Digital Watermarking

Michael Stumpfl

Dept. of Electronics and Computer Science University of Southampton e-mail: ms601@soton.ac.uk

Abstract

As more and more analogue systems are replaced by digital ones, the question of how to deal with piracy of copyrighted digital content is turning into a serious problem. For example, the speed with which the "Content Scrambling System" – the encryption standard used for DVD video – was broken, clearly shows, that encryption on its own is insufficient for many systems in protecting copyrighted material from misuse. Digital Watermarking, on the other hand, relies on the shortcomings of the "human visual system" (HVS) and is about embedding copyright information in a way that it is imperceivable by humans, while still being resistant to modifications of the cover media. Thus, digital watermarking can be considered to be a supplementary technique to well established encryption schemes.

1 Introduction

Along with the explosive growth of the Internet not only desirable new possibilities - like publicly available access to information databases around the world, distributed project work across different countries, or fast and reliable means of electronic communication - emerged, but the ease with which digital media can be duplicated and modified, or the fact that legislation is seemingly unable to cope with its rapid rate of change makes it also very attractive to people with dishonourable motives.

With these drawbacks of the "digital age" in mind, creators of multimedia content may wish for a digital analogy to the watermarks that have been used in bookmaking since the 13th Century [Kutt01]. This need for methods and tools to protect ones intellectual property rights initiated the relatively new research field of "digital watermarks". Someone familiar with encryption techniques might be tempted to ask why there is such an amount of interest in the research community to develop robust watermarking techniques, if numerous secure encryption algorithms are readily available. There are several reasons for this:

- Encryption alone often is insufficient to protect digital content, since unconsidered and erroneous usage by human operators often renders it useless.
- If somebody breaks the encryption (e.g. breaking the "content scrambling system" used on DVDs with tools like "DeCSS", "VobDec" or "SmartRipper"), copyright infringements can still be proven using the embedded watermark.
- The decryption process usually depends on the data being unmodified.
- Since rightful owners are to be allowed to access the data they paid for, the encryption needs to be undone at some point. As the unencrypted data is normally being held in the main memory of computers, it's not too difficult to devise tools for storing it onto a local hard disk (e.g. many DVD players for the Windows OS use DirectShow for video output. This proves to be useful for multiangle DVDs, where applications like "DeCSS" fail. By using tools that implement appropriate DirectShow filters to write the decoded images to a user-specified file instead of displaying them on the screen (e.g. "DVDRip"), the raw video data is still accessible.

Because of these shortcomings, digital watermarking is sometimes referred to as being "the last line of defence". Consequently, an effective watermark should normally have several properties, whose importance will vary depending upon the application [Kutt01]:

Robustness

- **Fragile Watermarks** are highly sensitive to any modifications, their sole purpose being to prove the authenticity of a document.
- **Robust Watermarks** should be embedded in a way, that they cannot be removed from the data without introducing noticeable defects.

• Perceptibility

- Visible Watermarks are added as a perceivable additional layer to the original data
 - Invisible Watermarks should not be perceivable by human senses.
- **Security** Unauthorized parties should not be able to read or alter the watermark, even if they have detailed knowledge about the used algorithms (Kerckhoffs's maxim).
- **Multiple Watermarks** Does the watermarking algorithm allow for multiple watermarks to be detected independently of each other?
- **Speed** In video distribution systems it may be necessary to use asymmetric algorithms, which offer very fast embedding methods, whereas watermark detection can take arbitrarily longer.

While encryption often is the best solution for establishing a secure communications channel, it can have the disadvantage of unintentionally raising the suspicion of third parties. Steganography, on the other hand, doesn't rely on encryption techniques, but tries to conceal the very existence of an ongoing communication. Consequently, a successful attack consists in detecting any covert communication. Although watermarking is conceptually equivalent to steganography, it usually has the additional requirement of being robust against possible attacks [Kutt01].

