A FEATURE-BASED FINGERPRINTING SCHEME ROBUST TO DESYNCHRONIZATION

BY

GUILLAUME GIGAUD

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2010

Urbana, Illinois

Adviser:

Professor Pierre Moulin

Abstract

Traitor-tracing (aka fingerprinting) has received much attention as a possible solution for protecting media copyrights. However, current schemes for image and video fingerprinting lack robustness against geometric attacks. We propose a novel semi-blind fingerprinting scheme that can cope with such attacks. The scheme improves a state-of-the-art high-rate fingerprinting code that can resist tens of colluders and Gaussian noise but has no resistance against geometric attacks.

Our scheme uses compressed SURF (Speeded-Up Robust Features) image features as side information in order to estimate and invert any geometric attack in a given class. We consider simple linear attacks (affine transforms), and more complex ones (homography and image warping). Our Estimation-Elimination algorithm estimates the attack parameters by matching image features and eliminating iteratively suspected outliers. We also compare this method to an adapted version of RANSAC (Random Sample Consensus).

The fingerprints are embedded securely and invisibly using Spread Transform Dither Modulation (STDM) applied to the intermediate level of a Laplace decomposition of the image. The fingerprints are robust against common attacks such as averaging, interleaving, addition of Gaussian noise, JPEG compression (with quality factor $Q = 45$), cropping (50% of the image area), affine transforms, homography and image warping.

To Amélie

Acknowledgments

I would like to thank first my adviser, Professor Pierre Moulin, for the trust he put in me and his constant pool of ideas that guided me throughout my research. He has always reviewed and carefully corrected my works – including this thesis – with much rigor, in a constant search for perfection. He introduced me to the world of academic research, and showed me that this activity is as fascinating as it can be challenging.

My girlfriend Amélie was also a great source of inspiration; she always tried hard to understand my work and advised me wisely on decisions I had to take. I thank her for bringing happiness into my life and giving me an excellent reason to go back to my country.

I would also like to thank the friends I met in Urbana-Champaign, the students of my research group, and my roommates for making my stay so pleasant and beneficial. Particularly, I had a lot of fun and many interesting discussions about our respective research with Bruno.

Finally, my parents, sister, family and friends in France were all of great support while I was abroad, often calling me and reminding me that I should go back soon.

This work has been possible thanks to the financial support of the National Science Foundation under grant CCF 07-29061.

Previously published articles are reprinted with permission from:

- G. Gigaud and P. Moulin, "Traitor-tracing aided by compressed SURF image features," CISS 2010, Princeton, NJ ©2010 IEEE.

Table of Contents

Chapter 1

Introduction

The multimedia industry has never been so important worldwide: 2.5 millions jobs are related to the film and television industry in the US, directly or indirectly, according to the Motion Picture Association of America [1]. Moreover, this industry brings more than $80 billion annually to the US economy [2].

On the other side, copyright infringement has also never been so present: from in-theater camcording and illegal redistribution networks to CD/DVD illegal copies and peer-to-peer, the multimedia industry is attacked at every level [3]. The loss for the movie industry due to piracy in 2005 was estimated to be more than $18 billion worldwide, $6.1 billion in the US and $2.7 billion in China [4]. Similarly, the cost of piracy was estimated to be around $1.5 billion and 10,000 jobs in France in 2008 [5]. According to several reports, this loss is mainly imputable to recordings of movies in theaters, leading to illegal DVDs and files available for downloading. Moreover, the majority of awards-nominated movies can be downloaded illegally before they are officially released due to leaks in the rating process.

One of the main tools to prevent such copyright enfringements is traitor-tracing (aka fingerprinting or forensic watermarking), among other techniques like watermarking and robust hashing. The idea is to insert a unique mark in a multimedia content, which contains information about the user or the date, time, and location of broadcast. This mark, also called a fingerprint, is intended to be hard to remove in order to prevent illegal redistribution. The embedding process takes advantage of the human visual or auditory system to make the mark invisible. However, colluders (traitors) may process their individual copies and create a pirated copy in order to remove traces of their fingerprints, often accepting to alter the quality of the multimedia content, in exchange for lower probability of detection. When a pirated copy is found, extracting its fingerprint enables the distributor to trace at least

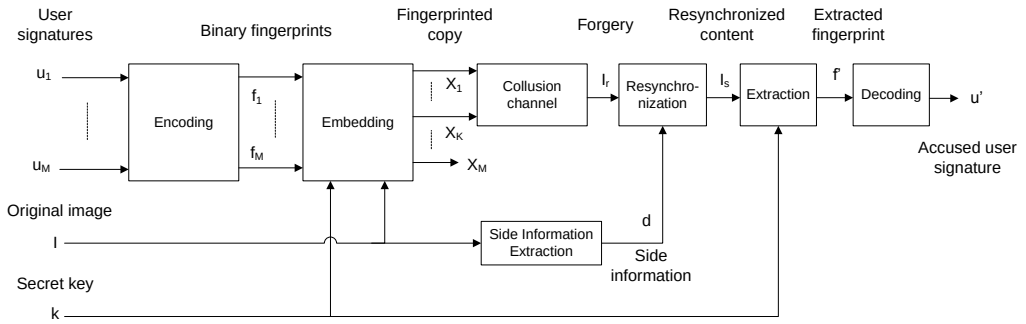


Figure 1.1: Fingerprinting scheme with K colluders and M users.

one of the colluders.

Such techniques have been used for other applications for centuries. For instance, in the 17th century, some of the least significant digits of logarithm tables were intentionally modified in order to identify users who tried to illegally reproduce the tables [6]. More recently, the use of steganography in printers enables to know where and when a document was printed and which model of printer was used [7]. In the movie industry, some motion picture distributors inserted visible dots in projected movies, identifying the cinema and the date of projection [8].

Nowadays, fingerprinting is used widely in several applications: digital cinema to prevent recordings in theaters, pre-release copies to prevent leaks, online pay-per-view, etc. Thereby, the use of traitor-tracing has helped dismantle several piracy networks, such as the 2005 arrest of William Sprague who stole several major motion picture copies prior to their theatrical or DVD release [9], or the 2008 apprehension of Robert Henderson who made counterfeit DVDs from recordings in a movie theater [10].

The global process of multimedia traitor-tracing is diagrammed in Fig. 1.1. The encoder encodes each user's binary bitstream and a secret key into a fingerprint that is designed to be robust against collusion. Each fingerprint is then inserted into a copy of the original content. The marked copies are then distributed to the corresponding users. Some of them might collude and process their copies to create a forgery. The colluders are assumed to know the entire fingerprinting scheme, except for the secret key. They can use all kinds of operations provided the content is not too much altered: addition of white noise, compression, averaging or interleaving of several copies, geometric attacks, etc. Once the forgery is retrieved by the distributor, it is first

resynchronized using side information about the original content. Then, the fingerprint can be extracted from the resynchronized content and decoded in order to find one of the colluders.

Several efficient traitor-tracing codes have been presented recently. For example, Jourdas designed two codes, one high-rate and random-like [11, 12] and another one low-rate with high minimimum distance [13]. The first one is short, can accomodate millions of users and can deal with tens of colluders. He and Wu proposed two joint coding-embedding schemes. The first one, called GRACE (Group-Based Joint Coding and Embedding), assumes that users are gathered in social groups inside which they are more likely to collude. Orthogonal fingerprints are used inside each group [14]. The second scheme uses a very long code based on Reed-Solomon which can accomodate millions of users [15]. Trappe et al. introduced Anti-Collusion Codes (ACC) [16], which are applications of Superimposed Codes (SIC) [17] to traitor-tracing. Those codes have the property that any composition of K fingerprints is unique, and thus are efficient under the Boneh and Shaw marking assumption [18]. More recently and similarly, Korzhik et al. designed random Superimposed Codes (SIC) [19] which have been shown to outperform ACC under a distortion assumption rather than a marking assumption.

Most of these codes are efficient against additive white noise, averaging of several copies and some non-linear operations (interleaving, median attacks, etc). However, none of them was designed to be robust against desynchronization. For instance, while Jourdas' code [11] is near-capacity achieving under averaging plus Gaussian noise attacks, the code fails against a rotation by just 1° or a scaling of factor 1.1. Our goal is to enhance such schemes with a layer that protects the fingerprint against such geometric attacks. Jourdas' code [11] will be used as a proof of concept.

Geometric attacks have only been studied in the restricted application of watermarking. While traitor-tracing is a multi-user version of watermarking and presents much greater challenges due to the efficiency of collusion attacks, the design of geometrically resilient watermarking schemes is very relevant to our study. Four different approaches have been studied to create geometrically resilient watermarks.

The first approach consists in performing an *exhaustive search* for the attack parameters inside a given class of attacks. The parameters chosen are

those that maximize a measure of confidence (*e.g.* likelihood score) for the decoder. This algorithm is computationally expensive as it usally requires full decoding for each set of parameters being tried.

The second idea is still based on exhaustive search for the attack parameters but avoids using the decoder for each iteration [20]. A fixed *synchronization mark* is inserted in the watermarked copy and for each set of parameters being tried, the synchronization mark is used to determine if the content is well synchronized. This method is often less expensive than running the decoder multiple times, but synchronization marks are hard to design and cause a distortion overhead.

