

A Digital Watermarking Algorithm for Authentication and Tamper Detection

Jeremy Thurgood and Roger Peplow

School of Electrical, Electronic and Computer Engineering
University of KwaZulu-Natal (Howard College Campus)
Email: {thurgood, roger}@ukzn.ac.za

Abstract

An important element of a communications system is the trustworthiness of the data being received, especially where the data must travel through insecure channels or pass through several hands. Digital watermarking provides a partial solution to the problem of trustworthiness, especially with regard to image and video data.

This paper describes a hybrid watermarking algorithm that combines robust and fragile watermarking to allow source authentication of an image and detection of malicious tampering. The embedded watermark is relatively immune to moderate JPEG and SPIHT compression. Encryption is used to ensure that the watermark cannot be forged without knowledge of a secret key.

The watermark is generated from the host image and is a *content hash* containing a single bit of data for every sixty-four pixels of the host image. This watermark is embedded in a region of the discrete wavelet transform of the image. An encrypted version of the watermark is embedded in another region for use as a source authenticator. The embedding is performed by the quantisation of groups of discrete wavelet transform coefficients. The watermark generation requires a symmetric session key which is randomly determined during watermark generation and stored (in an encrypted form) in a robust watermark in a third region of the discrete wavelet transform.

1. Introduction

Digital watermarking is a relatively new and increasingly important area of signal processing. The basic idea behind it is to embed data into a signal imperceptibly in such a way that it can be extracted at a later date.

It is useful to classify watermarks according to their application. Firstly, a watermark may be used to carry data, such as copyright information about the host signal. In this case, it should be robust to accidental, benign or malicious alterations to the host signal and is known as a *robust watermark*. A *fragile watermark*, on the other hand, is used to detect alterations in an image and hence should be fragile to image alterations. A more useful variant of the fragile watermark is the *semi-fragile watermark* which is fragile to malicious tampering but robust to useful data transformations such as lossy compression and certain image enhancements.

The next section will discuss semi-fragile watermarking algorithms in more detail and develop a set of criteria for design and testing of an algorithm. Section 3 will describe the algorithm currently under test. Section 4 will discuss implementation details. Section 5 will present a testing framework for watermarking algorithms in general and the one described in this paper in particular and section 6 will discuss the results of testing the algorithm.

2. Watermarking Methods

Watermarking algorithms differ in three main areas: watermark design, embedding, and extraction. The watermark is typically a pseudorandom signal with a uniform or gaussian distribution [1] [2]. In order to reduce visibility, an implicit or explicit spectral or spatial filter is usually applied with the goal of attenuating the watermark where it would otherwise be visible.

A watermark is embedded in an image by either modifying the pixel data in the spatial domain or by taking a transform of the data and modifying transform coefficients. The most popular transforms used are the discrete cosine transform and the discrete wavelet transform as these are particularly advantageous in terms of security and imperceptibility [3].

Extraction of a watermark depends in large part on the embedding method. Usually some form of correlation detection or hypothesis testing method is used to check for the presence of a known watermark signal. Some methods, however, are able to use arbitrary information as the watermark and extract it without requiring either the original image or the watermark.

2.1. Semi-Fragile Watermark Requirements

For a semi-fragile watermark to be useful, it has to fulfil several basic requirements. Firstly, and most importantly, it must be fragile to malicious image alterations such as replacement of portions of the image data.

Secondly, it should be secure against forgery. There is little point to watermarking an image if the watermark can easily be re-embedded into a forged image.

The essence of a semi-fragile watermark, however, is the fact that it is robust to alterations that don't essentially change the content of the host image, such as lossy compression.

It is also important that the presence of the watermark does not degrade the host image significantly—there should be minimal visual impact.

Unfortunately, these criteria conflict somewhat. In particular, robustness and imperceptibility oppose each other, as robustness requires a high amplitude in the embedded signal while imperceptibility requires low amplitude. Thus, a balance must be struck between the various requirements that depends on the priorities of the application.

3. Watermarking Algorithm

The watermarking algorithm described here is based on work in [4]. The watermark generation is the same, but the embedding method has been modified to suit more general authentication and tamper detection requirements.

