Computing and Informatics, Vol. 35, 2016, 1189–1209

# COPYRIGHT PROTECTION FOR DIGITAL IMAGES ON PORTABLE DEVICES

Raffaele Pizzolante, Bruno Carpentieri

*Dipartimento di Informatica*
*Università degli Studi di Salerno*
*I-84084, Fisciano (SA), Italy*
*e-mail:* rpizzolante@unisa.it, bc@dia.unisa.it

**Abstract.** The astonishing rapid diffusion of portable devices (i.e. smartphones, tablets, etc.) has had a big, and often positive, impact on our every-day life. These devices have new advanced features developed specifically because of user demand. For example, it is now possible to publish directly the pictures obtained by means of the internal camera of a smartphone on our social network accounts, or on an image hosting service. It is therefore important to have tools, on the portable devices, that can prove the ownership of the pictures and to use them before publishing images. Digital watermarking techniques are commonly used for the copyright protection of images and videos. We have developed a tool for portable devices based on the Android OS that allows the embedding of a digital visible or invisible watermark into a digital image.

**Keywords:** Portable device security, digital watermarking, copyright protection, invisible digital watermarking, visible digital watermarking

## 1 INTRODUCTION

Digital watermarking involves hiding data into a digital object to protect its copyright. In these days there is a growing interest in digital watermarking, because of the huge digital information explosion that internet and multimedia are bringing to our society. The digital watermarking techniques are certainly part of a possible solution for the copyright problems of digital media. Watermarking meets requirements that are desirable for copyright protection such as invisibility, unchanged compressibility, high detection reliability, low cost, robustness and security (see [14]). The real

problem is that the security and robustness requirements depend on the application domain and there is no "general" watermarking technique that is suitable to solve copyright issues in all the possible application domains.

During the last few years portable devices, such as tablets and smartphones, have increased their potentialities in terms of hardware (i.e. display resolution, processor speed, etc.) and software. They are widely used by different typologies of end-users, therefore the built-in operating systems are designed to allow different personalization of the user interface in order to permit an efficient usage of the devices. It is easy to install third-party applications directly on the devices through dedicated online stores. Portable devices are currently used for a wide range of purposes: for example for internet browsing, for sending and receiving emails, Multimedia Message Service (MMS) messages, and for other different services. It is also possible to directly publish the photos that we obtain by means of the internal camera of our portable device on online social networks, image hosting services, and so on.

These latter scenarios have prompted us to focus our attention to the security problems related to the images obtained via portable devices. An important aspect related to portable devices security is to guarantee the ownership of an image, before publishing or sending the image to another party. The common solution is to embed in the image hidden information related to the producer through the digital watermarking techniques.

There are two types of digital watermarking approaches: visible or invisible watermarking. If the watermarking is visible then a logo or a text string is visibly embedded in the image, and it is a visible proof of copyright or ownership. Usually the watermark is a text or a simple logo, which identifies the owner of the multimedia data. As, for example, a TV channel is identifiable through its logo. In this case the logo can be considered as visible watermark which provides information on the ownership to the end-user.

Often this typology of watermark is used for pictures and videos on the web. In this scenario the owner, through the visible watermark, immediately informs the end-user of the property information regarding the digital object. Invisible watermarking instead applies modifications to a few small features of the multimedia data. In this case the logo or text information are not visible to the end-user but the owner can prove the ownership of the image through an algorithm which extracts the invisible watermark. There are many methods for embedding the invisible information (for example the spread-spectrum methods and the amplitude methods). In the next Section we will briefly review some of them. In this paper, we present a tool for portable devices which allows the embedding of a visible or an invisible digital watermarking into an image. The proposed tool is developed for Google Android OS [5] and it takes as input a picture and a string, and the watermarked image gives as output. This paper is organized as follows: in Section 2 we review the backgrounds of digital watermarking. Section 3 presents and describes our Android-based digital watermarking tool. Section 4 reports the simulation results achieved by our tool and Section 5 draws our conclusions and highlights future work directions.

## 2 DIGITAL WATERMARKING ON IMAGES

A digital watermark might be a logo or a text message or in general a bit sequence, that is permanently embedded into the digital image and that should remain present within the data after any standard manipulation of the image itself (i.e. compression, scaling, etc.).

Formally, a digital watermark $W$ can be defined as in Equation (1).

$$W = \langle w_0, w_1, \ldots, w_{N-1} \rangle \tag{1}$$

where $w_i \in \{0, 1\}$ and $0 \leq i \leq N - 1$.

Image watermarking therefore involves hiding data into a digital object to protect its value. A watermark can be visible or perceptually transparent (invisible), it can be fragile and always break down if the image is altered or robust and resists to alterations (when the alterations do not significantly reduce the value of the image itself).

The image watermarking techniques could be divided into two groups, depending on the fact that the watermark is embedded directly in the original pixels of the image (Spatial Domain techniques) or that the embedding is done by using a transformed version of the image (Frequency Domain techniques) [16].

Examples of Spatial Domain techniques are the statistical labeling method presented by Bender et al. in [7] ("Patchwork") and by Pitas and Kaskalis in [19].

A watermarking algorithm in the Spatial Domain, derived by the algorithm in [19], is presented by Langelaar et al. in [11] and [12]. The goal is to provide a copy protection system that has the capacity of embedding a watermark up to a few hundreds bits. One important advantage of this method is that it is a blind watermarking scheme. In fact, it is possible to extract the embedded watermark without using the original, unlabeled, image.

In [11] and [12], the image is segmented in fixed size, $8 \times k$ times $8 \times k$ blocks, for a chosen constant $k$ (the blocks dimensions are multiples of 8, to make the algorithm robust under JPEG compression). Each watermark bit is embedded in a pseudo-randomly selected block of luminance values. The bit embedding is achieved by randomly dividing the pixels in the block into two subsets and by modifying the luminance values so that the difference between the average luminance in the two subsets determines the value of the inserted bit. The advantage of this method is that there is no need of the original image to extract the watermark, the disadvantage is that the method is not as robust as desirable: it has been experimentally shown that this watermarking technique is not resistant if the image undergoes geometric transformation (rotation, translation, scaling, etc.) or cropping.

