a network; (6) shuffling the database at the start of each period in order to have better behavior of batch normalization and help generalization; and (7) potentially the use of an ensemble of networks of the same type with a pooling of the outputs by majority vote. With such expertise, PNet beat the classic two-step machine learning approaches in non-SCA and SCA (Classifier set + GFR) scenarios.

The Xu-Net-Jpeg (Xu 2017) is much more attractive as the approach is slightly better than PNet and does not require a strong domain inspiration, as you do for PNet. Xu-Net-Jpeg is heavily inspired by ResNet (He *et al*. 2016), a well-established network in the machine learning community. ResNet allows the use of deeper networks, thanks to the use of shortcuts. In Xu-Net, the preprocessing block takes a de-quantized image (real values) as an input, convolves the image with 16 DCT bases (in the same way as the Zeng *et al.* network (Zeng *et al*. 2017, 2018)) then applies an absolute value, a truncation and a set of BN convolutions, ReLU, until obtaining a feature vector of dimension 384, which is given to a fully connected block. Note that "max pooling" or "average pooling" is replaced by convolutions. This network is really simple and in 2017 it was the most effective method. In general, the networks proposed by the machine learning community are often very competitive, with little specific knowledge of the steganalysis field to integrate into the topology of a network to obtain a very efficient network[18].

In 2018, the state-of-the-art CNN for JPEG steganalysis (which can also be used for spatial steganalysis) was SRNet (Boroumand *et al*. 2019). Note that for the SRNet version, which knows about the selection channel (SCA), the modification probability by DCT coefficient is first re-expressed in the spatial domain by applying an inverse DCT, and by using the absolute values for the DCT base. The *selection map* obtained then enters the network and is convolved with each kernel (this first convolution is equivalent to the preprocessing block). Note that convolutions in this first block of this *selection map* are such that the kernels of the filters are modified to their absolute values. After passing the convolution, the *feature maps* are added with the square root of the previously convolved values of the *selection map*. Note that this idea goes back over what was presented in the SCA Ye-Net version (SCA-TLU-CNN) (Ye *et al*. 2017), with integration of adjacent information, and the recent proposal for steganalysis with awareness of the selection channel (SCA) in

18 The recent ALASKA#2 competition (Cogranne *et al*. 2020a) has also shown that networks from the *Deep Learning* community can be directly used in steganalysis producing excellent results.

JPEG with *Rich Models* (Denemark *et al*. 2016), or the construction of the *selection map*, and more specifically, the quantity $\delta_{uSA}^{1/2}$[19].

8.6. Current avenues of research

We finish this introductory chapter to steganalysis with a brief list of the open problems that seem to be the most interesting and the most important in this field.

8.6.1. *The problem of* Cover-Source mismatch

The first problem that we will describe is *Cover-Source mismatch* (CSM); this problem is actually a fairly general case in the field of statistical learning, where it is referred to as "generalization". In practice, this is the mismatch between the learning base, on which the classifier is trained, and the test base on which we want to detect the presence of hidden information. The defining feature of steganalysis is double, compared to CSM. On the one hand, in steganalysis we want to detect very weak signals; on the other hand, we generally work with characteristics of (very) large dimensions. Together, these two features mean that the detection methods will be very sensitive to the learning bases and difficult to generalize.

A fairly exhaustive study concerning the evaluation of the factors leading to CSM was carried out in Giboulot *et al*. (2020), showing, for example, the major role of image processing. Unfortunately, there is no work to understand the root causes yet, and, as a result, it remains very difficult to overcome this problem.

