A Novel Watermarking Algorithm for Image Authentication: Robustness Against Common Attacks and JPEG2000 Compression

Marco Aguzzi, Maria Grazia Albanesi, and Marco Ferretti

Università degli Studi di Pavia, Pavia PV 27100, Italy
marco.aguzzi, mariagrazia.albanesi, marco.ferretti@unipv.it
http://orfeo.unipv.it

Abstract. This paper presents an authentication algorithm based on robust watermarking techniques. In particular, the proposed method is based on a *self - embedding* scheme that is able not only to authenticate noisy images, but also to recover areas modified by a software pirate. The attack method investigated are semantic (altering the meaning of what the image is about) tampering, Gaussian white noise superposition, and JPEG2000 compression. The results are checked against the TAF function, which measure the distance between the inserted and the extracted watermark, and compared to similar algorithms in literature.

1 Introduction

The paper is organized as follows: section 2 is about the Golay code [1], [2], [3], which serves as a foundation to the watermarking algorithm, section 3 is dedicated to introducing the watermarking algorithm, focusing on the wavelet transform, the embedding and the retrieval of the watermark; section 4 is about how the attacks to the watermarked image have been carried out, and section 5 is about the conclusion of the paper and the future works that we are going to develop.

2 The Golay Code

Our watermarking algorithm relies on Golay code. This theoretic foundation allows two main features: a) embedding the watermark and b) if an attack is performed on the watermarked images, reconstructing a new recovered image. This section serves as a theoretical introduction to the argument in order to support the next sections. The Golay code belongs to the family of the linear correcting codes. Let's assume that a message x of n digit is sent by a transmitter, while a message y is received by a receiver: generally we will assume $y = x + e$, where e is the error introduced either by a noisy channel or by a software attack. All equations written in this section are modulo 2, in order to use matrix notation and to adopt the convention $1 + 1 = 0$.

F. Roli and S. Vitulano (Eds.): ICIAP 2005, LNCS 3617, pp. 559–567, 2005.

The *codeword* x does not contain only the information, but it is divided into k digits of actual information and $n-k$ check digits. The way the check digits are calculated depends on the particular code implementation, and it can be asserted by specifying the equation used for going from u (the actual data) to the codeword x or by constructing a k by n matrix in which the position i, j is 1 if the ith digit of u is used to form the jth digit of x. The matrix built in this way is called the *generator matrix* G and for this class of code it can always be constructed as

$$G = [I_k|P] \tag{1}$$

where I_k is the identity matrix of dimension k, P tells how the check digits have to be built, and $[\cdot|\cdot]$ denotes matrix concatenation. So building the codeword can be written as:

$$x = uG \tag{2}$$

On the other side of the communication channel, recalling that the decoder has received y instead of x a vector s, called the *syndrome*, is built using the equation

$$s = yH^T \tag{3}$$

where H is obtained as

$$H = [P^T|I_{n-k}] \tag{4}$$

The syndrome s is a $n-k$ element vector and it contains a 1 where the corresponding parity check has failed, 0 otherwise. If we want to use this kind of codes to correct the errors introduced by transmission or attacks, the information contained in the syndrome are not sufficient, because they signal only if an error is present and what the corresponding parity check failed is, but not where the actual error is. Recalling that $y = x + e$ and, relying on the properties of this class of codes, that $xH^T = 0$, Equation 3 can be rewritten as

$$s = eH^T \tag{5}$$

For each given syndrome the possible associated error patterns are 2^k: this means that even for low n and k it is not possible to store and look up all the error patterns associated with all the syndromes. So only a subset of all the error patterns will be actually correctable. In order to minimize the decoding error produced when the actual error that corresponds to the obtained syndrome differs from the correctable one, the choice of the error patterns eligible for correction points to the ones with the minimum weight, that is, that contain the least number of 1's. The table containing the syndromes and their selected error patterns is called *standard array*.