An efficient watermarking algorithm in the Frequency Domain is presented in [8]. The watermark, a sequence of $n$ real numbers, is embedded in the image by computing the Discrete Cosine Transform (DCT) of the image and by adding the watermark to the $n$ highest magnitude coefficients of the transform matrix (exluding the most important). These $n$ coefficients are the perceptually significant coefficients. To

extract the watermark, the DCT transform of the original image is subtracted from the DCT transform of the watermarked image and the watermark is recovered from the highest coefficients.

A disadvantage of this technique is that it is a non-blind method. In other words, the original image is always needed to extract the watermark and this limits the field of applications of this method. For example it cannot be used for metadata tagging. However, for copyright protection, this is considered as one of the most robust and secure methods available. Other approaches based on the spread-spectrum techniques [23, 13] consist in different additive modification on the image spectral signal. The quantization technique [10] obtains the marked signal by quantizing the original spectral signal.

### 2.1 Watermark Attacks: StirMark

There are two main objectives of a watermark attack. First, an attack tries to defeat the digital watermarking algorithm and, second, it tries to maintain the perceptual quality of the attacked image. Hence, the robustness of a watermarking scheme can be evaluated by simulating the most common attacks. In order to evaluate the similarities between the original watermark $W$ and the watermark $W^*$ extracted from the watermarked image, we considered the following similarity measure [8]:

$$\mathrm{sim}(W, W^*) = \sum_{i=0}^{N-1} \frac{w_i^* \cdot w_i}{\sqrt{w_i^* \cdot w_i^*}} \tag{2}$$

where $N$ is the length of the original watermark $W$ and of the extracted watermark $W^*$.

It is possible to simulate attacks by using a benchmark suite, as for example StirMark [18, 17]. StirMark applies different image manipulations including lossy compression, different filters, geometrical transformations, etc., in order to test the robustness of the watermark when a watermarked image is altered. The following subsections shortly describe the most common attack techniques that are included in StirMark.

#### 2.1.1 Lossy Compression

The lossy compression algorithms for still images alter the original image by modifying the image quality in a way that is not generally perceptible by the human visual system. One of the most known and used algorithms is the Joint Photographic Experts Group (JPEG) algorithm. JPEG allows to set the quality factor, generally expressed in percentage. The percentage of the quality factor is directly proportional to the perceived quality of the compressed image and, in general, it is inversely proportional to the compression ratio.

JPEG can be considered also as an attack to the digital watermarking schemes, in fact, the information that is lost to obtain the desired compression factor could

contain essential parts of the watermark string. StirMark simulates the behavior of the JPEG algorithm, by using different quality factors. In this way it is possible to analyze the robustness against lossy compression of the watermarking technique.

### 2.1.2 Median Filter

The median filter is a non-linear spatial filter [9] commonly used for removing the noise from an image by preserving its edges. Each pixel $I(x, y)$ is replaced with the median value of the neighboring pixels, which are generally contained in a $n \times n$ pattern, denoted as *window* $N_{xy}$ (see Figure 1 for an example).

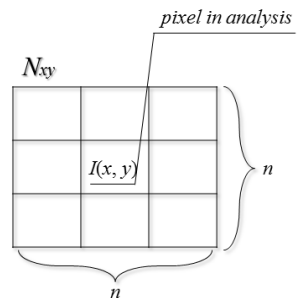


Figure 1. An example of $3 \times 3$ window, $N_{xy}$, for the pixel $I(x, y)$

The computation of the median consists into the sorting of the neighboring pixels intensities (including the intensity of the pixel in analysis) and then the median value is the output value.

In mathematical terms, we can define the median filter as defined in Equation (3).

$$I'(x, y) = \text{median}\left\{I(u, v) \mid \forall I(u, v) \in N_{xy}\right\}. \tag{3}$$

The median filter can be interpreted as a watermark attack, since the filtered image obtained loses information and the information lost could partially contain the embedded string.

### 2.1.3 Scaling

The scaling/rescaling image transformation, a geometric transformation, is widely used for the subsampling and the upsampling of an image.

Subsampling permits to reduce the resolution of the original image: for example to produce a thumbnail of the image. Upsampling permits to enlarge the resolution of the original image.

Depending on the scaling factor the scaling transformation produces a subsampled or a upsampled image with respect to the original image.

This process alters the image. Therefore, if the image embeds an invisible watermark, the embedded watermark could be altered too.

## 3 A NEW ANDROID-BASED TOOL

The tool we have developed is based on Google Android OS [5] (2.2 or later). The tool permits to select as input a picture that is stored into the portable device to embed a watermark, and then to indicate the output path where the image affected by the watermark will be stored.

The end-user can select one of the two main options that define the methodology of the digital watermarking that shall be embedded into the image:

- digital visible watermarking, or
- digital invisible watermarking.

A preliminary version of this tool was presented in [20]. In this paper we have enhanced the tool presented in [20] by using a more robust procedure for the embedding, maintaining the computational complexity unaffected. The preliminary version of the embedding procedure is also used in [1, 2].

### 3.1 Digital Visible Watermarking

If the user chooses to embed a digital visible watermark, the tool takes as input the text string that will be inserted as watermark, and it produces the visible watermarked image as output. The tool allows the configuration of different options such as the font size, the typeface (Serif, Sans Serif or one of them randomly selected), the number of repetitions and the color. The algorithm converts the watermark text string into a bitmap image and then it merges the obtained bitmap with the input image in randomly selected positions. Figure 2 shows the User Interface (UI) of the tool for the configuration of the options on the SDK emulator [4, 6, 3] with screen resolution of $320 \times 640$. Moreover, the tool permits a preview of the watermarked image on the smartphone display and the owner of the device can save the resulting image in a user defined position on the device itself.

Figure 3 shows a preview of the image "Splash" affected by a visible watermark by the tool on the Android SDK Emulator in portrait mode.