8.6.2. *The problem with steganalysis in real life*

Similarly, steganalysis *in real life* been not been explored much (Ker and Pevný 2014). In fact, as we mentioned in the introduction, steganalysis was developed with the main aim of evaluating the different methods of steganography. To do this, the community generally works using the targeted steganalysis scenario and with specific image databases, in particular the BOSS (Bas *et al*. 2011) database consisting of images from seven reflex cameras (no smartphone or compact), all processed the same, starting from the RAW file. Work has shown that developing these RAW images in different ways can lead to very different results (Sedighi *et al.* 2016b). A first competition was launched for this purpose with a much more heterogeneous image base (Cogranne *et al*. 2019) and proposals show the difficulty of carrying out steganalysis on heterogeneous images. The winners, in particular, relied on a increase of learning, according to the JPEG compression parameters (Yousfi *et al*. 2019), but many questions about steganalysis in real-life remain open.

19 uSA means *upper bounded Sum of Absolute values*.

8.6.3. *Reliable steganalysis*

A problem that is fairly similar to the previous two concerns reliable steganalysis. In fact, as we mentioned in section 8.3.2, a test is inevitably full of errors (false-positives and false-negatives), and the result depends greatly on the application context. In steganalysis, it is important to minimize false-positives, given the large number of images that we may have to analyze. On the other hand, methods based on statistical learning generally aim to minimize the overall error rate, P_E. However, assuming that it is possible to design a detection method where the probabilities of false-positive and false-negative are both about 1%, it is clearly unrealistic to use this method in practice. It is clear to argue that, with a Bayesian approach, an image considered as being steganographed by a detector like this would actually be more likely from a false-positive, given that on an image-sharing site such as Flickr, probably less than 1% of images actually contain hidden information.

Therefore, the crucial question arises of how to design a *reliable* steganalysis method that would guarantee a very low probability of false-positive; typically, a probability of false-positive less than 10^{-6} (i.e. less than 1 in 1 million). We have seen that the theory of hypothesis tests makes it possible to design methods of steganalysis like this, but relies on a statistical model of the images which is not necessarily exact, even more so when certain parameters have to be estimated. A very interesting study in Pevný and Ker (2015) studies the possibility of learning with the criterion of minimizing the probability of false-positive, if the power ς is fixed at 50%. However, this preliminary work deserves to be looked at in more depth, and steganalysis methods providing a precise "p-value"[20] are sorely lacking.

8.6.4. *Steganalysis of color images*

As indicated in Chapter 4, the vast majority of images today are in color; for practical applications, it is therefore necessary to study this type of image. However, academic work differs from practical use since the vast majority of work in steganalysis relates to "grayscale" images (see Cogranne *et al.* 2019, 2020a, 2020b).

On the contrary, detection of hidden information in color images seems more interesting in practice. This subject remains largely undiscovered. In fact, a significant number of researchers think that a color image can be represented by three images in grayscale (one red, one green and one blue), which can be analyzed separately.

However, it seems intuitive to consider that the color channels are statistically correlated, in particular due to the "demosaicing" during the acquisition and

20 That is, a probability that this detection corresponds to a false-positive for an given image.

processing of images. However, work which has been carried out in this field has not shown a great improvement by analyzing the components together (Goljan *et al*. 2014; Abdulrahman *et al*. 2016). For example, the first work in this field (Goljan *et al*. 2014) proposed adding histograms obtained from pixel values in different colors. This certainly allowed a slight improvement in detection, but below what was expected.

Finally, this field of study has not been studied very much for color images compressed with the JPEG standard. This is more harmful, because JPEG color images do not represent green, red and blue, but luminance/chrominance components whose statistical correlations are less significant. On the other hand, color channels are treated separately in the JPEG standard, so they should be analyzed in different ways. Unfortunately, detection of steganography in JPEG color channels has hardly ever been studied (see, for example, (Taburet *et al*. 2018) and the articles (Cogranne *et al*. 2019; Yousfi *et al*. 2019) relating to the ALASKA steganalysis challenge and about JPEG color images).

