End of Course Project

Copyrigth Protection Platform

Author: Roger Pau Monné
Advisor: Joan Tomàs-Buliart

Department of Telematics Engineering

This document has been produced using $\LaTeX$ and gnuplot.
The objective of this project is to provide a safe and reliable Content Distribution Platform, to allow users without any specific knowledge to download and view movies anywhere, and to assure the holders of the copyright that dishonest users, that redistribute the copyrighted material can be traced and identified.

The aim of this project is not to prevent the users from physically distributing the content they can download through the platform, but to discourage them from doing so, because they can be identified and brought to the authorities competent in that matter.

The technologies used to make this possible are described in detail, and a fully functional implementation of a secure Content Distribution Platform is developed.

Since each movie downloaded by a user contains a special content embedded inside the video stream, a modification of a well known open source video encoder was used. This encoder is called FFmpeg, and is used because of its speed and easy to understand code. The modification developed in this project allows to modify the discrete cosine transform coefficients, and to inspect them in order to recover the modified ones.

During the project a couple of contributions to the FFmpeg project where done, to allow the use of filters to modify the contents of certain frames. This modifications are now included in the stable FFmpeg code, and allow filter developers to discriminate different types of frames.
# Contents

1 Introduction ................................................. 1
   1.1 Current situation ........................................ 1
   1.2 Objectives ............................................. 2

2 Previous concepts ........................................ 5
   2.1 Differences between watermarking and cryptography ....... 5
   2.2 Classical image watermarking systems .................... 8
   2.3 Common concepts in actual watermarking schemes .......... 9
   2.4 Spread Spectrum Modulation ............................. 11
   2.5 MPEG-2 Video Stream ................................... 13
   2.6 Watermarking process .................................. 13

3 Marking videos using a custom FFmpeg modification ....... 17
   3.1 Previous work .......................................... 17
   3.2 General improvements ................................... 18
   3.3 Addition of the mark .................................... 20
   3.4 Extraction of the mark .................................. 20
   3.5 Confabulation .......................................... 22
   3.6 Analysis of the video stream before adding the mark ..... 24
   3.7 Usage of FFmpeg ...................................... 24
   3.8 Benchmark results ..................................... 27

4 Marking videos using an avfilter ............................ 31
   4.1 Libavfilter introduction ................................ 31
   4.2 Implementation ........................................ 32
   4.3 Results and study of common attacks to the system ....... 34
   4.4 Study of the degradation of image quality after embedding a mark ... 37

5 Design of the Content Distribution Platform ................. 41
   5.1 Technical requirements of the platform .................. 41
   5.2 End-user features of the platform ....................... 48
   5.3 Study of the technology used in the platform .......... 52

6 RPC Interfaces .............................................. 57
   6.1 Marker Interface ....................................... 57
   6.2 FileServer Interface ................................... 58
   6.3 Tracker Interface ...................................... 59
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Differences between watermarking and cryptography</td>
<td>6</td>
</tr>
<tr>
<td>2.2</td>
<td>Generic embedding of marks</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Generic mark extraction</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Black coefficients represents those used to embed the mark</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>DCT Matrix</td>
<td>14</td>
</tr>
<tr>
<td>2.6</td>
<td>Watermarking scheme</td>
<td>14</td>
</tr>
<tr>
<td>2.7</td>
<td>Non-null coefficients in DCT luminance matrix for sample movie A</td>
<td>15</td>
</tr>
<tr>
<td>2.8</td>
<td>Non-null coefficients in DCT luminance matrix for sample movie B</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>Watermarking algorithm ([CKLS97b])</td>
<td>20</td>
</tr>
<tr>
<td>3.2</td>
<td>Differences between marking and distortion assumptions</td>
<td>22</td>
</tr>
<tr>
<td>3.3</td>
<td>Memory used during the addition of a mark</td>
<td>28</td>
</tr>
<tr>
<td>3.4</td>
<td>Memory used during the extraction of a mark</td>
<td>29</td>
</tr>
<tr>
<td>3.5</td>
<td>Memory used during the merging of two marked videos</td>
<td>29</td>
</tr>
<tr>
<td>3.6</td>
<td>Time of the different operations</td>
<td>30</td>
</tr>
<tr>
<td>4.1</td>
<td>Libavfilter working flow</td>
<td>32</td>
</tr>
<tr>
<td>4.2</td>
<td>Frames marked with different alpha values</td>
<td>38</td>
</tr>
<tr>
<td>4.3</td>
<td>Frames from a marked movie with alpha 20 post-processed with a Gaussian filter</td>
<td>39</td>
</tr>
<tr>
<td>4.4</td>
<td>Frames from a marked movie with alpha 20 post-processed with a Blur filter</td>
<td>39</td>
</tr>
<tr>
<td>4.5</td>
<td>Relation between $\alpha$ and PSNR</td>
<td>40</td>
</tr>
<tr>
<td>5.1</td>
<td>Layout of the components of the system</td>
<td>42</td>
</tr>
<tr>
<td>5.2</td>
<td>Flow of the request of an item</td>
<td>44</td>
</tr>
<tr>
<td>5.3</td>
<td>Flow of the addition of an item</td>
<td>45</td>
</tr>
<tr>
<td>5.4</td>
<td>Flow of the mark of an item</td>
<td>46</td>
</tr>
<tr>
<td>5.5</td>
<td>Block diagram of a RPC server</td>
<td>54</td>
</tr>
<tr>
<td>7.1</td>
<td>Scheme of the database</td>
<td>70</td>
</tr>
</tbody>
</table>
LIST OF TABLES

4.1 Tests results for Sample A ......................................................... 35
4.2 Tests results for Sample B ......................................................... 36
4.3 Tests results for Sample C ......................................................... 36
4.4 Tests results for Sample D ......................................................... 37
### LIST OF LISTINGS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>ff_mpeg2_watermark function</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>cox_algorithm function</td>
<td>20</td>
</tr>
<tr>
<td>3.3</td>
<td>copy_block function</td>
<td>21</td>
</tr>
<tr>
<td>3.4</td>
<td>compare_block function</td>
<td>21</td>
</tr>
<tr>
<td>3.5</td>
<td>copy_merge_block function</td>
<td>22</td>
</tr>
<tr>
<td>3.6</td>
<td>merge_block function</td>
<td>23</td>
</tr>
<tr>
<td>3.7</td>
<td>analize function</td>
<td>24</td>
</tr>
<tr>
<td>3.8</td>
<td>code to plot the DCT matrix</td>
<td>24</td>
</tr>
<tr>
<td>3.9</td>
<td>How to mark a video using FFmpeg</td>
<td>24</td>
</tr>
<tr>
<td>3.10</td>
<td>How to extract a mark from a video using FFmpeg</td>
<td>25</td>
</tr>
<tr>
<td>3.11</td>
<td>Convert a video to I-Frames only using FFmpeg</td>
<td>26</td>
</tr>
<tr>
<td>3.12</td>
<td>Merge two videos using FFmpeg</td>
<td>26</td>
</tr>
<tr>
<td>3.13</td>
<td>Analyze a video using FFmpeg</td>
<td>27</td>
</tr>
<tr>
<td>4.1</td>
<td>How to mark a video</td>
<td>33</td>
</tr>
<tr>
<td>4.2</td>
<td>How to recover a mark from a video</td>
<td>33</td>
</tr>
<tr>
<td>4.3</td>
<td>calculate MSE function</td>
<td>40</td>
</tr>
<tr>
<td>7.1</td>
<td>Extract key and certificate from PKCS#12</td>
<td>64</td>
</tr>
<tr>
<td>7.2</td>
<td>Fetch and reload PostgreSQL CRL</td>
<td>64</td>
</tr>
<tr>
<td>7.3</td>
<td>Add SSL support to PostgreSQL</td>
<td>64</td>
</tr>
<tr>
<td>7.4</td>
<td>Allow users to connect to the PostgreSQL server</td>
<td>64</td>
</tr>
<tr>
<td>7.5</td>
<td>Database schema</td>
<td>65</td>
</tr>
<tr>
<td>7.6</td>
<td>Set Tomcat KeyStore and TrustStore</td>
<td>70</td>
</tr>
<tr>
<td>7.7</td>
<td>Add a Tomcat manager user</td>
<td>71</td>
</tr>
<tr>
<td>7.8</td>
<td>Set up an SSL connector on port 8443</td>
<td>71</td>
</tr>
<tr>
<td>7.9</td>
<td>/etc/hosts</td>
<td>71</td>
</tr>
<tr>
<td>7.10</td>
<td>Java Service Wrapper configuration template</td>
<td>72</td>
</tr>
<tr>
<td>7.11</td>
<td>Marker Service config.yml</td>
<td>76</td>
</tr>
<tr>
<td>7.12</td>
<td>Install Marker service</td>
<td>77</td>
</tr>
</tbody>
</table>
The objective of this project is to provide a safe and reliable Content Distribution Platform, to allow users without any specific knowledge to download and view movies anywhere, and to assure the holders of the copyright that dishonest users, that redistribute the copyrighted material can be traced and discovered.

The aim of this project is not to prevent the users from physically distributing the content they can download through the platform, but to discourage them from doing so, because they can be discovered and brought to the authorities competent in that matter.

The technologies used to make this possible are described in detail, and a fully functional implementation of a secure Content Distribution Platform is developed.

1.1 Current situation

As a consequence of the recent improvements in Information Technologies, and the absence of barriers in the virtual world, a new market based on the fact that there's no intellectual property has appeared. A new market where a user with enough skills can find any kind of content easily, fast and free of charge. Using P2P\(^1\) networks, or pages that contains lists of links for downloading the desired file directly from a remote web server it is possible to find and download any kind of information without having to pay for it. Users of this kind of services are used to low-quality content and long waiting times, but it is not important as long as there's no need to pay for the desired content.

According to the Observatorio de piratería y hábitos de consumo de contenidos digitales\cite{Ach10}, during the last six months of 2009 the contents downloaded by users without the proper copyright license where priced at 5121 millions of euros, more than three times the profit that the industry

\(^1\)Peer-to-peer: distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.
receives from legal transactions, which is taxed at 1653 millions of euros. All efforts made to stop this problem have proven useless, and the download of content without the appropriate copyright permission increases every year.

It is clear that there isn't any method to prevent this increase in the download of illegal content, but technologies like watermarking and fingerprinting allows to trace the dishonest user, and act as a deterrent factor.

1.2 Objectives

The main objective of this project is to develop a secure and usable platform for delivering digital content over the Internet. The main problem of the distribution of content over the Internet is the protection of the copyright of the distributed media.

This work exposes how to create, deliver and detect possible threats to the system. The platform uses a protection system in order to ensure that the users cannot share the copies of their videos. The system is based on the principles of watermarking and fingerprinting codes. A watermark is a message that is inserted into a document, that provides information about the author or the work itself. Since every user will have a different message embedded inside the video, it is easy to point out which user has distributed his copy without authorization. Fingerprinting algorithms are used to generate the marks that will be embedded inside the original document, this algorithms are used to provide robustness and detection of traitors that perform confabulations to obtain fraudulent marks.

This protection does not block the user from sharing the content physically, since the movie that the user downloads is a normal video, and it does not require any special player, hardware or secret key to be played back. The use of watermarking and fingerprinting allows the content distributor to trace which user or users distributed the unauthorized copy, and is a deterrent factor, because the dishonest user knows that the content distributor can trace the origin of the copy if he finds it on unauthorized places.

The main objectives of this work were to create a fast and robust watermarking process, based in the FFmpeg open source video editing library\textsuperscript{2} and to develop a distributed platform with a web interface that allows the users to easily download the desired movies.

The distribution platform has to be scalable and distributed, because the embedding of a mark in a movie has a high computational cost, in an ideal system, there should be many entities in charge of marking videos, to be able to cope with the demand. This also allows the possibility of adding new entities while the system is running, if there is a peak in the demand of video files.

To coordinate all the elements of the system, some kind of authority also has to be developed, to ensure that all the entities that conform the system will cooperate between them and work together to provide the services the user expects.

One of the main objectives of the system also is to be secure, regarding communications from outside of the system, and internal communications of the different entities. To ensure this kind of se-

\textsuperscript{2}http://www.ffmpeg.org
1.2. OBJECTIVES

Security, the TLS/SSL protocol has to be used in all transactions, and each entity in the system needs a valid certificate.

It is obvious that communications from the outside should be protected against sniffer and other kind of attacks, that’s why they need to be encrypted, but it is also important to ensure that communications inside the system are secured, to prevent malicious users from impersonating servers and gaining access to the system.

Since the connection between servers is performed using TLS/SSL, it would be feasible to have the system distributed across several data centers without the need of creating virtual private networks across them, greatly simplifying the deploying of new servers to cope with the demand.
The definition of watermarking is simple. A watermark is a message that is inserted into a document, that provides information about the author or the work itself. The reader can easily see that this vague definition applies to a great variety of fields. From the format of the original work; audio, video, text... to the topology of the incrusted mark, the spectrum of possibilities is endless.

2.1 Differences between watermarking and cryptography

Cryptography should not be confused with watermarking (the differences can be seen in the Figure 2.1). When a content is ciphered the user or client is forced to have a specific key to be able to view the original data (the term key in this paragraph has to be understood as a software, code...). When watermarking is used, the intention is not to block the user from displaying the content, the objective is to prevent a dishonest user from removing the mark (a mark of property or a copyright) from a content. If we take an email as example, the encryption of it would prevent anybody except the authorized receiver from reading it. On the other hand, watermarking the email would prevent a dishonest user from deleting the signature and forwarding the email changing the authorship of it.

The main technical objective of digital watermarking schemes is to build a robust and secure watermark. To accomplish this goal a basic requirements have to be set to discriminate a good watermarking technique. This requirements are:

1. **Fidelity**: a watermark should be perceptually invisible, it should not cause the degradation of the original content that’s being marked.

2. **Robustness**: a watermark should be difficult to remove. Specially, it should be resistant to the distortions caused by the typical signal processing mechanisms (conversion for analog
3. **Capacity**: A watermarking system has to be capable of embedding a relative high volume of information. Being capable of storing a large volume of information inside a mark makes the watermarking algorithm more flexible and versatile.

Some of this requirements are incompatible between them, for example it’s not feasible to design a watermarking system that can provide a high robustness and capacity without introducing a high degree of distortion to the resulting document. On the other hand, an invisible and robust system is probably not able to offer a high capacity. As a result of this constrains, the design of a watermarking schema is a trade-off between all this parameters. Also, not all kinds of watermarking systems are designed to maximize this requirements, as an example the fragile watermarking is designed to detect any kind of modification made to a document, so it is very fragile, and when a document is modified the recovered mark is different from the original. This watermarking technique is based

---

**Figure 2.1**: Differences between watermarking and cryptography
on a low robustness algorithm, in fact, the low robustness of the algorithm is what proves the validity of this method.

Other complementary requirements also exist, and they vary according to the application. The most relevant ones are the following:

1. **Undetectable**: impossibility to prove the presence of a hidden message in a certain content. This concept is strongly tied to the statistical model of the original document. It is important to remark the the capacity to detect the presence of a mark doesn't directly imply the possibility to remove it, but in some cases the possibility to apply a certain watermarking schema depends of this factor. Sometimes this requirement is mistaken with fidelity, it is important to remark that fidelity is based on perceptual measurements, on the other hand undetactability is based on statistical methods.

2. **Complexity**: the process of generating a mark should not be trivial, since a dishonest user may take advantage of this and be able to generate false mark, and the system might not be able to differentiate false marks from original ones.

3. **Access key**: embedded information should not be extracted, not even with attacks designed knowing the algorithm used in the embedding. The extraction of the mark requires knowing the algorithm and having at least a marked copy and the mark embedded in the copy. This property exactly the same as the one applied in cryptographic systems.

4. **Low error probability**: it is important to minimize the probability of incorrectly detecting a mark. It is also important to differentiate this situation from the one that happens when it is not possible to detect any mark. Recovering an erroneous mark may incriminate a honest user, since the recovered mark may match with mark of this user due to an error in the extraction. This presents a subtle issue, that has to the examined with caution.

5. **Computational cost of the embedding and extraction**: usually the input files of this systems have a high size (a DVD movie), but it should be feasible to embed the mark in a short amount of time using appropriate and optimized algorithms. This factor is usually not taken into account, since the computational power of devices is increased every day.

The scope of application of watermarking techniques is very large, and this project is centered on applying watermarking systems to video streams, in a similar way as still images, video streams are a set of coded still images, usually using the DCT\(^1\).

\(^1\)The Discrete Cosine Transform or abbreviated as DCT is a transformation based on the Discrete Fourier Transform but using only real numbers. Formally the Discrete Cosine Transformation is an invertible lineal function with an equivalent square matrix. The DCT used in the image domain is a 2D-DCT with a size of 8x8 pixels. The formal formulation is the following:

\[
X_{k_1,k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1,n_2} \cos \left[ \frac{\pi}{N_1} \left( n_1 + \frac{1}{2} \right) k_1 \right] \cos \left[ \frac{\pi}{N_2} \left( n_2 + \frac{1}{2} \right) k_2 \right]
\]

where \(X\) is the transformed matrix of \(x\) and \(N_1 = N_2 = 8\) when using images.
2.2 Classical image watermarking systems

Currently there’s a great number of programs able to embed information inside of an image. Most of them work by adding the desired information in the less significant bits of every pixel [vSTO94]. When using this method, the information embedded in the system remains invisible to the human eye [KM92], but it is trivial for a third party to detect and remove this information without introducing any distortion in the original image. Improved systems use a shared key between the sender and the receiver, and a pseudo-random cryptographic generator [Sch95] to select the bits that will hold the encrypted data somehow [FJM96].

It is logical to think that not all pixels in an image will be suitable for embedding the watermark: modifications made on pixels that belong to large areas of the same color (monochromes) or on sharp edges are more likely to cause visual distortion of the image. That is why some systems have algorithms designed to choose the best pixels for embedding extra information. This algorithms are based on the calculation of the variance of the luminance in the surrounding pixels (if the result of this difference is very high, it means the pixel belong to an edge, and if it very low or null it means the pixel belongs to a monochrome region). If a pixel passes this test, it is capable of holding the desired information in it’s least significant bits.