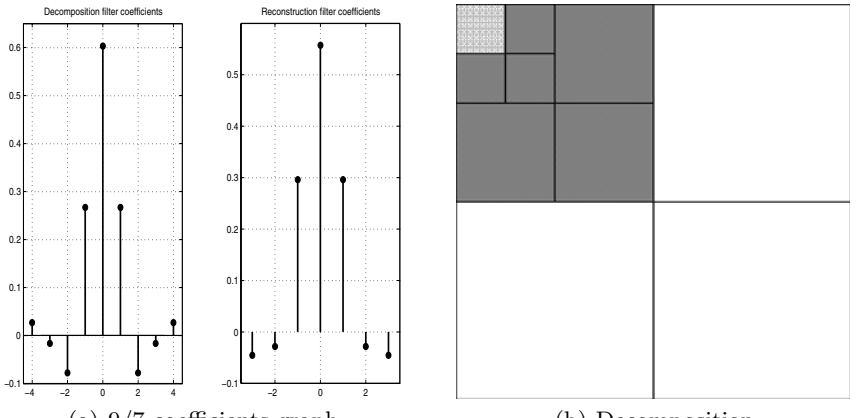
The Golay code is part of a subset of this class of codes: the study of the structure of G, H, and standard array is beyond the scope of this paper. What has to be known is that the code has $n = 23$ and $k = 12$ and is able to correct triple errors.

3 Description of the Novel Algorithm

3.1 Embedding the Watermark

We will focus in more detail in the embedding of the watermark because most of the transformations of this phase are used in the extracting process without modifications. The steps that have to be performed to insert the watermark are: a) wavelet decomposition, b) coefficients quantization, c) watermark calculation on baseband coefficients (see light gray area in Figure 1(b)), and d) watermark embedding in coefficients coming from the next two levels of the decomposition. (see dark gray area in Figure 1(b))

Wavelet Decomposition. The wavelet transform chosen for embedding the watermark uses the Daubechies 9,7 [4] filters. The properties of these filters [5] are: they are compactly and not infinitely supported, they have the shortest basis function for wavelets with 4 vanishing moments, and they form a nearly orthogonal basis. The filter coefficients belonging to the 9,7 filter are reported in Figure 1(a), using their traditional graphic representation. The wavelet decomposition is done over three levels, as depicted in Figure 1(b).



(a) 9/7 coefficients graph (b) Decomposition

Fig. 1. The coefficients used for wavelet decomposition and its subband representation

Coefficients Quantization. The coefficients produced by the wavelet transform are quantized accordingly to the Watson [6] step. Indeed, the thresholds found by Watson are strictly adapted to the wavelet transform and chosen to make the quantization noise perceptually invisible. The quantization matrix is calculated using the formula reported in [6].

Watermark Computation. The baseband coefficients are ported to a suitable data structure containing only their binary digits, scrambled according to a

random permutation and divided into k-element ($k = 12$) blocks. Each k-element block generates the bit string u that builds the watermark in an incremental fashion. The equation used is similar to (2), but here only the parity digits are produced; thus the actual equation used is

$$\hat{x} = uP \tag{6}$$

So $\hat{x}$ is the $n - k$ digit vector containing the check digits of the codeword x. The watermark is formed by successive concatenations of vector $\hat{x}$, which is generated for each bit plane of the baseband coefficient and for each k-element block.

Watermark Embedding. The embedding process does not wait for the whole watermark to be produced, but the insertion goes along as the $\hat{x}$ vectors are ready. The coefficients of these sub bandsare also converted in the "binary digits" structure, scrambled according to a random permutation, divided into $(n - k)$-element blocks ($n - k = 11$) and the least significant bit plane of these blocks is substituted with the corresponding piece of watermark, that is the vector $\hat{x}$.

3.2 Extracting the Watermark

The extraction of the watermark partly follows the same steps as the embedding. The wavelet transform is applied to the watermarked image, and the coefficients are quantized in the same way as when the watermark was being embedded. The quantized coefficients are grouped accordingly to the scheme in Figure 1(b), and on each of them a random permutation is performed, initializing the random seed in the same way as it was initialized when embedding the watermark; this to assure that the permutations in this step are carried out in the same way as the ones made in the embedding process. From the baseband coefficients the vector u is extracted and from the watermarked "embedding" coefficients the vector $\hat{x}$ is extracted. The codeword x is built by the concatenation of u and $\hat{x}$ which are, respectively, of k and $n - k$ elements. The syndrome s is calculated using Equation 3. Now if one ore more digits of s are different from 0, the codeword received has some errors: the corrected codeword is obtained subtracting modulo 2 from the mistaken codeword the value of the standard array indexed by the syndrome.