The third approach consists in inserting the watermark into an *invariant domain* where the attacks considered cannot affect the fingerprint [21]. Theoretically, the watermark can then be extracted and decoded correctly even if the content is desynchronized using one of those attacks. However, invariant domains exist for a very limited range of attacks and do not take advantage of the visual or audible human system.

These three approaches have been mainly used in a *blind* context where the decoder does not have information about the original content. In contrast, *non-blind* systems use the original content on the decoder side, and this facilitates resynchronization. However, the presence of the original content as side information implies some security and storage issues.

The fourth idea, *registration*, is an intermediate approach, and can be advantageously used in a *semi-blind* context. Here, limited information about the original content is available to the decoder. The side information is either a hash [22] — a short description of the content — or a collection of local feature descriptions [23]. In both cases, one tries to register the original content and the pirated copy in order to estimate and invert the attack. We decided to follow this last approach and apply it to fingerprinting because side information can improve dramatically the robustness of the fingerprint. However, the size and security of the side information are also important constraints for semi-blind systems.

We chose to insert the fingerprint in the medium frequencies of the multimedia content. This allows us to avoid the high frequencies, which can be entirely erased in the case of compression, and low frequencies, which are very sensible. Moreover, a fundamental concept for both blind and semi-blind schemes is the use of host-signal interference rejection methods such as

Spread Transform Dither Modulation [24, 6] as opposed to spread-spectrum modulation methods which are more suitable for non-blind schemes. Conventional wisdom is that quantization-based methods are brittle against geometric attacks; however, we show this is not the case when the decoder is provided with appropriate side information.

A natural choice for the side information in the case of images is the SURF (Speeded-Up Robust Features) [25] features, an enhanced version of the SIFT (Scale Invariant Feature Transform) [26] features. The SURF features are robust to a wide range of geometric attacks and to additive noise. Furthermore, a suitable compression of the SURF features is convenient for storage and transmission and retains the discriminative properties of these features.

The main steps of our traitor-tracing algorithm are the following: (i) match the features of the forgery and those of the original image, (ii) estimate the parameters of a given class of attacks, (iii) invert the attack, and (iv) extract the fingerprint.

This thesis is organized as follows. Chapter 2 presents the encoder and decoder. Chapter 3 describes our STDM-Laplace embedding/extracting scheme. Chapter 4 presents our choise of image features and our compression method. Chapter 5 explains how we used this side information in order to resynchronize the received image. Chapter 6 presents some experimental results for different classes of attacks.

Chapter 2

High-Rate Encoder and Decoder

2.1 Encoder

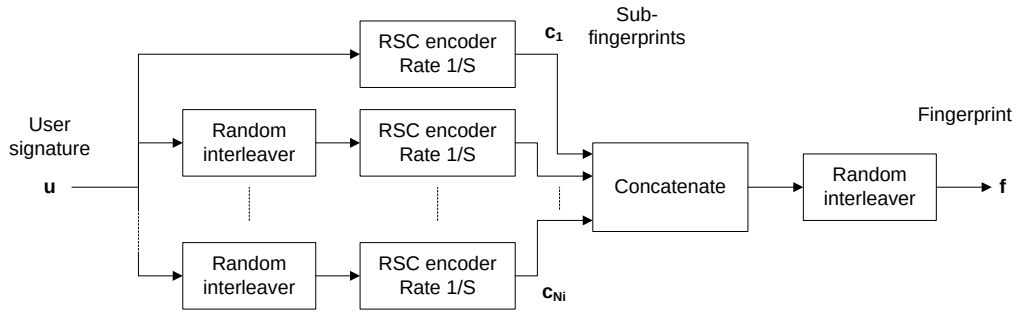


Figure 2.1: Encoder scheme for one user.

Each user is identified by a binary bitstream $\mathbf{u}$ of length n, so the code can accomodate $M = 2^n$ users. The binary sequences $\{\mathbf{u}_i\}_{1 \leq i \leq M}$ are input to the encoder of [11], which, analogously to a turbo code, is composed of N_i RSC (Recursive Systematic Convolutional) encoders preceeded by random interleavers. The rate of each RSC encoder is $1/S$ and the generator polyomials coefficients are randomly drawn from a Bernoulli distribution of parameter p. The memory of the encoders is denoted by M_e. As shown in Fig. 2.1, the outputs of the RSC encoders are then concatenated to create binary fingerprints $\{\mathbf{f}_i\}_{1 \leq i \leq M}$ of length $N = (n + M_e) \times S \times (N_i + 1)$.

In order to have symmetric fingerprints for the embedding part, we switch the domain of $\{\mathbf{f}_i\}_{1 \leq i \leq M}$ from $\{0, 1\}^N$ to $\{-1, 1\}^N$. In the original implementation of [11], an orthogonal code followed the RSC encoders in order to map the fingerprint to a real-valued mark. This is not needed here because our embedding scheme requires only a binary sequence. However, we added a random interleaver that shuffles the fingerprint. This precaution prevents the erasure of clusters of the fingerprint if part of the image is cropped.

For illustration, the parameters of the code we used are the following: $n = 25$, $N_i = 4$, $S = 7$, $p = 0.7$ and $M_e = 3$. With those values, the maximum number of users is $M = 33,554,432$, the length of the code is $N = 784$, and the total rate of the code is $R = \frac{n}{N} = 0.032$.

2.2 Decoder

The fingerprint $\mathbf{f}'$ extracted from the image I_s (see Fig. 1.1) is often referred to as the *forgery* and is in general real-valued. The decoder used in [11] seeks the fingerprint that has the highest correlation with the forgery.

The forgery is first deinterleaved and then split into N_i segments of equal length, each input to a list Viterbi decoder. The outputs of the decoders (after deinterleaving) are lists of D suspect users. The final accused user is determined by the highest correlation between the forgery and the suspect fingerprints. There are at most $D \times N_i$ such fingerprints.

Chapter 3

Fingerprint Embedding and Extraction

3.1 STDM-Laplace embedding

The fingerprint is embedded in the medium frequencies of the image, because a slight modification of the low frequencies would imply a significant visual alteration, and the high frequencies can be strongly attacked without any major visual modification on the image. We chose the Laplacian pyramid of the image as the transform domain to embed the fingerprint.

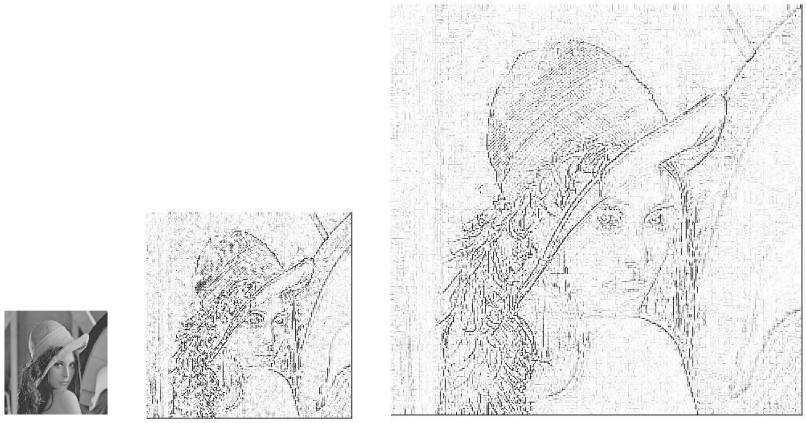
The embedding technique is based on STDM (Spread Transform Dither Modulation) [24, 6]. It spreads the fingerprint bits over a large set of values in the transform domain.

Given a host image I, we first compute its 3-level Laplacian pyramid which results in three frequency images $\{LP_1(I),\ LP_2(I),\ LP_3(I)\}$, respectively the fine, intermediate and coarse descriptions of the image I (see Fig. 3.1). We use STDM on the intermediate description $LP_2(I)$ of the pyramid.

Given the length of the fingerprints to embed in the intermediate description $LP_2(I)$, we consider N non-overlapping blocks of equal size. Each fingerprint bit will then be embedded in one of those blocks. The size of those blocks is chosen in order to maximize the embedding area and we do not embed bits on the border of the image, which may be cropped later. For illustration, in the case where the fingerprint length is $N = 784$ and the size of the intermediate description $LP_2(I)$ is 256×256, the size of each block is set to 9×9. Those N blocks are then transformed into column vectors, and we denote this set by $\{\mathbf{b}^j\}_{1 \leq j \leq N}$.

We project each block $\mathbf{b}^j$ onto a random direction $\mathbf{v}^j$ (generated using the secret key), identical for all users:

$$p^j = (\mathbf{v}^j)^{\mathrm{T}} \mathbf{b}^j, \quad 1 \leq j \leq N. \tag{3.1}$$



(a) (b) (c)

Figure 3.1: Laplacian pyramid of Lena image: (a) Coarse description (size 128×128); (b) intermediate description (size 256×256); and (c) fine description (size 512×512). Contrast has been enhanced for (b) and (c).