This techniques can easily be neutralized by an attacker that has the permissions to manipulate the marked image. As an example, almost every filtering process can modify the value of the less significative bits of an image, effectively removing all the extra information and deleting the mark without affecting the quality of the image. A way to palliate this attack is to add an error correction code to the mark, or to add the mark multiple times in the same image. The ”Patchwork” algorithm, developed by Bender et al. and exposed in [BGML95], modify the luminance of pairs of pixels selected in a pseudo-random way to embed a mark. A similar system was proposed by Pitas on [Pit96]. Many similar techniques can be used to mark digital audio.

Another attack to this systems is to break the synchronization needed to detect the zones of the image where the information was embedded. Regarding images, it is possible to crop a region of the image. On the audio field, an attack developed by Anderson et al. on [AP98] to accomplish a desynchronization of the audio stream is described as removing random parts of the samples and duplicating the others. This attack introduces a jitter of several microseconds that is capable of fooling the typical embedding systems.

Another way to attach information to an image is to embed this information inside the color palette. A great number of the images distributed over the internet are coded using formats based on color palette mechanism, like GIF or PNG. The advantage of this method is that it is easier to design a secure system based on images with a certain degree of noise, like images coming from converted analog sources (scanner, camera...). The main disadvantage is that the length of the mark is restricted to the size of the palette and not the size of the image.

On [Kwa] it is suggested that a message might be hidden in a secure way on a color palette by shifting the order of the colors instead of actually changing the colors itself. This method doesn’t change the visual aspect of the image perceived by the viewer, which is an advantage, but it’s safety is at least questionable, because most image processing programs order the palette in an specific manner (usu-

---

2 Cropping: process where a portion of the image is removed from the original.
3 Jitter: deviation in or displacement of some aspect of the pulses in a high-frequency digital signal
ally regarding the luminance of the colors or their frequency. A randomly ordered palette would probably seem suspicious at least, and just opening the image and saving it again using any common image processing program would destroy the hidden message because the palette will be reordered.

More practical methods can obtain capacities that depend on the size of the image or the number of pixels. There is existing software that first decreases the depth of the color palette of a GIF image to 128, 64 or 32 and then modifies the less significant bits of this palette to make it grow back to its original size. Using this method it is possible to embed from one to three bits per pixel without creating much visual distortion. However, the newly created palette will have a similar group of colors, and the detection of this kind of marks is feasible because of this fact [JJ98b, JJ98a].

One of the most popular methods to embed information in images based on a color palette was proposed by Machado in [Mac]. The proposed method, known as **EZ Stego** the color palette is first ordered according their luminance. In the re-ordered palette adjacent colors are very similar and **EZ Stego** embeds the message in a binary way using the less significant bits of the pixels that point to the color palette.

The algorithm is based on the premise that adjacent colors in a luminosity ordered the palette are similar. However, as luminosity is a lineal combination of the three colors R (red), G (green) and B (blue), eventually adjacent colors in a palette can represent different colors. To avoid this problem, Fridich proposes to embed this information in the parity bit (the calculation of the parity bit is as follows: \( R + B + G \mod 2 \)) found on adjacent colors [Fri99]. Based on the color of every pixel that is going to be used to embed information a search is performed to find similar colors on the palette until a color with the desired parity is found. According to the author, it is realistic to assume that parity bits across a palette are distributed in a random way.

Currently all these systems based on pixel or color palette modifications have been put aside in favor of embedding the information on the transformed domain. During the project none of this systems where implemented.

### 2.3 Common concepts in actual watermarking schemes

Usually the process of embedding a watermark follows four steps schematized in Figure 2.2. The first step is to transform the original document to a more appropriate domain to perform the embedding process. It is common to use the Discrete Cosine Transform, the Fast Fourier Transform or the Discrete Wavelet Transform. In this project the Discrete Cosine Transform was used because most video compression systems are based on this transformation and the watermarking algorithm will be more efficient and easy to implement. The second step consist in coding the original message or the information to a mark that is suitable for embedding. The third step is the embedding of the mark inside the DCT coefficients, and the last step is the inverse DCT transformation, to recover the image.

The process of recovering a watermark consists of three steps, as show in Figure 2.3. The first step is to convert the document to the domain where the mark was embedded and extract the mark from it. The second pass determines if the extracted information is a watermark, and if the result of this step is positive, it decodes the embedded message. Depending on the information available to perform the extraction, we can distinct between two different detection schemes:
CHAPTER 2. PREVIOUS CONCEPTS

1. **Informed detection**: the original unmarked document is available.

2. **Blind detection**: the decoder works without using the original document.

Usually, the availability of the original document during the detection process facilitates the decoding, and better results are obtained using informed detection.

As explained before, the marks are embedded directly into the DCT domain, modifying the coefficients of the transformation. It is important to notice that not all the coefficients of the domain are modified, on the image field, the DCT matrix has sixty four coefficients (structured in a $8 \times 8$ square matrix), but usually only some of them are used. The coefficients are not chosen randomly, the coefficients used are the ones where a small modification doesn’t represent a big impact on image quality, but that they are enough important that they can not be removed from the image without a significant image quality loss. Different coefficients have been used for various studies, as an example the author of [CKLS97a] use the coefficients indicated in Figure 2.4(a) while the authors of [MDC04] use the coefficients of Figure 2.4(b).

Regardless of the coefficients to be marked, the marking process applies the chosen transformation and selects the coefficients where the watermark information is going to be embedded. This positions can be permuted to increase the robustness of the algorithm.
2.4 Spread Spectrum Modulation

The Spread Spectrum modulation proposed by Cox et al. on [CKLS97a] is one of the most famous watermarking schemes, probably because it was one of the first ones to embed the mark in the transformed domain (using the DCT). The article advocated that adding the mark to the most significant coefficients in the DCT would provide a most robust watermark. Also, this watermark should be generated using independent samples and equally distributed, generated by a Gaussian distribution (N(0,1)). So any attempt to attenuate or remove the watermark would result in a severe degradation of the content.

This article defines the following notation: given a content to mark (cover work) \( C = \{c_1, c_2, \cdots, c_N\} \), the length of the embedded mark \( W = \{w_1, w_2, \cdots, w_N\} \) so that \( C \) can be converted to a marked vector \( C_w = \{c'_1, c'_2, \cdots, c'_N\} \). On the proposal it is thought that this marked document \( C_w \) is sent using a given channel that produces a set of attacks. Eventually, the document given to the detector is represented as \( C_w^* \). One simple way to embed the mark could be

\[
c'_j = c_j + \alpha \cdot w_j,
\]

(2.1)

where \( \alpha \) is the scaling parameter that controls the trade-off between the robustness and the visibility of the watermark. The equation 2.1 is always invertible, so given \( C_w^* \) and having \( C \) it is feasible to extract \( W^* \). It seems logical that the next step is to put together \( W \) and \( W^* \) in order to determine if the extracted watermark is the same as the one that has been embedded. This evaluation is performed using the similarity function represented in the equation

\[
sim(W, W^*) = \frac{W \cdot W^*}{\sqrt{W \cdot W^*}}.
\]

(2.2)
The next step is to settle a threshold $T$ so that if $\text{sim}(W, W^*) > T$, $W$ is considered to be similar to $W^*$ and the received document contains the mark $W$. On the other hand, if $\text{sim}(W, W^*) \leq T$, the document does not contain the mark $W$. The threshold level $T$ has to be chosen having in mind that a value too high may cause false negatives (the detector incorrectly states that the document does not contain $W$), and a value too low may cause false positives (the detector incorrectly states that the document contains $W$ when it actually does not).

Another way to calculate the similarity between $W$ and $W^*$, which is usually more efficient regarding systems implementations is

$$\text{sim}(W, W^*) = \frac{1}{N} W \cdot W^* = \frac{1}{N} \sum_i w_i \cdot w_i^*$$  \hspace{1cm} (2.3)

If a bit of the message ($m = 0$ or $m = 1$) is going to be inserted inside a document $c_0$, a reference mark $w_r$ of the same length as the document $c_0$. The watermark is defined using the following function

$$w_m = \begin{cases} w_r & \text{if } m = 1 \\ -w_r & \text{if } m = 0 \end{cases}$$  \hspace{1cm} (2.4)

once we have obtained $w_m$ it is inserted inside the original document $c_0$ as the equation 2.1 states.

Taking into account the noise injected during the process, the received content $c_w$ can be defined as $c_w = c_0 + \alpha w_m + n$, where $n$ is considered an additive Gaussian noise. The relation between the watermark and the noise can be measured using what the authors define as a Watermark-to-Noise relation (WNR), where $\delta_w^2$ is the strength of the mark and $\delta_n^2$ is the strength of the noise.

$$\text{WNR} = 10 \times \log_{10} \left( \frac{\delta_w^2}{\delta_n^2} \right).$$  \hspace{1cm} (2.5)

To detect the watermark, a substraction of the original document $c_0$ from $c_w$ is performed to obtain $w_m^*$. Then the lineal correlation between $w_m^*$ and $w_r$ is calculated using the following function

$$z_{lc} \left( w_m^*, w_r \right) = \frac{1}{N} (\alpha w_m \cdot w_r + n \cdot w_r).$$  \hspace{1cm} (2.6)

Taking into account that $n$ has a Gaussian distribution, $n \cdot w_r$ can be considered negligible because the term $\alpha w_m \cdot w_r = \pm \alpha w_r \cdot w_r$ is going to be greater. So $z_{lc} \left( w_m^*, w_r \right) \approx \alpha w_r \cdot w_r$ if the watermark is $m = 1$ and $z_{lc} \left( w_m^*, w_r \right) \approx -\alpha w_r \cdot w_r$ if the watermark is $m = 0$. The detector should set the threshold $T$ to $z_{lc} \left( w_m^*, w_r \right)$ and the result will be:

$$\text{message} = \begin{cases} 1 & \text{if } z_{lc} \left( w_m^*, w_r \right) > T \\ \text{no watermark} & \text{if } -T \leq z_{lc} \left( w_m^*, w_r \right) \leq T \\ 0 & \text{if } z_{lc} \left( w_m^*, w_r \right) < -T \end{cases}.$$  \hspace{1cm} (2.7)

If the original document is divided in several segments and a vector is generated for each segment, various bits of the mark can be embedded. It is important to notice that the smaller the segment, the greater the distortion generated by the noise will be.
2.5 MPEG-2 Video Stream

A basic overview of the MPEG-2 stream is needed to understand the watermarking process, first of all we need to know that every video file is just a bunch of frames\textsuperscript{4}. The most common frame rates are 25fps for the PAL standard and 29.97fps or 23.976fps for NTSC standard. Since each frame is an image and taking into account that just one second of video needs 25 images, it's easy to see that some compression algorithm should be implemented in order to reduce the size of the resulting data. This is where the MPEG-2 format comes to play, it allows compression of video streams using several techniques.

To accomplish the desired compression MPEG-2 format classifies the frames of a video into three categories, I-Frames, P-Frames and B-Frames. A brief description of each frame type and it's task is described above:\textsuperscript{[Tud95]}

- I-Frames are coded without reference to other pictures. Moderate compression is achieved by reducing spatial redundancy, but not temporal redundancy. They can be used periodically to provide access points in the bitstream where decoding can begin.

- P-Frames can use the previous I- or P-frame for motion compensation and may be used as a reference for further prediction. Each block in a P-frame can either be predicted or intra-coded. By reducing spatial and temporal redundancy, P-frame offer increased compression compared to I-frames.

- B-Frames can use the previous and next I- or P-Frame for motion-compensation, and offer the highest degree of compression. Each block in a B-Frame can be forward, backward or bidirectionally predicted or intra-coded. To enable backward prediction from a future frame, the coder reorders the pictures from natural 'display' order to 'bitstream' order so that the B-Frame is transmitted after the previous and next pictures it references. This introduces a reordering delay dependent on the number of consecutive B-Frames.

2.6 Watermarking process

In our use case the main interest is to mark I-Frames, because they suffer less distortion and are more resistant to future compressions. The mark however is not embedded directly into the image, instead it is placed in the DCT\textsuperscript{5} coefficients used during the encoding of the video. A two-dimensional DCT is performed on small blocks (8 pixels by 8 lines) of each component of the picture to produce blocks of DCT coefficients (Figure 2.5). The magnitude of each DCT coefficient indicates the contribution of a particular combination of horizontal and vertical spatial frequencies to the original picture block. The coefficient corresponding to zero horizontal and vertical frequency is called the DC coefficient.

At least four of this blocks are generated for every 8 pixel by 8 lines of video, we will modify the luminance component because it's the one that has less visual impact on the video once the mark has been inserted. P- and B-Frames cannot be marked because they don't describe and image but

\textsuperscript{4}A frame is a still image inside a video
\textsuperscript{5}Discrete Cosine Transformation
a difference between frames, often using movement vectors and other complex techniques, trying to mark those frames will introduce visual artifacts, rendering the video stream unsuitable for playback. At this point we already know that I-Frames should be marked, and the watermarking algorithm used in this project is the Cox Algorithm, which modifies the coefficients of the DCT luminance blocks [CKLS97b]. In Figure 2.6 we can see a graphical representation about how the mark is embedded into an image. This process is applied to a subset of I-Frames present in the original stream, that are then encoded back to the stream and saved.

To know which coefficients are more resistant to compression we have performed an analysis of the presence of non-null coefficients using different bitrates, the Figure 2.7 and 2.8 shows the result of this analysis. As we can see, coefficients near the [0,0] position\(^6\) tend to be more resistant to

\(^6\)This is the DC component
compression, thus we are going to mark those coefficients.

In Figure ?? the selected coefficients for compression are shown. This selection has been done taking into account that coefficients near the DC component have a big impact in the distortion, so coefficients [0,1], [1,0] and [0,0] are not marked to preserve image integrity.
MARKING VIDEOS USING A CUSTOM FFmpeg MODIFICATION

This watermarking process requires a considerable amount of CPU and memory, so it is important to use an efficient encoding/decoding process to add the mark to the video. This was the most important factor when deciding how to implement a functional marking software. Since implementing an MPEG2 encoder/decoder is difficult and would require a lot of knowledge to make it efficient. To solve this problems we decided to implement the marking algorithm on an already working MPEG2 encoder/decoder, FFmpeg. FFmpeg was chosen because it is licensed under the GPL, and because it is highly efficient, as the FFmpeg webpage states FFmpeg is a complete, cross-platform solution to record, convert and stream audio and video. It includes libavcodec - the leading audio/video codec library. Our work will be mainly focused in adding a modification to the MPEG2 encoder to allow the embedding of a mark during the encoding process. This work will be performed trying to modify the less possible parts of FFmpeg, always putting the major part of the code in outside files that will be compiled together with FFmpeg to provide the required functionality. This is done to allow the portability of the generated code across different versions of FFmpeg.

3.1 Previous work

The main objective of this modification is to allow a safe, multi-platform and fast watermarking implementation, capable of adding a mark, retrieving it and also allowing the merge of two different marked videos to generate a third video, whose mark will be a mix between the marks from the two input videos. There was a previous implementation of a marker and mark retriever in FFmpeg[PP08], and the work started by understanding this previous modification. This modification was based on an obsolete version of FFmpeg, which used assembly instructions that where no longer valid on modern systems.
The first step was to port the modification to a newer version of FFmpeg, so we could test the functionality of this modification and the possible improvements to the system. The new code is based on FFmpeg release 6.1. Once the original code was ported to the new version, some aspects of it were modified in order to improve it's speed, memory usage and understandability.

### 3.2 General improvements

The code that was used in general relied too much on self created structures, that where linked to AVCodecContext, the main FFmpeg structure. Instead of using this self created structured, that often contained counters, this implementation used as much data as possible from existing FFmpeg structures, like the MotionEstContext, that contains the position of the current block inside of the DCT block matrix and the frame number. With this elements we can get a unique identifier for each block, thus allowing for multiple threads to run concurrently during the marking and mark extraction processes, since no counters are used to know the position. Also the code regarding the watermark implementation was moved to a different file, called mpeg2watermark.c. This file contains the initialization and clean up functions (which will not be explained here, because they just perform reservations and frees of memory mainly), and the necessary code for doing the addition, extraction and confabulation of marks. The most important function in this file is `ff_mpeg2_watermark` (the contents of this function can be found in Listing 3.1), that is in charge of calling the necessary subroutines needed to perform the actions explained before. This function is only called from mpeg12.c, and only one time for each processed block.

```c
void ff_mpeg2_watermark(MpegEncContext *s, DCTELEM block[12][64]) {
  switch(s->avctx->watermark->action) {
    case FF_WATERMARK_ENC:
      switch(s->avctx->watermark->algo_type) {
        case FF_COX_ALGO:
          if(s->pict_type==FF_I_TYPE && ((s->avctx->frame_number / s->avctx->gop_size) + 1) % s->avctx->watermark->qdct == 0) {
            // av_log(s->avctx, AV_LOG_DEBUG, "Marcar frame: %d, gop_size: %d, qdct: %d\n", s->avctx->frame_number, s->avctx->gop_size, s->avctx->watermark->qdct);
            cox_algorithm(s, block);
            if((s->mb_x+s->mb_y*s->mb_width) == 0) {
              s->avctx->watermark->frames[s->avctx->watermark->frames_size++] = s->avctx->frame_number;
              s->avctx->watermark->frames = av_realloc(s->avctx->watermark->frames, (s->avctx->watermark->frames_size*1)*sizeof(int));
            }
            break;
          }
        break;
      }
      break;
    case FF_WATERMARK_DEC:
      switch(s->avctx->watermark->algo_type) {
        case FF_COX_ALGO:
          for(int i=0; i < s->avctx->watermark->frames_size;i++) {
            if(s->avctx->watermark->frames[i] == s->avctx->frame_number) {
              if(s->avctx->check_watermark->save == 1) {
              
            }
          }
        break;
      }
```

1 It can be found here: [http://www.ffmpeg.org/releases/ffmpeg-0.6.1.tar.bz2](http://www.ffmpeg.org/releases/ffmpeg-0.6.1.tar.bz2)
3.2. GENERAL IMPROVEMENTS

The parameters received by the function are the context that is currently being processed and the blocks of the DCT matrix, twelve 8x8 16bit integers\[Ker88\] blocks that contain all the coefficients. This function chooses which frames to mark, extract or merge and calls the specific functions that perform the modifications of the coefficients.