Digital watermarking is still a very young research area, with its first academic conference held in 1996 [Ande99]. Numerous algorithms have been proposed and dismissed since then. Therefore, this paper aims at describing one popular watermarking technique in detail ("Spread Spectrum Watermarking"), rather than providing a short and unsatisfactory explanation of a larger number of different methods.

2 Spread Spectrum Watermarking

Pickholtz et al. [Ruan97] define spread spectrum communication as follows:

Spread spectrum is a means of transmission in which the signal occupies a bandwidth in excess of the minimum necessary to send the information; the band spread is accomplished by a code, which is independent of the data, and a synchronized reception with the code at the receiver is used for despreading and subsequent data recovery.

When used in digital image watermarking, this translates to inserting the watermarking bits at more than one location in the image. Thus, even if subsequent image operations may remove the watermark in some parts of the image, it is very likely that the embedded copyright is still detectable.

2.1 Variant 1

There are several variants of spread spectrum watermarking. Possibly the easiest way to embed copyright bits into an image is to add them in the spatial domain without the need to do costly preand post-processing steps like cosine-, Fourier- or Wavelet transformations.

2.1.1 Embedding

The first step is to assign -1 and +1 to the watermarking bits 0 and 1 respectively. The resulting bit stream is then arranged either in rows, columns or tiles across the whole image (Figure 1, "key2": logical '0': white areas, logical '1': black areas; each tile corresponds to one bit in the watermark).

To accomplish the band spread, "key2" is then multiplied by the output of a "pseudo-noise generator" like a "linear feedback shift register" (LFSR) or other random generators.

Lastly, the values of the watermark are adjusted by multiplying them with a "rescaling factor" and added to the original image in a pixel wise manner. The "rescale factor" determines the strength with which the watermark is embedded (usually depending on the characteristics of different parts of the image, e.g. some parts of the image might allow for a rescale factor of 5 or more, while others suffer a visible degradation in image quality even if the rescale factor is reduced to 1).



Figure 1: Embedding a spread spectrum watermark

2.1.2 Detection

The obvious problem in the watermark detection step is how to retrieve the embedded watermark from a (possibly) watermarked image. There are several approaches to this problem:

- "private/non-blind watermarking": The embedded watermark is recovered by subtracting the original image
- "semi-blind watermarking": The detector requires access to the published/unmodified watermarked image
- "public/blind watermarking": The embedded watermark can be detected without the original image

The described watermark is an example for a public/blind watermarking scheme, because the embedded copyright can be retrieved without the original image by taking advantage of the specifics of the algorithm described in 2.1.1. Since the watermark is added as a pseudo-noise sequence having a high spatial frequency, it can be recovered by applying a high-pass filter such as the following convolution mask:

high-pass filter.
$$\frac{1}{5} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

To calculate the filtered image, the convolution mask is shifted over the whole image. In each step the pixel below the "-8" is replaced by the inner product of the convolution mask and the underlying pixel values. The resulting image is then multiplied by password-dependent pseudo-noise sequence "key1" to get the "correlation image". After summing the values of the correlation image for each copyright bit, the actual bit values (key2) are finally recovered using the following formula (see Figure 2):

$$OutputBit(corrSquare[i][j]) \coloneqq \begin{cases} 0 \dots avg(corrSuqare[i][j]) = -detectionThreshold \\ 1 \dots avg(corrSuqare[i][j]) = +detectionThreshold \\ detection failed \dots else \end{cases}$$



Figure 2: Detection of a spread spectrum watermark

2.2 Variant 2

Many of today's predominant video and image compression algorithms make use of the "Discrete Cosine Transformation" (DCT, see Equation 1) to transform a signal from the spatial- into the frequency domain, where compression can be gained by quantizing the DCT coefficients and applying a Huffman- or arithmetic code to the result.