8.6.5. *Taking into account the adaptivity of steganography*

A very different problem, already mentioned in section 8.5.2.3 in the context of neural networks, is about the consideration of the "selection channel" for steganalysis. The channel selection shows the probabilities of using each pixel, $\beta_{m,n}$. Clearly, it seems interesting (as shown by equations [8.14] and [8.16]) that a detector can use a, possibly misleading, estimate of these probabilities of use. But this is not very effective with current detection methods. For example, one of the first advances proposed in the field (Denemark *et al*. 2014) involves replacing the calculation of co-occurrence histograms (see section 8.4.1) by adding, for each of the blocks considered, the maximum probability of modification, $\beta_{m,n}$, of this sample. In this study (Denemark *et al*. 2014), the authors indicate that they have tried several weighting functions and finally found that the *maximum* obtains the best results, in terms of performance. In addition, the authors indicate that this function is also robust to a misestimation of the selection channel, $\beta_{m,n}$. This is particularly important, since this quantity is not directly attainable and therefore must be estimated. This requires knowing the insertion method and size of the message and, in practice, the performances obtained from the estimated probability or from the real probability are very similar. In all cases, SCA methods (*SCA*) remain very *ad hoc*, and they are empirically explained, mainly by results in terms of statistical detection performance.

8.6.6. *Grouped steganalysis (batch steganalysis)*

A problem that has also been studied very little concerns "grouped" steganography, also called "batch" steganography (Ker and Pevný 2012). This is a

more general scenario where Alice can "spread out" her message in many different images, which are then sent to Bob. The steganalyst, Eve, must therefore observe a batch of images, q, and make a "grouped" decision about all of these images: they may all be Cover images, or some may contain a variable part of the message to hide. For steganography[21], this case can be modeled as a problem of message allocation, which must be dispersed in several images in order to minimize the probability of detection. For the steganalyst, Eve, the job of analyzing a batch of images is much more tricky. Many questions remain open, in particular: How can you analyze a lot of images, unknown at first? Is it better to use a method designed to analyze several images together, or should you carry out q independent image analyzes? Finally, should not the order of the images be taken into account? Some works have been proposed, in particular (Cogranne *et al*. 2017; Pevný and Nikolaev 2015), which are based on a detector adapted to the analysis of each image individually. The first article studies how to regulate the "(continuous) scores" obtained for each image. The second work (Pevný and Nikolaev 2015) carries out a detection in two stages: a histogram of the same "scores" is first constructed, then a second classifier is trained to identify the batches of Cover images. More recently, a study relating to the lack of knowledge of the strategy of spreading the message has been proposed in Zakaria *et al*. (2019).

8.6.7. *Universal steganalysis*

The last issue that we want to tackle is universal steganalysis. Again, this is a more general framework than targeted binary decision, which, as we have shown, makes it possible to assess a specific method of steganography. In practice, it is interesting to design steganalysis methods that work when neither the size of the message nor the insertion algorithm are known. Steganalysis without knowing the size of the message refers to so-called "quantitative" approaches, that is, aiming to estimate the size of the inserted message (Pevný *et al*. 2009). The study of the performance of the general classifier (Cogranne and Fridrich 2015), in the context where the size of the message is unknown, specifically shows that the latter (Kodovskỳ *et al*. 2012) cannot be used directly, since the detection threshold is fixed according to the steganographed images used during training. The approach proposed in Pevný *et al*. (2009) essentially involves using images containing different sized messages during training.

Similarly, multi-class steganalysis has not received much attention. Again, the extension of the general classifier has been considered with this in mind (Cogranne and Fridrich 2015) by using two simple and relatively efficient approaches; the first, known as "Cover media *versus* the rest", involves creating an alternative hypothesis

21 See also Chapter 5.

gathering all the possible insertion algorithms and, in the end, carrying out a statistical learning aimed at detecting any of the steganography methods. On the other hand, the second approach involves considering $H + 1$ hypotheses, relating to the H insertion algorithms considered, then the case of Cover images is added. To detect steganography and identify the insertion method, it is possible to create many classifiers that are specifically trained to distinguish between two hypotheses. Analyzing an image therefore requires using all the classifiers to then choose the hypothesis that gathers the most "votes" from the set of all the classifiers.