This algorithm is designed to allow the originator of an image (who has the required keys) or an associate to verify the authenticity of the image.

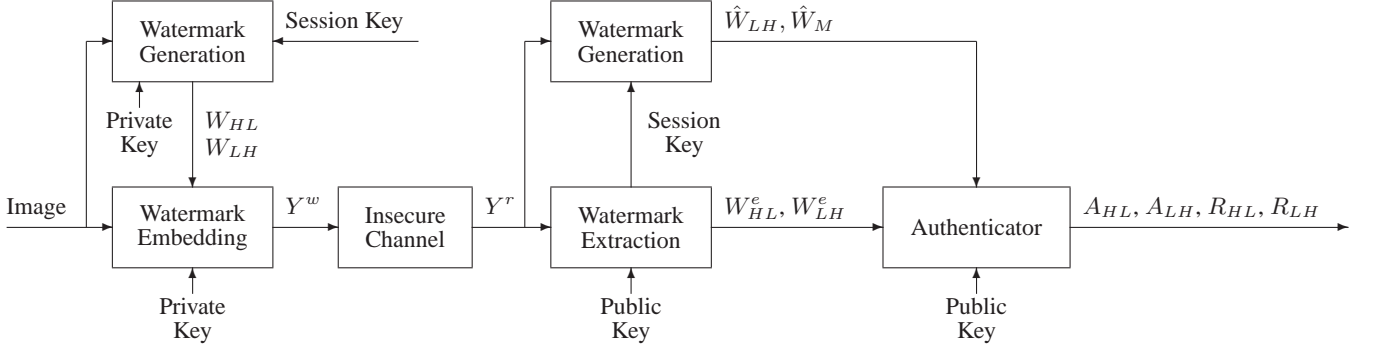


Figure 1: Watermark algorithm block diagram

3.1. General Properties and Overview

The aim of this algorithm is to allow an image from a remote source, such as a camera, to be verified and authenticated once it has been transmitted over an insecure channel. The structure of the watermark allows the content of the image to be verified even if the original is unavailable.

The algorithm watermarks the host image with a *content hash*, which is basically an error-tolerant cryptographic hash of the image. The content hash is based on the relationships between various DC components of the 8×8 Discrete Cosine Transform (DCT) of the original image. Zhao, et. al. [4] have experimentally verified that this hash is invariant for 70% JPEG compression and moderate SPIHT compression.

As well as embedding the content hash, the algorithm also embeds an encrypted version of the hash as an authenticator. The authenticator is used as a digital signature to verify the source of the watermarked image and prevent forgery.

A block diagram of the watermarking algorithm is shown in figure 1 and explained in detail in the rest of this section.

3.2. Computing the Watermark

The watermark is generated in several steps, some of which use randomised parameters to prevent forgery. As repeatability is required to allow verification of the watermark, the pseudorandom number generator is seeded with a *session key*. This session key is required to be the same on both the embedding and extraction sides. The watermark generation process is shown diagrammatically in figure 2.

The first step in generating the watermark is to obtain the luminance of the host image. Next, the 8×8 DCT is taken. This uses the $M_x \times M_y$ luminance image to produce $f_{D_{i,j}}(u, v)$ where (i, j) for $1 \leq i \leq \frac{M_x}{8}$, $1 \leq j \leq \lceil \frac{M_y}{8} \rceil$ denotes the particular 8×8 block and (u, v) for $1 \leq u, v \leq 8$ is the frequency index where $(1, 1)$ represents the DC coefficient.

Feature extraction is performed by creating a $\frac{M_x}{8} \times \lceil \frac{M_y}{8} \rceil$ matrix D of DC coefficients as shown in equation 1.

$$D(i, j) = f_{D_{i,j}}(1, 1) \text{ for all } i, j \quad (1)$$

The session key is used to pair each element $D(i, j)$ in D with another coefficient $D(i', j')$ within a 3×3 neighbourhood. If $|D(i, j) - D(i', j')| < 16$ then the next coefficient in a clockwise direction is chosen. A binary matrix B is generated from these pairs as shown in equation 2.

$$B(i, j) = \begin{cases} 0 & \text{if } D(i, j) - D(i', j') > 16 \\ 1 & \text{if } D(i, j) - D(i', j') < -16 \\ 0 & \text{if no valid pairing could be found} \end{cases} \quad (2)$$

The threshold of 16 was chosen as the relationship $|D(i, j) - D(i', j')| > 16$ is usually preserved under JPEG compression of 70% and moderate SPIHT compression. B is used as the first part of the watermark, denoted W_{LH} .