### 3.2 Digital Invisible Watermarking

The design of a scheme for the embedding of a digital invisible watermark into an image for portable devices needs to satisfy the following requirements:

**Low resources usage:** It is important to use resources as parsimoniously as possible, since the portable devices could have reduced capabilities in terms of CPU power, memory, etc.

**Power energy saving:** Commonly, by using low computational complexity models, the power energy, which is an important resource for portable devices, is also preserved.

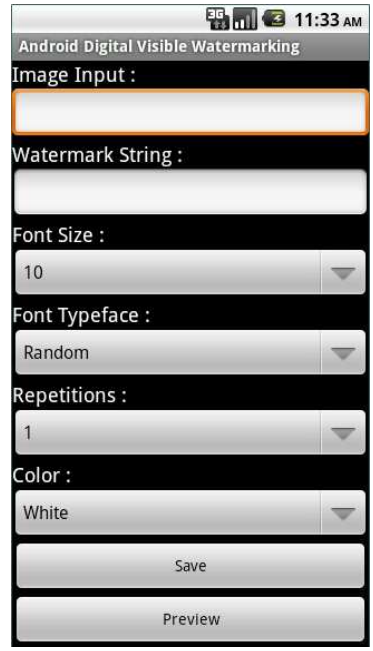


Figure 2. The configuration panel for the embedding of digital visible watermarking

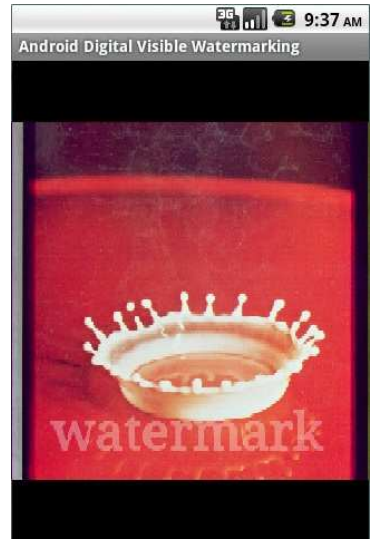


Figure 3. Preview of the "Splash" image on the SDK Android emulator (portrait mode)

**Blind watermarking scheme:** It can be preferable to design a blind watermarking scheme, in order to store only the watermarked images (especially when the image is affected by a watermark directly after the shoot with the integrated camera). In this way, it is possible to save storage space.

Our tool uses reduced resources in terms of CPU power and memory usage. Moreover, the embedding procedure is a blind technique, allowing the user not to save the original image.

### 3.2.1 The Embedding Procedure

The procedure for the embedding of a digital invisible watermark into a still image is based on the technique proposed by Langelaar et al. [12].

The parameters that the embedding algorithm takes as input are:

**inputImage:** The input image, in which the algorithm will embed the watermark string.

**watermarkString:** The string of $N$ bits that will be embedded into *inputImage* at the end of the process.

**seed:** A numeric PIN that will be used for the embedding of the watermark.

**blockSize:** The size of the blocks, in which the algorithm will incorporate each bit of the watermark string.

**$\lambda$:** An integer value ($> 0$) that will be used to modify the blocks selected by the algorithm.

**T:** An integer value that will be used for the robustness of the embedded watermark with respect to some attacks.

The algorithm outputs a watermarked image, which can be compressed through a lossless compressor (PNG) or through the JPEG lossy compression algorithm. Both the JPEG and the PNG implementations are directly provided by the APIs of the Android OS.

At the beginning of the processing state, *inputImage* is converted from the RGB domain to the YUV domain, by using Equations (4).

$$\begin{aligned} Y &= 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B, \\ U &= 0.492 \cdot (B - Y), \\ V &= 0.877 \cdot (R - Y) \end{aligned} \tag{4}$$

where $R$, $G$ and $B$ are respectively the red, green and blue components of each pixel of the image.

Once *inputImage* is processed and the watermark string is embedded, *inputImage* is reconverted into the RGB domain from the YUV domain, as explained in Equations (5).

$$\begin{aligned} R &= Y + \frac{V}{0.877}, \\ G &= Y - (0.395 \cdot U) - (0.581 \cdot V), \\ B &= Y + \frac{U}{0.492} \end{aligned} \tag{5}$$

where $Y$, $U$ and $V$ are respectively the luminance and the two chrominances components of each pixel of the image.

The following pseudo-code introduces step-by-step the proposed embedding procedure:

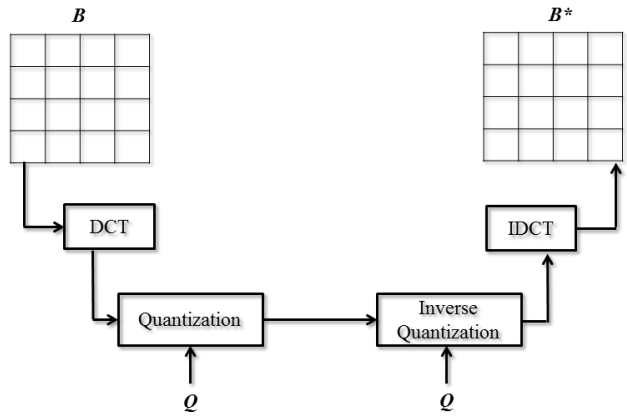


Figure 4. How the block $B^*$ is computed

1. A block $B$ of *blockSize* × *blockSize* pixels is pseudo-randomly selected from *inputImage* to embed one bit of the watermark string (using *seed*).
2. A fixed pseudo-random pattern $B^R$ of the same size of $B$ is generated (using *seed*). Each entry of $B^R$ can be 0 or $\lambda$.
3. $I_0$, $I_\lambda$ and $D$ are calculated. $I_0$ and $I_\lambda$ are obtained by computing the average between the luminance values (Y components) in $B$, where the random pattern $B^R$ is equal to 0 and to $\lambda$, respectively. $D$ is the difference between $I_\lambda$ and $I_0$.
4. $B^*$ is a reduced quality block, which is obtained by following these steps (graphically reported in Figure 4):
   (a) $B^{Coeffs} = DCT(B)$;
   (b) $B_Q^{Coeffs} = \text{quantize}(B^{Coeffs}, Q)$;
   (c) $B_R^{Coeffs} = \text{inverseQuantize}(B_Q^{Coeffs}, Q)$;
   (d) $B^* = IDCT(B_R^{Coeffs})$

where $Q$ is a quantization matrix.