Image Reconstruction. From the corrected coefficients and the error mask built while performing the watermark extraction according to the calculated syndrome values, the coefficients of the reconstructed image, before applying the inverse wavelet transform, are calculated in this way:

$$u_r = y_c v + y\overline{v} \tag{7}$$

where u_r is the overall reconstructed coefficients, y_c the received coefficients after correction and v is the error mask. The result from Equation 7 is joined with unchanged (and uncorrectable) coefficients from the other bands and on them the inverse wavelet transform is performed in order to obtain the reconstructed image.

4 Robustness to Attacks

When a robust watermark scheme is adopted, a possible measure of how good the watermark is relies on the Tamper Assessment Function (TAF) [7] shown in Equation 8: given the embedded watermark w^M and the extracted watermark w^X, both NW bit long, it measures how much the two watermarks differ one from the other.

$$TAF(w^M, w^X) = \frac{\sum_{i=1}^{NW} w_i^M \oplus w_i^X}{NW} \tag{8}$$

This ratio floats between 0 and 1, and among these two a threshold th can be set in order to discriminate between tampered or not tampered images. In the next subsections various types of attack made to the image will be described, and it will be shown the effects that those attacks have on the TAF value.

As a quality measure also the Peak Signal to Noise Ratio (PSNR) will be considered. The formula used for calculating the PSNR is shown in Equation 9, given the reference signal s^r and the noisy signal s^n, both N^2 samples long. We consider a square image (N rows by N columns) and 255 as the maximum image sample value.

$$PSNR(s^r, s^n) = 10 \log_{10} \left(\frac{255^2}{\frac{1}{N^2} \sum_{i=1}^{N^2} (s_i^r - s_i^n)^2} \right) \tag{9}$$

It will be shown that as the PSNR increases, the TAF decreases consequently, even though the former measure seems to be less "sensitive" than the latter.

4.1 Semantic Tampering

With the term "semantic tampering" we address those kinds of attack aimed at corrupting the semantic meaning of the image. For example, Lena's nose can be substituted with an "ugly" one. Three examples of this kind will be shown: for each attack will we show: a) the original image, b) the watermarked and attacked image, and c) the reconstructed image. (In the case of Lena the original image is not shown because it is already well known.) In Table 1 we report, for each attack, the PSNR loss and the calculated TAF.

Table 1. Semantic attacks summary

	PSNR	PSNR loss	TAF
Bird without feather	37.0305	2.3971	0.0126
Bird with two feathers	39.4276	1.8787	0.0108
"Ugly" *Lena*	37.9413	1.2891	0.0051

4.2 Adding White Noise

For the second attack, we superimposed a white noise, generated by a Gaussian distribution with mean value $\bar{x} = 0$ and varying σ, over the watermarked image.

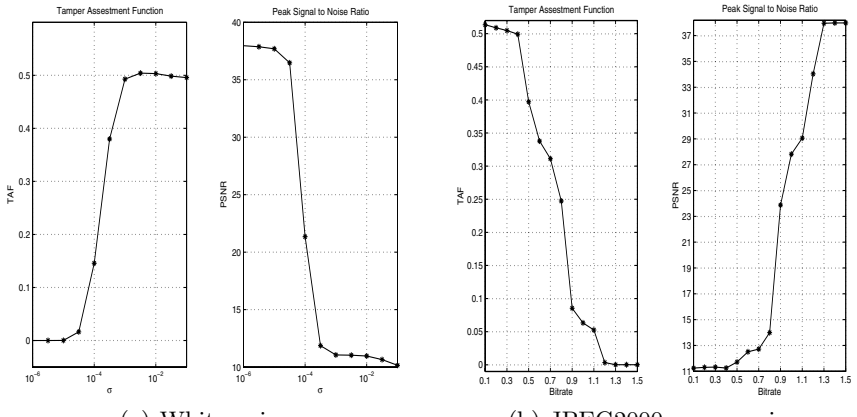
The alteration is spread all over the image, and for increasing σ the TAF increases and the PSNR decreases. σ ranges from $10^{-}1$ to $10^{-}5$, and the curves, as shown in Figure 2(a), highlight the breakpoint around $\sigma = 10^{-}4$, where the quality of the reconstruction becomes unacceptable.