Given the fingerprint $\mathbf{f}_i$ of the ith user, the projection p^j — which is a scalar — is quantized using the uniform scalar quantizers $Q_{-1}(\cdot)$ or $Q_1(\cdot)$, respectively, when the corresponding fingerprinting bit f_{ij} is -1 or 1. The lattices used for the quantizers $Q_{-1}(\cdot)$ and $Q_1(\cdot)$ are respectively $\mathcal{L}_{-1} = \{\delta(-\frac{1}{4}+k),\, k \in \mathbb{Z}\}$ and $\mathcal{L}_1 = \{\delta(\frac{1}{4}+k),\, k \in \mathbb{Z}\}$. The two possible quantizers are shown on Fig. 3.2. The quantized value of the projection of $\mathbf{b}^j$ is denoted by q_i^j:

$$q_i^j = Q_{f_{ij}}(p^j), \quad 1 \leq j \leq N,\, 1 \leq i \leq M. \tag{3.2}$$

The fingerprint bit f_{ij} is embedded in the block $\mathbf{b}^j$ by replacing the projection p^j by its quantized value q_i^j. The jth block of the ith fingerprinted copy is denoted by $\mathbf{b}_i^j$:

$$\mathbf{b}_i^j = \mathbf{b}^j + \mathbf{v}^j(q_i^j - p^j), \quad 1 \leq j \leq N,\, 1 \leq i \leq M. \tag{3.3}$$

We then convert the column vectors $\{\mathbf{b}_i^j\}_{1 \leq j \leq N}$ into square blocks and concatenate them to form the new second detail description $LP_2(X_i)$. We finally form the fingerprinted image X_i for user i from the Laplacian pyramid composed of the original fine and coarse descriptions $LP_3(I)$ and $LP_1(I)$ and the fingerprinted intermediate description $LP_2(X_i)$.

This spreading technique, unlike a simple embedding of each bit in one pixel of $LP_2(I)$, is more secure because the colluders cannot determine easily

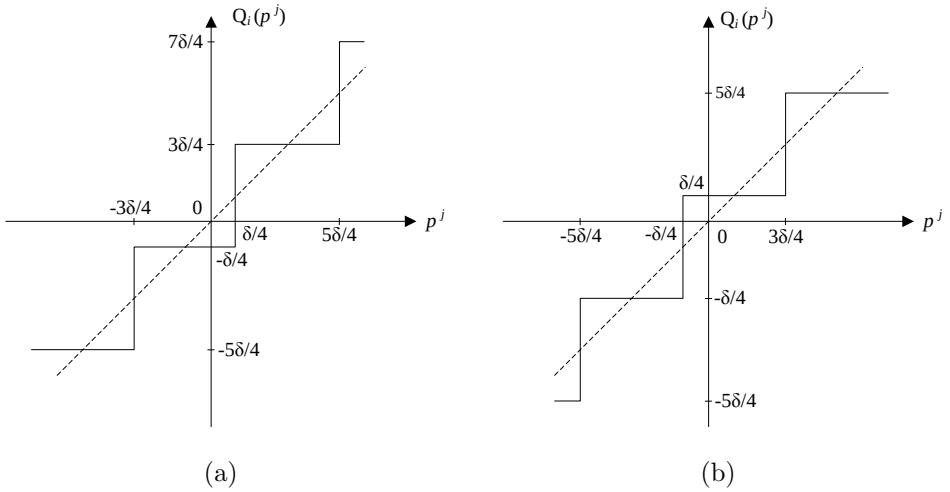


Figure 3.2: The two different quantizers: (a) $Q_{-1}(\cdot)$, and (b) $Q_1(\cdot)$.

the location of the fingerprint. Moreover, the visual distortion due to the embedding can be directly controlled via the quantization step δ.

3.2 Fingerprint extraction

The fingerprint extraction is straightforward. Given the received image I_s, we compute its 3-level Laplacian pyramid $\{LP_1(I_s),\ LP_2(I_s),\ LP_3(I_s)\}$ and consider the N blocks $\{\mathbf{b}'_j\}_{1\leq j\leq N}$ of the intermediate description $LP_2(I_s)$. Each block $\mathbf{b}'_j$ is projected onto the random direction $\mathbf{v}^j$, similarly to (3.1):

$$p'_j = (\mathbf{v}^j)^{\mathrm{T}}\mathbf{b}'_j, \quad 1 \leq j \leq N. \tag{3.4}$$

The noisy fingerprint $\mathbf{f}' = \{f'_j\}_{1\leq j\leq N}$ is extracted in two different ways depending on the kind of decoding performed afterwards: hard-bit ($f' \in \{-1,1\}^N$) or soft-bit ($f' \in [-1,1]^N$), as shown in Fig. 3.3.

In the hard-bit case, we look for the minimum distance between the projected value p'_j and the points of the two quantizer lattices $\mathcal{L}_{-1}$ and $\mathcal{L}_1$. The extracted bit f'_j is the state of the quantizer for which the minimum distance was found:

$$f'_j = \underset{m\in\{-1,1\}}{\operatorname{argmin}} \min_k \left| \left(k - \frac{m}{4}\right)\delta - p'_j \right|, \quad 1 \leq j \leq N. \tag{3.5}$$

In order to handle image cropping, we set the components of $\mathbf{f}'$ to 0

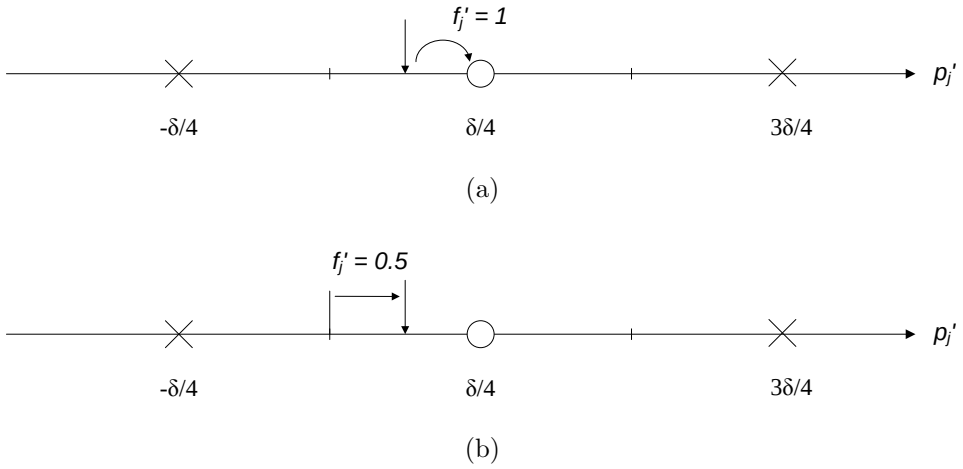


Figure 3.3: An example of extraction: (a) in the hard-bit case, and (b) in the soft-bit case. Crosses and circles represent respectively the elements of the lattices $\mathcal{L}_{-1}$ and $\mathcal{L}_1$.

whenever more than half of the corresponding block of the image is erased.

In the soft-bit case, the value of $-1 \leq f_j' \leq 1$ is given by the following formula:

$$f_j' = 1 - \left| \frac{4}{\delta} \left((p_j' + \frac{\delta}{4}) \bmod \delta \right) - 2 \right|, \quad 1 \leq j \leq N, \tag{3.6}$$

where $\left|f_j'\right| = 1$ indicates a reliable bit: p_j' is equal to an element of $\mathcal{L}_{-1}$ or $\mathcal{L}_1$.

Cropping is handled automatically in the soft-bit case: an extracted value f_j' close to 0 indicates that the corresponding block of pixels in the image I_s is not reliable, and thus the extracted bit is treated as an erasure by the decoder. If a region of I_n is erased (and thus set to a constant value), the extracted fingerprint components corresponding to this region will be zero (or close to zero if noise was added) and this region will not bring any information to the decoder. More generally, soft-decoding performs better because more information is conveyed by the extracted fingerprint, and the decoding algorithm is well adapted to soft values. The extraction is always performed in a soft way in our simulations, except as otherwise specified.

In order to quantify the distortion due to the embedding, we define the

average embedding power d:

$$d = \frac{1}{K} \sum_{k=1}^{K} ||LP_2(X_k) - LP_2(I)||_2^2 . \tag{3.7}$$

3.3 Large quantization step

The value of the quantization step δ is set to the "just-noticeable power." It means that we choose the maximum value with the constraint that the fingerprint remains invisible in the embedded images $\{X_i\}_i$. We chose the value $\delta = 60$ in our experiments.

We noticed during the implementation of STDM-Laplace that the embedding/extraction scheme we described can be simplified practically, for such value of the quantization step. Indeed, for all the images we tested, the empirical standard deviation of the projected blocks components $\{p_j\}_j$ is around 10, which is inferior to $\delta/4 = 15$, and thus the projection values $\{p_j\}_j$ are most of the time located in the first quantization bin of either $Q_{-1}(\cdot)$ or $Q_1(\cdot)$. Considering this observation, the quantization which takes place while embedding can be approximated to a simple replacement of the projected value by $\delta/4$ or $-\delta/4$, respectively, when the fingerprint bit considered is 1 or -1. Equation (3.3) becomes:

$$\mathbf{b}_i^j = \mathbf{b}^j + \mathbf{v}^j(\frac{\delta}{4} \cdot f_i^j - p^j), \quad 1 \leq j \leq N, 1 \leq i \leq M. \tag{3.8}$$

The extraction is then also simplified for both the hard-bit and soft-bit case. In the first case, the extracted bit f'_j is determined by the sign of the projected value p'_j:

$$f'_j = \text{sgn}(p'_j), \quad 1 \leq j \leq N. \tag{3.9}$$

For the soft-bit case, Equation (3.6) is approximated by the following formula to determine the extracted fingerprint bits:

$$f'_j = p'_j \cdot \frac{4}{\delta}, \quad 1 \leq j \leq N. \tag{3.10}$$

This approximation is exact – and thus $\left|f'_j\right| < 1$ – most of the time because p'_j is in the first quantization bin most of the time. Even if the components of the extracted fingerprint f' are no longer always in $[-1, 1]$ – because the noise

can statistically take any real value – we avoid with this simplified extraction method some quantization errors. Moreover, since the extracted fingerprint bits are now proportional to the projection values, the fingerprinting scheme is more robust against amplitude scaling and histogram equalization.