5. $I_0^*$, $I_\lambda^*$ and $D^*$ are calculated. $I_0^*$ and $I_\lambda^*$ are obtained by calculating the average of the luminance values in $B^*$, where the random pattern $B^R$ is 0 and $\lambda$, respectively. $D^*$ is the difference between $I_\lambda^*$ and $I_0^*$.
6. If the bit to embed has value 1 go to step 8.
7. In order to embed the bit with value 0, the pseudo-random pattern $B^R$ is subtracted from the block $B$, if one of the differences, $D$ and $D^*$, is greater than $-T$. The steps 3–5, and 7 are repeated iteratively until both differences are less than $-T$. Go to step 9.
8. In order to embed the bit with value 1, the pseudo-random pattern $B^R$ is added to the block $B$, if one of the differences, $D$ and $D^*$, is less than $T$. The steps 3–5, and 8 are repeated iteratively until both the differences are greater than $T$.
9. The steps from 1 to 8 are applied to all pseudo-randomly selected blocks until all bits of the watermark string are embedded.

The Discrete Cosine Transform (DCT), coupled with the quantization, is used in order to improve the robustness against the JPEG lossy compression attacks.

Using the DCT, the JPEG algorithm exploits its advantages, since the main feature of the DCT is to concentrate the energy of the input data just in the first few coefficients. The definition (6) formally defines the one-dimensional version of the DCT.

Let $S = \langle s_1, s_2, \ldots, s_n \rangle$ be a sequence of $n$ samples (pixels, audio, etc.), then

$$G_i = \sqrt{\frac{2}{n}} C_i \sum_{h=0}^{n-1} s_h \cos\left[\frac{(2h+1)i\pi}{2n}\right] \tag{6}$$

where

$$C_i = \begin{cases} \frac{1}{\sqrt{2}}, & i = 0, \\ 1, & i > 0. \end{cases} \tag{7}$$

The resulting $n$ DCT *transform coefficients* are named $AC$ coefficients except for the first, which is named $DC$ coefficient (the most important). It is important to note that the coefficients could be negative or positive and could be real number even in the case all the sample are integers [21]. From these coefficients it is possible to *reconstruct* the $n$ values of the input through the *inverse* DCT (IDCT), as defined in Equation (8).

$$s_j = \sqrt{\frac{2}{n}} \sum_{h=0}^{n-1} C_h G_h \cos\left[\frac{(2j+1)h\pi}{2n}\right] \tag{8}$$

where $j = 0, \ldots, n-1$.

Regarding the images, applying the DCT, the early coefficients contain the important image information and the later coefficients maintain the information, which are less important (i.e. details of the images, etc.).

It is easier to scale the DCT transform to bi-dimensional DCT. Therefore, scaling the Equation (6), it is possible to define the 2-D DCT as in Equation (9), for an input $m \times n$ matrix, $\mathbf{S}_{m,n}$.

$$G_{i,j} = \sqrt{\frac{2}{m}}\sqrt{\frac{2}{n}} C_i C_j \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} s_{x,y} \cos\left[\frac{(2y+1)j\pi}{2m}\right] \cos\left[\frac{(2x+1)i\pi}{2n}\right], \qquad (9)$$

for $0 \leq i \leq n-1$, $0 \leq j \leq m-1$. Similarly to the one-dimensional DCT, the coefficient $G_{0,0}$ is referred as $DC$ coefficient and the other coefficients are referred as $AC$ coefficients, in similar way, it is possible to re-define the 2-D inverse DCT (2-D IDCT).

### 3.2.2 The Extraction Procedure

The procedure for the extraction of a digital invisible watermark, embedded into a watermarked image, uses the following input parameters:

**watermarkedImage:** The image from where the procedure will extract the embedded watermark,

**seed:** The same value used for the embedding,

**blockSize:** The same value used for the embedding,

**N:** The number of bits that composes the embedded watermark.

As output, the procedure returns the extracted watermark $W^*$.

As in the embedding procedure, also the extraction procedure needs to convert the image from the RGB domain to the YUV domain, using the Equations (4). Once the watermark is extracted, it is possible to re-convert the image from the YUV domain to the RGB domain, using the Equations (5).

The following pseudo-code reports the procedure for the extraction of the digital invisible watermarking embedded into an image:

1. A block $B$ of *blockSize* $\times$ *blockSize* pixels is pseudo-randomly selected from *inputImage* to embed one bit of the watermark string (using *seed*).
2. A fixed pseudo-random pattern $B^R$ of the same size of $B$ is generated (using *seed*). Each entry of $B^R$ can be 0 or $\lambda$.
3. $I_0$, $I_\lambda$ and $D$ are calculated. $I_0$ and $I_\lambda$ are obtained by calculating the average of the luminance values (Y components) in $B$, where the random pattern $B^R$ is equal to 0 and to $\lambda$, respectively. $D$ is the difference between $I_\lambda$ and $I_0$.
4. The embedded bit has value 1, if $D > 0$, the embedded bit has value 0, otherwise.
5. The steps from 1 to 4 are repeated for all the $N$ bits, composing the embedded watermark.

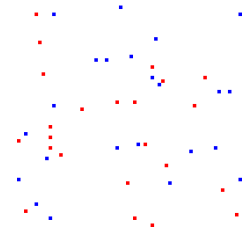

Figure 5. The blocks of the images modified by the embedding algorithm for the digital invisible watermarking

Optionally, it is possible to produce a false-colors image in order to depict the positions of the blocks, which are affected by the embedding algorithm, as shown in Figure 5.

The white part of the image is the part that is not modified by the algorithm. The red blocks indicate the blocks where a bit with value 1 is embedded and the blue blocks indicate the blocks where a bit with the value 0 is embedded.