The second part of the watermark is an encrypted authenticator to verify the source of the image and prevent forgery. The majority function, shown in equations 3 and 4, generates the authenticator, W_A .

$$W_A(k) = \begin{cases} 1 & \text{if } \sum_{j=1}^{\lceil \frac{M_y}{8} \rceil} \tilde{B}(k, j) > \lceil \frac{M_y}{8} \rceil / 2 \\ 0 & \text{otherwise} \end{cases} \quad \text{for } k = 1, \dots, \frac{M_x}{8} \quad (3)$$

$$W_A(k) = \begin{cases} 1 & \text{if } \sum_{i=1}^{\lceil \frac{M_x}{8} \rceil} \tilde{B}(i, k - \frac{M_x}{8}) > \frac{M_x}{8} / 2 \\ 0 & \text{otherwise} \end{cases} \quad \text{for } k = 1 + \frac{M_x}{8}, \dots, \frac{M_x}{8} + \frac{M_y}{8} \quad (4)$$

W_A is encrypted using the private key of an asymmetric encryption algorithm, such as DES, and error-correction coding is applied to produce the binary matrix W_{HL} as the second part of the watermark. The code rate of the error-correction coding should be chosen such that the W_{HL} matrix is $\frac{M_x}{8}$ by $\lceil \frac{M_y}{8} \rceil$ bits.

3.3. Embedding the Watermark

The watermark is embedded in the Discrete Wavelet Transform (DWT) domain, as shown in figure 3. First, the 2-level Haar DWT is computed to obtain the $\frac{M_x}{4} \times \lceil \frac{M_y}{4} \rceil$ second level LH and HL bands, denoted Y_{2LH} and Y_{2HL} respectively. The binary watermarks W_{LH} and W_{HL} are embedded into Y_{2LH} and Y_{2HL} respectively to produce Y_{2LH}^w and Y_{2HL}^w as shown in equations 5, 6 and 7.

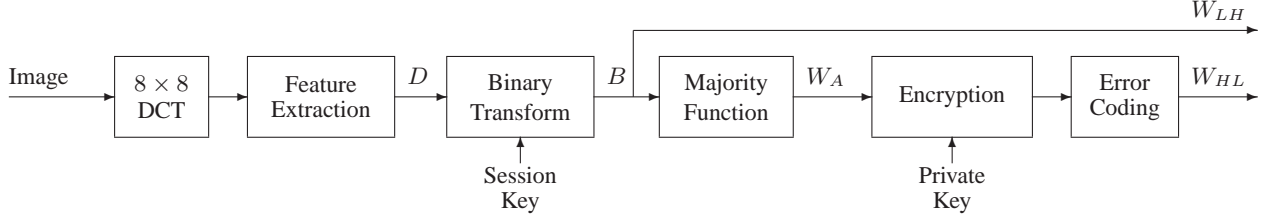


Figure 2: Watermark generation block diagram

$$S_g(i, j) = \sum_{m=1}^2 \sum_{n=1}^2 Y_{2LH/HL}(n + 2(i - 1), m + 2(j - 1)) \quad (5)$$

$$q(i, j) = \frac{S_g(i, j)}{4\delta} \quad (6)$$

$$Y_{2LH/HL}^w(n + 2(i - 1), m + 2(j - 1)) = Y_{2LH/HL}^w(n + 2(i - 1), m + 2(j - 1)) + f(\delta)$$

where

$$f(\delta) = \begin{cases} 0 & \text{if } \text{mod}(q(i, j), 2) = W_{LH/HL}(i, j) \\ s\delta & \text{if } \text{mod}(q(i, j), 2) \neq W_{LH/HL}(i, j) \end{cases}$$

$$\text{and } s = \text{sgn}(Y_{2LH/HL}^w(n + 2(i - 1), m + 2(j - 1))) \quad (7)$$

The parameter δ is a user-defined positive integer that controls the amplitude of the embedded watermark. A higher δ results in larger coefficient quantisation steps and thus a more robust watermark. However, higher δ also results in more image distortion.