We used this approximation of STDM for all our simulations.

Chapter 4

Channel Model

The channel can be modeled in general as a combination of collusion attacks involving several copies, also called copy-merging attacks (averaging, interleaving, median attacks), non-desynchronizing attacks (addition of white noise, compression, histogram modification, cropping) and geometric attacks (rotation, geometric scaling, translation, image warping).

As diagrammed in Fig. 4.1, we decided for our experiments to use the following cascade of attacks: a copy-merging attack, addition of Gaussian noise, and another attack (cropping, JPEG compression, histogram equalization, affine transform, homography or image warping).

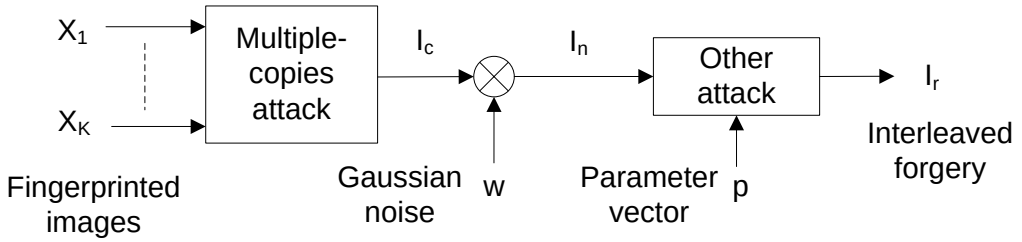


Figure 4.1: Channel model.

4.1 Non-desynchronizing attacks

4.1.1 Copy-merging attacks

Such attacks are performed by merging several fingerprinted copies in order to create a forgery where all the fingerprints are also mixed. We considered three different startegies: averaging, median and interleaving attack. For the averaging attack, each pixel value of the forgery I_c is the average of the pixel values of the fingerprinted copies $\{X_1, \cdots, X_K\}$ at the same location (l, m):

$$I_c(l,m) = \frac{1}{K}\sum_{k=1}^{K} X_k(l,m). \tag{4.1}$$

Similarly, the median attack assigns to each pixel value $I_c(l,m)$ the median of the pixel values $\{X_1(l,m), \cdots, X_K(l,m)\}$ of the fingerprinted copies.

In the case of interleaving, each colluder contributes a fraction of its pixels to the "interleaved forgery" I_c. Specifically, each pixel of I_c is given by:

$$I_c(l,m) = X_{k(l,m)}(l,m), \tag{4.2}$$

where $k(l,m)$ is chosen randomly and uniformly from $\{1, 2, \cdots, K\}$, independently for all pixel locations (l,m).

4.1.2 Additive Gaussian noise

The colluders add the white noise directly to the intermediate description of the Laplacian pyramid $LP_2(I_c)$. Each coefficient of the intermediate description $LP_2(I_n)$ is given by:

$$LP_2(I_n)(l,m) = LP_2(I_c)(l,m) + \lambda \cdot w(l,m), \tag{4.3}$$

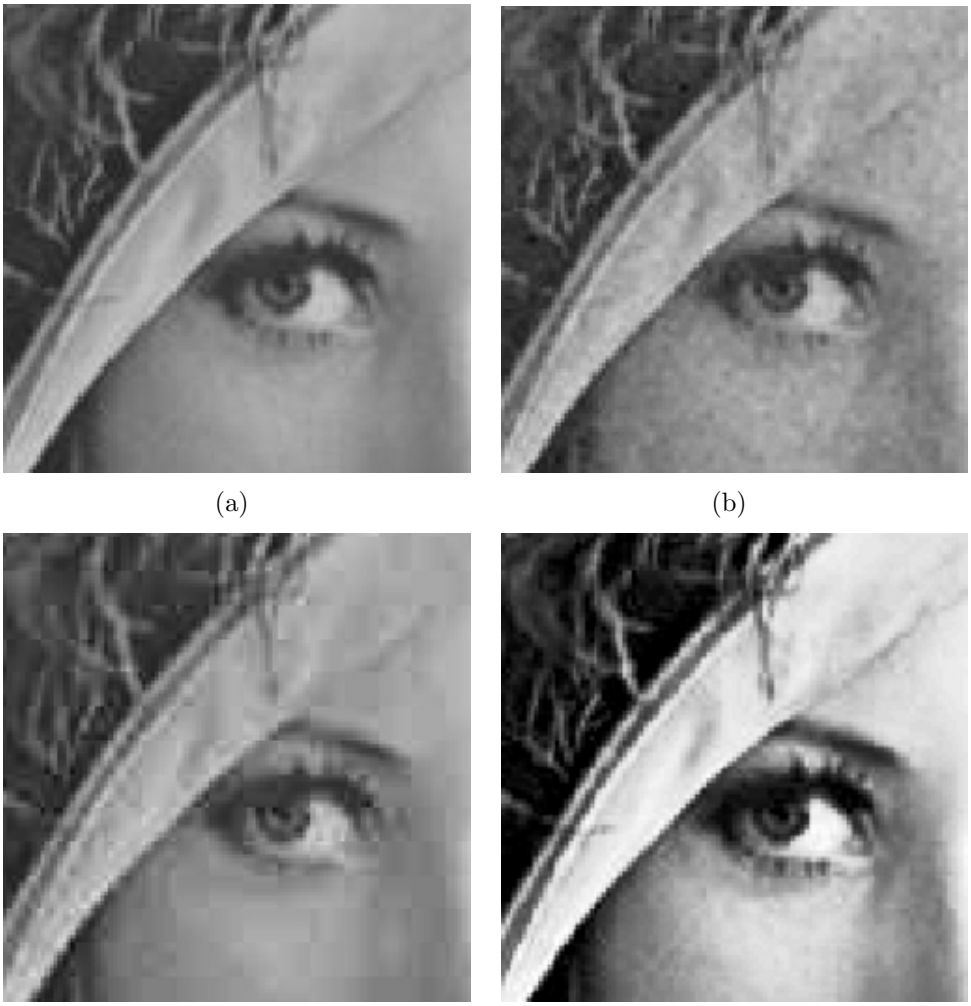
where $w(l,m)$ is drawn from the standard normal distribution $\mathcal{N}(0,1)$, independently for all pixel locations (l,m), and λ is the standard deviation of the Gaussian noise.

Instead of choosing a value for λ in our experiments, we rather set the value of the fingerprint-to-noise ratio $FNR = d/\lambda^2$, where d is the previously defined average fingerprint power.

An example of image corrupted with additive Gaussian noise is shown in Fig. 4.2(b).

4.1.3 Other attacks

We have studied additional attacks: JPEG compression, histogram equalization, and cropping. With JPEG compression, the image is first divided in blocks and the DCT coefficients of those blocks are quantized with respect to a quality factor. The smaller the quality factor, the more coarsely the co-



(a) (b)

(c) (d)

Figure 4.2: Examples of attacks on part of Lena image: (a) original image; (b) additive Gaussian noise with $FNR = 1/20$; (c) JPEG compression with quality factor 20; (d) histogram equalization.

efficients are quantized. Some preprocessing (subsampling of color channels) and postprocessing (entropy coding) operations are also performed. Fig. 4.2(c) shows an example of JPEG compression with quality factor 20. Histogram equalization modifies the intensity of the image in a nonlinear way. An example of histogram equalization is shown in Fig. 4.2(d). Cropping is performed by deleting entire regions of the image I_n and replacing them by black borders (see Fig. 4.3(a)).

4.2 Geometric attacks

4.2.1 Affine transform

Orthographic projection is a way of representing 3D objects in 2D, where each point is projected orthogonally to the projection plane. This model is an approximation of the process of taking a picture of an object, when the object depth is small compared to the distance of the object to the objective of the camera. The consequence of this model is that every plane in 3D is projected in 2D through an affine transform.

An affine transform is the composition of a rotation of angle θ, a translation by $(\Delta x, \Delta y)$ pixels, a scaling of factor c, a shearing of angle α, and an aspect ratio change where r is the new ratio of the height of the image to its width. The affine transform is then described by a set of six parameters $\mathbf{p} = (\theta, \Delta x, \Delta y, c, \alpha, r)$. Examples of affine transforms are shown in Fig. 4.3.