### 3.2.3 Implementation Optimizations

As it is possible to observe, the proposed model for the embedding of the invisible watermark, does not modifies the entire input image. Similarly, the extraction model does not uses the entire watermarked image. Just the selected $N$ blocks (of $blockSize \times blockSize$ pixels) are used by the algorithms.

Considering this aspect, it is unfruitful to convert from the RGB domain to the YUV domain and, subsequently, from the YUV domain to the RGB domain the entire image when embedding, and the entire watermarked image when extracting the watermark.

For these reasons, we further optimized both algorithms. Once the embedding/extraction algorithm has selected a block, denoted as $B$, just this block will be converted from the RGB domain to the YUV domain. Once finished, the block will be re-converted from the YUV domain to the RGB domain.

## 4 SIMULATION RESULTS

We tested our tool on a test set composed of 16 RGB images (available online [22]), 24 bits per pixel, described in Table 1. For each image, Table 1 indicates the filename, a short description of the image, the dimensions and the size in terms of kilobytes (KB) respectively on the first, second, third and fourth column. All the images were originally in TIFF format which is not directly supported by the tool, therefore, we converted them into BMP format.

The experiments have been performed by using the following parameters: $\lambda = 6$, $blockSize = 16$, $seed = 1\,234\,567\,890$ and $T = 1$. The watermark we have used is

a string of 50 bits, pseudo-randomly generated. As quantization matrix $Q$, we used the one defined in Equation (10).

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 2 & 2 & 4 \\ 1 & 1 & 1 & 1 & 1 & 2 & 2 & 4 \\ 1 & 1 & 1 & 1 & 2 & 2 & 2 & 4 \\ 1 & 1 & 1 & 1 & 2 & 2 & 4 & 8 \\ 1 & 1 & 2 & 2 & 2 & 2 & 4 & 8 \\ 2 & 2 & 2 & 2 & 2 & 4 & 8 & 8 \\ 2 & 2 & 2 & 4 & 4 & 8 & 8 & 16 \\ 4 & 4 & 4 & 4 & 8 & 8 & 16 & 16 \end{bmatrix} \tag{10}$$

Peak signal to noise (PSNR) [9], defined in Equation (11), is used for the evaluation of the perceptual distortion.

$$PSNR = 10 \cdot \log_{10} \left( \frac{255^2}{MSE} \right) \tag{11}$$

where the $MSE$ quantity is the *Mean Squared Error*, defined as

$$\begin{aligned} MSE = \frac{1}{3 \cdot W_I \cdot H_I} \sum_{x=1}^{W_I} \sum_{y=1}^{H_I} \Big[ & \left( I(R,x,y) - I^*(R,x,y) \right)^2 \\ & + \left( I(G,x,y) - I^*(G,x,y) \right)^2 \\ & + \left( I(B,x,y) - I^*(B,x,y) \right)^2 \Big] \end{aligned} \tag{12}$$

where $W_I$ and $H_I$ are the width and the height of the image $I$, respectively, and $I(C,x,y)$ and $I^*(C,x,y)$ are the intensity of the component $C$ (which can be $R$, $G$ or $B$) of the pixel with coordinates $(x,y)$ of the original image $I$ and the watermarked image $I^*$, respectively.

Table 2 reports the PSNR for each test image obtained by comparing the original image and the watermarked image. The trend is graphically represented in Figure 6.

In order to test the robustness of our watermarking approach with respect to the most commons attacks (such as scaling, JPEG lossy compression, etc.), we have used the StirMark benchmark suite.

Once the watermarked images have been modified by StirMark, we have extracted the watermark string $W^*$ and we have computed the similarity *sim* (see Section 2.1), between the embedded watermark $W$ and the extracted watermark $W^*$.

### 4.1 Lossy Compression

Table 3 reports the results achieved in terms of similarity between the original watermark $W$ and the extracted watermark $W^*$ after JPEG lossy compression at different quality factors (15 %, 25 %, 35 %, 50 %, 60 %, 80 % and 100 %). Figure 7 draws the trend of the *sim* metric for Table 3.

| Image | Description | Width x Height | Size (KB) |
|---|---|---|---|
| 4.1.01 | Girl (1) | 256 × 256 | 192 |
| 4.1.02 | Couple | 256 × 256 | 192 |
| 4.1.03 | Girl (2) | 256 × 256 | 192 |
| 4.1.04 | Girl (3) | 256 × 256 | 192 |
| 4.1.05 | House | 256 × 256 | 192 |
| 4.1.06 | Tree | 256 × 256 | 192 |
| 4.1.07 | Jelly beans (1) | 256 × 256 | 192 |
| 4.1.08 | Jelly beans (2) | 256 × 256 | 192 |
| 4.2.01 | Splash | 512 × 512 | 768 |
| 4.2.02 | Girl (Tiffany) | 512 × 512 | 768 |
| 4.2.03 | Mandrill (or Baboon) | 512 × 512 | 768 |
| 4.2.04 | Girl (Lena or Lenna) | 512 × 512 | 768 |
| 4.2.05 | Airplane (F-16) | 512 × 512 | 768 |
| 4.2.06 | Sailboat on lake | 512 × 512 | 768 |
| 4.2.07 | Peppers | 512 × 512 | 768 |
| house | House | 512 × 512 | 768 |

Table 1. Description of the used test set

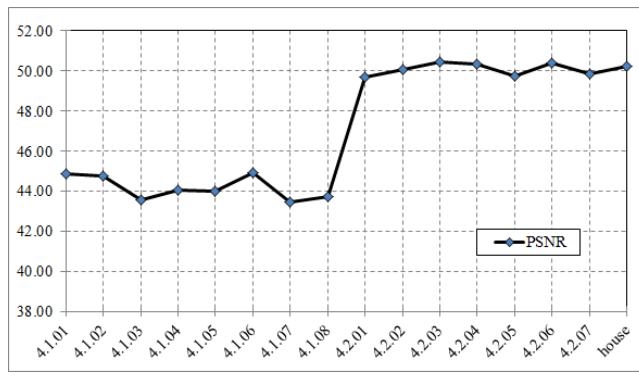


Figure 6. The trend of the PSNR for each image of the test set obtained by comparing the original image and the watermarked image