The embedding technique described in [4] was designed to embed colour information in the image as well as the authentication watermark, so the embedding scheme has been modified to embed the session key, appropriately encrypted, and other metadata relating to the image. This data is encrypted using the private key of an asymmetric encryption algorithm and the session key can thus be generated randomly at the image source instead of being required on both the embedding and extraction sides.

3.4. Embedding the Metadata

The metadata, including the session key used to generate the watermark, needs to be encrypted using an asymmetric encryption algorithm, such as DES, using the private key. As this encrypted metadata block is highly sensitive to error, it then has a high-rate error-correcting code applied. The rate of the code depends on the amount of metadata that needs to be embedded and the number of bits that can be embedded, N_M .

$$N_M = 2 \frac{M_x}{4} \frac{M_y}{4} \quad (8)$$

This error-coded, encrypted block is referred to as W_M . The encrypted metadata are embedded in the first level LH and HL bands, Y_{LH} and Y_{HL} respectively, using the same method as for the watermark data.

The Inverse DWT is then taken to produce Y^w , the watermarked image.

3.5. Extracting the Watermark

Since the watermark is embedded in the DWT domain, the first step in extraction is to take the 2-level Haar DWT of the watermarked image to obtain the second level bands Y_{2LH}^w and Y_{2HL}^w . The watermarks W_{LH}^e and W_{HL}^e are extracted as shown in equations 9, 10 and 11.

$$S_g(i, j) = \sum_{m=1}^2 \sum_{n=1}^2 Y_{2LH/HL}(n + 2(i - 1), m + 2(j - 1)) \quad (9)$$

$$q(i, j) = \frac{S_g(i, j)}{4\delta} \quad (10)$$

$$W_{LH/HL}^e(i, j) = \text{mod}(q(i, j), 2) \quad (11)$$

The encrypted session key and metadata are extracted from the Y_{LH}^w and Y_{HL}^w bands, error-corrected and decrypted using the public key.

3.6. Authentication

Once the watermarks have been extracted, the watermark and authenticator, \hat{W}_{LH} and \hat{W}_A , are computed from the received image as in section 3.2. W_{HL}^e is decrypted to create W_A^d using the appropriate key. Authentication matrices A_{LH} and A_A are generated as shown in equations 12 and 13.

$$A_{LH}(i, j) = \hat{W}_{LH}(i, j) \otimes W_{LH}^e(i, j) \quad (12)$$

$$A_A(i, j) = \hat{W}_A(i, j) \otimes W_A^d(i, j) \quad (13)$$

Visual inspection of A_{LH} can provide some information about the spacial locality of image distortions. Authentication statistics, R_{LH} and R_A may be calculated using equations 14 and 15.

$$R_{LH} = \frac{\sum_{i=1}^{\lceil \frac{M_x}{8} \rceil} \sum_{j=1}^{\lceil \frac{M_y}{8} \rceil} A_{LH}(i, j)}{\frac{M_x}{8} \cdot \lceil \frac{M_y}{8} \rceil} \quad (14)$$

$$R_A = \frac{\sum_{i=1}^{\lceil \frac{M_x}{8} \rceil} \sum_{j=1}^{\lceil \frac{M_y}{8} \rceil} A_A(i, j)}{\frac{M_x}{8} \cdot \lceil \frac{M_y}{8} \rceil} \quad (15)$$

A threshold can be applied to R_{LH} and R_A to quantify the amount of damage to the image. Errors in R_A will be near 50% in the case of errors in W_{HL}^e or the key used to decrypt it. If there is a small error, it is due to differences in the relationships between the DCT coefficients used to calculate the watermark. If $R_A \approx 0$ the sender can be positively identified.