To perform the affine transform, the colluders create an image I_r where the value $I_r(x', y')$ of the pixel at every integer location (x', y') is related to the image I_n by the following formulas:

$$I_r(x', y') = I_n(x, y) \tag{4.4}$$

$$\begin{pmatrix} x \\ y \end{pmatrix} = A^{-1} \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{4.5}$$

$$A = \begin{pmatrix} c \cdot cos(\theta + \alpha) & -c \cdot sin(\theta + \alpha) \\ r \cdot c \cdot sin(\theta) & r \cdot c \cdot cos(\theta) \end{pmatrix} \tag{4.6}$$

where the image indices are defined with respect to Cartesian coordinates

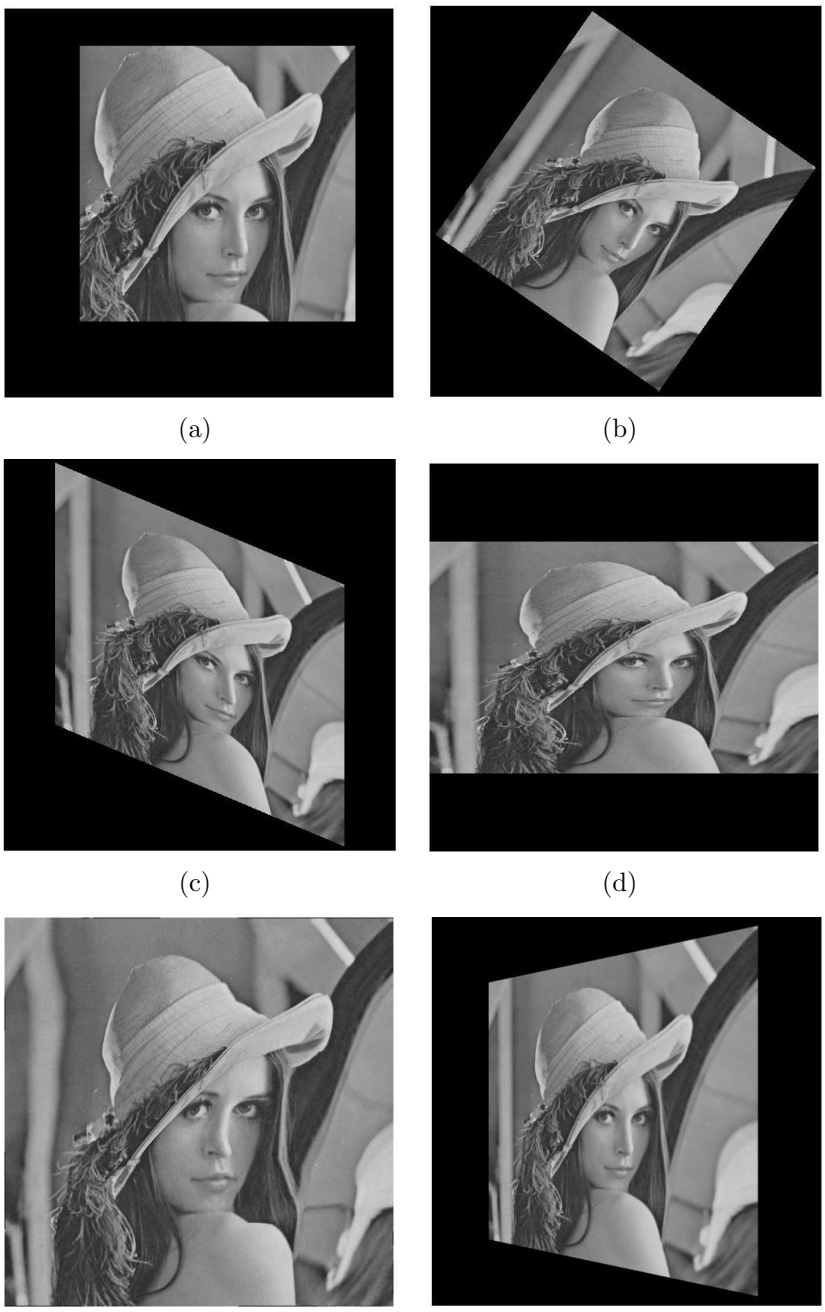


Figure 4.3: Examples of attacks on Lena image: (a) cropping of 50% of image area; (b) scaling of factor 0.7 and rotation by 35°; (c) scaling of factor 0.74 and shearing by 25°; (d) aspect ratio of 1:0.6; (e) image warping with a grid of size 5×5 and maximum shift of 30 pixels; (f) scaling of factor 0.6 and rolling by 30°.

where the origin is the center of the image. When x or y are not integers,

$I_n(x, y)$ is obtained with bilinear interpolation with the nearest neighbors of (x, y).

4.2.2 Homography

Perspective projection is a more sophisticated model than orthographic projection. Based on the pinhole camera model [27], perspective projection takes into account the focal length and geometry of the optical captors. The whole model contains eleven parameters but can be simplified to eight parameters when the object projected is a 2D plane (when a movie is camcorded for instance). The relation between the 3D plane and its projection is referred to as a homography.

The eight parameters of a homography are the six parameters of an affine transform, the angle of camera tilt ϕ, and the angle of camera roll ψ. A homography is thus described by a set of eight parameters $\mathbf{p} = (\theta, \Delta x, \Delta y, c, \alpha, r, \phi, \psi)$ (see Fig. 4.3(f)).

A homography is not an affine operation in Cartesian coordinates, but it becomes linear if we consider the transform in homogeneous coordinates. We denote by H the 3×3 homography matrix, which is defined up to a scale factor. The eight freedom degrees of H can be linked to the set of parameters p [27].

Similarly to affine transforms, homography is performed by creating an image I_r where the value $I_r(x', y')$ of the pixel at every integer location (x', y') is related to the image I_n by the following formulas:

$$I_r(x', y') = I_n\left(\frac{x}{z}, \frac{y}{z}\right) \tag{4.7}$$

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = H^{-1} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \tag{4.8}$$

where the image indices are defined with respect to Cartesian coordinates and the origin is the center of the image. When x/z or y/z are not integers, $I_n(x/z, y/z)$ is obtained with bilinear interpolation with the nearest neighbors of $(x/z, y/z)$.

4.2.3 Image warping

Unlike affine transforms and homography which are global operations, image warping is a local geometric attack parametrized by a large number of attack parameters.

The image is divided into a mesh, each vertex of which is randomly shifted. The regions of the mesh are then stretched according to the surrounding vertices. More specifically, the image I_n is triangulated using a regular grid of size $l \times l$. To avoid cropping, the borders are left intact. Each interior node of the mesh is shifted randomly and uniformly within the range $[-s_m, s_m]$. The set p of parameters is then characterized by the horizontal and vertical shifts of the interior nodes: $p = (s_1^h, s_1^v, \cdots, s_{(l-1)^2}^h, s_{(l-1)^2}^v)$. The total number of parameters is $2(l-1)^2$.

Given a triangle T in this mesh, with vertices coordinates (X_1, Y_1), (X_2, Y_2) and (X_3, Y_3), we denote by T' the resulting triangle after shifting the vertices, with vertices coordinates (X_1', Y_1'), (X_2', Y_2') and (X_3', Y_3'). The colluders iterate over all pixels locations $\{(x_i', y_i')\}_i$ which lie in T' and compute their barycentric coordinates $(\lambda_{1,i}, \lambda_{2,i}, \lambda_{3,i})$:

$$\begin{pmatrix} \lambda_{1,i} \\ \lambda_{2,i} \\ \lambda_{3,i} \end{pmatrix} = \begin{pmatrix} X_1' & X_2' & X_3' \\ Y_1' & Y_2' & Y_3' \\ 1 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix}. \tag{4.9}$$

Since the barycentric coordinates are invariant to linear shifts of the vertices, the colluders find the equivalent location (x_i, y_i) of (x_i', y_i') in the triangle T:

$$\begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} X_1 & X_2 & X_3 \\ Y_1 & Y_2 & Y_3 \end{pmatrix} \begin{pmatrix} \lambda_{1,i} \\ \lambda_{2,i} \\ \lambda_{3,i} \end{pmatrix}. \tag{4.10}$$

The value of $I_r(x_i', y_i')$ is then equal to $I_n(x_i, y_i)$. Similarly to an affine transform, if (x_i, y_i) are not integers, the colluders perform a bilinear interpolation with the nearest neighbors. Figure 4.3(e) shows an example of strong image warping.

Chapter 5

Features Computation and Compression

In order to resynchronize the received image, we compare its SURF features to those of the original image.

5.1 Image features

Image features are used in a wide variety of applications: to recognize specific instances of an object or a class of objects, to clusterize and segment an image and to register images. More specifically, image features are interest points (aka keypoints) in an image. They should be visually significant and robust to usual image transformations. For example, edges, corners and blob structures are often considered as interest points in an image.

Features are defined in two stages: first, a feature detector extracts interest points from the input image and stores their location, orientation and scale; second, a feature descriptor is associated with a neighborhood of each interest point. The descriptors are designed to be robust against basic affine transforms, and can then be used to match features, recognize objects and register images.

Detected feature points need to be *repeatable* — the same points are detected when the image is modified up to a certain limit — and *distinctive* — we need to be able to distinguish them from each other in order to avoid matching errors. There are trade-offs associated with both the localization and description of those keypoints.