8.7. Conclusion

In this chapter, we have presented different steganalysis methods. Generally, we have presented "signature" detection methods, which essentially aim to detect traces linked to the use of specific software. Although very efficient, these methods are not very generalizable and not very interesting from a methodological point of view.

We have also studied statistical steganalysis methods, which require an exact model of statistical distribution of the Cover and Stego images. These methods are not very efficient in terms of detection, but allow us to understand how we can carry out steganalysis. This makes it possible to improve the steganananalysis in particular, or even to come up with a steganography method.

Finally, we have seen that, in practice, the most effective methods are based on statistical learning. This can be done by extraction of characteristics, and then classification. Recently, methods based on deep neural networks have made it possible to considerably improve the performance of steganalysis methods, at the expense of increased calculation complexity.

We have also seen that steganalysis is a constantly evolving field. In particular, we insisted on the fact that most academic work is difficult to use in practice. In particular, the main obstacles that remain to be studied have been described. In this field, the recent steganalysis "challenges" that we organized in 2019 (Cogranne *et al*. 2019) and in 2020 (Cogranne *et al*. 2020a) have made it possible to make significant progress and will certainly impact work in the coming years.

8.8. References

Abdulrahman, H., Chaumont, M., Montesinos, P., Magnier, B. (2016). Color image steganalysis based on steerable Gaussian filters bank. *Workshop on Information Hiding and Multimedia Security*, Vigo, 109–114.

Auguste, K. (1883). La cryptographie militaire. *Journal des sciences militaires*, 9(538), 5.

Bas, P., Filler, T., Pevný, T. (2011). "Break our steganographic system": The ins and outs of organizing boss. In *Information Hiding*, Filler, T., Pevný, T., Craver, S., Ker, A. (eds). Springer, Prague.

Bengio, Y., Courville, A.C., Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798–1828.

Boroumand, M., Chen, M., Fridrich, J. (2019). Deep residual network for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 14(5), 1181–1193.

Butora, J. and Fridrich, J. (2020). Reverse JPEG compatibility attack. *IEEE Transactions on Information Forensics and Security*, 15, 1444–1454.

Chaumont, M. (2020). Deep Learning in steganography and steganalysis. In *Digital Media Steganography: Principles, Algorithms, Advances*, Hassaballah, M. (ed.). Elsevier, Amsterdam.

Chen, M., Sedighi, V., Boroumand, M., Fridrich, J. (2017). JPEG-phase-aware convolutional neural network for steganalysis of JPEG images. *Workshop on Information Hiding and Multimedia Security*, Philadelphia, 75–84.

Cogranne, R. (2011). Détection statistique d'informations cachées dans une image naturelle à partir d'un modèle physique. PhD Thesis, University of Technology of Troyes.

Cogranne, R. (2020). Selection-channel-aware reverse JPEG compatibility for highly reliable steganalysis of JPEG images. *ICASSP 2020, International Conference on Acoustics, Speech and Signal Processing*, Barcelona, 2772–2776.

Cogranne, R. and Fridrich, J. (2015). Modeling and extending the ensemble classifier for steganalysis of digital images using hypothesis testing theory. *IEEE Transactions on Information Forensics and Security*, 10(12), 2627–2642.

Cogranne, R., Sedighi, V., Fridrich, J., Pevný, T. (2015). Is ensemble classifier needed for steganalysis in high-dimensional feature spaces? *International Workshop on Information Forensics and Security*, Rome, 1–6.

Cogranne, R., Sedighi, V., Fridrich, J. (2017). Practical strategies for content-adaptive batch steganography and pooled steganalysis. *International Conference on Acoustics, Speech and Signal Processing*, New Orleans, 2122–2126.

Cogranne, R., Giboulot, Q., Bas, P. (2019). The ALASKA steganalysis challenge: A first step towards steganalysis "into the wild". *Workshop on Information Hiding and Multimedia Security*, New York, 125–137.

Cogranne, R., Giboulot, Q., Bas, P. (2020a). Challenging academic research on steganalysis with realistic images. In *International Workshop on Information Forensics and Security*, New York.