### 4.2 Median Filter

Table 4 (graphically represented in Figure 8) reports the achieved results in terms of similarity between the original watermark $W$ and the extracted watermark $W^*$ after altering by the median filter, using different *window* sizes: $3 \times 3$, $5 \times 5$, $7 \times 7$ and $9 \times 9$.

| Image | PSNR |
|---|---|
| 4.1.01 | 44.87 |
| 4.1.02 | 44.74 |
| 4.1.03 | 43.54 |
| 4.1.04 | 44.03 |
| 4.1.05 | 44.00 |
| 4.1.06 | 44.90 |
| 4.1.07 | 43.43 |
| 4.1.08 | 43.70 |
| 4.2.01 | 49.67 |
| 4.2.02 | 50.06 |
| 4.2.03 | 50.46 |
| 4.2.04 | 50.35 |
| 4.2.05 | 49.76 |
| 4.2.06 | 50.37 |
| 4.2.07 | 49.83 |
| house | 50.22 |

Table 2. The PSNR for each image of the test set obtained by comparing the original image and the watermarked image

### 4.3 Scaling

Table 5 reports the achieved results in terms of similarity between the original watermark $W$ and the extracted watermark $W^*$ after the scaling transformation at different factors: 50 %, 75 %, 90 %, 110 %, 150 % and 200 %. Before the extraction of

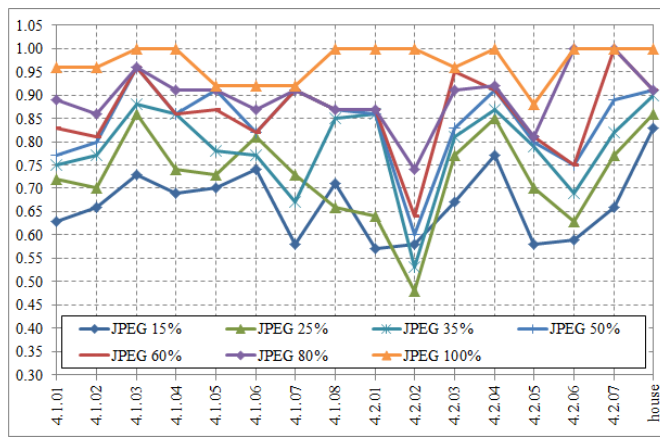


Figure 7. Trend of the similarities between the original watermark $W$ and the extracted watermarks from the watermarked images altered by the JPEG lossy compression algorithm

| **Image/*Quality*** | **15 %** | **25 %** | **35 %** | **50 %** | **60 %** | **80 %** | **100 %** |
|---|---|---|---|---|---|---|---|
| 4.1.01 | 0.63 | 0.72 | 0.75 | 0.77 | 0.83 | 0.89 | 0.96 |
| 4.1.02 | 0.66 | 0.70 | 0.77 | 0.80 | 0.81 | 0.86 | 0.96 |
| 4.1.03 | 0.73 | 0.86 | 0.88 | 0.96 | 0.96 | 0.96 | 1.00 |
| 4.1.04 | 0.69 | 0.74 | 0.86 | 0.86 | 0.86 | 0.91 | 1.00 |
| 4.1.05 | 0.70 | 0.73 | 0.78 | 0.91 | 0.87 | 0.91 | 0.92 |
| 4.1.06 | 0.74 | 0.81 | 0.77 | 0.82 | 0.82 | 0.87 | 0.92 |
| 4.1.07 | 0.58 | 0.73 | 0.67 | 0.91 | 0.91 | 0.91 | 0.92 |
| 4.1.08 | 0.71 | 0.66 | 0.85 | 0.87 | 0.87 | 0.87 | 1.00 |
| 4.2.01 | 0.57 | 0.64 | 0.86 | 0.86 | 0.87 | 0.87 | 1.00 |
| 4.2.02 | 0.58 | 0.48 | 0.53 | 0.60 | 0.64 | 0.74 | 1.00 |
| 4.2.03 | 0.67 | 0.77 | 0.81 | 0.83 | 0.95 | 0.91 | 0.96 |
| 4.2.04 | 0.77 | 0.85 | 0.87 | 0.91 | 0.91 | 0.92 | 1.00 |
| 4.2.05 | 0.58 | 0.70 | 0.79 | 0.80 | 0.81 | 0.81 | 0.88 |
| 4.2.06 | 0.59 | 0.63 | 0.69 | 0.75 | 0.75 | 1.00 | 1.00 |
| 4.2.07 | 0.66 | 0.77 | 0.82 | 0.89 | 1.00 | 1.00 | 1.00 |
| house | 0.83 | 0.86 | 0.90 | 0.91 | 0.91 | 0.91 | 1.00 |

Table 3. Similarities between the original watermark $W$ and the extracted watermarks from the watermarked images altered by the JPEG lossy compression algorithm

the watermark $W^*$ we have re-scaled the image to the original image size, in order to permit the correct extraction by the algorithm. Figure 9 draws the trend of the *sim* metric for Table 5.

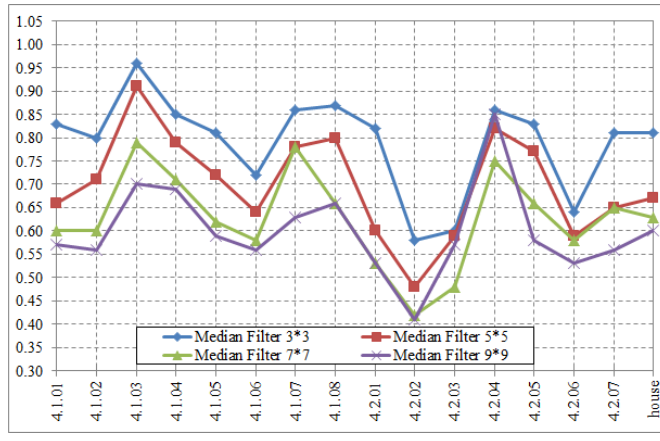


Figure 8. Trend of the similarities between the original watermark $W$ and the extracted watermarks from the watermarked images altered by the median filter