A variety of feature detectors and descriptors exists, each of them with different properties. The most common and used ones are SIFT (Scale-Invariant Feature Transform) [26], and SURF (Speeded-Up Robust Features) [25], an enhanced version of SIFT. Both are robust and perform very well in image registration applications [28], but we selected SURF which is computationally

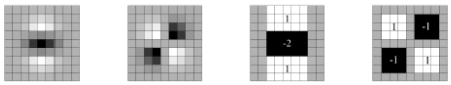


Figure 5.1: From left to right: the second-order derivatives of a Gaussian filter (cropped and discretized) in the vertical and diagonal directions respectively, and their respective approximations using box filters [25].

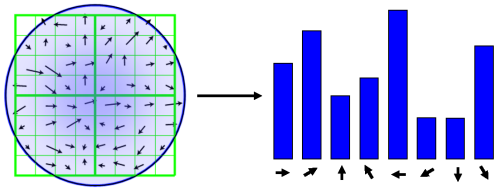


Figure 5.2: Computation of the weighted gradient orientation histogram in each feature neighborhood.

more efficient [29].

5.2 Feature computation

The SURF algorithm first detects scale-space extrema. This is done by approximating Laplacian-of-Gaussian using box filters and integral images (an exact implementation would be too computationally expensive). Examples of those approximations are shown in Fig. 5.1.

The algorithm then tries to enhance the precision of each extremum location by performing bilinear interpolation within the scale-space images. Moreover, extrema with low contrast or a strong edge response in one direction only are rejected.

The next step is called orientation assignment: a main orientation is computed in the neighborhood of each interest point in order to make the future descriptors robust to rotation. For each interest point, at the correct scale, the gradient magnitude and orientation are computed in an 8×8 neighborhood and a magnitude-weighted gradient orientation histogram is constructed (see Fig. 5.2). The orientation assigned to the interest point is the interpolated maximum of this histogram.

The result of those operations is a set of features (or interest points) described by four parameters: the coordinates (x, y) of the feature, the scale σ at which the extremum has been found, and the main orientation θ of the

feature.

The descriptor for each feature is computed in the following way: the neighborhood of each feature is rotated and rescaled (according to the assigned orientation and scale) to form a 20×20 pixels region. This region is partitioned into sixteen 5×5 cells. In each cell, the Haar wavelet gradients are computed in both directions (we denote by d_x and d_y respectively the horizontal and vertical gradients). For each 5×5 cell, the SURF descriptor consists of the sums of gradients and sums of absolute gradients: $\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|$. The descriptor for each feature is then a real-valued vector of size $16 \times 4 = 64$.

The number of extracted features N_f depends on the size and the content of the image but generally ranges from 500 to 2000. The ouput of the algorithm is a list of the feature locations and a list of 64-dimensional feature descriptors.

5.3 Feature compression

As an example, the SURF algorithm extracts 1493 features for the 512×512 grey-level Lena image, and thus the size of the feature descriptors (382 kB when each descriptor component is stored on 32 bits) is larger than the size of the image (40 kB in JPEG format, with quality factor 75). Therefore, having SURF features as side information is only an advantage if we can heavily compress them.

In order to compress efficiently the feature descriptors, we first study their correlation. Figure 5.3 shows the covariance matrix of the feature descriptors of a set of 44 images (from the "Miscellaneous" category of USC-SIPI image database). To remove the correlation between the descriptors, we compute the eigenvalues ($\sigma_1^2 > \cdots > \sigma_{64}^2$) and the corresponding eigenvectors ($\mathbf{e}_1, \cdots, e_{64}$) of the covariance matrix and multiply the matrix of descriptors D by the matrix of eigenvectors E:

$$D' = E^{\mathrm{T}} D, \tag{5.1}$$

where D' is the matrix of transformed descriptors.

Following this operation, the transformed descriptors $\{(d'_{i,1}, \cdots d'_{i,64}),\ 1 \leq$

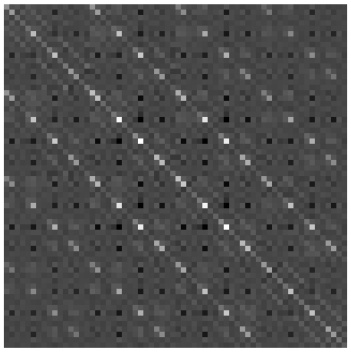


Figure 5.3: Covariance matrix of the SURF features of a set of 44 images (the white color represents highest values).

$i \leq N_f\}$ are statistically orthogonal and the first components of the descriptors are the ones with highest variance. We then assign to each dimension $i \in \{1, \cdots 64\}$ of the new descriptor a number of quantization bits B_i such that the number of quantization bins 2^{B_i} for the dimension i is almost proportional to the square root of the corresponding eigenvalue σ_i^2, with the constraint that the total number of bits for a descriptor is N_{bits}:

$$B_i = \left\lfloor \log_2(\sigma_i) + \frac{N_{bits} - \sum_{i=1}^{64} \log_2(\sigma_i)}{64} \right\rfloor, \quad 1 \leq i \leq 64. \tag{5.2}$$

Due to the floor operation, the total number of bits after this assignement is generally not N_{bits}. Hence, we add one bit to the first Δ_{bits} dimensions, where Δ_{bits} is defined as follows:

$$\Delta_{bits} = N_{bits} - \sum_{i=1}^{64} B_i. \tag{5.3}$$

We then quantize the transformed descriptors using a Max-Loyd quantizer with the given number of bits. The parameter N_{bits} is chosen by simulation to be as small as possible while resulting in a probability of error that is approximately the same as in the absence of quantization. We obtained $N_{bits} = 64$ bits, and the compression rate of the descriptors is thus $\frac{64}{64*32} = \frac{1}{32}$. The compressed SURF file size is only 12 kB. Table 5.1 shows the values of $\{B_i\}_{1 \leq j \leq 64}$ for $N_{bits} = 64$ bits.

One may object that the side information should also include the eigen-

Table 5.1: From top to bottom, and from left to right: number of bits $\{B_i\}_{1\leq j\leq 64}$ assigned to each dimension, from the highest eigenvalue to the lowest.

4	3	2	1	0	0	0	0
3	3	2	1	0	0	0	0
3	3	2	1	0	0	0	0
3	3	1	1	0	0	0	0
3	2	1	1	0	0	0	0
3	2	1	1	0	0	0	0
3	2	1	1	0	0	0	0
3	2	1	1	0	0	0	0

vectors of the covariance matrix and the quantization codebook and partition (in order to compress the SURF features of the original image I and the received image I_r in the same way). However, we noticed that these quantities are almost independent of the image used. We computed those three matrices over the same set of 44 images and used them to compress the descriptors of any image. The effect of using fixed transform and quantization on the resulting error probability is negligible.

A similar compression method for SURF descriptors was proposed by Chandrasekhar et al. [30], but the application of the features was more general, and they limited the compression rate to 1/16.

5.4 Features matching

Given the compressed SURF features from both the original image I and the received image I_r, we need to match them, prior to resynchronization. This is done identically to [25]. Specifically, for each feature of the original image I, the algorithm looks for the closest and second-closest features from the received image I_r, based on the L_1 distance of their descriptors. If the ratio of those distances is smaller than a given threshold t_{match}, then the feature from image I is matched to its closest neighbor in image I_r.

The resulting match set is a list of pairs of points $\{(\Gamma_i(I), \Gamma_i(I_r))\}_{1\leq i\leq N_m}$ where $\Gamma_i(I)$ and $\Gamma_i(I_r)$ denote pixel location, respectively, in the images I and I_r. The size of the match set N_m depends on the parameters of the attack and on the image.

We set the threshold to $t_{match} = 0.7$, as advised in the original implementation of SURF. For smaller values of the threshold, the matching algorithm does not provide enough points to be able to resynchronize in some cases, especially when the feature descriptors are noisy. For larger values of the threshold, too many outliers are present in the matches, and the resynchronization is not reliable.

Chapter 6

Resynchronization

We assume that the received image has been desynchronized through one of the following attacks: affine transform, homography or image warping. In an ideal case (*i.e.* without any other attack), the parameters of the geometric attacks can be retrieved by solving an overcomplete linear system using the feature matches. This linear system either uses Cartesian coordinates for affine transform and image warping, or homogeneous coordinates for homography.

The presence of other attacks (copy-merge attacks, additive noise, etc.) creates two types or errors in the process of feature detection, description and matching. The first error is related to localization: two features can be matched correctly but, since the feature detector is not perfectly robust to noise and other transforms, those features are not localized identically, relative to the image. We model the matching process in this case as follows:

$$\Gamma_i(I_r) = f\left(\Gamma_i(I)\right) + W_i, \quad 1 \leq i \leq N_m, \tag{6.1}$$

where f denotes the geometric attack, and W_i is the localization error, a 2-D vector drawn from an uncorrelated normal distribution with zero mean and variance σ_n^2.

The other kind of error is related to the feature descriptor: two features that should have been matched to each other have different descriptors in the two images and are matched incorrectly. In this case, we model the matching process as follows:

$$\Gamma_i(I_r) = O_i, \quad 1 \leq i \leq N_m, \tag{6.2}$$

where O_i is a location drawn uniformly over the image I_r. We denote by π the probability that $\Gamma_i(I_r)$ has an incorrect match (*i.e.* is an outlier).