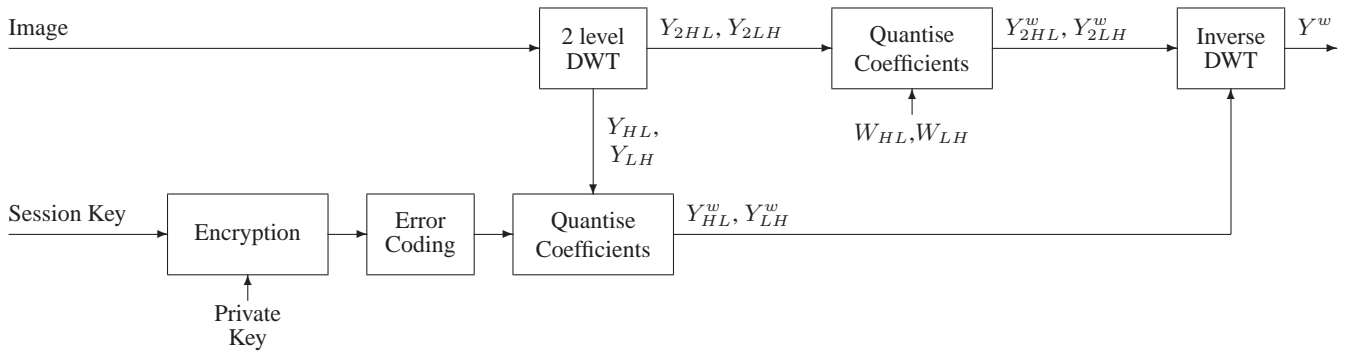


Figure 3: Watermark embedding block diagram

4. Implementation

An implementation of the algorithm described in section 3 is currently underway and in the final debugging stage. The primary implementation language is Matlab, but C libraries are used for the DCT, pseudorandom number generator and DES encryption modules.

ISAAC [5] was chosen as a pseudorandom number generator. It requires an 8192-bit seed (256 32-bit integers). However, a 256-bit session key is used, as it is reasonably secure and small enough to embed more robustly. To make the 8192-bit random seed, the session key is repeated 32 times.

DES encryption [6] is used as the asymmetric encryption algorithm.

Reed-Solomon codes are used for error-correction coding. The code rate depends on the amount of data to encode and the number of bits that can be embedded, as described in section 3.

5. Test Framework

A test framework has been developed to automate the testing of watermarks in general and the one described in this paper in particular.

This framework takes a range of different images and watermarks each several times with different amplitudes and keys. The watermarked images are subjected to a variety of transformations including JPEG compression, region replacement, region blurring and certain image enhancements, as shown in figure 4.

All the processed images have their watermarks extracted and statistics are collected and analysed. As well as the error between the extracted and regenerated watermarks, which is the output of the watermark authentication algorithm, the test framework gives the error between both the regenerated and extracted watermarks and the original embedded watermark. This information is useful in debugging the watermarking algorithm by showing where the errors are introduced.

6. Testing and Results

The algorithm was tested using three sample images, five different sets of keys and δ ranging from 1 to 5. An example of a watermarked test image is shown in figure 5. Each of these 75 watermarked images was then transformed in three different ways. The entire image was sharpened, as an example of a content-preserving enhancement, a portion of the image was blurred as an example of a subtle malicious alteration and a por-

tion of the image was replaced as an example of a less subtle malicious alteration.



Figure 5: Watermarked test image

There were a total of 300 watermarked images after this step. An example of an altered image is shown in figure 6, with Lena's face blurred somewhat in an attempt to make her less recognisable.



Figure 6: Forged image, face blurred

Once the images had been manipulated, they were JPEG compressed with qualities of 40%, 50%, 60%, 70%, 80%, 90%