| **Image**/$n \times n$ | $3 \times 3$ | $5 \times 5$ | $7 \times 7$ | $9 \times 9$ |
|---|---|---|---|---|
| 4.1.01 | 0.83 | 0.66 | 0.60 | 0.57 |
| 4.1.02 | 0.80 | 0.71 | 0.60 | 0.56 |
| 4.1.03 | 0.96 | 0.91 | 0.79 | 0.70 |
| 4.1.04 | 0.85 | 0.79 | 0.71 | 0.69 |
| 4.1.05 | 0.81 | 0.72 | 0.62 | 0.59 |
| 4.1.06 | 0.72 | 0.64 | 0.58 | 0.56 |
| 4.1.07 | 0.86 | 0.78 | 0.78 | 0.63 |
| 4.1.08 | 0.87 | 0.80 | 0.66 | 0.66 |
| 4.2.01 | 0.82 | 0.60 | 0.53 | 0.53 |
| 4.2.02 | 0.58 | 0.48 | 0.42 | 0.41 |
| 4.2.03 | 0.60 | 0.59 | 0.48 | 0.57 |
| 4.2.04 | 0.86 | 0.82 | 0.75 | 0.85 |
| 4.2.05 | 0.83 | 0.77 | 0.66 | 0.58 |
| 4.2.06 | 0.64 | 0.59 | 0.58 | 0.53 |
| 4.2.07 | 0.81 | 0.65 | 0.65 | 0.56 |
| house | 0.81 | 0.67 | 0.63 | 0.60 |

Table 4. Similarities between the original watermark $W$ and the extracted watermark after the median filter

### 4.4 Analysis of the Experimental Results

The simulation results have shown that the embedding algorithm is sufficiently robust with respect to the JPEG lossy compression algorithm. In particular, when the quality percentage is in the range from 50 % to 100 %. Only in few cases the similar-

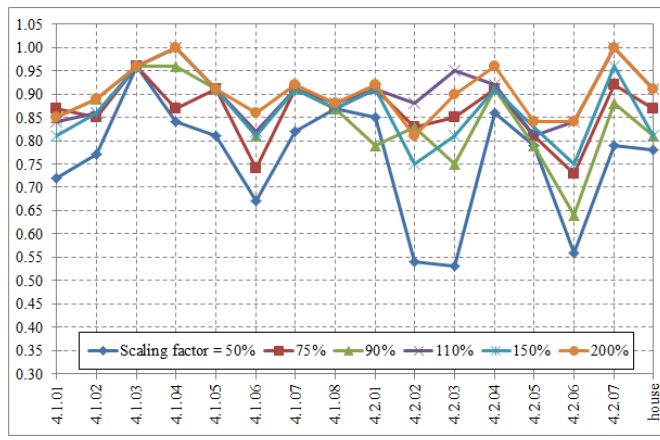


Figure 9. Trend of the similarities between the original watermark $W$ and the extracted watermarks from the watermarked images altered by the scaling transformation

| **Image/*Scaling Factor*** | **50 %** | **75 %** | **90 %** | **110 %** | **150 %** | **200 %** |
|---|---|---|---|---|---|---|
| 4.1.01 | 0.72 | 0.87 | 0.85 | 0.84 | 0.81 | 0.85 |
| 4.1.02 | 0.77 | 0.85 | 0.89 | 0.86 | 0.86 | 0.89 |
| 4.1.03 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 |
| 4.1.04 | 0.84 | 0.87 | 0.96 | 1.00 | 1.00 | 1.00 |
| 4.1.05 | 0.81 | 0.91 | 0.91 | 0.91 | 0.91 | 0.91 |
| 4.1.06 | 0.67 | 0.74 | 0.81 | 0.82 | 0.81 | 0.86 |
| 4.1.07 | 0.82 | 0.91 | 0.91 | 0.91 | 0.91 | 0.92 |
| 4.1.08 | 0.87 | 0.87 | 0.87 | 0.88 | 0.87 | 0.88 |
| 4.2.01 | 0.85 | 0.91 | 0.79 | 0.91 | 0.91 | 0.92 |
| 4.2.02 | 0.54 | 0.83 | 0.83 | 0.88 | 0.75 | 0.81 |
| 4.2.03 | 0.53 | 0.85 | 0.75 | 0.95 | 0.81 | 0.90 |
| 4.2.04 | 0.86 | 0.91 | 0.91 | 0.92 | 0.91 | 0.96 |
| 4.2.05 | 0.79 | 0.81 | 0.79 | 0.81 | 0.83 | 0.84 |
| 4.2.06 | 0.56 | 0.73 | 0.64 | 0.84 | 0.75 | 0.84 |
| 4.2.07 | 0.79 | 0.92 | 0.88 | 1.00 | 0.96 | 1.00 |
| house | 0.78 | 0.87 | 0.81 | 0.91 | 0.81 | 0.91 |

Table 5. Similarities between the original watermark $W$ and the extracted watermark after the scaling transformation

ity metric is less than 0.80, as for example for the image named 4.2.02, compressed at quality factor of 50 %, 60 % and 80 %.

Regarding the median filter, by using a window size of $3 \times 3$, the algorithm is sufficiently robust in 12 of the 16 cases, where the similarity metric is higher or at least equal to 0.80. On the other side, if the window size is wider than $3 \times 3$, the embedded watermark is less robust.

Furthermore, the embedded watermark is sufficiently robust if we do a moderate resizing of the image where the new image does not differ much from the original resolution (for example for scaling factors of 90 % and 110 %).

Considering the low computational complexity nature of the embedding and of the extraction procedures, the algorithm could be considered an equilibrated trade-off between complexity and robustness.

## 5 CONCLUSIONS AND FUTURE WORK

In this paper, we have investigated the possibility to guarantee the ownership of an image, acquired for example by a smartphone, by operating directly on the portable devices through digital watermarking techniques.

Our tool is designed for Android-based devices and it permits the immediate embedding of a visible watermark or an invisible watermark into an image. We implemented both procedures: the embedding and the extraction of an invisible watermark, by using as less resources as possible, since the hardware capabilities of a portable device are limited. Moreover, low computational complexity is strongly coupled with energy consumption savings.