6.1 Estimation-elimination algorithm

Our first algorithm estimates the attack parameters globally (*i.e.* by considering all the matches at once). It iterates between two steps: (i) estimation of the parameters of the attack based on the current matches, and (ii) elimination of inaccurate matches, given the current estimates of the parameters.

All the geometric transforms we consider can be expressed in homogeneous coordinates. Since we need to solve linear systems in our algorithm, we first normalize the Cartesian coordinates of the feature points before estimating the parameters [31] in order to have well-conditioned problems. This is done by shifting and scaling the coordinates of the feature points such that:

1. The centroid of the feature points in each image corresponds to the origin.

2. The average distance of a feature point to the origin is $\sqrt{2}$.

This normalization is applied in the two images I and I_r independently. Since the last coordinate in a homogeneous system is 1, this normalization enhances the stability of the linear system solution used in the algorithm.

When the geometric attack is an affine transform or a homography, the two steps take the following form:

1. Solve the following overcomplete linear system to get an estimate $\mathbf{p}^{(j)}$ of $\mathbf{p}$:
$$\Gamma_i(I_r) = f_a(\Gamma_i(I), \mathbf{p}^{(j)}), \quad 1 \leq i \leq N_m, \tag{6.3}$$
where $f_a(\cdot, \mathbf{p}^{(j)})$ denotes the affine transform or the homography with parameter $\mathbf{p}^{(j)}$ and $j \in \{1, 2, \cdots, N_{iter}\}$ is the current iteration index.

2. Eliminate the matches $(\Gamma_{1,i}, \Gamma_{2,i})$ that are too distant from $\mathbf{p}^{(j)}$, in the L^1 sense:
$$\left|\left|\Gamma_i(I_r) - f_a(\Gamma_i(I), \mathbf{p}^{(j)})\right|\right|_1 > \frac{100}{j}. \tag{6.4}$$

When the geometric attack is image warping, the algorithm is very similar. However, we add a preprocessing step: we delete the matches $(\Gamma_i(I), \Gamma_{2,i})$ for which the two matched points $\Gamma_i(I)$ and $\Gamma_i(I_r)$ are distant by more than twice the maximum allowed shift s_m, in the L^1 sense. Our experiments

showed that we need at least 5 matches in each triangle to estimate correctly the parameters. Then, for each triangle of the mesh, we repeat the following two steps:

1. Estimate the shift of the three vertices of the triangle by solving an overcomplete linear system, constrained by the assumption that the vertices on the borders of the image are not shifted.

2. Eliminate the matches $(\Gamma_i(I), \Gamma_i(I_r))$ that are not consistent with the estimated shifts, according to (6.4).

After each triangle has been processed, each node receives six estimates of its shift, from the six surrounding triangles. We average those values by weighting them with the number of matches in each surrounding triangle.

After estimating the unknown parameters, we invert the geometric transform using these estimates. This image is close to the original image but not perfectly synchronized.

We then apply a second time the same algorithm (feature computation, matching, parameter estimation and attack inversion) to this image. This second step improves the accuracy of the estimated attack parameters. The resulting image I_s is almost perfectly synchronized due to the fact that the SURF features are computed with an image that is fairly close to the original image (the SURF features are not perfectly robust to all affine transforms).

We noticed that the resulting image I_s is not systematically closer to the original image when we apply more than two times the resynchronization algorithm.

The main advantage of Estimation-Elimination is that the algorithm is fast. Indeed, we generally need fewer than 15 iterations to obtain accurate estimates.

6.2 Adapted RANSAC

The RANSAC algorithm — abbreviation of Random Sample Consensus — was first published by Fischler and Bolles in 1981 [32] and can be a good alternative to Estimation-Elimination. It estimates the attack parameters from a small number of matches instead of trying to fit a model to all the

matches at once. Therefore, outliers have no negative effect on the estimation when they are not part of the set of matches considered.

However, since the matches include outliers and some localization errors, the algorithm must try many small sets of matches before finally finding a model that fits most of the matches. Hence it is computationally more intense than Estimation-Elimination.

The same data normalization as for Estimation-Elimination is first applied to the feature points coordinates.

Our adaptation of RANSAC algorithm can be described as follows:

1. Choose a random set of N_{ran} matches.

2. Compute an estimate $\mathbf{p}^{(j)}$ of the geometric attack parameters using this set of matches by solving a linear system.

3. Count the number of matches S_j which agree with this set of parameters, within a given distance d_{ran}, *i.e.* which satisfy:

$$S_j = |\mathcal{E}_j| \tag{6.5}$$

and

$$\mathcal{E}_j = \left\{ i, \left|\left| \Gamma_i(I_r) - f\left(\Gamma_i(I), \mathbf{p}^{(j)}\right) \right|\right|_1 < d_{ran} \right\}. \tag{6.6}$$

4. Repeat steps 1-3 n_{ran} times. Keep the set $\mathcal{E}_k$ that corresponds to the largest number of matches:

$$k = \underset{1 \leq j \leq N_m}{\operatorname{argmax}}(S_j). \tag{6.7}$$

5. Compute an estimate $\hat{\mathbf{p}}$ of the geometric attack parameters using this set of matches $\mathcal{E}_k$.

In the original algorithm, the size N_{ran} of the random set is the minimum size to be able to fit a model. In the case of homography, eight parameters have to be estimated so four matches should be enough to get an estimate. However, we noticed that by setting $N_{ran} = 10$, we generally had better estimates and needed fewer iterations. Experimentally, we set $d_{ran} = 3$ and $n_{ran} = 1,000$.

The complexity of this algorithm is higher than Estimation/Elimination because we need to estimate the attack parameters twice per iteration, which is 2,000 times in total, compared to 15 times for Estimation/Elimination.

Chapter 7

Experimental Results

For our experiments, we used grey-level images of size 512×512. The parameters of our code are: $n = 25$, $N_i = 4$, $S = 7$, $p = 0.7$, $M_e = 3$, $M = 33,554,432$, and $N = 784$ (see Section 2.1 for a definition of these symbols). We tested the performance of our traitor-tracing scheme by running Monte Carlo simulations. Each simulation was repeated 1000 times for each considered attack.

The fingerprinting scheme is deemed "robust" if it can retrieve one of the colluders with probability of error lower than 5%. The extraction of the fingerprint is always performed in a soft way, except for Sections 7.1 and 7.2.

7.1 Copy-merging attacks

We first compare the robustness of our scheme against different copy-merging attacks and different numbers of colluders K. Table 7.1 shows the error probability for different copy-merging attacks, and Table 7.2 shows the effect of the number of colluders on the error probability. For both simulations, Gaussian noise is added after the copy-merging attack with $FNR = 1$. Our scheme is robust to 6 colluders for any copy-merging attack, when soft decoding is used. In order to design a code resistant to more colluders, one would need to change the parameters of the code to obtain a longer fingerprint.

7.2 AWGN

We then studied the impact of Gaussian noise on our scheme. The attacks considered consist of interleaving between 3 copies followed by addition of Gaussian noise with variable FNR. The different values of FNR and the corresponding probabilities of errors are given in Table 7.3. Our scheme

Table 7.1: Copy-merging attacks against which our scheme is robust, with the corresponding empirical probability of error. Gaussian noise is added with $FNR = 1$.

Attack	P_e
Interleaving $K = 6$	2.8%
Averaging $K = 6$	1.9%
Median attack $K = 6$	3%

Table 7.2: Effect of the number of colluders on the error probability. Gaussian noise is added with $FNR = 1$.

Attack	P_e (soft decoding)	P_e (hard decoding)
Interleaving $K = 3$	0.0	0.1%
Interleaving $K = 4$	0.1	0.2%
Interleaving $K = 5$	0.2	3.9%
Interleaving $K = 6$	2.8	17.7%
Interleaving $K = 7$	9.2	35.5%

Table 7.3: Comparison of the hard and soft extraction, for 3 colluders and different values of the FNR.

FNR	P_e (soft decoding)	P_e (hard decoding)
1	0.0%	0.0%
1/10	0.0%	0.1%
1/20	0.3%	3.7%
1/30	1.9%	24.5%

is highly robust to AWGN when soft decoding is perfomed, and is able to retrieve the fingerprint correctly even when the power of the noise is 30 times the embedding power. This simulation was done to see what is the "break point" of the algorithm as a function of FNR. An FNR larger than 1/10 is not very realistic: the quality of the noisy image I_r is very poor.

Tables 7.2 and 7.3 also clearly show the substantial performance improvements obtained by using soft decoding instead of hard decoding.

7.3 Other non-desynchronizing attacks

Table 7.4 presents the probability of error for JPEG compression, cropping, and histogram equalization, used in conjunction with the interleaving attack

Table 7.4: Some non-desynchronizing attacks used in conjunction with interleaving (with $K = 3$) and addition of Gaussian noise (with $FNR = 1$).

Attack	P_e
JPEG Compression ($Q = 45$)	2.0%
Cropping (50% of image area)	2.1%
Histogram equalization	3%

Table 7.5: Affine transforms against which our scheme is robust, using Estimation/Elimination algorithm.