Cogranne, R., Giboulot, Q., Bas, P. (2020b). Steganography by minimizing statistical detectability: The cases of JPEG and color images. *Information Hiding and MultiMedia Security*, Denver.

Denemark, T., Sedighi, V., Holub, V., Cogranne, R., Fridrich, J. (2014). Selection-channel-aware rich model for steganalysis of digital images. *International Workshop on Information Forensics and Security*, Atlanta, 48–53.

Denemark, T., Boroumand, M., Fridrich, J. (2016). Steganalysis features for content-adaptive JPEG steganography. *IEEE Transactions on Information Forensics and Security*, 11(8), 1736–1746.

Fridrich, J. and Kodovskỳ, J. (2012). Rich models for steganalysis of digital images. *IEEE Transactions on Information Forensics and Security*, 7(3), 868–882.

Fuji-Tsang, C. and Fridrich, J.J. (2018). Steganalyzing images of arbitrary size with CNNs. *Electronic Imaging*, 121(1)–121(8).

Giboulot, Q., Cogranne, R., Borghys, D., Bas, P. (2020). Effects and solutions of cover-source mismatch in image steganalysis. *Signal Processing: Image Communication*, 86, 115888.

Goljan, M. and Fridrich, J. (2015). CFA-aware features for steganalysis of color images. *Media Watermarking, Security, and Forensics 2015 – Proceedings of SPIE*, Alattar, A.M., Memon, N.D., Heitzenrater, C.D. (eds). SPIE Press, Bellingham.

Goljan, M., Fridrich, J., Cogranne, R. (2014). Rich model for steganalysis of color images. *International Workshop on Information Forensics and Security*, Atlanta, 185–190.

He, K., Zhang, X., Ren, S., Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. *European Conference on Computer Vision*, Zurich, 346–361.

He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition*, Las Vegas, 770–778.

Hinton, G.E. and Salakhutdinov, R.R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 504–507.

Holub, V. and Fridrich, J. (2015). Low-complexity features for JPEG steganalysis using undecimated DCT. *IEEE Transactions on Information Forensics and Security*, 10(2), 219–228.

Ker, A.D. and Pevný, T. (2012). Batch steganography in the real world. *Multimedia and Security*, New York, 1–10.

Ker, A.D. and Pevný, T. (2014). The steganographer is the outlier: Realistic large-scale steganalysis. *IEEE Transactions on Information Forensics and Security*, 9(9), 1424–1435.

Kodovskỳ, J., Fridrich, J., Holub, V. (2012). Ensemble classifiers for steganalysis of digital media. *IEEE Transactions on Information Forensics and Security*, 7(2), 432–444.

Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems 25*, Lake Tahoe, 1097–1105.

LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.

Lehmann, E. and Romano, J. (2005). *Testing Statistical Hypotheses*, 2nd edition, Springer, Berlin/Heidelberg.

Pevný, T. and Ker, A.D. (2015). Towards dependable steganalysis. In *Media Watermarking, Security and Forensics 2015 – Proceedings of SPIE*, Alattar, A.M., Memon, A.D. and Heitzenrater, C.D. (eds). SPIE Press, Bellingham.

Pevný, T. and Nikolaev, I. (2015). Optimizing pooling function for pooled steganalysis. *International Workshop on Information Forensics and Security*, Rome, 1–6.

Pevný, T., Fridrich, J., Ker, A.D. (2009). From blind to quantitative steganalysis. *IS&T Electronic Imaging Symposium*, San Jose.

Pevný, T., Bas, P., Fridrich, J. (2010). Steganalysis by subtractive pixel adjacency matrix. *IEEE Transactions on Information Forensics and Security*, 5(2), 215–224.

Qian, Y., Dong, J., Wang, W., Tan, T. (2015). Deep learning for steganalysis via convolutional neural networks. *IS&T Electronic Imaging Symposium*, San Francisco.