$$\begin{split} \mathbf{S}_{uv} &= \frac{1}{4} \cdot \mathbf{C}_{u} \cdot \mathbf{C}_{v} \cdot \sum_{x=0}^{7} \sum_{y=0}^{7} \mathbf{s}_{yx} \cdot \cos\left(\frac{(2 \cdot x + 1) \cdot \mathbf{u} \cdot \pi}{16}\right) \cdot \cos\left(\frac{(2 \cdot y + 1) \cdot \mathbf{v} \cdot \pi}{16}\right) \\ \mathbf{s}_{yx} &= \frac{1}{4} \cdot \sum_{u=0}^{7} \sum_{v=0}^{7} \mathbf{C}_{u} \cdot \mathbf{C}_{v} \cdot \mathbf{S}_{uv} \cdot \cos\left(\frac{(2 \cdot x + 1) \cdot \mathbf{u} \cdot \pi}{16}\right) \cdot \cos\left(\frac{(2 \cdot y + 1) \cdot \mathbf{v} \cdot \pi}{16}\right) \\ \text{where } \mathbf{C}_{u}, \mathbf{C}_{v} &= \begin{cases} \frac{1}{\sqrt{2}} & \text{for } \mathbf{u}, \mathbf{v} = 0 \\ 1 & \text{otherwise} \end{cases} \end{split}$$

Equation 1: DCT/IDCT as defined in the JPEG standard

The calculated DCT coefficients give a measure for the correlation between the original 8x8 block and the respective DCT basis image i.e. they represent the amplitudes of all cosine waves necessary to synthesize the original signal in the inverse process (see [Smit97] for further information).

A variant of the algorithm presented in the last section that fits into this framework, works in exactly the same way, except the watermarking bits are only embedded at certain locations in each 8x8 DCT-block (see Figure 3).

DC—	►AC ₀₁	AC ₀₂ -	→ W	W -	→W	W -	►AC ₀₇
AC ₁₀	AC ₁₁	w	w	W	W	AC ₁₆	AC ₁₇
AC ₂₀	w	W	w	W	AC ₂₅	AC ₂₆	AC ₂₇
W	w	w	w	AC ₃₄	AC ₃₅	AC ₃₆	AC ₁₇
¥ W	w	W	AC ₄₃	AC ₄₄	AC ₄₅	AC ₄₆	AC ₄₇
W	w	AC ₅₂	AC ₅₃	AC ₅₄	AC ₅₅	AC ₅₆	AC ₅₇
w	AC ₆₁	AC ₆₂	AC ₆₃	AC ₆₄	AC ₆₅	AC ₆₆	AC ₆₇
AC ₇₀ -	►AC ₇₁	AC ₇₂ -	+AC ₇₃	AC ₇₄	►AC ₇₅	AC ₇₆ -	►AC ₇₇

Figure 3: DCT block: 'W' depicts suitable coefficients for the watermarking process [Jeon01]

The range of suitable DCT coefficients is limited by two constraints [Jeon01]:

- (i) The human eye is especially sensitive to changes in lower frequency components
- (ii) Higher frequency components are unsuitable for watermarking due to quantization

Therefore, the only coefficients that should be used in the watermarking process are those depicted by a 'W' in Figure 3 (the middle-frequency range).

2.3 Variant 3

The algorithms presented so far suffer from one deficiency: although spread spectrum watermarks are robust against various filtering and compression techniques, they are easily defeated by operations involving scaling or rotation. To overcome this limitation the Discrete Fourier Transform can be used, which provides a very elegant way to make an embedded watermark RST-invariant (RST...rotation, scale, translation).