Future work will include extensions of the proposed tool, by considering other embedding approaches. We will investigate also the possibility of the invisible embedding into videos (e.g., the sensitive videos produced by surveillance systems [15]), by considering all the constraints imposed by the low resources usage needed on portable devices.

## REFERENCES


[1] Albano, P.—Bruno, A.—Carpentieri, B.—Castiglione, A.—Castiglione, A.—Palmieri, F.—Pizzolante, R.—You, I.: A Secure Distributed Video Surveillance System Based on Portable Devices. Multidisciplinary Research and Practice for Information Systems (CD-ARES 2012), Lecture Notes in Computer Science, Vol. 7465, 2012, pp. 403–415.

[2] Albano, P.—Bruno, A.—Carpentieri, B.—Castiglione, A.—Castiglione, A.—Palmieri, F.—Pizzolante, R.—Yim, K.—You, I.: Secure and Distributed Video Surveillance via Portable Devices. Journal of Ambient Intelligence and Humanized Computing, Vol. 5, 2014, No. 2, pp. 205–213.

[3] Android Developer Blog. Availaible on: `http://android-developers.blogspot.com/`.

[4] Android Developers. Availaible on: `http://developer.android.com`.

[5] Android Home Page. Availaible on: `http://www.android.com`.

[6] Android SDK Web Page. Availaible on: `http://developer.android.com/sdk/`.

[7] Bender, W.—Gruhl, D.—Morimoto, N.—Lu, A.: Techniques for Data Hiding. IBM Systems Journal, Vol. 35, 1996, No. 3-4, pp. 313–336.

[8] Cox, I. J.—Kilian, J.—Leighton, T.—Shamoon, T.: Secure Spread Spectrum Watermarking for Multimedia. IEEE Transition on Image Processing, Vol. 6, 1997, No. 12, pp. 1673–1687.

[9] Gonzalez, R.C.—Woods, R. E.: Digital Image Processing. 3rd Edition. Pearson Education, 2007.

[10] Jin, C.—Chao, Y.—Zhang, X.-L.: Semi-Fragile Watermark Based on Adaptive Quantization for Image Content Authentication. International Conference on E-Business and Information System Security (EBISS '09), May 23–24, 2009, pp. 1–5.

[11] Langelaar, G. C.—van der Lubbe, J. C. A.—Biemond, J.: Copy Protection for Multimedia Data Based on Labeling Techniques. Proceedings of the 17th Symposium on Information Theory in the Benelux, Enschede, The Netherlands, May 1996.

[12] Langelaar, G. C.—van der Lubbe, J. C. A.—Lagendijk, R. L.: Robust Labeling Methods for Copy Protection of Images. Proceedings of SPIE 1997, Storage and Retrieval for Image and Video Database V, February 1997.

[13] Liang, Q.—Ding, Z.: Spread Spectrum Watermark for Color Image Based on Wavelet Tree Structure. International Conference on Computer Science and Software Engineering, December 12–14, 2008, Vol. 3, pp. 692–695.

[14] Mintzer, F.—Braudaway, G. W.—Bell, A. E.: Opportunities for Watermarking Standards. Communications of the ACM, Vol. 41, 1998, No. 7, pp. 57–64.

[15] Lee, Y.-H.—Ahn, H.—Cho, H.-J.—Lee, J.-H.: Object Recognition and Tracking Based on Object Feature Extracting. Journal of Internet Services and Information Security (JISIS), Vol. 5, 2015, No. 3, pp. 48–57.

[16] Nin, J.—Ricciardi, S.: Digital Watermarking Techniques and Security Issues in the Information and Communication Society. AINA Workshops 2013, pp. 1553–1558.

[17] Petitcolas, F. A. P.: Watermarking Schemes Evaluation. IEEE Signal Processing, Vol. 17, 2000, No. 5, pp. 58–64.

[18] Petitcolas, F. A. P.—Anderson, R. J.—Kuhn, M. G.: Attacks on Copyright Marking Systems. In: Aucsmith, D. (Ed.): Information Hiding 1998 (IH '98). Lecture Notes in Computer Science, Vol. 1525, 1998, pp. 218–238.

[19] Pitas, I.—Kaskalis, T.: Signature Casting on Digital Images. Proceedings of the IEEE Workshop on Nonlinear Signal and Image Processing, Neos Marmaras, June 1995.

[20] Pizzolante, R.—Carpentieri, B.: Copyright Protection for Images on Mobile Devices. 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2012), 2012, pp. 585–590.

[21] Salomon, D.—Motta, G.: Handbook of Data Compression. 5th Edition. Springer 2010. ISBN 978-1-84882-902-2.

[22] SIPI Image Database. Availaible on: `http://sipi.usc.edu/database/database.php`.

[23] Wang, Y.-P.—Chen, M.-J.—Cheng, P.-Y.: Robust Image Watermark with Wavelet Transform and Spread Spectrum Techniques. Conference Record of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers, 2000, Vol. 2, pp. 1846–1850.

**Raffaele Pizzolante** received his Master degree (cum laude) in computer science and his Ph.D. degree in computer science from University of Salerno, Italy. Currently, he is a Research Fellow at the same university. His research interests include data compression, image processing, digital watermarking and information hiding.

**Bruno Carpentieri** received the Laurea degree in computer science from the University of Salerno, Italy, and the M.A. and Ph.D. degrees in computer science from the Brandeis University, Waltham, MA, U.S.A. Since 1991, he has been first Assistant Professor and then Associate Professor of computer science at the University of Salerno, Italy. His research interests include lossless and lossy image compression, video compression and motion estimation, information hiding. He has been, from 2002 to 2008, Associate Editor of the journal IEEE Transactions on Image Processing. He was recently the chair and organizer of the International Conference on Data Compression, Communication and Processing 2011, a co-chair of the International Conference on Compression and Complexity of Sequences, and, for many years, a program committee member of the IEEE Data Compression Conference and of other international conferences in the field. He has been responsible for various European Commission contracts regarding image and video compression.