One of the main improvements also accomplished in this implementation is that the memory used for the extraction of the mark is fixed, and doesn't grow over time, as it was with the previous implementation, that presented problems when extracting a mark from large video streams.
3.3 Addition of the mark

The addition of the mark consists in adding a positive or negative integer to certain coefficients in the DCT matrix. The formula used is found in Equation 3.1.

\[
\nu'(i) = \begin{cases} 
\nu(i) - \alpha & \text{if mark is 0} \\
\nu(i) + \alpha & \text{if mark is 1}
\end{cases}
\]

Since we want this implementation to be concurrent, we can no longer use a pointer to know the position of the next mark bit that we have to insert, that's why we use the information already present in FFmpeg structures.

Listing 3.2: cox_algorithm function

Listing 3.2 contains the function used to mark the blocks, note that it is defined as `inline`, to speed up the process of calling this functions several times for a single frame. The initial position of the part of the mark that has to be embedded into a block is calculated at the start of the function, and in every iteration it is increased.

3.4 Extraction of the mark

The extraction of the mark is done comparing the original video with a watermarked one. Then the difference in the coefficients is recovered and the mark is retrieved. During this process FFmpeg works with two videos at the same time, it jumps from one video to another, processing one frame each time, as an example, if we use as notation x.y, where x means the video number and y the frame number, FFmpeg will process this videos as: 1.1 - 2.1 - 1.2 - 2.2 - 1.3 - 2.3 ... In order to compare the DCT blocks of each video we need to save the block from the first video and then compare it with

\[^2\text{Inline functions don't need to push and pop things on/off the stack, so the execution is faster}\]
3.4. EXTRACTION OF THE MARK

the block from the second video. In this process the actions performed by one stream are different
that the actions performed by the other stream, so two different functions where needed in order to
accomplish this without much overhead.

In Listings 3.3 and 3.4 we can observe the functions related to the extraction of the mark, the function
`copy_block` simply copies the current block to the shared memory, and the second function

```
static inline void copy_block(MpegEncContext *s, DCTELEM block[12][64]){
    int mb_xy = s->mb_x+s->mb_y*s->mb_width;
    // Synchronization lock (active wait)
    while(__sync_sub_and_fetch(&s->avctx->check_watermark->ref_block[mb_xy].save, 1) < 0) __sync_add_and_fetch(&s->avctx->check_watermark->ref_block[mb_xy].save, 1);
    for(int i=0; i<4; i++){
        for(int j=0; j<s->avctx->watermark->coeff_size; j++){
            s->avctx->watermark->mark_coeffs[j].y*8 + s->avctx->watermark->mark_coeffs[j].x;}
    }
    __sync_add_and_fetch(&s->avctx->check_watermark->ref_block[mb_xy].read, 1);
}
```

```
static inline void compare_block(MpegEncContext *s, DCTELEM block[12][64], int frame_number){
    int mb_xy = s->mb_x+s->mb_y*s->mb_width;
    int pos = (frame_number*s->mb_num*4*s->avctx->watermark->coeff_size+s->avctx->watermark->coeff_size) % s->avctx->watermark->x_size;
    // Synchronization lock (active wait)
    while(__sync_sub_and_fetch(&s->avctx->check_watermark->other_stream->ref_block[mb_xy].read, 1) < 0) __sync_add_and_fetch(&s->avctx->check_watermark->other_stream->ref_block[mb_xy].read, 1);
    for(int i=0; i<4; i++){
        for(int j=0; j<s->avctx->watermark->coeff_size; j++, pos = (pos + 1) % s->avctx->watermark->x_size) {
            diff = s->avctx->watermark->other_stream->ref_block[mb_xy].block[i][j] - block[i][j];
            if(diff > 0) {
                __sync_fetch_and_sub(&s->avctx->check_watermark->result[pos], 1);
            } else {
                __sync_fetch_and_add(&s->avctx->check_watermark->result[pos], 1);
            }
        }
    }
    __sync_add_and_fetch(&s->avctx->check_watermark->other_stream->ref_block[mb_xy].save, 1);
}
```

Listing 3.3: copy_block function

Listing 3.4: compare_block function
22

CHAPTER 3. MARKING VIDEOS USING A CUSTOM FFmpeg MODIFICATION

*compare_block* compares the differences between the copied block and the block that is currently being processed and stores the result in an array.

### 3.5 Confabulation

The workflow of the confabulation process is very similar to the one used in the extraction of the mark, two videos are processed at the same time and the coefficients of each video are compared in order to find the marked position. If there's a difference between two coefficients, we can either use a marking assumption or a distortion assumption.

- **Marking assumption:** when using this method, two positions of the DCT matrix are compared, and if the differ one of them is used, choosing randomly between the two (or more) options. This creates a new mark, that will be a merge of the marks present in the different video files used in the confabulation process. This process attempts to create a new mark, that does not belong to any of the users that participated in the confabulation procedure.

- **Distortion assumption:** this method also compares the positions of the DCT matrix and when there's a difference in the value the coefficients are averaged and the resulting value is stored in the position of the previous one. This method differs from the previous one because here the dishonest users are no trying to create a new mark, but to delete the mark present in the content by averaging the different position of the DCT matrix.

Since we introduce the mark cyclically in our video file, both methods yield the same result if the probability in the marking assumption for each user is the same. This can be seen clearly, as the recovered mark is averaged during the recovery process. In Figure 3.2 we can see the differences of both methods, and how the averaging performed during the recovery obtains the same result.

![Figure 3.2: Differences between marking and distortion assumptions](image)

```c
static inline void copy_merge_block (MpegEncContext *s, DCTELEM block[12][64])
{
    int mb_xy = s->mb_x+s->mb_y*s->mb_width;
    // Synchronozation lock (active wait)
```
while(__sync_sub_and_fetch(&s->avctx->merge_watermark->ref_block[mb_xy].save, 1) < 0) __sync_add_and_fetch(&s->avctx->merge_watermark->ref_block[mb_xy].save, 1);
for(int i=0; i<4; i++){
    for(int j=0; j < 64; j++){
        s->avctx->merge_watermark->ref_block[mb_xy].block[i][j] = block[i][j];
    }
}
// Signal that the block has been saved
__sync_add_and_fetch(&s->avctx->merge_watermark->ref_block[mb_xy].read, 1);

Listing 3.5: copy_merge_block function

static inline void merge_block(MpegEncContext *s, DCTELEM block[12][64]) {
    int mb_xy = s->mb_x+s->mb_y*s->mb_width;
    // Synchronization lock (active wait)
    while(__sync_sub_and_fetch(&s->avctx->merge_watermark->other_stream->ref_block[mb_xy].read, 1) < 0) __sync_add_and_fetch(&s->avctx->merge_watermark->other_stream->ref_block[mb_xy].read, 1);
    for(int i=0; i<4; i++) {
        for(int j=0; j < 64; j++) {
            if(block[i][j] != s->avctx->merge_watermark->other_stream->ref_block[mb_xy].block[i][j]) {
                switch(s->avctx->watermark->algo_type) {
                case FF_AVERAGE_ALGO:
                    block[i][j] = ((int) block[i][j] + (int) s->avctx->merge_watermark->other_stream->ref_block[mb_xy].block[i][j])/2;
                    break;
                case FF_RANDOM_ALGO:
                    block[i][j] = ((av_lfg_get(&s->avctx->watermark->rnd) % 2) > 0) ? block[i][j] : s->avctx->merge_watermark->other_stream->ref_block[mb_xy].block[i][j];
                    break;
                }
            }
        }
    }
    // Signal that the block has been read
    __sync_add_and_fetch(&s->avctx->merge_watermark->other_stream->ref_block[mb_xy].save, 1);
}

Listing 3.6: merge_block function

Listings 3.5 and 3.6 display the code used in these functions, as a note, FFmpeg bans the use of rand(), so we have to use the internal FFmpeg functions for generation random numbers, in this code a Lagged Fibonacci generator is used to generate pseudo-random numbers. The rest of the code is the same as the one used in the extraction of the mark, but this time all the coefficients of the DCT blocks are used, because the position of the mark is unknown.
CHAPTER 3. MARKING VIDEOS USING A CUSTOM FFmpeg MODIFICATION

3.6 Analysis of the video stream before adding the mark

This option allows us to see the more significant DCT coefficients before adding the mark, so the right positions can be marked. The process is simple, for each I-Frame that we would like to mark the DCT coefficients are extracted and checked to detect non-null positions. This information will allow us to decide which is the better place to add a mark. Usually the coefficients with the higher number of non-null values should be chosen for embedding the mark, since it is more likely that the values in these positions will suffer less loss because of the compression. The code in Listing 3.7 shows the process of storing the number of non-null coefficients.

```
static inline void analize ( MpegEncContext *s, DCTELEM block[12][64]) {
    for (int i=0; i<4; i++) {
        for (int j=0; j < 64; j++) {
            if( block[i][j] != 0) __sync_fetch_and_add (&s->avctx->watermark->analize [j], 1);
        }
    }
}
```

Listing 3.7: analize function

Once all the coefficients are gathered and added together, we can plot the resulting graph, that will show us which coefficients have a higher probability of being non null, and thus provide a more robust position for adding a mark.

```
# Graph 1
set autoscale
set title "Title"
set pm3d map
set size square
set palette rgbformulae 30,31,32
set cblabel "\footnotesize Number of not null coefficients"
unset cbtics
splot "<path to coefficients file >" matrix with image title ""
```

Listing 3.8: code to plot the DCT matrix

The Listing 3.8 is an example of how to plot this results with Gnuplot\(^3\), a freely available graphics plotter.

3.7 Usage of FFmpeg

Adding a mark to a video

To add a mark to a video we have to pass the following parameters to FFmpeg:

```
$ ffmpeg -threads 4 -i in.mpeg -add_watermark "-x 101 -q 10 -a 40 -m
{1,2;1,3;2,1} -t 0" -vcodec mpeg2video -b 4000k -acodec copy out.mpeg
```

Listing 3.9: How to mark a video using FFmpeg

\(^3\)http://www.gnuplot.info/
3.7. USAGE OF FFmpeg

The meaning of these options are the following:

• \textit{-threads}: number of simultaneous worker threads to use. On machines with more than one CPU this can speed up the processing of the video file.

• \textit{-i}: Video to mark. This is the original copy of the video file, without any modification. It is recommended that the input file is in MPEG2 format, to avoid unexpected results.

• \textit{-x}: Binary mark to embed in the form of ’1011101’. This mark can be of any length, as long as it doesn’t overflow the character limit of input arguments for a program present in the operating system used.

• \textit{-q}: Desired interval between marked I-Frames. If we use long videos, we might desire to mark only some frames, instead of marking all the I-Frames, to avoid degrading the quality of the movie.

• \textit{-a}: Strength of the mark\(^4\). This value is directly added or subtracted from the DCT matrix present in the input movie.

• \textit{-m}: Positions of the DCT coefficients to mark. The position of the coefficients that the process will modify in order to embed a watermark in the video file.

• \textit{-t}: Marking algorithm (the only one implemented currently is 0, which is a modification of the Cox Algorithm[CKLS97b]).

• \textit{-b}: Bitrate of the output video in bits/s. A low value will difficult the extraction of the mark, and the quality of the generated video, in MPEG2 it is recommended to use a bitrate ranging from 2000k to 4000k for optimal results.

Once the codification has ended we will find a file called out.mpeg in the directory where the program was executed and a log file, called Wat_enc_DATE-HOUR.log, that contains the position of the marked frames inside the video stream. We need to keep this log file in order to retrieve the mark afterwards, because the watermarking algorithm implemented hasn’t a blind-detection algorithm.

**Obtaining a mark from a video file**

To retrieve a mark from a video we have to use the following options:

\begin{verbatim}
$ ffmpeg -threads 4 -i original.mpeg -check_watermark "-x 3 -m \{1,2;1,3;2,1\} -t 0 -f Wat_enc_DATE-HOUR.log" -i watermarked.mpeg -vcodec mpeg2video -an -f rawvideo -y /dev/null -vcodec mpeg2video -an -f rawvideo -y /dev/null
\end{verbatim}

Listing 3.10: How to extract a mark from a video using FFmpeg

The meaning of the options is the following:

• \textit{-threads}: number of simultaneous worker threads to use.

\(^4\)The strength of the mark is directly related to the degradation the video will suffer
CHAPTER 3. MARKING VIDEOS USING A CUSTOM FFmpeg MODIFICATION

-\textbf{-i:} Original video (without any watermark) or marked video.
-\textbf{-x:} Length of the mark.
-\textbf{-m:} Positions of the marked DCT coefficients.
-\textbf{-t:} Marking algorithm (the only one implemented currently is 0, Cox Algorithm).
-\textbf{-f:} Log file created during the addition of the mark, that contains the number of the marked frames.

Once the process has ended the mark will be printed on the screen.

**Confabulating with two videos**

The first step needed to perform a confabulation is to convert both video files to I-Frames only streams. This is done because I-Frames have a higher quality, and the resulting confabulation will be better and more accurate. Since MPEG2 streams are composed of various types of frames, and most of them only reflect the difference between the current frame and the previous one it is important to perform the conversion before the confabulation, so that all frames can be averaged correctly.

\begin{verbatim}
$ ffmpeg -threads 4 -i input1.m2v -vcodec mpeg2video -qscale 1 -qmin 1 -intra -an -acodec copy output1.m2v
\end{verbatim}

Listing 3.11: Convert a video to I-Frames only using FFmpeg

Once both files contain only I-Frames we are ready to merge them, the following command show how to merge both videos into a single one, so the marks are mixed.

\begin{verbatim}
$ ffmpeg -threads 4 -i marked_video_1.m2v -merge_watermark "-t 0" -i marked_video_2.m2v -vcodec mpeg2video -an -f rawvideo -y /dev/null -vcodec mpeg2video -b 4000k -acodec copy result.m2v
\end{verbatim}

Listing 3.12: Merge two videos using FFmpeg

The meaning of the options is the following:

-\textbf{-threads:} number of simultaneous worker threads to use.
-\textbf{-i:} First marked and converted video file (I-Frames only).
-\textbf{-t:} Decision algorithm 0 implies to use a distortion assumption algorithm, while 1 means to use a marking assumption algorithm. The process used in both methods is described in detail in the previous section.
-\textbf{-i:} Second marked and converted video file (I-Frames only).

The resulting video can be found in result.m2v, containing a mark that’s a mix between the mark from the first and the second input files.
3.8 Benchmark results

The following benchmarks were performed on an eight core Intel(R) Xeon(R) CPU E5450 server, using from one to four simultaneous threads. The memory used by FFmpeg during the processing of the video files is not important, since it is usually below 20MB, so the memory present in the physical machine used in the testing process is always sufficient to ensure that the swap memory is not used. The movie used for the test had the following properties:

<table>
<thead>
<tr>
<th>Movie properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>5400s</td>
</tr>
<tr>
<td>Size</td>
<td>2.66GB</td>
</tr>
<tr>
<td>Bitrate</td>
<td>4000kb/s</td>
</tr>
<tr>
<td>Framerate</td>
<td>25fps</td>
</tr>
<tr>
<td>GOP Size</td>
<td>12 frames</td>
</tr>
</tbody>
</table>

All the tests were performed using the same parameters and changing the number of threads. The parameters used were the following:

<table>
<thead>
<tr>
<th>FFmpeg Parameters</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark length</td>
<td>512bits</td>
</tr>
<tr>
<td>Distance between marks</td>
<td>10 I-Frames</td>
</tr>
<tr>
<td>Mark strength</td>
<td>5</td>
</tr>
<tr>
<td>Marked positions in the DCT</td>
<td>([0,2] [0,3] [1,1] [1,2] [2,0] [2,1] [3,0])</td>
</tr>
</tbody>
</table>

\(^5\)http://gnuplot.info/
The results show that the execution time is reduced as the number of threads is increased. Also, we can see that the memory used is increased as the number of threads grow, this was expected, because it means that more blocks are processed concurrently and a different region in memory is needed for each thread.

![Memory used in encoding](image)

Figure 3.3: Memory used during the addition of a mark

In Figure 3.3 the reduction of execution time is greatly reduced as more blocks are simultaneously processed, but in Figure 3.4 the effect is even more noticeable, since the extraction of a mark doesn't create any file in disk and we don't have the bottleneck of writing data to a physical medium.

The confabulation process is more complex, because we first have to reencode each video to a stream containing I-Frames only, once this is done we will compare the DCT coefficients of both videos and mix them in order to scramble the mark. This process requires the parsing of every frame in the video, and thus is the most costly in terms of CPU usage. Figure 3.5 shows the memory usage of this process, and the time taken to perform the merge of the videos.

Finally a graph to summarize the results is presented in Figure 3.6, the time taken to execute the processes using one, two or three threads is clearly different, while the time difference from using three to four threads doesn't make a big speed increase. It is safe to say that using three threads is a good trade-off between speed and memory usage.
3.8. BENCHMARK RESULTS

Figure 3.4: Memory used during the extraction of a mark

Figure 3.5: Memory used during the merging of two marked videos
CHAPTER 3. MARKING VIDEOS USING A CUSTOM FFmpeg MODIFICATION

Figure 3.6: Time of the different operations
MARKING VIDEOS USING AN AVFILTER

During a long time FFmpeg used a filter mechanism called "vhooks", but it was not standard and it leaked crucial features to develop plugins. Since FFmpeg and libavcodec are the de facto standard for video decoding and encoding, and it is used vastly on many other applications a decision was taken to write a new filter architecture from scratch, that allowed a more versatile and powerful plugin system, aiming for it to become the default standard for writing video filters and plugins. Much of the work of designing and implementing the system took place during the Google Summer of Code of 2010.