As opposed to the DCT, which composes a signal solely of cosine waves with different frequencies and amplitudes, the Fourier Transform uses both sine and cosine waves to decompose a given signal. Of the 4 different kinds of Fourier Transforms listed in Table 1, only the Discrete Fourier Transform (DFT) is commonly used with Computers, i.e. when working with discrete signals [Smit97]:

Type of Transform	Used for signals that			
Fourier Transform	are continuous and aperiodic			
Fourier Series	are continuous and periodic			
Discrete Time Fourier Transform	are discrete and aperiodic			
Discrete Fourier Transform	are discrete and periodic			

Table 1: Different types of Fourier Transforms

There are two different sub-types of the Discrete Fourier Transform [Smit97]:

a.) The **real DFT** transforms a time-domain signal consisting of N points (x_0 to x_{N-1}) into two frequency-domain signals consisting of $\frac{N}{2}$ +1 points, according to the following formulas:

Forward DFT:

$$a_{k} = C_{a} \cdot \sum_{i=0}^{N-1} x_{i} \cdot \cos\left(\frac{2 \cdot \pi \cdot k \cdot i}{N}\right)$$
where $C_{a} = \begin{cases} \frac{1}{N} & \text{for } i = 0 \text{ or } i = \frac{N}{2} \\ \frac{2}{N} & \text{else} \end{cases}$
where $C_{b} = \frac{2}{N}$

$$k = 0, ..., \frac{N}{2}$$
Inverse DFT:
 $x_{i} = \sum_{k=0}^{N/2} a_{k} \cdot \cos\left(\frac{2 \cdot \pi \cdot k \cdot i}{N}\right) + b_{k} \cdot \sin\left(\frac{2 \cdot \pi \cdot k \cdot i}{N}\right)$

b.) The **complex DFT** takes a N point complex signal (i.e. having a real and an imaginary part) and transforms it into two N point signals with the first $\frac{N}{2}$ +1 coefficients being equal to a_i and b_i . The calculation is usually performed using a method known as the Fast Fourier Transform (FFT). It is important to note, that most FFT algorithms require the input signal to have a length N=2^p where $p \in \mathbb{N}$, otherwise the input signal has to be padded with zeros.

Although the real DFT looks simpler at first, most imaging programs use the FFT for one simple reason: The algorithm in a.) has a computational complexity of $O(N^2)$ whereas the FFT is of O(N.log(N)). As it is rather complicated to derive the algorithm, this is left to books dealing specifically with Digital Signal Processing (DSP) such as [Smit97]. While these formulas apply to the one-dimensional case only, they can be used for images as well by first transforming all rows, followed by a transformation of all columns, or vice versa (if you are interested in implementing any of these algorithms, the routines in Appendix A may serve as a good starting point for further experiments).

If the result of a Fourier Transformation has to be displayed on a computer screen, the information contained in the amplitude values ("rectangular notation") is often converted into polar form where the coefficients are represented using a "Magnitude" and a "Phase":

$$m_{k} = Mag(a_{k}, b_{k}) = \sqrt{a_{k}^{2} + b_{k}^{2}} \qquad a_{k} = m_{k} \cdot \cos(\varphi_{k})$$

$$\varphi_{k} = Phase(a_{k}, b_{k}) = \arctan(\frac{b_{k}}{a_{k}}) \qquad b_{k} = m_{k} \cdot \sin(\varphi_{k})$$

Equation 3: Rectangular to polar conversion and vice versa

For example, if the FFT is applied to Figure 4, the Magnitude of the spectrum in polar form will look like Figure 5b with the white corners corresponding to low frequencies and the black centre corresponding to high frequencies.



Figure 4: 500x500 image of a Mandrill [Ruan97]

At this point it is important to remember, that the sine and cosine functions are periodic. Therefore, the spectrum is also repeated infinitely in all directions (although only N points in this spectrum are actually sampled by the FFT, Figure 5b). This gave rise to the common convention of shifting the spectrum by $\frac{N}{2}$ horizontally and vertically, hence placing the zero frequency in the centre (Figure 5c) and displaying it without the phase information (which is the case for all pictures in Figure 5):



Figure 5: Frequency spectrum of Figure 4 displayed in polar form

2.3.1 Embedding

Most watermarking algorithms that claim to be invariant with respect to some transformations, reach this goal by transforming the original image into a space that has the sought invariants, inserting the watermark in this space and inversing the process to get the watermarked image. To reach a RST-invariant space for the Fourier Transform, it is necessary to look at the properties of frequency spectra in polar form [Ruan97]:

Operation in the spatial (time) domain	Result in the frequency domain			
Circular Translation	Only the phase component is affected. If the translation is			
	not circular, this adds a cropping operation			
Scaling	Inverse Scaling			
Rotation	Rotation with same angle			
Table Q. Effects of different impering an exciting in the time demain				

Table 2: Effects of different imaging operations in the time domain

If the watermarking process relies solely on the magnitude component, only scaling and rotation need further attention. Fortunately, these operations can be reduced to translations by changing the coordinate system through a Log-Polar Mapping (LPM, Equation 4) [Ruan97]:

$$\begin{aligned} x &= e^{\mu} \cdot \cos(\theta) \\ y &= e^{\mu} \cdot \sin(\theta) \end{aligned} \quad \begin{array}{l} \mu \in \Re \\ 0 &\leq \theta < 2 \cdot \pi \end{aligned} \quad \begin{array}{l} F(\mu, \theta) &= f\left(e^{\mu} \cdot \cos(\theta), e^{\mu} \cdot \sin(\theta)\right) \\ f(x, y) &= F\left(\frac{1}{2} \cdot \ln\left(\frac{x^2 + y^2}{2}\right) \operatorname{atan}\left(\frac{y}{x}\right)\right) \end{aligned}$$
Equation 4: Log-Polar Mapping (LPM)

Figure 6 gives an example where a LPM has been applied to Figure 4. The process of applying a Fourier-Transform followed by a LPM is commonly referred to as a "Fourier-Mellin Transform".



Figure 6: Log-Polar map of Figure 4 [Ruan97]

In the last step, the resulting Log-Polar map is transformed by another FFT to defeat any translations caused by rotation or scaling operations. As a result, any watermark that is embedded in the magnitude of the resulting spectrum will not be harmed by RST operations.

It is important to note, that the resolution of the LPM is not required to match the size of the original image. This simplifies the watermarking process immensely, because the resolution of the LPM can be kept the same for all images. On the other hand, the coordinate mapping introduces the problem of sampling pixel values between the original grid lines. To prevent a visible degradation of the final image, it is usually advantageous to use algorithms similar to "Bilinear Interpolation" rather than simple methods like "Nearest-Neighbour" etc. (see Figure 7)



Figure 7: Log-Polar Mapping and Bilinear Interpolation

Conclusions

After stating the requirements for digital watermarks, 3 variations on the "Spread Spectrum" watermarking scheme were presented. Specifically the last method showed, how such watermarks can be tailored to be resistant against a pre-chosen set of attacks. However, there are still many research problems to solve until attributes like security or robustness can be ascribed to any of the existing watermarking techniques and according the authors of the freely available watermark-attacking tool "Stirmark"^a, it is questionable if any such algorithm can exist, provided the distortion attack is chosen appropriately.

Bibliography

[Ande99]	"Information Hiding–A survey", Ross J. Anderson, Markus G. Kuhn, Fabien A. P. Petitcolas, Proceedings of the IEEE, special issue on protection of multimedia content, 87(7):1062-1078, July 1999
	http://www.cl.cam.ac.uk/~fapp2/publications/ieee99-infohiding.pdf
[ISO93]	"Information Technology-Digital Compression and Coding of continuous-tone still im-
	ages-Requirements and Guidelines", ISO/IEC 10918-1, CCITT Recommendation T.81,
	Description of the full JPEG Standard, 1993
	http://www.wotsit.org/search.asp@page=6&s=graphics
[Kutt01]	"Digital Watermarking Frequently Asked Questions", Martin Kutter, 2001
	http://www.watermarkingworld.org/faq.html
[Jeon01]	"Dual Detection of a Watermark Embedded in the DCT Domain", Sangoh
	Jeong, Kihyun Hong, EE368A Project Report, Stanford University
	http://www.stanford.edu/class/ee368a/dropbox/project06/
[Ruan97]	Rotation, Scale and Translation Invariant Digital Image Watermarking, Joseph J.K. Ó
	Ruanaidh, Thierry Pun, Centre Universitaire d'Informatique, Universeté de Genéve,
	Switzerland
[Smit97]	"The Scientist and Engineer's Guide to Digital Signal Processing", Steven W. Smith,
	California Technical Publishing, ISBN 0-9660176-3-3, can be downloaded at:
	http://www.dspguide.com/
[Su98]	"Data Embedding and Digital Watermarking", Jonathan K. Su, Telecommunications Laboratory I, University of Erlangen-Nuremberg, 1998