4.3 JPEG2000 Compression

The third attack is related to JPEG2000 compression. The original image is watermarked and then compressed with a bit rate varying from 0.1 bpp (bit per pixel) to 1.5 bpp. As the graphs plotted in Figure 2(b) show, the compression becomes no longer acceptable when the compression rate goes under 1 bpp.

A second piece of information that is visible in Figure 2(b) is in correspondence of the final plateau in the two graphs: they show when the PSNR and the TAF become no longer "sensible" to a relaxation in the compression rate: the TAF remains equal to zero (this means that the watermark is perfectly extracted) after 1.2103 bpp, the PSNR becomes stable after 1.3 bpp. The stable value that the PSNR assumes is slightly different from the original obtained value of 37.94 dB, this is because the compression rate specified as a parameter from the JPEG2000 encoder command line is not strictly equal to the actual gain obtained during compression, so the reconstructed image seems even better than the watermarked one.

Compared to the values reported in work [7], the TAF values we obtained are much better. In Table 2 are reported TAF values obtained with JPEG and



(a) White noise (b) JPEG2000 compression

Fig. 2. Tamper assestment function and Peak Signal to Noise Ratio for gaussian white noise superimposal with a σ varying from 10^{-6} to 10^{-1}. and for JPEG2000 compression varying from 0.1 to 1.5 bits per pixel

Table 2. TAF values comparison with results in [7]. Conversion between bpp and Compression Ratio is made speculating that gray level images in [7] are taken at 8 bpp

bpp	Compression Ratio	TAF [7]	our TAF
4	2	0.0615	0
2.67	3	0.0732	0
2.29	3.5	0.1357	0
1.78	4.5	0.1455	0
1.6	5	0.1729	0
1.33	6	0.25	0
1.14	7	0.3027	0.0544
0.94	8.5	0.4004	0.0891
0.8	10	0.4229	0.2472
0.7	11.43	n.a.	0.3111
0.6	13.33	n.a.	0.3377
0.5	16	n.a.	0.3970
0.4	20	n.a.	0.4991
0.3	26.66	n.a.	0.5046
0.2	40	n.a.	0.5088
0.1	80	n.a.	0.5134

JPEG2000 compression. Bit rate ranges from 1.333 bpp to 0.8 bpp. A wider range would not have been possible because a) for compressions lighter than 1.3 bpp our algorithm returns always a TAF value equal to 0, and b) the results reported in [7] go no further than 0.8 bpp, while ours do.

5 Conclusions and Future Work

This work has shown a *self-embedding* robust watermark inserted into images based on the Golay code using wavelet transform. The method has been tested over various kinds of attack, and the most interesting thing is that the performances over JPEG2000 compression are quite good, allowing compression toward at almost 1 bit per pixel while preserving watermark extraction. Other kinds of attack, such as semantic ones, are well localized by the algorithm so that substitution with lower resolution data is well localized as well. As future work, we are planning to use this approach in order to embed watermarking and public / private key criptography in authentication of bio-metrical data.

References

1. Benedetto, S., Biglieri, E., Castellani, V.: Digital Transmission Theory. Prentice - Hall, Inc. (1987)
2. Gallagher, R.G.: Information theory and reliable communications. John Wiley & Sons (1968)

3. Golay, M.J.E.: Notes on digital coding. Proceedings of IEEE **37** (1949) 637
4. Daubechies, I.: Ten lectures on wavelet. Society for Industrial and Applied Mathematics (1992)
5. Unser, M., Blu, T.: Mathematical properties of the jpeg2000 wavelet filters. IEEE Transaction on Image Processing **12** (2003) 1080–1090
6. Watson, A.B., Yang, G.Y., Solomon, J.A., Villasenor, J.: Visibility of wavelet quantization noise. IEEE Transaction on Image Processing **6** (1997) 1164–1175
7. Kundur, D., Hatzinakos, D.: Digital watermarking for telltale tamper proofing and authentication. Prooceedings of IEEE **87** (1999) 1167–1180 Invited paper.

(a) Original *Bird*

(b) Watermarked and tampered *Bird*

(c) Reconstructed *Bird*

Fig. 3. *Bird* without feather

(a) Watermarked and tampered *Bird* (b) Reconstructed *Bird*

Fig. 4. *Bird* with two feathers

(a) Watermarked and tampered *Lena* (b) Reconstructed *Lena*

Fig. 5. "Ugly" *Lena*: the nose is heavily tampered