This work was later included in FFmpeg and now is a part of the main distribution. At the time of writing this document not many filters exists, but there is an undergoing migration of MPlayer, OpenCV and other systems plugins to the AVFilter format.

To develop a DCT Watermarking plugin some modifications were made to the AVFilter source during this PFC, mainly to add some properties to the picture passed to the filter and be able to detect where the picture is a key frame or not. This modifications where sent to the official FFmpeg developers and are now a part of the FFmpeg code. The commits and information about the modifications can be found on [PM11a] and [PM11b].

4.1 Libavfilter introduction

Libavfilter is an FFmpeg library, designed to allow the creation of filters that can be applied to video processing, regardless of the format of the the input and output videos. This is accomplished by decoding the input stream and passing to the filter a single frame each time, then the processed frame is coded and saved in the output stream, this flow is represented in Figure 4.1.

This flow allows developers of filters to focus on the processing of the image, without handling any of the decoding, encoding or muxing. This is specially interesting when the filter has to be applied
in the spatial domain, but in our case the filter has to be applied in the DCT domain. This might seem like a problem, since AVFilter provides us with an image divided in 4 planes. The more usual procedure for modifying the frame would be to perform the DCT of the pixels, modify the desired coefficients and perform the IDCT to convert the image back to the spatial domain. This however can be simplified with the properties of the DCT.

Given the integrable functions \( f(x), g(x) \) and \( h(x) \) denote their Fourier transforms by \( \hat{f}(\xi), \hat{g}(\xi) \) and \( \hat{h}(\xi) \) respectively: [Pin01]

For any complex numbers \( a \) and \( b \), if \( h(x) = a \cdot f(x) + b \cdot g(x) \), then \( \hat{h}(\xi) = a \cdot \hat{f}(\xi) + b \cdot \hat{g}(\xi) \).

Based on this property the embedding process is less costly, since we no longer need to perform the DCT of the original image, instead we can create and empty matrix, fill the desired positions with the marking coefficients, perform the IDCT of the matrix and add it to the original image as seen in (4.1).

\[
\begin{bmatrix}
20 & \cdots & 0 \\
\vdots & \ddots & \vdots \\
-30 & \cdots & 40 \\
\end{bmatrix} \xrightarrow{\text{IDCT}} \begin{bmatrix}
5 & \cdots & 2 \\
\vdots & \ddots & \vdots \\
-3 & \cdots & 9 \\
\end{bmatrix} \xrightarrow{\text{Add to original}} \begin{bmatrix}
200 & \cdots & 154 \\
\vdots & \ddots & \vdots \\
8 & \cdots & 76 \\
\end{bmatrix}
\] (4.1)

4.2 Implementation

The implementation has been done using the Libavcodec and Libavutil libraries only, that are part of FFmpeg, so there are no external dependencies from using this filters. Since the most crucial part of the process is the DCT and IDCT transformations, optimized versions of this algorithms have been used. Currently FFmpeg comes with a DSPContext struct, that contains, when possible, hardware optimized versions of the most common video transformations. The watermark embedding and recovery filters have their own instance of a DSPContext, to be able to perform this operations as fast as possible.

Watermark embedding process

In the case of MPEG2 video streams (and most video codecs) the DCT of the image is performed in regions of 8x8 pixels, resulting in a 2D-DCT of 8x8 coefficients also. In this filter we are only going to modify the luminosity plane, so we can discard the other 3 planes corresponding to the chrominance values. When a new frame is fully decoded it is passed to the filter chain, to give each
4.2. IMPLEMENTATION

filter the opportunity to process the image and pass it to the filter that goes after. It is possible to chain various filters, so the input of one filter is the output of the filter that precedes it.

In our implementation, each time we get a new input frame that is a key frame, we create the corresponding 8x8 IDCT matrices with the values of the mark that are then added to the image. After that the image is passed to the next filter in the chain and the filter waits for a new input. Since marks are usually longer than the number of positions to mark, the mark is embedded in a circular way, this procedure also adds robustness\(^1\) to the marking, since the mark is not the same for each 8x8 block of pixels and is embedded more than one time in the stream.

The correct way of embedding a mark into a video stream is the following:

```bash
$ ffmpeg -threads 2 -i input.mpg -vf \\
"[in] watermark =20:0.2|0.3|1.1|1.2|2.0|2.1|3.0:101001011100000011100010 , setpts=PTS-STARTPTS [out]" -vcodec mpeg2video -b 4000k -acodec copy marked.mpg
```

Listing 4.1: How to mark a video

The watermark filter expects three parameters, the first one is the strength of the mark in the DCT domain, the second one corresponds to the positions of the DCT coefficients to mark, the syntax is y.x and each position is separated with a \|. The last parameter is the mark itself, that can be of any length. The setpts filter is used together with the watermark filter to make sure the timestamps of the video are not changed. Before the marking process starts, the parameters are printed on the screen.

**Watermark recover process**

The recovery process is performed comparing the DCT positions of the marked frame with the original video frame. In this case, instead of doing two DCTs, first the difference of the pixels between the original frame and the marked one is performed, and then the DCT is performed on this matrix of pixel differences. Once we have the DCT of the matrix of differences, we only need to get the values of the desired positions to recover the mark.

The correct way of recovering a mark is the following:

```bash
$ ffmpeg -threads 4 -i original.mpg -vf "movie=marked.mpg , setpts=PTS-STARTPTS [marked]; [in] setpts=PTS-STARTPTS , [marked] recovermark =7:0.2|0.3|1.1|1.2|2.0|2.1|3.0 [out]" -vcodec rawvideo -f null -an -y /dev/null
```

Listing 4.2: How to recover a mark from a video

The recover watermark filter expects two parameters, the first one is the length of the mark, and the second one corresponds to the position of the marked coefficients, the syntax is y.x and each position is separated with a \|. Additional setpts filters are added, to make sure the timestamps of both movies match.

---

\(^1\)ability of a system to cope with errors
4.3 Results and study of common attacks to the system

To prove the robustness of the watermarking system some tests were conducted, using other filters provided with FFmpeg. Here are the properties of the sample movies used for testing:

<table>
<thead>
<tr>
<th>Sample A</th>
<th>Sample B</th>
<th>Sample C</th>
<th>Sample D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format: Color cartoons</td>
<td>Format: Color live action movie</td>
<td>Format: Color live action movie</td>
<td>Format: Black and white live action movie</td>
</tr>
<tr>
<td>Duration: 101s</td>
<td>Duration: 128s</td>
<td>Duration: 149s</td>
<td>Duration: 120s</td>
</tr>
<tr>
<td>Size: 47.6MB</td>
<td>Size: 58.2MB</td>
<td>Size: 71.9MB</td>
<td>Size: 54.5MB</td>
</tr>
<tr>
<td>Bitrate: 4000kb/s</td>
<td>Bitrate: 4000kb/s</td>
<td>Bitrate: 4000kb/s</td>
<td>Bitrate: 4000kb/s</td>
</tr>
<tr>
<td>Framerate: 23.98fps</td>
<td>Framerate: 23.98fps</td>
<td>Framerate: 23.98fps</td>
<td>Framerate: 23.98fps</td>
</tr>
<tr>
<td>Dimensions: 848x480</td>
<td>Dimensions: 852x480</td>
<td>Dimensions: 848x352</td>
<td>Dimensions: 848x480</td>
</tr>
<tr>
<td>GOP Size: 12 frames</td>
<td>GOP Size: 12 frames</td>
<td>GOP Size: 12 frames</td>
<td>GOP Size: 12 frames</td>
</tr>
</tbody>
</table>

This samples correspond to movie trailers freely available on the Internet. All tests were done trying to simulate a real environment, so the samples where re-encoded several times, for example in the scale tests, the movie was first watermarked, scaled to the desired size, then it was scaled back to the original size and finally it was processed to recover the mark (the movie was decoded and encoded 3 times). The tests where performed this way to simulate the behavior of a dishonest user, who might try to scale the movie in order to remove the mark and then share it on the internet. Once the movie is recovered, it has to be scaled back and then compared with the original to extract the mark.
Table 4.1: Tests results for Sample A

<table>
<thead>
<tr>
<th>Alpha</th>
<th>40</th>
<th>35</th>
<th>30</th>
<th>25</th>
<th>20</th>
<th>15</th>
<th>10</th>
<th>5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark and recover (no filter)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.01</td>
</tr>
<tr>
<td>Gaussian filter factor 3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
</tr>
<tr>
<td>Gaussian filter factor 5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.32</td>
</tr>
<tr>
<td>Gaussian filter factor 7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
</tr>
<tr>
<td>Gaussian filter factor 9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.41</td>
</tr>
<tr>
<td>Gaussian filter factor 11</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.01</td>
<td>0.44</td>
</tr>
<tr>
<td>Gaussian filter factor 13</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.004</td>
<td>0.02</td>
<td>0.12</td>
<td>0.48</td>
</tr>
<tr>
<td>Gaussian filter factor 15</td>
<td>0.0</td>
<td>0.006</td>
<td>0.01</td>
<td>0.02</td>
<td>0.07</td>
<td>0.11</td>
<td>0.17</td>
<td>0.3</td>
<td>0.53</td>
</tr>
<tr>
<td>Blur filter factor 3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.3</td>
</tr>
<tr>
<td>Blur filter factor 5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.07</td>
<td>0.45</td>
</tr>
<tr>
<td>Blur filter factor 7</td>
<td>0.28</td>
<td>0.23</td>
<td>0.26</td>
<td>0.18</td>
<td>0.16</td>
<td>0.24</td>
<td>0.37</td>
<td>0.42</td>
<td>0.49</td>
</tr>
<tr>
<td>Scale to 706x400 (5/6)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.21</td>
</tr>
<tr>
<td>Scale to 636x360 (3/4)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.32</td>
</tr>
<tr>
<td>Scale to 424x240 (1/2)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.37</td>
</tr>
<tr>
<td>Scale to 282x160 (1/3)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.01</td>
<td>0.51</td>
</tr>
</tbody>
</table>

The tables with the results of the simulations contains the BER\(^3\) of each recovered watermark, that is the number of bit errors divided by the total number of bits that a mark contains. As an example, given the following mark ‘00101’, suppose the recovered mark is ’10001’, the number of erroneous bits is two, and the total number of bits is five, the BER would be

\[
BER = \frac{2}{5} = 0.4
\]

So the BER in this example would be 0.4, or 40%.

**Examples**

In Figure 4.2 the distortion introduced by the marking process can be seen, and the maximum recommended mark strength is 40, since anything above this value clearly deteriorates image quality.

Since the user is able to apply several distortions to the video, this algorithm has been tested for robustness, to make sure the implemented filter is able to resist several attacks. The video taken as example for this distortion tests has been marked with an alpha of 20 and a random mark of length 512bits. In Figure 4.3 we can see the distortion created by applying a gaussian filter to the marked movie, and in Figure 4.4 a blur filter is applied to the same marked movie.

In this examples the viewer can observe that the distortion introduced while trying to remove the mark clearly leaves the movie unwatchable, because a Gaussian filter with a factor greater than 11 or a Blur filter with a factor of 5 smooths the image so much that most text in the movie is unreadable and figures became difficult to differentiate.

---

\(^3\)Bit error rate
Table 4.2: Tests results for Sample B

<table>
<thead>
<tr>
<th>Alpha</th>
<th>40</th>
<th>35</th>
<th>30</th>
<th>25</th>
<th>20</th>
<th>15</th>
<th>10</th>
<th>5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark and recover (no filter)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.008</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.013</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.022</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.039</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.048</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 11</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.046</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 13</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.01</td>
<td>0.01</td>
<td>0.049</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 15</td>
<td>0.0</td>
<td>0.004</td>
<td>0.006</td>
<td>0.01</td>
<td>0.04</td>
<td>0.04</td>
<td>0.08</td>
<td>0.18</td>
<td>0.49</td>
</tr>
<tr>
<td>Blur filter factor 3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.15</td>
</tr>
<tr>
<td>Blur filter factor 5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
</tr>
<tr>
<td>Blur filter factor 7</td>
<td>0.24</td>
<td>0.23</td>
<td>0.25</td>
<td>0.19</td>
<td>0.15</td>
<td>0.18</td>
<td>0.23</td>
<td>0.3</td>
<td>0.48</td>
</tr>
<tr>
<td>Scale to 710x400 (5/6)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.19</td>
</tr>
<tr>
<td>Scale to 639x360 (3/4)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.23</td>
</tr>
<tr>
<td>Scale to 426x240 (1/2)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.37</td>
</tr>
<tr>
<td>Scale to 284x160 (1/3)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.45</td>
</tr>
</tbody>
</table>

Table 4.3: Tests results for Sample C

<table>
<thead>
<tr>
<th>Alpha</th>
<th>40</th>
<th>35</th>
<th>30</th>
<th>25</th>
<th>20</th>
<th>15</th>
<th>10</th>
<th>5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark and recover (no filter)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.001</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.26</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.004</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 11</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.042</td>
<td></td>
</tr>
<tr>
<td>Gaussian filter factor 13</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.002</td>
<td>0.02</td>
<td>0.07</td>
<td>0.15</td>
</tr>
<tr>
<td>Gaussian filter factor 15</td>
<td>0.01</td>
<td>0.02</td>
<td>0.03</td>
<td>0.05</td>
<td>0.1</td>
<td>0.12</td>
<td>0.15</td>
<td>0.28</td>
<td>0.5</td>
</tr>
<tr>
<td>Blur filter factor 3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.26</td>
</tr>
<tr>
<td>Blur filter factor 5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.004</td>
<td>0.43</td>
</tr>
<tr>
<td>Blur filter factor 7</td>
<td>0.25</td>
<td>0.19</td>
<td>0.19</td>
<td>0.2</td>
<td>0.19</td>
<td>0.25</td>
<td>0.3</td>
<td>0.33</td>
<td>0.5</td>
</tr>
<tr>
<td>Scale to 706x293 (5/6)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.14</td>
</tr>
<tr>
<td>Scale to 636x264 (3/4)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.24</td>
</tr>
<tr>
<td>Scale to 424x176 (1/2)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.32</td>
</tr>
<tr>
<td>Scale to 282x117 (1/3)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.38</td>
</tr>
</tbody>
</table>
Table 4.4: Tests results for Sample D

<table>
<thead>
<tr>
<th>Alpha</th>
<th>40</th>
<th>35</th>
<th>30</th>
<th>25</th>
<th>20</th>
<th>15</th>
<th>10</th>
<th>5</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mark and recover (no filter)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.07</td>
</tr>
<tr>
<td>Gaussian filter factor 3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.18</td>
</tr>
<tr>
<td>Gaussian filter factor 5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.31</td>
</tr>
<tr>
<td>Gaussian filter factor 7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.38</td>
</tr>
<tr>
<td>Gaussian filter factor 9</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.006</td>
<td>0.04</td>
</tr>
<tr>
<td>Gaussian filter factor 11</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.03</td>
<td>0.05</td>
<td>0.16</td>
<td>0.46</td>
</tr>
<tr>
<td>Gaussian filter factor 13</td>
<td>0.0</td>
<td>0.0</td>
<td>0.002</td>
<td>0.006</td>
<td>0.03</td>
<td>0.07</td>
<td>0.07</td>
<td>0.2</td>
<td>0.48</td>
</tr>
<tr>
<td>Gaussian filter factor 15</td>
<td>0.02</td>
<td>0.03</td>
<td>0.04</td>
<td>0.06</td>
<td>0.13</td>
<td>0.09</td>
<td>0.28</td>
<td>0.23</td>
<td>0.48</td>
</tr>
<tr>
<td>Blur filter factor 3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.24</td>
</tr>
<tr>
<td>Blur filter factor 5</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.008</td>
<td>0.06</td>
</tr>
<tr>
<td>Blur filter factor 7</td>
<td>0.3</td>
<td>0.24</td>
<td>0.24</td>
<td>0.26</td>
<td>0.28</td>
<td>0.33</td>
<td>0.36</td>
<td>0.5</td>
<td>0.33</td>
</tr>
<tr>
<td>Scale to 706x293 (5/6)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.23</td>
</tr>
<tr>
<td>Scale to 636x264 (3/4)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.24</td>
</tr>
<tr>
<td>Scale to 424x176 (1/2)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.35</td>
</tr>
<tr>
<td>Scale to 282x117 (1/3)</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.43</td>
</tr>
</tbody>
</table>

4.4 Study of the degradation of image quality after embedding a mark

To analyze the impact of a mark in a video sample we have used the peak signal-to-noise ratio (PSNR) calculation. The PSNR is the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale. In our case we have taken the original image as reference, and the marked image as "noise", this way we have a objective measurement of the degradation of the image once the mark as been applied. The equation of the PSNR is displayed in 4.3 and 4.4. $I$ and $K$ represent each frame of a video sequence, and $MAX_I$ is the maximum value a pixel (or a position in a component plane) can take (usually 255 or 240, depending on the image format).

$$PSNR = 10 \cdot \log_{10} \left( \frac{MAX_I^2}{MSE} \right)$$  (4.3)

$$MSE = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (I(i, j) - K(i, j))^2$$  (4.4)

Some tools exists to calculate the PSNR between two videos, like vidprofile⁴, but they are slow, because every frame has to be extracted, saved and the processed independently. That’s why a new FFmpeg filter was implemented, to help calculate the PSNR between two videos using an easy and fast approach. Since FFmpeg filters already work with images, the only step necessary to perform the PSNR calculation was to define a new filter that takes two inputs (video sequences) and compares them, to calculate the MSE of each frame. This MSE is then averaged across all frames, and finally the PSNR is calculated.

⁴http://vidprofile.berlios.de/vidprofile.html
CHAPTER 4. MARKING VIDEOS USING AN AVFILTER

(a) Sample A with alpha 10

(b) Sample B with alpha 20

(c) Sample C with alpha 30

(d) Sample D with alpha 40

Figure 4.2: Frames marked with different alpha values

Using this filter we have been able to obtain the PSNR of the different samples used in the previous section, and to have an objective measurement of the impact of a mark on a given video sequence.