^a Download Address: <u>http://www.cl.cam.ac.uk/~fapp2/software/StirMark 3 1 79.zip</u>

[Su99] "Digital Watermarking of Text, Image, and Video Documents", Jonathan K. Su, Frank Hartung, Bernd Girod, Telecommunications Laboratory, University of Erlangen-Nuremberg <u>http://www.cq.cs.tu-bs.de/v3d2/pubs/diwa-shg98.pdf</u>

All URLs were last checked on October 4, 2001 and found to be valid!

Appendix A

}

The following C++ code can be used to calculate the (Cooley-Tukey-)FFT and its inverse [Smit97]:

```
int FFT1D (float *re_x, float *im_x, int sigLen)
{ double re_temp, im_temp;
    int    i, j, k, l, N;
  // N <= 1?
  if ((N=sigLen) < 2)
   return (0);
  // N = 2^{?}
  int exp = (int)(log(N+0.5)/log(2));
  if ((1 << exp) < N)
   return (-1);
  // bit reversal sorting:
  int Nd2 = N >> 1;
  for (i=1, j=Nd2; i < N-1; i++)
  {
    if (i < j)
    {
      re_temp = re_x [j];
im_temp = im_x [j];
      re_x [j] = re_x [i];
im_x [j] = im_x [i];
      re_x [i] = (float)re_temp;
      im_x [i] = (float)im_temp;
    }
    k = Nd2;
    while (j >= k)
    {
      j -= k;
      k >>= 1;
    }
    j += k;
  }
  // loop for each stage:
  double re_u, im_u, cwav, swav;
  for (l=2; l <= N; l <<= 1)
    re u = 1.0;
    im_u = 0.0;
    cwav = +cos (PI2/l);
    swav = -sin (PI2/1);
    // loop for each sub DFT:
    for (j=0; j < (l>>1); j++)
    {
      k = (1>>1) + j;
      // loop for each butterfly:
      for (i=j; i < N; i+=1, k+=1)
      {
         // butterfly calculcation:
        re_temp = re_x [k]*re_u - im_x [k]*im_u;
im_temp = re_x [k]*im_u + im_x [k]*re_u;
        re_x [k] = (float)(re_x [i] - re_temp);
        im_x [k] = (float)(im_x [i] - im_temp);
         re_x [i] += (float)re_temp;
        im_x [i] += (float)im_temp;
      }
      re_temp = re_u;
      re_u = re_temp*cwav - im_u*swav;
im_u = re_temp*swav + im_u*cwav;
    }
  }
  // FFT calculated successfully:
  return (0);
```

```
int IFFT1D (float *re_x, float *im_x, int sigLen)
{ int N, iRetVal;
  // N <= 1?
 if ((N=sigLen) < 2)
   return (0);
  // N = 2^p?
  int exp = (int)(log(N+0.5)/log(2));
  if ((1 << exp) < N)
    return (-1);
  // change the sign of 'imag[]': for (int k=0; k < N; k++)
  {
    im_x [k] = -im_x [k];
  }
  // calculate the forward FFT:
  iRetVal = FFT1D (re_x,im_x,N);
  // divide the time domain by 'sigLen':
  for (int i=0; i < N; i++)
   re_x [i] = +re_x [i] / sigLen;
im_x [i] = -im_x [i] / sigLen;
  }
  return (iRetVal);
}
```