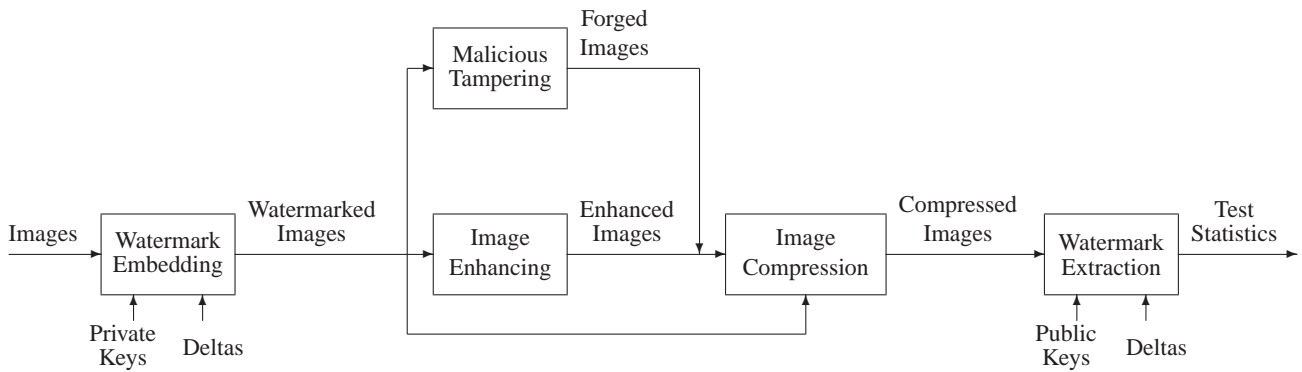


Figure 4: Testing framework block diagram

and 100% for a total of 2400 images, including compressed and uncompressed versions of each of the manipulated images.

Analysis of this data showed that the algorithm could consistently authenticate JPEG compressed images of 80%, but ran into trouble with lower qualities. No images compressed at less than 60% were able to be authenticated, however. Increasing the watermark amplitude (δ) did increase robustness to JPEG compression somewhat, but resulted in higher levels of visual degradation.

Localised changes in the image data, such as the blurring and replacement of portions of the image, result in a lower overall error rate in the extracted watermark, but the errors are clustered in the areas which have been altered and this can be used to determine both that the image has been tampered with and which portions of the image are forged, as shown in figure 7, where the white pixels indicate a bit error in the extracted watermark.



Figure 7: Watermark error pattern in forged image

7. Conclusion

While malicious alterations to an image often result in smaller overall errors in the extracted watermark than lossy compression, they are localised to the areas in the image that have been modified, making forgeries easier to distinguish from compression-damaged images.

While watermarking of images does go some way toward detecting tampering and authenticating the source of an image, it should be noted that it is not foolproof and should be used

cautiously. Stronger forms of authentication, such as digital signatures, should be used where possible.

8. References

- [1] A. Tirkel, G. Rankin, R. van Schyndel, W. Ho, N. Mee, and C. Osborne, "Electronic water mark," in *Proc. DICTA'93 - Digital Image Computing: Techniques and Applications*, December, pp. 666–672.
- [2] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne, "A digital watermark," in *IEEE Signal Processing Society 1994 International Conference on Image Processing (ICIP'94)*, pp. 86–89.
- [3] M. Barni, F. Bartolini, V. Cappellini, A. Piva, and F. Rigacci, "A M.A.P. identification criterion for D.C.T.-based watermarking," in *9th European Signal Processing Conference (EUSIPCO'98)*, Island of Rhodes, Greece, 8–11 1998, pp. 17–20. [Online]. Available: citeseer.ist.psu.edu/barni98mp.html
- [4] Y. Zhao, P. Campisi, and D. Kundur, "Dual domain watermarking for authentication and compression of cultural heritage images," *IEEE Transactions on Image Processing*, vol. 13, no. 3, pp. 430–448, 2004.
- [5] R. J. Jenkins, Jr., "ISAAC," in *Fast software encryption: third international workshop Cambridge, UK, February 21–23, 1996: proceedings*, ser. Lecture Notes in Computer Science. New York, NY, USA: Springer-Verlag Inc., 1996.
- [6] National Institute of Standards and Technology, *FIPS PUB 46-3: Data Encryption Standard (DES)*. Gaithersburg, MD, USA: National Institute for Standards and Technology, Oct. 1999. [Online]. Available: <http://www.itl.nist.gov/fipspubs/fip186-2.pdf>
- [7] F. Hartung and M. Kutter, "Multimedia watermarking techniques," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1079–1107, July 1999.