As can be seen from the graph in Figure 4.5, the PSNR decreases according to the power of the mark, and monochrome movies (Sample D) seem to be less altered by this artifacts. Usual PSNR values range from 60 to 20dB, considering 20dB the minimum acceptable PSNR for an image sequence, so it can be proven that the mark, although introduces a degradation of quality, this degradation is not as high as to render the video sequence unwatchable.

The PSNR filter is based on the overlay filter, that merge two images together. In the PSNR filter, instead of adding the images, the difference between the two inputs is calculated, using the MSE, and then all the corresponding MSE values are accumulated and the PSNR is calculated. The code in Listing 4.3 is used to calculate the MSE of each frame and channel.
4.4. STUDY OF THE DEGRADATION OF IMAGE QUALITY AFTER EMBEDDING A MARK

Figure 4.3: Frames from a marked movie with alpha 20 post-processed with a Gaussian filter

Figure 4.4: Frames from a marked movie with alpha 20 post-processed with a Blur filter
40

CHAPTER 4. MARKING VIDEOS USING AN AVFILTER

Figure 4.5: Relation between $\alpha$ and PSNR

```
void compute_images_mse(const uint8_t * ref_data[4],
                        const uint8_t * data[4], const int linesizes[4],
                        int w, int h, const AVPixFmtDescriptor *desc,
                        double mse[4], uint16_t * line1, uint16_t * line2)
{
    int i, c, j = w;
    memset(mse, 0, sizeof(*mse)*4);
    for (c = 0; c < desc->nb_components; c++) {
        int w1 = c == 1 || c == 2 ? w >> desc->log2_chroma_w : w;
        int h1 = c == 1 || c == 2 ? h >> desc->log2_chroma_h : h;
        for (i = 0; i < h1; i++) {
            av_read_image_line(line1, ref_data, linesizes, desc, 0, i, c, w1, 0);
            av_read_image_line(line2, data, linesizes, desc, 0, i, c, w1, 0);
            for (j = 0; j < w1; j++)
                mse[c] += pow2(line1[j] - line2[j]);
        }
        mse[c] /= w1*h1;
    }
}
```

Listing 4.3: calculate MSE function
5.1 Technical requirements of the platform

Once the watermarking algorithm has been developed and tested, we would like to show an use case of this marking system. It is important to demonstrate that this system could be used to distribute video online. One of the main problems of distributing marked video, is that the marking process is costly, and marking videos on demand could mean the user has to wait some time before being able to download the requested movie.

The distribution system designed and developed allows an easy and fast video download, while being scalable and secure. The different parts that conform this system have been separated in several services, to allow the system to scale very well on high load environments.

Layout of the system

In Figure 5.1 we can see the components of the system, the only service that is accessible for the users is the web server, which is in charge of performing all the interaction with the user. This decision was taken to allow the simplification of the system and the reduction of the risks the platform is exposed to. Although we could isolate the Marker, Tracker, Database and Fileserver services from the external networks, all the communications between these servers make use of the TLS secure protocol and authentication mechanisms.

There are some servers which have a higher demand of computational power, like the Marker server, this is why we have decided to allow multiple instances of this servers, to distribute the tasks and be able to cope with the demand. The system can also have multiple instances of the Fileserver, since it has to store large amounts of data. Other services are limited to only one running instance, since
they coordinate the workflow of the system.

The number of running Marker and File servers can be changed at run time, without affecting the behavior of the system. This is interesting, because we are able to start more marking services during a period of time with high demand, and stop them once the demand is lower, allowing us to save energy and thus reducing expenses.

The system will treat with two different kind of files, ones will be the original ones, that are not marked neither modified. In this document this kind of files are called *items*. To distribute this items amongst the users of the platform, they have to pass the marking process, to obtain a new file. This file differ from the original because it has an embedded mark. This kind of marked items will be called *products* and will be the files that the end user receives.

**Services provided by the platform**

Even if the system is split in several components it doesn’t mind there isn’t a central control unit, that would be the Tracker. There can only be one Tracker instance in the whole system and coordinates all the other components, to allow them to work flawlessly and prevent concurrency issues. This is also the only component that has write access to the Database. All other elements depend on the Tracker, and communicate with it to obtain the necessary information.
This system allows the following operations:

- Associate a marked item with a user and provide the download to the user.
- Add new items to the database.
- Control the stock of marked items and mark more items when necessary.
- List the items currently present in the system.
- Control all the parameters of the system from the web interface.

**Associate a marked item with a user and provide the download to the user**

When there are marked items available the user will be able to proceed with the download of them, provided he has the correct permissions to view the movie. The download process is started by the web server, and the Tracker is in charge of assigning the marked item to the user that requested the download. Once the item is assigned to the user, the item is requested from the corresponding Fileserver and delivered to the user. In Figure 5.2 we can see the flow of this procedure.

**Add new items to the database**

The adding of new movies is also started from the web server, where a user with the appropriate permissions uploads an unmarked item. This item is stored temporary in the web server, while a transaction is established with the Tracker, that provides the web server with a suitable Fileserver to upload the movie to. Once the movie is uploaded the Tracker is notified of the new addition, and the item is added to the database. Figure 5.3 show the flow of this operation.

The flow contains some messages, that are described here as *ip:port*, this is because the system uses another port for binary transfers. The approach used to design this protocol is similar to the one used in FTP and related file transfer protocols. One transmission channel is used for signaling, and the other one is used to transfer the data requested. This last channel can only be used for a single transfer, once the transfer is finished the channel is closed by both ends. The message *ip:port* means that the previous message was a file transfer request, and the other end is giving the IP address and port were this transfer is going to take place.

**Control the stock of marked items and mark more items when necessary**

Since the Tracker is the only element of the system capable of adding movies and assign marked items to users, we can implement a database driven even that triggers the marking of new items. This event is launched every time the table of items is modified, then the stock of each item is checked and if necessary the Tracker requests an idle marker service or more to start marking items. If the process is successful, the marked items are added to the database and users can download them.

Figure 5.4 shows the communications that take place when the Tracker requests a Marker to create a new marked item.
List the items currently present in the system

All the presentation related processes are performed directly from the web server, that has a read-only access to the database. This interface is in charge of listing the items and showing them to the user. The designed system allows the following sorting:

- List all items.
- List items by category.
- Show items the user has already downloaded.
- Search based on text keywords present on the database.
- Show a random selection of movies.
Control all the parameters of the system from the web interface

If the user has the appropriate permissions it is possible to modify the working parameters of the system directly from the web interface, this allows for an easy control and maintenance of the system, since the most usual configuration changes can be performed from the same web page. The authentication system is role based, and the following roles exist:

- **Guest**: this is the most basic user, and only has the permission to view the movies but not
The Stock of a certain movie has reached its limit

Generate new fingerprint

Figure 5.4: Flow of the mark of an item

download them.

- **User**: a user has the permissions to view and download items.

- **Uploader**: this role is able to add new items to the system and is used by content managers.

- **Admin**: a user with admin permissions can perform any action, and is the only role that can change the system configuration parameters.
5.1. TECHNICAL REQUIREMENTS OF THE PLATFORM

Communication between the components of the system

Since the system has a lot of components, a communication protocol was chosen to unify all the components and allow an easy message exchange between them. This system also has to provide a good security, so the chosen protocol has to be able to operate on top of an encrypted channel.

Securing the communications between the components of the system

We assume an scenario where the different parts of the system (the Marker, Tracker and other services) might not be inside an isolated network, so we need to protect the communication between them to prevent attacks and data leakage. To solve this situation we encrypt all the connections using TLS. This solution provides encryption of the channel and also allows us to perform validation of the authenticity of the client and server before establishing the connection.

Authentication is accomplished using X.509 signed certificates, real time validated against the certification authority using the OCSP mechanism. All communications between the elements of the system make use of this feature, and if the system is unable to validate the identity of the entity the communication is immediately closed.

Because of this feature, the system always needs to be in contact with the certification authority, making it indispensable for the correct operation of the whole system. Users are also required to have a valid certificate to enter the system, this way we can also have a secure channel with the user.

The intensive use of the certification authority also has it’s drawbacks, if the authority is down, the system will be fully blocked, and impossible to perform any actions.

This is why the certification authority has to be protected from attacks. It is important to have in mind that the certification authority has to be accessible from outside users, so that they can check the validity of the server certificate, but it is also worthy to mention that this process is not currently implemented in any modern browser, so the user has to manually check the validity of the certificate. To prevent attacks like DoS\(^1\) we have to limit the maximum number of petitions we can get from the same IP; a good value would be 1 petition and hour, or even one petition a day, because there’s no need for the user to be constantly check for the authenticity of the certificate.

Communications protocol

The protocol agreed to use for inter system communication has to accomplish the following requirements:

- Function on top of a TLS connection.
- Easy to add new messages to.
- Portable between different programming languages.
- Fast.

\[^1\]A denial-of-service attack involves saturating the target machine with external communications requests, such that it cannot respond to legitimate traffic, or responds so slowly as to be rendered effectively unavailable
To allow for an easy implementation and use, we decided to use a RPC protocol, this allows us to call functions on a remote host like if it was running on the local machine. Doing so we can accomplish the abstraction between the real protocol and the functions called by the user, making the development and debugging easier.

There are many RPC protocols for Java, but most implementations don't support TLS sockets or OCSP validation, that's why we had to implement our custom RPC service to suit our needs. We chose to implement the protocol using the JSON-RPC 2.0 standard specification [Cro] [Gro10]. This is a lightweight yet fully featured RPC protocol, that allows the call of remote functions from local machines. Java also provides a very extensive reflection system, that allows us to make the RPC calls transparent to the user.

Since the messages that are used to intercommunicate the various elements of the system are encoded in the JSON standard, we decided to use an already implemented JSON parser to encode and decode the messages. The library used is called Jackson\(^2\) and allows a fast encoding and decoding of JSON data.

**Interaction with FFmpeg from Java**

The implementation of the marking and recovery of the mark is implemented in C to improve it's speed, but the rest of the system is developed in Java, because it more suitable for this task. Because of this fact, we need to somehow connect the C code with the Java application, that why we had to use the JNI library, which allows the calling of native C and C++ functions from Java code.

Using this interface, we are able to call FFmpeg code from Java without having to launch an external process to run the C code, and get the results of the called functions right into the Java code. This is an important factor, because we are able to link both codes in a transparent way. The JNI is only used in the Marker Server, that has to execute FFmpeg code to embed a given mark into a video stream.

### 5.2 End-user features of the platform

This chapter describes the main features of the platform that we are going to develop to distribute the marked videos online, with detailed explanations about how it works. Those features are grouped as administration features, and features offered to the customers (users) of the platform.

The platform will be implemented as an online catalogue of video files, available to the user for download. The main difference with any other distribution platform available will be that the served video files will contain a different mark, depending on the user that requested them.

**Administration features**

One of the main features of the administration is the possibility to change the parameters of the system, without having to modify any configuration files or reloading the services. The administration

\(^2\)http://jackson.codehaus.org/
5.2. END-USER FEATURES OF THE PLATFORM

zone allows the configuration of the following parameters.

Services

The platform for multimedia content management consists of different services, each with a distinct responsibility. By managing services, the administrator configures the deployment of the platform as required, balancing the load of components and adding more components on demand. This flexibility allows the administrator to define various marking services, or multiple file servers, sizing the platform for a high volume of requests or reducing it when the volume of requests is low.

This service management takes place through the Web portal access in the platform, and is only accessible to users with an administrator role, previously registered in the system.

Users

The content management platform includes four different user roles; administrators, clients, guests and content managers. Each one of these roles can access various functionalities of the system. Through user management functionality, the administrator (enabled to access this feature), can view the profile of registered users, and manage his / her access, license serial number, certificate name, status.

Currently, the system allows access to users with valid certificate previously unrecorded. At the time of first access, the user is automatically added to the system with a client profile. Later the profile of this user can be modified, to add administration or content management attributes.

Parameters

The administrator profile can change the settings of the platform and thus modify the functioning of it. These parameters are defined in the database during the deployment and configuration phase, but administrators may be interested in changing them at runtime according to the context of operation.

Different services query the system configuration parameters at runtime, so it is feasible to change them at any time without interrupting the service. These include marking parameters from the FFmpeg module; distance between marked frames, the strength of the embedded mark, encoding related parameters, DCT coefficients to modify. Also display parameters for the web page shown to the clients can be changed at runtime; products per page, product categories...

Publication of new contents

Both the administrator and content manager profiles have access to the publishing functionality. With this functionality, users can publish new products in the catalog. To publish a new content, the uploader needs to define the new product with all information visible to other users, and the system will generate marked products to meet the required minimum level of stock.

The publishing mechanism works as follows:
• A user with an appropriate role accesses the contents publication formulary. In this formulary the user will define all the characteristics of the new product; name, description, cover, categories it belongs to and the multimedia file that has to be served.

• Once the formulary has been filled correctly, the user requests the creation of the new item by submitting it. The web front-end will start the process of contacting the necessary services for creating this new item.

• If the information supplied is correct, and all the necessary services are available, the process will conclude by creating a new item and adding it to the catalogue of available items.

• At this time, the creation of the stock of marked products will start, so that when one client requests a new product, it can be served without any delays.

• When the stock reaches the defined minimum stock the generation of new products will end, and the system enters an idle state until a level of stock decreases or new content is added to the system.

**Stock control**

Although the system has an automatic stock control, which guarantees the existence of a minimum stock level for each product, the web front-end provides an interface to monitor these levels. This interface, available to the administrator profile, provides information on existing products as well as the stock level of each.

As mentioned in the section about the publication of new contents, there is an automatic control stock levels for each product, which should provide sufficient stock levels to cope with the demand. This mechanism is triggered in the publishing process and product procurement. During the first, the system does not have stock of the product advertised and must generate a minimum. In the second case, when a user purchases a product, the stock level decreases, requiring a corresponding replenishment to maintain the desired levels. The minimum stock level will respond to concurrent requests from different users, as the stock build process is not immediate.

Since the stock control is transparent to the administrator users, the web interface provides a section to view and modify the minimum stock level for each product, and also allows the regularization of stock levels.

**Mark recovery**

The mark recovery service is a functionality that completes the main system capacity, fingerprinting. It wouldn't be helpful to distribute marked content without the possibility to trace the generated marks, and discover the user that has that mark associated. This functionality is accessible to the administrator profile, and is capable of extracting the mark of a given file and compare it with the database of associated marks previously given to the users.

The system assumes, of course, that the product that is going to be analyzed for the mark extraction, is the result of a process of confabulation of one or more registered users. That is, one or more users, with products generated by the system, create a new product. This new product is created by
merging both user files, in an attempt to create a new mark that can’t be associated to any user. This process creates a product which mark does not match the original marks, in fact, this is the main aim of the confabulators, generate a product that can’t, apparently, be traced.

But the mark is not a random array of bits, it is designed to resist confabulation processes, so at least one of the users that confabulate is detected.

Thus, once the administrator has a confabulated product, that has been recovered for example, from a P2P network, it can be uploaded to the platform for analysis and mark recover, and the administrator will get the users involved in the fraudulent process, and a link to the product used to generate the confabulation. Currently the system can recover the involved users in a confabulation process with no more than two participants.

End User Features

Browse the catalogue of products

The platform has an interface that allows users and administrators to interact with the system using a Web browser that supports authentication based on client certificates. Once the end user is logged with a valid certificate installed in the browser, the system displays the contents through a catalog of products.

The catalog allows the user to access the published content easily. Initially, all products are sorted by name and structured through the browser with categories. The user can select a product to obtain more information and proceed with the purchase if interested.

Also the user can access the list of previously purchased products, and download them again if desired.

Both through the catalog of available products and products previously acquired, the user may sort the products using the attributes name, price and date of publication. If the system has a large number of products, these will be presented on different pages that the user may slide with the controls located at the bottom of the page.

Acquisition and Product Download

Browsing through the catalog of products is aimed at presenting the existing catalogue for later purchase. Thus, once the user selects a product, it triggers the allocation process and discharge. The allocation process is as follows;

- The user selects the option of downloading a new product, a product that has not been purchased previously.

- The system queries the inventory level of the ordered product, if the level is not sufficient to assign a marked product, the user is informed and proceeds to update inventory, allowing for further successful download once the stock is refilled.
• If the stock level is sufficient, the system assigns a marked product to the user and logs the transaction. Currently the purchasing process is not implemented, so the download doesn't require any monetary transaction.

• As described in previous paragraphs, this allocation process triggers the automatic control of stocks that should generate a new product, to replace the one that has just been assigned.

• In this same petition, after the allocation of a product to the user, the downloading process starts. The user can store the downloaded product on his local machine for later viewing.

• Subsequent access to products purchased do not require the allocation process and the download is directly started, providing the product that has already been downloaded.

5.3 Study of the technology used in the platform

Many technologies are used together to create this distributed and scalable system, we should provide a little insight on the technologies used and the given use in our case. Most of this technologies are already well documented on the Internet, but we are going to explain why we use them and how they interact with other components on the system.

TLS

The Transport Layer Security (TLS) and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that provide communications security over the Internet. TLS and SSL encrypt the segments of network connections above the Transport Layer, using symmetric cryptography for privacy and a keyed message authentication code for message reliability. Several versions of the protocols are in widespread use in applications such as web browsing, electronic mail, Internet faxing, instant messaging and voice-over-IP (VoIP). TLS is an IETF standards track protocol [DR08] and is based on the earlier SSL specifications developed by Netscape Corporation.

TLS is very important in our system, it assures us that all the data transferred can not be eavesdropped by a third party. If someone was able to get a copy of the original movie, he could distribute it without any watermark. Also it could happened that a third party might be able to obtain an item marked with the mark of a honest user, and then distribute this item illegally.

It also provides authentication since every piece of the system, from the servers to the users, has an unique certificate, that guarantees the authenticity of the entity in the other end of the communication channel.