Attack and parameters	P_e
Scaling $\times 0.6$	4.0%
Scaling $\times 0.7$ and rotation of 35°	3.0%
Scaling $\times 0.74$ and shearing of 25°	3.9%
Aspect ratio of 1:0.6	3.1%

Table 7.6: Image warping against which our scheme is robust, using Estimation/Elimination algorithm.

Attack and parameters	P_e
Warping: 4×4 grid, 50 pixels max. shift	2.5%
Warping: 5×5 grid, 30 pixels max. shift	0.8%
Warping: 6×6 grid, 20 pixels max. shift	1.9%
Warping: 7×7 grid, 10 pixels max. shift	1.0%

and addition of Gaussian noise with $FNR = 1$.

7.4 Desynchronizing attacks

This section quantifies the performance of our resynchronization algorithms. For all those results, the cascade of attacks is the following: interleaving with 3 colluders, addition of Gaussian noise with $FNR = 1$, and a geometric attack.

Tables 7.5 and 7.6 show the probability of error when an affine transform or an image warping is used. We notice our scheme can resist those strong attacks very well. In those two cases, Estimation/Elimination is used for resynchronization.

The results presented in Table 7.7 show a comparison between Estima-

Table 7.7: Comparison of Estimation/Elimination and RANSAC algorithms for two attack models: affine transform (first 3 lines) and homography (last three lines).

Attack and parameters	Resynchronization algorithm	# iterations (Resync. alg.)	P_e
Scaling $\times 0.6$	Est./Elim.	15	4.0%
Scaling $\times 0.6$	RANSAC	1,000	4.3%
Scaling $\times 0.6$	RANSAC	100	11.4%
Scaling $\times 0.6$, Rolling 30°	Est./Elim.	15	5.0%
Scaling $\times 0.6$, Rolling 30°	RANSAC	1,000	3.2%
Scaling $\times 0.6$, Rolling 30°	RANSAC	100	30.2%

tion/Elimination and RANSAC as synchronization algorithms. The three first lines correspond to the affine transform model, and the last three to the homography model. Both algorithms result in approximately the same probability of error in the case of a scaling attack (in the affine transform model), but RANSAC needs 1,000 iterations when Estimation/Elimination only needs 15. Thus, there is a trade-off between the complexity of RANSAC and the resulting error probabilty since the latter increases when the number of iterations for RANSAC is decreased.

However, in the case of homography, RANSAC performs better than Estimation/Elimination when the number of iterations for RANSAC is large enough.

Chapter 8

Conclusion

In this paper, we have proposed a complete, practical, robust, and secure image fingerprinting scheme. By using Jourdas' high-rate code and our Laplace-STDM embedding scheme, we can accomodate more than 33 million users and resist coalitions of up to six colluders, while using a short fingerprint. We showed that our scheme is robust against the most common copy-merging attacks (averaging, interleaving and median attack), additive Gaussian noise (with a fingerprint-to-noise ratio of 1/30), JPEG compression, intense cropping, and histogram equalization. Moreover, the embedding scheme ensures invisibility, security and reliability of the fingerprint.

Our SURF-based resynchronization algorithm is also very efficient: it can handle geometric attacks that happen in media recording (affine transforms and homography), and other attacks involving more parameters like image warping. Estimation/Elimination seems more adapted to affine transforms and image warping, and RANSAC is better suited to to homography. However, RANSAC is globally more complex than Estimation/Elimination because it needs more iterations. To our knowledge, this is the first fingerprinting scheme that can cope with so many users and severe geometric attacks.

Future work could include a study of the impact of digital-to-analog and analog-to-digital transforms on this fingerprinting scheme and solutions to deal with them.

References

[1] The Motion Picture Association of America, "Piracy and the law," May 2010. [Online]. Available: www.mpaa.org/piracy_AndLaw.asp

[2] ——, "Research statistics," May 2010. [Online]. Available: www.mpaa.org/researchStatistics.asp

[3] ——, "2005 US piracy fact sheet," May 2010. [Online]. Available: www.mpaa.org/USPiracyFactSheet.pdf

[4] ——, "Piracy economies," May 2010. [Online]. Available: http://www.mpaa.org/piracy_Economies.asp

[5] Tera Consultants, "Impact économique de la copie illégale des biens numérisés en france," Nov. 2008. [Online]. Available: www.droit-technologie.org/upload/dossier/doc/179-1.pdf

[6] P. Moulin and R. Koetter, "Data-hiding codes," *Proc. of the IEEE*, vol. 93, no. 12, pp. 2083–2126, 2005.

[7] M. Musgrove, "Sleuths crack tracking code discovered in color printers," Oct. 2005. [Online]. Available: www.washingtonpost.com/wp-dyn/content/article/2005/10/18/AR2005101801663.html

[8] Warner Bros Entertainment Inc., "Motion picture anti-piracy coding," U.S. Patent 7,206,409, April 2007. [Online]. Available: http://www.patentstorm.us/patents/7206409/description.html

[9] U.S. Department of Justice, "Chicago man arrested for criminal copyright infringement in connection," Jan. 2004. [Online]. Available: http://www.justice.gov/criminal/cybercrime/spragueArrest.htm

[10] Federal Bureau of Investigation, Kansas City, "Two men plead guilty to trafficking in counterfeit DVDs, CDs," Oct. 2009. [Online]. Available: http://kansascity.fbi.gov/dojpressrel/pressrel09/kc100709a.htm

[11] J.-F. Jourdas and P. Moulin, "A high-rate fingerprinting code," *Proc. of the SPIE*, vol. 6819, p. 68190A, 2008.

[12] ——, "High-rate random-like fingerprinting codes with linear decoding complexity," *IEEE Trans. on Information Forensics and Security*, vol. 4, no. 4, pp. 768 – 780, 2009.

[13] ——, "A low-rate fingerprinting code and its application to blind image fingerprinting," *Proc. of the SPIE*, vol. 6819, p. 681907, 2008.

[14] S. He and M. Wu, "Joint coding and embedding techniques for multimedia fingerprinting," *IEEE Trans. on Information Forensics and Security*, vol. 1, no. 2, pp. 231–247, 2006.

[15] ——, "Collusion-resistant video fingerprinting for large user group," *IEEE Trans. on Information Forensics and Security*, vol. 2, no. 4, pp. 697–709, Dec. 2007.

[16] W. Trappe, M. Wu, Z. Wang, and K. Liu, "Anti-collusion fingerprinting for multimedia," *IEEE Trans. on Signal Processing*, vol. 51, no. 4, pp. 1069–1087, 2003.

[17] W. Kautz and R. Singleton, "Nonrandom binary superimposed codes," *IEEE Trans. on Information Theory*, vol. 10, no. 4, pp. 363–377, Oct. 1964.

[18] D. Boneh and J. Shaw, "Collusion-secure fingerprinting for digital data," *IEEE Trans. on Information Theory*, vol. 44, no. 5, pp. 1897–1905, Sep. 1998.

[19] V. Korzhik, A. Ushmotkin, A. Razumov, G. Morales-Luna, and I. Marakova-Begoc, "Collusion-resistant fingerprints based on the use of superimposed codes in real vector spaces," *Int. Multiconference on Computer Science and Information Technology*, pp. 487–491, Oct. 2009.

[20] D. Delannay and B. Macq, "Watermarking relying on cover signal content to hide synchronization marks," *IEEE Trans. on Information Forensics and Security*, vol. 1, no. 1, pp. 87–101, 2006.

[21] M. Pawlak and Y. Xin, "Robust image watermarking: an invariant domain approach," *IEEE Canadian Conf. on Electrical and Computer Engineering*, vol. 2, pp. 885–888, 2002.

[22] B. Coskun and M. Mihcak, "Perceptual hash-based blind geometric synchronization of images for watermarking," *Proc. of the SPIE*, vol. 6819, p. 68191G, 2008.

[23] V.-Q. Pham, T. Miyaki, T. Yamasaki, and K. Aizawa, "Robust object-based watermarking using feature matching," *IEICE Trans. on Information and Systems*, vol. E91-D, no. 7, pp. 2027–2034, 2008.

[24] B. Chen and G. Wornell, "Quantization index modulation: a class of provably good methods for digital watermarking and information embedding," *IEEE Trans. on Information Theory*, vol. 47, no. 4, pp. 1423–1443, May 2001.

[25] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features," *CVIU*, vol. 110, no. 3, pp. 346–359, 2008.

[26] D. Lowe, "Object recognition from local scale-invariant features," *Proc. of the 7th IEEE Int. Conf. on Computer Vision*, vol. 2, pp. 1150–1157, 1999.

[27] D. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 2002, pp. 28 – 35.

[28] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[29] J. Bauer, N. Sünderhauf, and P. Protzel, "Comparing several implementations of two recently published feature detectors," in *Proc. of the International Conference on Intelligent and Autonomous Systems*, Toulouse, France, 2007.

[30] V. Chandrasekhar, G. Takacs, D. Chen, S. Tsai, J. Singh, and B. Girod, "Transform coding of image feature descriptors," *Visual Communications and Image Processing 2009*, vol. 7257, no. 1, p. 725710, 2009.

[31] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, UK: Cambridge University Press, 2003, pp. 107 – 108.

[32] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381 – 395, June 1981.