Sedighi, V. and Fridrich, J. (2015). Effect of imprecise knowledge of the selection channel on steganalysis. *Workshop on Information Hiding and Multimedia Security*, Portland, 33–42.

Sedighi, V., Cogranne, R., Fridrich, J. (2016a). Content-adaptive steganography by minimizing statistical detectability. *IEEE Transactions on Information Forensics and Security*, 11(2), 221–234.

Sedighi, V., Fridrich, J.J., Cogranne, R. (2016b). Toss that BOSSbase, Alice! *IS&T International Symposium on Electronic Imaging*, San Francisco.

Sharma, G. and Bala, R. (2017). *Digital Color Imaging Handbook*. CRC Press, Boca Raton.

Simmons, G. (1984). The prisoners' problem and the subliminal channel. In *Proceedings of CRYPTO '83*, Chaum, D. (ed.). Plenum Press, New York.

Song, X., Liu, F., Yang, C., Luo, X., Zhang, Y. (2015). Steganalysis of adaptive JPEG steganography using 2D gabor filters. *Workshop on Information Hiding and Multimedia Security*, Portland, 15–23.

Taburet, T., Filstroff, L., Bas, P., Sawaya, W. (2018). An empirical study of steganography and steganalysis of color images in the JPEG domain. *International Workshop on Digital Forensics and Watermarking*, Jeju.

Tomàš, P. and Andrew D.K. (2015). Towards dependable steganalysis. In *Media Watermarking, Security, and Forensics – Proceedings of SPIE*, Alattar, A.M., Memon, N.D., Heitzenrater, C.D. (eds), SPIE Press, Bellingham.

Westfeld, A. (2001). F5 – A steganographic algorithm. *Lecture Notes in Computer Science*, 2137, 289–302.

Westfeld, A. and Pfitzmann, A. (1999). Attacks on steganographic systems. *International Workshop on Information Hiding*, Dresden, 61–76.

Xu, G. (2017). Deep convolutional neural network to detect J-UNIWARD. *Workshop on Information Hiding and Multimedia Security*, Philadelphia, 67–73.

Xu, G., Wu, H.Z., Shi, Y.Q. (2016). Structural design of convolutional neural networks for steganalysis. *IEEE Signal Processing Letters*, 23(5), 708–712.

Ye, J., Ni, J., Yi, Y. (2017). Deep learning hierarchical representations for image steganalysis. *IEEE Transactions on Information Forensics and Security*, 12(11), 2545–2557.

Yedroudj, M., Chaumont, M., Comby, F. (2018a). How to augment a small learning set for improving the performances of a CNN-based steganalyzer? *International Symposium on Electronic Imaging*, Burlingame, 7.

Yedroudj, M., Comby, F., Chaumont, M. (2018b). Yedroudj-Net: An efficient CNN for spatial steganalysis. *International Conference on Acoustics, Speech and Signal Processing*, Calgary, 2092–2096.

Yousfi, Y., Butora, J., Fridrich, J., Giboulot, Q. (2019). Breaking Alaska: Color separation for steganalysis in JPEG domain. *Workshop on Information Hiding and Multimedia Security*, Paris, 138–149.

Zakaria, A., Chaumont, M., Subsol, G. (2019). Pooled steganalysis in JPEG: How to deal with the spreading strategy? *International Workshop on Information Forensics and Security*, Delft, 1–6.

Zeng, J., Tan, S., Li, B., Huang, J. (2017). Pre-training via fitting deep neural network to rich-model features extraction procedure and its effect on deep learning for steganalysis. *IS&T Symposium on Electronic Imaging*, Burlingame, 6.

Zeng, J., Tan, S., Li, B., Huang, J. (2018). Large-scale JPEG image steganalysis using hybrid deep-learning framework. *IEEE Transactions on Information Forensics and Security*, 13(5), 1200–1214.

Zhang, R., Zhu, F., Liu, J., Liu, G. (2020). Depth-wise separable convolutions and multi-level pooling for an efficient spatial CNN-based steganalysis. *IEEE Transactions on Information Forensics and Security*, 15, 1138–1150.