This is why it is so important to use TLS and secure communications between all the pieces of the whole system.

OCSP

The Online Certificate Status Protocol (OCSP) is an Internet protocol used for obtaining the revocation status of an X.509 digital certificate. It is described in RFC 2560 [MAM+99] and is on the Internet
standards track. It was created as an alternative to certificate revocation lists (CRL), specifically addressing certain problems associated with using CRLs in a public key infrastructure (PKI).

When using CRL, we cannot provide real-time status checks of certificates, since the CRL is generated at fixed intervals of time, and if after the generation there has been a revocation the user is not aware of it, since the new list has not yet been published. Also CRL presents a bandwidth problem, because the user has to download the full CRL each time it is published.

Messages communicated via OCSP are encoded in ASN.1 and are usually communicated over HTTP. The "request/response" nature of these messages leads to OCSP servers being termed OCSP responders.

The OCSP technology is used because it is preferred over the CRL method. The OCSP assures us that once the certificate is revoked it can no longer be used within the system, and the restriction is applied at the same moment. With CRL list we have to wait until the new CRL is generated, then download and apply it. Using OCSP validation we can isolate any compromised server or user from our network, thus limiting the effect of this problem and allowing the system to resume normal operation.

**JNI**

JNI enables one to write native methods to handle situations when an application cannot be written entirely in the Java programming language, e.g. when the standard Java class library does not support the platform-specific features or program libraries. It is also used to modify an existing application—written in another programming language—to be accessible to Java applications. Many of the standard library classes depend on JNI to provide functionality to the developer and the user, e.g. file I/O and sound capabilities. Including performance- and platform-sensitive API implementations in the standard library allows all Java applications to access this functionality in a safe and platform-independent manner.

The JNI framework lets a native method use Java objects in the same way that Java code uses these objects. A native method can create Java objects and then inspect and use these objects to perform its tasks. A native method can also inspect and use objects created by Java application code.

JNI is sometimes referred to as the "escape hatch" for Java developers because it enables them to add functionality to their Java application that the standard Java APIs cannot otherwise provide. It can be used to interface with code written in other languages, such as C and C++. It is also used for time-critical calculations or operations like solving complicated mathematical equations, because native code can be faster than JVM code.

Since adding a mark to a movie is a complex and computationally expensive process we had to use a native library to perform this task, this way we can add a mark to a movie faster, and reduce the number of marking servers needed to cope with demand. Using native libraries also has its drawbacks, since we have to compile and deploy a different library for every operation system and architecture, and there’s no garbage collector, so all the memory management has to be done very carefully to prevent running out of memory and crashing.
Remote Procedure Call

RPC is an inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space (commonly on another computer on a shared network) without the programmer explicitly coding the details for this remote interaction. That is, the programmer writes essentially the same code whether the subroutine is local to the executing program, or remote.

For the needs of our application we decided to use the JSON-RPC protocol, and RPC protocol that uses JSON encoding to send messages across different computers in a network. JSON is a lightweight text-based open standard designed for human-readable data interchange. It is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects. Despite its relationship to JavaScript, it is language-independent, with parsers available for most scripting languages.

JSON messages are in a human readable form that is easy to debug, the drawback if this is that the traffic in the network is a little higher than if we used a binary protocol, that usually have a best compression ratio but are difficult to debug without some set of special tools. JSON messages are encoded in a dictionary fashion, and can contain any kind of data as long as there isn’t any repeated key.

Reflection

In computer science, reflection is the process by which a computer program can observe (do type introspection) and modify its own structure and behavior at runtime.

In many computer architectures, program instructions are stored as data, hence the distinction between instruction and data is merely a matter of how the information is treated by the computer and programming language. Normally, instructions are executed and data is processed; however, in some languages, programs can also treat instructions as data and therefore make reflective modifications. Reflection is most commonly used in high-level virtual machine programming languages like Smalltalk and scripting languages, and less commonly used in manifestly typed and/or statically typed programming languages such as Java, C, ML or Haskell.

Reflection is very important in our application, because is allows us to decode the RPC JSON messages and thus build the whole RPC system. With Java Reflection we can discover the types an anonymous class implements and perform the calls to the appropriate methods when a certain message is received.

![Figure 5.5: Block diagram of a RPC server](image)

If we take a look at Figure 5.5 we can observe a block diagram of an RPC server. The key components of this system are:
• **JSON Parser**: this element receives a RAW unprocessed data stream (in our case from a TLS Socket) and parses the information to obtain a Java Object, which can be passed to the next element in the chain. This conversion allows the next blocks in the diagram to not be tied to any specific data encoding, and if the protocol in which messages are encoded is changed, we will only need to replace this block in order to accommodate the necessary modifications for the new chosen data encoding.

• **Generic RPC Server**: the generic RPC server receives a Java Object (message) and tries to find a method in an unknown Object that matches the data given in the message. This procedure is done using Java Reflection, listing the methods and parameters that the unknown Object has and finding one that matches the request. If there's suitable method, a call to this method is performed, using the parameters provided in the message.

• **Class implementing the service**: this is the real implementation of the server, and has the methods that are called on the RPC interface. It is provided, paired with a data stream (a TLS Socket in our case) to the constructor of the server.

**Hibernate**

Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.

Hibernate's primary feature is mapping from Java classes to database tables (and from Java data types to SQL data types). Hibernate also provides data query and retrieval facilities. Hibernate generates the SQL calls and attempts to relieve the developer from manual result set handling and object conversion and keep the application portable to all supported SQL databases with little performance overhead.

Hibernate has been used in this project, so that the platform is unaware of the specific database server that holds all the data of the system, allowing the development of a portable and efficient application, that can run on compatible SQL databases. The database used to store all the data is a PostgreSQL, that provides TLS/SSL support and can hold large volumes of data.

PostgreSQL is an object-relational database management system (ORDBMS), used to store the most relevant information about the system and it's contents. A database is used, to allow for an easy data manipulation and fast retrieval of results. Only one element in the system is allowed to modify the database (the Tracker), and all the other elements have read only permissions. Also the database connection is protected using a TLS mechanism.
Each service in the system provides a secure socket with a JSON-RPC server, that handles access to the different methods this service has made public. This interface is used to communicate with the system, and to execute the necessary instructions for the correct work flow needed in the system.

6.1 Marker Interface

The Marker interface provides access to the mechanisms used to add and recover the mark of a movie.

- `create_product` method called when a new product is added to the content distribution platform, it has the following parameters:
  - `BigDecimal product_id`: ID of the product to create.
  - `BigDecimal item_id`: ID of the original item.
  - `long size`: Length of the item.
  - `String mark`: Mark to apply to the item.
  - `String server`: IP:port of the fileserver that holds the item.
  - `int frameDistance`: qdct between marked frames.
  - `int markDepth`: strength of the mark.
  - `String dctMatrix`: positions of the DCT to mark.
  - `String vcodec`: codec to use to encode the resulting product.
  - `returns Map<String, String>`: location of the marked frames.
• **recover_mark** used to call the fingerprinting algorithm and recover the original mark, this method has the following parameters:
  - `BigDecimal item_id`: ID of the original item.
  - `String upload_hash`: Hash of the marked product.
  - `String server`: IP:port of the fileserver that holds the item.
  - `int frameDistance`: qdct between marked frames.
  - `int markLength`: length of the mark.
  - `String dctMatrix`: DCT positions to recover the mark from.
  - `String markedFrames`: position of the marked frames.
  - returns `String`: recovered mark.

### 6.2 FileServer Interface

The FileServer interface provides access to the underlying filesystem in an abstract manner, is capable of serving items and products to other components of the system on request.

• **store_item** used to request the start of a new binary transfer to store the item that has been added to the system. The parameters of this method are:
  - `String id`: ID of the item to store.
  - `byte[] session_key`: key to use in the binary connection to transfer the item.

• **store_product** used to request the start of a new binary transfer to store the product that has been marked. The parameters of this method are:
  - `BigDecimal id`: ID of the product to store.
  - `byte[] session_key`: key to use in the binary connection to transfer the product.

• **download_item** used to request the start of a new binary transfer to download an item in order to create more products to distribute to users. The parameters of this method are:
  - `BigDecimal id`: ID of the item to retrieve.
  - `byte[] session_key`: key to use in the binary connection to transfer the item.

• **download_product** used to request the start of a new binary transfer to download the product that the user has requested. The parameters of this method are:
  - `BigDecimal id`: ID of the product to download.
  - `byte[] session_key`: key to use in the binary connection to transfer the product.

• **rename_item** used to request the rename of an item that has been added to the system. The parameters of this method are:
  - `String old_id`: old ID of the item to rename.
  - `BigDecimal new_id`: new ID of the item.
6.3 Tracker Interface

The Tracker interface provides access to the Tracker service, which is in charge of coordinating the whole system. The methods available are:

- **request_product** this method is called when a user requests a new product, and is in charge of assigning a product to the user and providing the binary transfer that would allow the user to download it. The parameters of this method are:
  - `BigDecimal user_id`: ID of the user that requested the item.
  - `BigDecimal item_id`: ID of the requested item.
  - `returns boolean`: success of the operation.

- **upload_item** method parameters:
  - `return List<String>`: returns the address of a fileserver to upload the item to.

- **upload_product** this method is called when a content provider wants to check for the presence of a mark inside an unknown product. It handles the upload of the movie to the storage servers and starts the recovery of the mark. The parameters of this method are:
  - `return List<String>`: returns the address of a fileserver to upload the product to.

- **add_item** this method is called when a content provider wants to upload a new item to the system. It handles the upload to the item to the storage servers and triggers the creation of new products for the recently added item. The parameters of this method are:
  - `BigDecimal user_id`: ID of the user adding the item.
  - `String name`: name of the item.
  - `String description`: short description about the item.
  - `Set<BigDecimal> categories`: set of categories the item belongs to.
  - `byte[] cover`: image of the cover.
  - `String server`: IP:port of the server where the temporary item resides.
  - `int price`: price of the item.
  - `int minStock`: minimum stock of products for this item.
  - `String tmp_id`: temporary ID of the item.
  - `BigDecimal size`: size of the uploaded item.
  - `returns BigDecimal`: ID of the new item.

- **edit_item** this method is called when a content provider wants to edit the information of an item already present on the system. The parameters of this method are:
  - `String hashId`: ID of the item to modify.
  - `String name`: new name of the item.
  - `String description`: new description of the product.
– Set<BigDecimal> categories: new categories the item belongs to.
– byte[] cover: new cover of the item.
– int price: new price of the item.
– int minStock: new minimum stock for this item.
– returns boolean: success of the operation.

• add_product this method is called when a marker wants to add a new product to the system (a product that has just been marked). The parameters are:
  – BigDecimal item_id: ID of the original item.
  – String mark: mark embedded to the product.
  – String server: server where the product is stored.
  – return boolean: success of the operation.

• find_dishonest_users this method is in charge of analyzing the mark in a product and obtaining the user or users that were assigned to this product. The parameters are:
  – String item_hash: ID the original item.
  – String product_hash: ID of the fraudulent copy.
  – String ipport: server where the upload product is stored.
  – return List<String>: list of dishonest users this copy belongs to.

• recalculate_stocks method parameters:
  – return boolean: success of the operation.
It is very important to detail the process that has to be followed to install this software, to ensure that possible uses or further development of this system by diverse users. The process requires of external software, to be installed prior to setting the system. All the components of the system where deployed in a FreeBSD 8.1-RELEASE amd64 standard installation using the Java JDK provided by the diablo-jdk 1.6 port\(^1\), running inside a Xen\(^2\) environment.

As a side note, a very good and in-depth guide about how to secure a FreeBSD installation can be found on [http://www.littlewhitedog.com/content-72.html](http://www.littlewhitedog.com/content-72.html) and it is really recommended to follow that guide before installing any other software if the base system is FreeBSD. Other guides about securing Linux and *BSD systems can be found over the Internet, and it is recommended to read and apply them before installing any other software on the system.

### 7.1 EJBCA

A certification entity is needed to generate the certificates and to provide the OCSP and CRL services. EJBCA is probably the most robust and versatile Open Source certification authority that provides all the needs of the system. It has a web interface that allows the user for an easy and intuitive administration.

The following components are needed to set up a working EJBCA instance:

- JDK 1.6
- Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files for your JDK.

\(^1\)[http://cvsweb.freebsd.org/ports/java/diablo-jdk16](http://cvsweb.freebsd.org/ports/java/diablo-jdk16)
\(^2\)[http://www.xen.org](http://www.xen.org)
• JBoss Application Server 5.1.x or later for JDK6.
• Apache Ant 1.7.1 or later to build.

Resources about how to install and configure JBoss can be found in the JBoss Community page [Coma] and specific installation guides for different distributions can be found in the official distribution page or on the Internet. The version used to deploy the test system was 5.1.0.GA, which is known to work fine with EJBCA 3.11.1 and Java JDK 1.6.

Once JBoss is installed we need to set up the Certification Authority, which is EJBCA in our test system. The EJBCA site has a very detailed and easy to follow guide about how to install EJBCA using JBoss [CP], This guide covers all the topics regarding the installation of EJBCA and also includes several topics about how to secure it successfully.

Once the system is set up, and the certificate SuperAdmin.p12 is installed on the browser the user can access the EJBCA administration pages. Usually the url of the main page is https://host:8443/ejbca, where "host" is the hostname or IP address where JBoss is listening. The browser will ask for a certificate to use, which in our case would be the "SuperAdmin" certificate. Once inside the administration zone, one of the first actions to perform is to create a new Certification Authority, (go to "Edit Certificate Authorities" which can be found on the menu located in the left margin of the page) input the desired name for your CA and click on "Create...". Fill in the form, and make sure to set the OCSP Service to "active", it is also recommended to set the key size to 2048.

Once the CA is up and running, it is recommended to create two certificate profiles, one to generate Server Certificates (used by the internal components of the system) and one to generate User Certificates (used to identify and allow users to browse our catalogue). To perform this action, go to "Edit Certificate Profiles" and add a new profile called "MyServer" (or any other name you choose, just make sure to remember which one is for users and which one is for servers). Once the Certificate Profile is added to the system, you will see it on the list, select it and click on "Edit Certificate Profile". You will be prompted with a form, the most important part of this process is to select the correct key usages, in our case, the server certificate need the following Key Usages:

• Digital Signature
• Non-repudiation
• Key encipherment
• Data encipherment
• Key agreement
• Key certificate sign

The Extended Key Usage should be set to:

• Server Authentication
• Client Authentication
Fill the rest of the form with the desired values and click on "Save". The next step is to create another Certificate Profile, this time for the users of the system and with the following Key Usages:

- Digital Signature
- Non-repudiation
- Key encipherment

And the Extended Key Usages:

- Client Authentication

The last step in this process is to create "End Entity Profiles", so you can select your newly created Certificate Profiles when adding an End Entity. Go to "Edit End Entity Profiles" and add a new one, fill the form and select the certificate profiles you have created before in the "Available Certificate Profiles" section. Now when you add a new End Entity for your CA you should see the two certificate profiles you have added to your system.

Now you should create at least one certificate for each component of the system, this is a list of the "Server Certificates" you need to create:

- Tracker Server
- Marker Server
- Fileserver Server
- PostgreSQL Server (Database)
- Web front-end (Tomcat)

To log in into the system you can use the SuperAdmin certificate if it belongs to the same CA as the other certificates, but it is recommended to create a new one with the user profile. Also, you should create a new "User Certificate" for each user that wants to log in into the system.

### 7.2 PostgreSQL

The PostgreSQL Database is widely used in many software projects and web pages around the world, and is one of the most known object-relational database management systems. Documents about how to install and configure PostgreSQL can be found on [Comb]. The version used in our test system is 8.4.7 and here we will explain how to configure it properly to support SSL connections. First of all we need the following files (all in PEM format):

- `root.crt`: certificate of the CA
• server.crt: certificate of the PostgreSQL server
• server.key: private key of the PostgreSQL server
• root.crl: certificate revocation list of the CA

The easiest way to obtain these files is to generate a certificate for the PostgreSQL entity in PKCS#12 format (p12 extension) and extract the private key and certificate from it using the commands in Listing 7.1.

```bash
$ openssl pkcs12 -nocerts -in mycert.p12 -out server.key
$ openssl pkcs12 -clcerts -nokeys -in mycert.p12 -out server.crt
```

Listing 7.1: Extract key and certificate from PKCS#12

The CA certificate and CRL can be downloaded directly in PEM format, so there’s no need to convert it. The permissions of all these files should be set to "0600" and the owner of the files has to be the same as the one running the PostgreSQL service. Place all the files in the PostgreSQL configuration folder, where the configuration files reside (pg_hba.conf, postgresql.conf).

It is also a good idea to add a small bash script to cron, to update the CRL periodically (it is recommended to update it at least once a day, but that depends on the configuration of EJBCA, and the refresh of the CRL). In Listing 7.2 the version of the script used in our test system can be seen.

```bash
#!/usr/local/bin/bash
wget -O /usr/local/pgsql/data/root.crl "http://EjbcaHost:8080/ejbca/publicweb/webdist/certdist?cmd=crl&format=PEM&issuer=CN%3dCPP%2cO%3dUPC%2cC%3dSE"
chown pgsql:pgsql /usr/local/pgsql/data/root.crl
/usr/local/etc/rc.d/postgresql restart
```

Listing 7.2: Fetch and reload PostgreSQL CRL

Note that PostgreSQL will be restarted during the process of updating the CRL, so use this script with caution.

Now it’s time to modify the PostgreSQL configuration files, start with postgresql.conf, and add the line displayed in Listing 7.3. PostgreSQL uses an authentication system based on a combination of host and authentication methods, so we will need to add a line to pg_hba.conf to allow outside connections to our PostgreSQL server as shown in Listing 7.4. Notice the "hostssl" prefix and the "clientcert=1" option, this makes connections from the 192.168.0.0/16 range only possible if using SSL and a client certificate.

```bash
ssl = true
hostssl all all 192.168.0.0/16 md5 clientcert=1
```

Listing 7.3: Add SSL support to PostgreSQL

Listing 7.4: Allow users to connect to the PostgreSQL server

There is only one small step to perform before finishing the installation, be should configure EJBCA properly to generate a new CRL periodically. To do this, log in into the EJBCA Administration zone
and go to "Edit Services", add a new service and configure it as a "CRL Updater". Don't forget to mark the box "Active" to run the updater.