List of Authors

David ALLEYSSON
LPNC
Grenoble Alpes University
Savoie Mont Blanc University
CNRS
Grenoble
France

Laurent AMSALEG
IRISA
University of Rennes
Inria
CNRS
France

Gildas AVOINE
INSA Rennes
University of Rennes
IRISA
CNRS
France

Yannis AVRITHIS
IRISA
University of Rennes
Inria
CNRS
France

Quentin BAMMEY
Centre Borelli
ENS Paris-Saclay
University of Paris-Saclay
CNRS
Gif-sur-Yvette
France

Patrick BAS
CRIStAL
CNRS
University of Lille
France

Sébastien BEUGNON
LIRMM
Université de Montpellier
CNRS
France

Philippe CARRÉ
XLIM
CNRS
University of Poitiers
France

Multimedia Security 1: Authentication and Data Hiding,
First Edition. William Puech.

Marc CHAUMONT
LIRMM
Université de Montpellier
CNRS
University of Nîmes
France

Rémi COGRANNE
LIST3N
University of Technology of Troyes
France

Miguel COLOM
Centre Borelli
ENS Paris-Saclay
University of Paris-Saclay
CNRS
Gif-sur-Yvette
France

Thibaud EHRET
Centre Borelli
ENS Paris-Saclay
University of Paris-Saclay
CNRS
Gif-sur-Yvette
France

Teddy FURON
IRISA
University of Rennes
Inria
CNRS
France

Philippe GABORIT
XLIM
CNRS
University of Limoges
France

Marina GARDELLA
Centre Borelli
ENS Paris-Saclay
University of Paris-Saclay
CNRS
Gif-sur-Yvette
France

Rafael GROMPONE
Centre Borelli
ENS Paris-Saclay
University of Paris-Saclay
CNRS
Gif-sur-Yvette
France

Vincent ITIER
CRIStAL
University of Lille
CNRS
IMT Lille Douai
France

Pascal LEFÈVRE
XLIM
CNRS
University of Poitiers
France

Jean-Michel MOREL
Centre Borelli
ENS Paris-Saclay
University of Paris-Saclay
CNRS
Gif-sur-Yvette
France

Tina NIKOUKHAH
Centre Borelli
ENS Paris-Saclay
University of Paris-Saclay
CNRS
Gif-sur-Yvette
France

Denis PERRAUD
Technical and Scientific Police
Central Directorate of the Judicial Police
Lyon
France

William PUECH
LIRMM
Université de Montpellier
CNRS
France

Cédric RICHARD
CNRS GdR ISIS
Côte d'Azur Observatory
University of Côte d'Azur
Nice
France

Hanwei ZHANG
IRISA
University of Rennes
Inria
CNRS
France

Index

Multimedia Security 1: Authentication and Data Hiding,
First Edition. William Puech.

E

F, G

H, I

J, L

M, N

Q, S

T, V, W

WILEY END USER LICENSE AGREEMENT

Go to www.wiley.com/go/eula to access Wiley’s ebook EULA.

Today, more than 80% of the data transmitted over networks and archived on our computers, tablets, cell phones or clouds is multimedia data – images, videos, audio, 3D data. The applications of this data range from video games to healthcare, and include computer-aided design, video surveillance and biometrics.

It is becoming increasingly urgent to secure this data, not only during transmission and archiving, but also during its retrieval and use. Indeed, in today's "all-digital" world, it is becoming ever-easier to copy data, view it unrightfully, steal it or falsify it.

Multimedia Security 1 analyzes the issues of the authentication of multimedia data, code and the embedding of hidden data, both from the point of view of defense and attack. Regarding the embedding of hidden data, it also covers invisibility, color, tracing and 3D data, as well as the detection of hidden messages in an image by steganalysis.

William Puech is Professor of Computer Science at Université de Montpellier, France. His research focuses on image processing and multimedia security in particular, from its theories to its applications.

www.iste.co.uk

WILEY

9 781789 450262