Once the connections to the database work properly, create a new database called "cpp", and the user called "cppdb" with password "cppdb-pass" (if you want to change this parameters edit the file persistence.xml found in cpp-db/src/META-INF and recompile the applications). After this the tables needed for the system can be added, execute the Listing 7.5 that contains the SQL commands to create all the necessary tables.

```sql
CREATE SEQUENCE public.configuration_id_sqc;

CREATE TABLE public.CONFIGURATION (
    CONFIGURATION_ID NUMERIC NOT NULL DEFAULT nextval('public.configuration_id_sqc'),
    KEY VARCHAR NOT NULL,
    VALUE VARCHAR NOT NULL,
    LASTUPDATE_DT TIMESTAMP NOT NULL,
    CONSTRAINT configuration_id PRIMARY KEY (CONFIGURATION_ID)
);

ALTER SEQUENCE public.configuration_id_sqc OWNED BY public.CONFIGURATION.

CREATE UNIQUE INDEX configuration_ux1
    ON public.CONFIGURATION
    ( KEY );

CREATE SEQUENCE public.authority_id_sqc;

CREATE TABLE public.AUTHORITY (
    AUTHORITY_ID NUMERIC NOT NULL DEFAULT nextval('public.authority_id_sqc'),
    NAME VARCHAR NOT NULL,
    ROLE VARCHAR NOT NULL,
    CONSTRAINT authority_id PRIMARY KEY (AUTHORITY_ID)
);

ALTER SEQUENCE public.authority_id_sqc OWNED BY public.AUTHORITY.AUTHORITY_ID;

CREATE UNIQUE INDEX authority_ux1
    ON public.AUTHORITY
    ( ROLE );

CREATE SEQUENCE public.server_type_id_sqc;

CREATE TABLE public.SERVER_TYPE (
    SERVER_TYPE_ID NUMERIC NOT NULL DEFAULT nextval('public.server_type_id_sqc'),
    TYPE INTEGER NOT NULL,
    DESCRIPTION VARCHAR NOT NULL,
    CONSTRAINT server_type_id PRIMARY KEY (SERVER_TYPE_ID)
);
```
ALTER SEQUENCE public.server_type_id_sqc OWNED BY public.SERVER_TYPE.
  SERVER_TYPE_ID;

CREATE UNIQUE INDEX server_type_ux1
  ON public.SERVER_TYPE
  ( TYPE );

CREATE SEQUENCE public.server_id_sqc;

CREATE TABLE public.SERVER
  ( SERVER_ID NUMERIC NOT NULL DEFAULT nextval('public.
    server_id_sqc'),
   SERVER_TYPE_ID NUMERIC NOT NULL,
   IP VARCHAR NOT NULL,
   PORT INTEGER NOT NULL,
   NAME VARCHAR NOT NULL,
   ACTIVE BOOLEAN NOT NULL,
   BUSY BOOLEAN NOT NULL,
   CERT_SN VARCHAR NOT NULL,
   LASTUPDATE_DT TIMESTAMP NOT NULL,
   CONSTRAINT server_id PRIMARY KEY (SERVER_ID) );

ALTER SEQUENCE public.server_id_sqc OWNED BY public.SERVER.SERVER_ID;

CREATE UNIQUE INDEX server_ux1
  ON public.SERVER
  ( IP, PORT );

CREATE SEQUENCE public.category_id_sqc;

CREATE TABLE public.CATEGORY
  ( CATEGORY_ID NUMERIC NOT NULL DEFAULT nextval('public.
    category_id_sqc'),
   NAME VARCHAR NOT NULL,
   DESCRIPTION VARCHAR NOT NULL,
   LASTUPDATE_DT TIMESTAMP NOT NULL,
   CONSTRAINT category_id PRIMARY KEY (CATEGORY_ID) );

ALTER SEQUENCE public.category_id_sqc OWNED BY public.CATEGORYCATEGORY_ID;

CREATE UNIQUE INDEX category_ux1
  ON public.CATEGORY
  ( NAME );

CREATE SEQUENCE public.user_id_sqc;

CREATE TABLE public.USERS
  ( USER_ID NUMERIC NOT NULL DEFAULT nextval('public.user_id_sqc'),
   CN VARCHAR(255),
   MASTER BOOLEAN,
   SERIAL VARCHAR NOT NULL,
   STATE INTEGER NOT NULL,
   CONSTRAINT users_pkey PRIMARY KEY (USER_ID) );
ALTER SEQUENCE public.user_id_sqc OWNED BY public.USERS.USER_ID;

CREATE UNIQUE INDEX users_serial_key
ON public.USERS USING BTREE
( SERIAL );

CREATE TABLE public.USERS_AUTHORITY (  
    USER_ID NUMERIC NOT NULL,
    AUTHORITY_ID NUMERIC NOT NULL,
    CONSTRAINT users_authority_id PRIMARY KEY (USER_ID, AUTHORITY_ID)
);

CREATE SEQUENCE public.item_id_sqc;

CREATE TABLE public.ITEM (  
    ITEM_ID NUMERIC NOT NULL DEFAULT nextval('public.item_id_sqc'),
    AUTHOR_ID NUMERIC NOT NULL,
    SERVER_ID NUMERIC NOT NULL,
    NAME VARCHAR NOT NULL,
    DESCRIPTION VARCHAR NOT NULL,
    PRICE INTEGER NOT NULL,
    COVER BYTEA,
    SIZE INTEGER DEFAULT 0,
    MIN_STOCK INTEGER,
    RAW_MARK_SQC NUMERIC,
    HASHID VARCHAR NOT NULL,
    DCT_MATRIX VARCHAR,
    FCCN INTEGER,
    FCCR INTEGER,
    FRAME_DISTANCE INTEGER,
    CODEC VARCHAR,
    MARK_DEPTH INTEGER,
    CREATION_DT TIMESTAMP NOT NULL,
    LASTUPDATE_DT TIMESTAMP NOT NULL,
    CONSTRAINT item_id PRIMARY KEY (ITEM_ID)
);

ALTER SEQUENCE public.item_id_sqc OWNED BY public.ITEM.ITEM_ID;

CREATE TABLE public.ITEM_CATEGORY (  
    ITEM_ID NUMERIC NOT NULL,
    CATEGORY_ID NUMERIC NOT NULL,
    CONSTRAINT item_category_id PRIMARY KEY (ITEM_ID, CATEGORY_ID)
);

CREATE SEQUENCE public.product_id_sqc;

CREATE TABLE public.PRODUCT (  
    PRODUCT_ID NUMERIC NOT NULL DEFAULT nextval('public.product_id_sqc'),
    AUTHOR_ID NUMERIC NOT NULL,
    SERVER_ID NUMERIC NOT NULL,
    NAME VARCHAR NOT NULL,
    DESCRIPTION VARCHAR NOT NULL,
    PRICE INTEGER NOT NULL,
    COVER BYTEA,
    SIZE INTEGER DEFAULT 0,
    MIN_STOCK INTEGER,
    RAW_MARK_SQC NUMERIC,
    HASHID VARCHAR NOT NULL,
    DCT_MATRIX VARCHAR,
    FCCN INTEGER,
    FCCR INTEGER,
    FRAME_DISTANCE INTEGER,
    CODEC VARCHAR,
    MARK_DEPTH INTEGER,
    CREATION_DT TIMESTAMP NOT NULL,
    LASTUPDATE_DT TIMESTAMP NOT NULL,
    CONSTRAINT product_id PRIMARY KEY (PRODUCT_ID)
);
ITEM_ID NUMERIC NOT NULL,
MARK VARCHAR NOT NULL,
HASHID VARCHAR NOT NULL,
MARKED_FRAMES VARCHAR,
SIZE INTEGER,
USER_ID NUMERIC,
SERVER_ID NUMERIC NOT NULL,
CREATION_DT TIMESTAMP NOT NULL,
LASTACCESS_DT TIMESTAMP,
LASTUPDATE_DT TIMESTAMP NOT NULL,
CONSTRAINT product_id PRIMARY KEY (PRODUCT_ID)
);

ALTER SEQUENCE public.product_id_sqc OWNED BY public.PRODUCT.PRODUCT_ID;

ALTER TABLE public.USERS_AUTHORITY ADD CONSTRAINT authority_users_authority_fk FOREIGN KEY (AUTHORITY_ID)
REFERENCES public.AUTHORITY (AUTHORITY_ID)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.SERVER ADD CONSTRAINT server_type_server_fk FOREIGN KEY (SERVER_TYPE_ID)
REFERENCES public.SERVER_TYPE (SERVER_TYPE_ID)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.ITEM ADD CONSTRAINT server_item_fk FOREIGN KEY (SERVER_ID)
REFERENCES public.SERVER (SERVER_ID)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.PRODUCT ADD CONSTRAINT server_product_fk FOREIGN KEY (SERVER_ID)
REFERENCES public.SERVER (SERVER_ID)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.ITEM_CATEGORY ADD CONSTRAINT category_item_category_fk FOREIGN KEY (CATEGORY_ID)
REFERENCES public.CATEGORY (CATEGORY_ID)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.PRODUCT ADD CONSTRAINT users_product_fk FOREIGN KEY (USER_ID)
REFERENCES public.USERS (USER_ID)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;
ALTER TABLE public.ITEM ADD CONSTRAINT users_item_fk
FOREIGN KEY (AUTHOR_ID)
REFERENCES public.USERS (USER_ID)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.USERS_AUTHORITY ADD CONSTRAINT users_users_authority_fk
FOREIGN KEY (USER_ID)
REFERENCES public.USERS (USER_ID)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

ALTER TABLE public.PRODUCT ADD CONSTRAINT item_product_fk
FOREIGN KEY (ITEM_ID)
REFERENCES public.ITEM (ITEM_ID)
ON DELETE NO ACTION
ON UPDATE NO ACTION
DEFERRABLE INITIALLY IMMEDIATE;

ALTER TABLE public.ITEMCATEGORY ADD CONSTRAINT item_item_category_fk
FOREIGN KEY (ITEM_ID)
REFERENCES public.ITEM (ITEM_ID)
ON DELETE NO ACTION
ON UPDATE NO ACTION
NOT DEFERRABLE;

--- INSERT Basic configuration

INSERT INTO authority VALUES (50, 'Administrator', 'ROLE_ADMIN');
INSERT INTO authority VALUES (51, 'Guest', 'ROLE_GUEST');
INSERT INTO authority VALUES (52, 'User', 'ROLE_USER');
INSERT INTO authority VALUES (53, 'Uploader', 'ROLE_UPLOADER');

INSERT INTO configuration VALUES (50, 'marker.framedistance', '10', NOW());
INSERT INTO configuration VALUES (51, 'marker.markdepth', '40', NOW());
INSERT INTO configuration VALUES (52, 'marker.dctmatrix', '0,2;0,3;1,1;1,2;2,0;2,1;3,0', NOW());
INSERT INTO configuration VALUES (53, 'marker.vcodec', 'mpeg2video', NOW());
INSERT INTO configuration VALUES (54, 'marker.bitrate', '4000000', NOW());
INSERT INTO configuration VALUES (55, 'web.itemsPerPage', '16', NOW());
INSERT INTO configuration VALUES (56, 'tracker.fcctimeout', '5', NOW());

INSERT INTO server_type VALUES (50, 1, 'File Server');
INSERT INTO server_type VALUES (51, 0, 'Mux Server');
INSERT INTO server_type VALUES (52, 2, 'Tracker Server');
INSERT INTO server_type VALUES (53, 3, 'Web Server');

INSERT INTO users VALUES (150, 'SuperAdmin', true, '26ec63e92b91f4b2', 0);

Listing 7.5: Database schema

Figure 7.1 is a visual representation of the database, with all the connections that exist between the different fields and tables.
The last part of the SQL code in Listing 7.5 adds all the basic parameters needed for the system to work properly. The last line of this code adds the first user to the system and contains the serial number of the administrator certificate, remember to change this serial number to the one that you have installed in your browser, to be able to access the web page and add the necessary content and servers. The marker parameters can be change now, before insertion, but they can also be changed after the insertion, using the more friendly web front-end that has been developed.

7.3 Tomcat

Tomcat can usually be installed using the package manager provided with your distribution, and doesn't require much extra work to get running. In this guide we will assume that you have a working and properly configured Tomcat instance. The version used in your test system is Tomcat 7.0.6, and has been installed using the FreeBSD port system. The first step in properly configuring a Tomcat instance consists in placing all the necessary files in the configuration folder. We will need the certificate of the CA and the certificate to be used in Tomcat and in the web front-end, both of them in JKS format. The first file we will edit is `cataline.properties` and we will add the lines as shown in Listing 7.6. Remember to change the paths to the one used in your system. The last line in this listing has been commented, and can be used to debug SSL connection problems, uncomment it if you are experiencing issues with the SSL connections.

```java
javax.net.ssl.keyStore=/usr/local/apache-tomcat-7.0/conf/web.jks
javax.net.ssl.keyStorePassword=web-pass
javax.net.ssl.trustStore=/usr/local/apache-tomcat-7.0/conf/CA.jks
javax.net.ssl.trustStorePassword=ca-pass
# javax.net.debug=ssl.handshake
```

Listing 7.6: Set Tomcat KeyStore and TrustStore
To ease the deploying and managing of running applications, Tomcat provides an Application Manager, to active it, edit the `tomcat-users.xml` file and add the lines in Listing 7.7, setting the password correctly.

```xml
<role rolename="manager-gui"/>
<user username="tomcat" password="xxxxxxx" roles="manager-gui"/>
```

Listing 7.7: Add a Tomcat manager user

The manager interface can be reached using the URL [http://tomcatHost:8080/manager/](http://tomcatHost:8080/manager/). Now the only remaining steps are to configure the SSL connector on port 8443 and to deploy the applications. Configuring an SSL socket in Tomcat is easy, the only steps necessary are to open the `server.xml` file and add the line displayed in Listing 7.8 inside the "Service" section.

```xml
<Connector SSLEnabled="true" clientAuth="want" keystoreFile="${catalina.base}/conf/cp.jks" keystorePass="cp-pass" maxThreads="150" port="8443" protocol="HTTP/1.1" scheme="https" secure="true" sslProtocol="TLS" truststoreFile="${catalina.base}/conf/CA.jks" truststorePass=""/>
```

Listing 7.8: Set up an SSL connector on port 8443

Remember to change the keystoreFile, keystorePass, truststoreFile and truststorePass to the ones that match your configuration. Once you have modified all the files you can restart Tomcat and proceed with the deployment of the system.

Now the Tomcat server is ready to support the CPP-WEB and CPP-CONFABULATOR applications. To ease the deployment of this applications two WAR files are provided, that just need to be uploaded to the Tomcat Server using the manager interface, usually at: [http://tomcatHost:8080/manager/](http://tomcatHost:8080/manager/).

For the deployment to be successful the PostgreSQL and EJBCA servers need to be active and reachable from the Tomcat instance.

### 7.4 Deployment of the system components

To allow for an easy and painless installation the system uses a predefined set of hosts names, which you should add to your `/etc/hosts` file. This hostnames are the places where the PostgreSQL and EJBCA reside. In Listing 7.9 is an example of the lines to be added to the `/etc/hosts` file if EJBCA and PostgreSQL are listening on the IP address 192.168.1.122. This hostnames can also be changed inside the xml configuration files of Hibernate.

```plaintext
192.168.1.122 postgres
192.168.1.122 ejbca
```

Listing 7.9: `/etc/hosts`

All the services that conform the system come packed in a JAR file[^3] that contains all the specific code of the application. Libraries are not packed inside the JAR, because most services share the same libraries, so a single folder can contain all the needed libraries for all the services, thus avoiding duplication of libraries and easy library management.

Java Service Wrapper

In our system, we have used Java Service Wrapper\(^4\) to run the JARs as native Unix services (daemons), this has several advantages compared to the normal launch of the JAR file:

- Services are started at boot time
- Services are restarted if an error occurs
- Services can be easily controlled using the standard daemon manager tools (update-rc.d, rc.conf...)

Creating and executing Java applications using the Java Service Wrapper requires some modifications, and a configuration file. The first step to accomplish this task is to generate all the JARs for your services, and place them in a folder together with the necessary runtime libraries. It is also necessary to download a Java Service Wrapper appropriate for your Operating System/Distribution.

The downloaded Service Wrapper comes with a native library, called libwrapper.so on *nix systems, we placed this library in /usr/local/lib/javaservicewrapper/lib. We also placed all the libraries and application inside /usr/local/cpp-common/lib. The next step is to create a different folder for each application that you want to launch inside /usr/local, in our case we need three folders, marker, tracker and fileserver. Create a bin and a conf folder inside each one and copy the wrapper executable in the bin folder. Rename this executable to the name of the service (for example the marker executable should be in /usr/local/marker/bin/marker).

Also a configuration file should be created for each service, a template is provided in Listing 7.10, notice that all the libraries have been added to the classpath, and that only the set.APP and probably some paths need to be changed in order for the wrapper to work correctly.

\[^4\]http://wrapper.tanukisoftware.com/
# The following property will output information about which License Key(s)
# are being found, and can aid in resolving any licensing problems.
#wrapper.license.debug=TRUE

set.APP=marker

# Wrapper Localization
# Specify the locale which the Wrapper should use. By default the system
# locale is used.
#wrapper.lang=en_US # en_US or ja_JP

# Specify the location of the Wrapper’s language resources. If these are
# missing, the Wrapper will default to the en_US locale.
wrapper.lang.folder=../lang

# Wrapper Java Properties
# Java Application
# Locate the java binary on the system PATH:
wrapper.java.command=/usr/local/bin/java

# Specify a specific java binary:
# set.JAVA_HOME=/java/path
# wrapper.java.command=%JAVA_HOME%bin/java

# Tell the Wrapper to log the full generated Java command line.
#wrapper.java.command.loglevel=INFO

# Java Main class. This class must implement the WrapperListener interface
# or guarantee that the WrapperManager class is initialized. Helper
# classes are provided to do this for you. See the Integration section
# of the documentation for details.
wrapper.java.mainclass=org.tanukisoftware.wrapper.WrapperSimpleApp

set.APP_HOME=/usr/local.marker
set.COMMON_DIR=/usr/local/cpp-common/
wrapper.working.dir=%APP_HOME%

# Java Classpath (include wrapper.jar) Add class path elements as
# needed starting from 1
wrapper.java.classpath.1=%COMMON_DIR%/lib/wrapper.jar
wrapper.java.classpath.2=%COMMON_DIR%/lib/%APP%.jar
wrapper.java.classpath.3=%COMMON_DIR%/lib/antlr-2.7.6.jar
wrapper.java.classpath.4=%COMMON_DIR%/lib/bcprov-jdk15-138.jar
wrapper.java.classpath.5=%COMMON_DIR%/lib/commons-collections-3.1.jar
wrapper.java.classpath.6=%COMMON_DIR%/lib/d5m4j-1.6.1.jar
wrapper.java.classpath.7=%COMMON_DIR%/lib/hibernate-jpa-2.0-api-1.0.0.Final.jar
wrapper.java.classpath.8=%COMMON_DIR%/lib/hibernate-testing.jar
wrapper.java.classpath.9=%COMMON_DIR%/lib/hibernate3.jar
wrapper.java.classpath.10=%COMMON_DIR%/lib/jackson-core-asl-1.6.6.jar
wrapper.java.classpath.11=%COMMON_DIR%/lib/jackson-mapper-asl-1.6.6.jar
wrapper.java.classpath.12=%COMMON_DIR%/lib/javaxassist-3.12.0.GA.jar
wrapper.java.classpath.13=%COMMON_DIR%/lib/jta-1.1.jar
wrapper.java.classpath.14=%COMMON_DIR%/lib/junit.jar
wrapper.java.classpath.15=%COMMON_DIR%/lib/jyaml-1.1.jar
wrapper.java.classpath.16=%COMMON_DIR%/lib/log4j-1.2.16.jar
wrapper.java.classpath.17=%COMMON_DIR%/lib/org.hamcrest.core_1.1.0.v20090501071000.jar
wrapper.java.classpath.18=%COMMON_DIR%/lib/postgresql-8.4-702.jdbc4.jar
wrapper.java.classpath.19=%COMMON_DIR%/lib/slf4j-api-1.6.1.jar
wrapper.java.classpath.20=%COMMON_DIR%/lib/cpp-utils.jar
wrapper.java.classpath.21=%COMMON_DIR%/lib/cpp-protocol.jar
wrapper.java.classpath.22=%COMMON_DIR%/lib/cpp-securesocket.jar
wrapper.java.classpath.23=%COMMON_DIR%/lib/cpp-db.jar
wrapper.java.classpath.24=%COMMON_DIR%/lib/c3p0-0.9.1.2.jar
wrapper.java.classpath.25=%COMMON_DIR%/lib/cpp-fcc.jar
wrapper.java.classpath.26=%COMMON_DIR%/lib/cpp-ffmpeg.jar

# Java Library Path (location of Wrapper.DLL or libwrapper.so)
wrapper.java.library.path.1=/usr/local/lib/javaservicewrapper/lib

# Java Bits. On applicable platforms, tells the JVM to run in 32 or 64-bit mode.
wrapper.java.additional.auto_bits=TRUE

# Java Additional Parameters
wrapper.java.additional.1=

# Initial Java Heap Size (in MB)
# wrapper.java.initmemory=256

# Maximum Java Heap Size (in MB)
wrapper.java.maxmemory=512

# Application parameters. Add parameters as needed starting from 1
wrapper.app.parameter.1=cpp.marker.Main

# Wrapper Logging Properties

# Enables Debug output from the Wrapper.
# wrapper.debug=TRUE

# Format of output for the console. (See docs for formats)
wrapper.console.format=PM

# Log Level for console output. (See docs for log levels)
wrapper.console.loglevel=INFO

# Log file to use for wrapper output logging.
wrapper.logfile=/var/log/marker.log

# Format of output for the log file. (See docs for formats)
wrapper.logfile.format=LPTM

# Log Level for log file output. (See docs for log levels)
wrapper.logfile.loglevel=INFO

# Maximum size that the log file will be allowed to grow to before
# the log is rolled. Size is specified in bytes. The default value
# of 0, disables log rolling. May abbreviate with the 'k' (kb) or
# 'm' (mb) suffix. For example: 10m = 10 megabytes.
7.4. DEPLOYMENT OF THE SYSTEM COMPONENTS

wrapper.logfile.maxsize=200k

# Maximum number of rolled log files which will be allowed before old
# files are deleted. The default value of 0 implies no limit.
wrapper.logfile.maxfiles=0

# Log Level for sys/event log output. (See docs for log levels)
wrapper.syslog.loglevel=NONE

#********************************************************************
# Wrapper General Properties
#********************************************************************
# Allow for the use of non-contiguous numbered properties
wrapper.ignore_sequence_gaps=TRUE

# Title to use when running as a console
wrapper.console.title=@app.long.name@

#********************************************************************
# Wrapper JVM Checks
#********************************************************************
# Detect DeadLocked Threads in the JVM. (Requires Standard Edition)
wrapper.check.deadlock=TRUE
wrapper.check.deadlock.interval=60
wrapper.check.deadlock.action=RESTART
wrapper.check.deadlock.output=FULL

# Out Of Memory detection.
# (Simple match)
wrapper.filter.trigger.1000=java.lang.OutOfMemoryError
# (Only match text in stack traces if -XX:+PrintClassHistogram is being used.)
wrapper.filter.trigger.1000=Exception in thread "" java.lang.OutOfMemoryError
wrapper.filter.allow_wildcards.1000=TRUE
wrapper.filter.action.1000=RESTART
wrapper.filter.message.1000=The JVM has run out of memory.

#********************************************************************
# Wrapper Email Notifications. (Requires Professional Edition)
#********************************************************************
# Common Event Email settings.
#wrapper.event.default.email.debug=TRUE
#wrapper.event.default.email.smtp.host=<SMTP_Host>
#wrapper.event.default.email.smtp.port=25
#wrapper.event.default.email.subject=[%WRAPPER_HOSTNAME%:%WRAPPER_NAME%:%
#WRAPPER_EVENT_NAME%] Event Notification
#wrapper.event.default.email.sender=<Sender email>
#wrapper.event.default.email.recipient=<Recipient email>

# Configure the log attached to event emails.
#wrapper.event.default.email.attach_log=TRUE
#wrapper.event.default.email.maillog.lines=50
#wrapper.event.default.email.maillog.format=LPTM
#wrapper.event.default.email.maillog.loglevel=INFO

# Enable specific event emails.
#wrapper.event.wrapper_start.email=TRUE
#wrapper.event.jvm_prelaunch.email=TRUE
Listing 7.10: Java Service Wrapper configuration template

Inside the conf folder there should also be the certificate of the CA (CA.jks) and the unique certificate of each service (marker.jks, tracker.jks...) and the configuration file of the service itself (config.yml). Listing 7.11 contains an example of the config.yml file of the marker service (notice that slibPath should only be set if the service is a marker). An example of each service config.yml is supplied with it’s source code.

```yaml
publicPort: 9801
maxClients: 5
keyStore: conf/marker.jks
keyStorePass: marker-pass
trustStore: conf/CA.jks
```
7.4. DEPLOYMENT OF THE SYSTEM COMPONENTS

```yaml
trustStorePass: ca-pass
tmpDir: /opt/cpp-files/marker-tmp
ocspUrl: "http://ejbca:8080/ejbca/publicweb/status/ocsp"
slibPath: "/usr/local/marker/cpp-ffmpeg-watermark/
```

Listing 7.11: Marker Service config.yml

To install the services as daemons, execute the launcher with the argument "install", in the case of the marker, we will execute the code in Listing 7.12 as root. After that, the service can be started and stopped as any other service in the Operating System.

```
# /usr/local/marker/bin/marker install
```

Listing 7.12: Install Marker service

Before starting the services you should check that they can connect to the EJBCA and PostgreSQL database, if there is no connection to the database the services will refuse to start. Also the Java Service Wrapper provides a very useful log file inside the default log folder (usually /var/log).
The work done in this project has proven that it is feasible to create a fast implementation of a modified Cox Algorithm for marking MPEG2 videos. Also the platform of distribution of this marked videos has proven to be the best option when it comes to sharing movies over the Internet.

One of the key factors that made possible this project is that every technology used in it is clearly studied and defined before being implemented, and all the possible options are taken into account before making a choice. This allowed us to design a system that was balanced, and to build it with the right tools from the start.

The tests that where performed with the marking process clearly demonstrate that the algorithm used is robust and can resist several attacks, and the only option to remove the embedded mark is to perform attacks that degrade the image quality so much that render the movie unwatchable. Also the PSNR of embedding a mark has been demonstrated to be tolerable, and that even movies marked with a strong factor are watchable.

A reliable watermarking, recovery and confabulation algorithms where implemented based on the FFmpeg project, together with a content platform distribution capable of providing a simple downloading service for users. The use of both technologies together allowed the creation of a reliable and easy to use online video download service, that allows the tracing of unauthorized copies back to the original distributor of the fraudulent copy.

The front-end of the system, where the users access to browse the online movie catalogue is secured using SSL client certificates, that the client uses to authenticate with the platform. Also, the internal connections that happen between the distributed components of the platform are secured, using SSL client and server certificates. With this, the possibility of rogue servers being added to the system is dissipated, and if a service of the platform is compromised it can be easily isolated from the rest, just by revoking it's certificate.

With the system developed during this project we have proven that a secure, yet highly usable video
distribution platform is reliable.

8.1 Contributions to FFmpeg

During the implementation of the mark embedder and detector some changes were committed to the FFmpeg project, to be added to improve the code of the FFmpeg project. After the approval of this patches, they were added to the main FFmpeg source code, and can now be found at [PM11a] and [PM11b].

This patches include modifications to detect the type of the frame that is passed to a chain of video filters. This modification was needed in order to detect and mark only I-Frames, instead of marking all the frames. Although FFmpeg video filters are a quite new feature, some other filter authors have also benefited from this improvement.

8.2 Future topics of research

This chapter presents proposals to improve system capacity, these proposals are thought in a way that they would benefit the system, increasing its functionality, stability, scalability and robustness. Thus, there are two major proposals: the first will provide new features while the second aims to provide stability, scalability and robustness of the platform.

Marking products with H.264

The MPEG2 compression standard which the system uses has been replaced in most areas in favor of efficient codecs that provide higher quality with a lower bit rate. Amongst these formats, the H.264 codecs have become really popular, this format is developed by the ITU-T as a variant of MPEG-4 format. This new codec is predominant regarding online video publishing, as it can be reproduced from Flash components or directly if the browser itself supports the new HTML5 standard.

Due to this evolution in compression formats and the advantages they offer for online content distribution, it would be interesting to study the possibility of adapting the marking process performed by the platform to this new format. Thus, it would be possible to offer the users of the content distribution platform the option of downloading the video in higher quality and also play it without problems in various devices that support HTML5 and H.264, enabling the use of the platform in smart phones and tablets.

Because the H.264 coding differs greatly from the MPEG2 standard it is necessary to carry out some research, in order to study what's the best way to embed the marks in H.264 streams to ensure the robustness and viability of the system. Having examined the migration and if found possible, a new marker could be implemented, capable of making videos in this format.

Finally, a new option to reproduce the videos directly from the browser should be added to the platform, this option is possible because the new HTML5 standard allows the playback of multimedia
content directly from the browser. This solution would be suitable for users using a mobile device, a smart phone, to interact with the platform.

**Cloud Architecture**

The current design of the platform has some restrictions regarding its deployment and scalability due to the nature of the services that comprise it. It also true that the current design meets all the safety requirements, and they shall be maintained in the new proposed architecture.

The evolution of technologies in the field of cloud computing and the requirements of the software that can be build based on this architecture, make this project a good candidate to be adapted to the cloud architecture.

What characteristics does a cloud platform meet? Why are beneficial for this platform? And what changes are required to the current platform? Well, the first step is to define the objective, knowing the characteristics and the benefits of the implemented services and the structure of a cloud environment. The following is a list of the main features of cloud platforms:

- **Scalability**: ability to change the morphology of the platform with a very low cost. Platforms that are not designed this way are rigid and static. The initial deployment is not easy to change, and the variability of services is low or absent, and in any case, changes are costly.

- **Interoperability**: services are independent of the platform on which they are deployed. Interoperability is one of the main features that a platform must ensure to become successful.

- **Standardization**: the processes of packaging and deploying the services should be standardized, so that is not being tied to proprietary systems that do not allow the needed flexibility. The speed of the deployment and startup of the services is one of the strengths of the cloud environment.

- **Administration**: a cloud platform has at his disposal a large number of distributed services. The management of these resources is necessary for the proper functioning of the platform.

- **Reliability**: a platform with distributed replicated services is much more reliable than one with a rigid and centralized services, obviously ensuring their availability through management mechanisms. Also services that are in a non-functional state, should not disturb the services that are running normally.

- **Security**: security is a derived characteristic of this architecture, since it is not possible or advisable to have a distributed data management. In fact may even be considered a negative factor, there is a natural centralization that guarantees more reliable safety. On the other hand, it should be studied what information and how it travels over the cloud.

- **Accessibility**: services in the cloud are more accessible than the old static infrastructure, the architecture requires clearly defined interfaces and how they are published, resulting in a more transparent and accessible structure. On the other hand, we must pay special attention to safety, to prevent denial of service attacks, phishing, etc.
Given these characteristics, it is obvious that some of them are extremely beneficial to any system, but they do not apply to any context, and is therefore needed to identify the benefits this characteristics bring to our specific platform:

- **Scalability**: the platform for multimedia content management is designed as a set of services that work together to accomplish one goal, the system has multiple instances of the same service to ensure availability and scalability. While services are designed to provide low coupling and high cohesion, with this scenario the benefits of a system with high scalability is very important.

- **Interoperability**: for the platform we are deploying it is recommended to have a private architecture topology. In this case, services are not deployed in a public setting, but in a controlled environment, with known clients. This architecture does not force the definition of services, letting developers use different programming languages which ensure interoperable interfaces.

- **Standardization**: the benefits of a standards-based design are endless and in our case standardization should facilitate the deployment and implementation of new services, reducing effort and cost.

- **Administration**: it is a requirement derived from the distributed architecture, with many services, there must be a mechanism for managing them to successfully administrate them in a comprehensive way. Otherwise, we can have multiple replicated services, none of them being active.

- **Reliability**: derived from the previous point, the system will be more reliable if multiple replicated services are available. The platform may have multiple markers, the marking process is costly in time and resources, and distributing the computational cost of this process can be extremely beneficial to the system and end users, which will spend less time waiting for new products to be available.

- **Security**: it is obvious that the management of the data should be centralized, possibly through the Tracker Server, which is unique in the system. Certainly the system should benefit from the security features of having the platform in the cloud, regarding this matter, it is necessary to define a private cloud and a public cloud, and the interconnection between them. Special consideration to other inherent safety requirements considered in the current platform should be taken into account, the use of certificates and encrypted communications are one example.

- **Accessibility**: the current platform has an object serialization mechanism for the communication of the various services that conform the system. This object serialization mechanism continues to add limitations in the design and scalability, the services must agree at the design phase which interfaces are shared and must have the same version of the objects to be exchanged. In an ideal environment, public services publish their interface, and the clients (other services) may be adapted to these new interfaces without requiring anything else. While it is true that the number of serialized objects will not evolve out of control, designing services with this in mind will allow greater accessibility as appropriate standard protocols are used for the exchange of messages in distributed environments.
8.2. FUTURE TOPICS OF RESEARCH

- **Costs:** during the workflow of the existing platform, there are times when it is necessary to use a vast amount of computing resources, particularly in the process of refilling the stocks. This situation will occur with a lower frequency compared to processes such as downloading content. In this sense, the cost model associated with the paradigm of cloud computing is well suited to the business model of the platform, the occasional moments where it requires more computing power coincide with sales and therefore, there is a linear dependence between operating costs and system turnover.

How should the current platform evolve in order to take advantage of the characteristics defined in a cloud computing environment? Here are some aspects that should be taken into account for the correct development of the platform to suit a cloud computing deploying:

- **Standalone service migration to Web services:** web services, are a key tool for development-oriented software services (SaaS, Software as a Service). One of the initial actions should be the migration of existing services to Web Services, this includes the Tracker, File Server and Marking service, maintaining the existing security features.

- **Development of a global service management:** service management is a key point for the platform, and is required to achieve success. A management system that covers the publication and monitoring of services is needed in this kind of environments.

- **Data centralization:** in the security section, we stated that there must be a single repository of information, ideally the service should not access the repository as it may be private and be guarded by strict security rules. Thus, a redesign of the distribution and access to information should be done to accommodate the new topology of services.

- **Adjustment distributed environment:** it is possible that the logic implemented in current services, as well as the design of activity diagrams is not valid for a fully distributed environment with multiple instances of each service. It should be studied, for each service, the load balancing in an environment with multiple instances of the same service, so that they can live without collision and that the workload is correctly distributed amongst all instances. Also the reliability of the system when there are unavailable instances should be taken into account. This can lead to the implementation of synchronization mechanisms, which by nature the paradigm of cloud computing already incorporates.
BIBLIOGRAPHY


[PM11a] Roger Pau Monné. Add key_frame and pict_type to avfilterbufferrefvideo. http://git.videolan.org/?p=ffmpeg.git;a=commit;h=09a1416db78f49fb3a39cdf5a4bd0f5c5f849ce0, 2011.


