A modular framework for color image watermarking

This copy represents the peer reviewed and accepted version of paper:

Marco Botta, Davide Cavagnino, Victor Pomponiu, "A Modular Framework for Color Image Watermarking", Signal Processing (2016), pp. 102-114

DOI 10.1016/j.sigpro.2015.07.018

A Modular Framework for Color Image Watermarking

Marco Botta[1], Davide Cavagnino[*1], Victor Pomponiu[2]

[1] Dipartimento di Informatica, Università degli Studi di Torino
Corso Svizzera 185, 10149 Torino, Italy

{marco.botta, davide.cavagnino}@unito.it

[2] Information Systems Technology and Design, Singapore University of Technology and Design, 138682, Singapore

victor.pomponiu@ieee.org, victor.pomponiu@gmail.com

Abstract. We present an algorithm for fragile watermarking of color, or multi-channel, images either in uncompressed format, in lossless compressed format, or in compressed format with locally compressed units (like JPEG). The watermark is embedded into the Karhunen-Loève transform (KLT) coefficients of the host image, and the security of the method is based on the secrecy of the KLT basis computed from a corresponding secret key image. The watermark bits may be embedded with various methods, in particular the use of syndrome coding has proven flexible and effective. Genetic Algorithms (GAs) are used to find the optimum pixel modification leading to the watermark embedding. The resulting watermarked images have shown to have both a high objective quality (in terms of PSNR and SSIM) and a high subjective quality (tested by a group of observers). At the same time, the watermark extraction is very sensitive to small (±1 intensity level for single pixel) modifications, ensuring image authentication with very high probability.

Keywords: color image; compressed image; information hiding; fragile watermarking; genetic algorithms; Karhunen-Loève Transform.

1. Introduction

Different types of data, like images, videos and sounds are evermore widely distributed thanks to computing power and network capacity. At the same time new malware may cancel the security and integrity of these media. For example, unauthorized copy, forgery and tampering are possible attacks to the digital objects.

* Contact author.

To contrast the possible risks many techniques have been developed in the field of computer security: for example, digital signatures guarantee integrity and authenticity; message authentication codes may also be used when non-repudiation is not required.

Digital watermarking defines a broad family of methods having in common the characteristic of embedding a signal into a digital object. Depending on the requirements of the application at hand, this signal may be designed to be altered at the minimum modification of the object or, conversely, to resist modifications aimed at its removal.

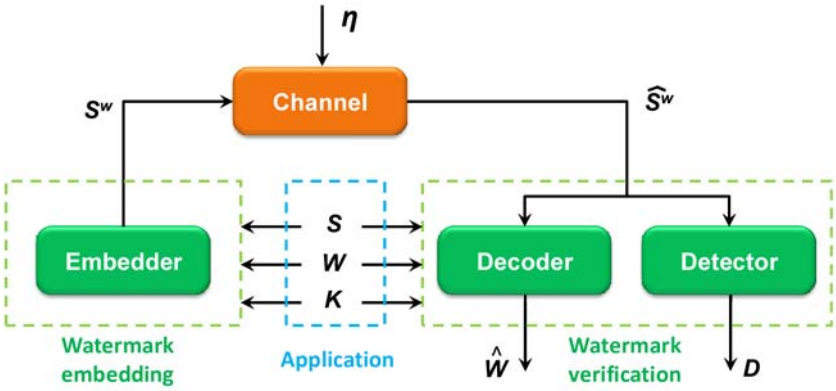

Figure 1: The general representation of the watermarking from a communication point of view: s denotes the host signal, W is the watermark signal, K is the secret key, S^w is the watermarked signal, η is the generic modification, $\widehat{S^w}$ is the possibly perturbed signal, $\widehat{W}$ is the extracted watermark and D is the detector decision.

It is common to give a high level description of watermarking techniques required by an application by defining two phases: the embedding phase and the verification phase, separated by a transmission, as shown in Figure 1; in this context, transmission possibly means sending data over a communication channel and/or storing data on a media. In the embedding phase a function is used to modify the host signal S (or some of its features, like linear transform coefficients) with the objective of hiding a watermark W: typically, to improve the security of the scheme a secret key K is used to control how W is stored in S. The output of the embedder is a watermarked signal S^w. During the transmission, S^w may be subject to modifications η due, for example, to noise, compression and/or filtering. At the receiving end, the application for which the method is developed may require one or both of two possible results from the watermark verification phase (note that these results may be obtained sequentially, independently or at the same time according to the specific watermarking algorithm): 1) an estimation $\widehat{W}$ of the watermark by a decoder, and/or 2) a Boolean decision D whether or not W is correctly present in the possibly modified signal $\widehat{S^w}$. The verification phase

always requires the key K and the signal $\widehat{S^W}$, but some methods may also need the watermark W and/or the original signal S.

The application domain defines the characteristics the watermark must satisfy. Watermarks may be *robust* or *fragile*. A robust watermark is designed to be detectable even if the digital object containing it is (maliciously) modified. Conversely, a fragile watermark is designed to be altered at the minimal modification of the object. Typical applications are copyright protection for robust watermarking and content authentication for fragile watermarking.

Watermarking algorithms that need the host object (i.e. the original un-watermarked object) when verifying the presence of the watermark are called *non-blind* (or *informed*) whilst methods that only use the watermarked object are called *blind*.

Digital objects may be represented in different domains: for example, a sound may be represented as samples in the time domain, or in the frequency domain obtained through Fourier analysis. As a direct consequence of this, a watermark may be embedded considering one (or more) of the possible representations of the object. In the case of images, watermarks are typically inserted in the *spatial domain*, i.e. the pixels of the image, or in a *frequency domain*, like the Fourier Transform domain or the Discrete Cosine Transform domain. Also, other domains have been exploited for images, like the *fractal* and the *Singular Value Decomposition* (SVD) domains.

A further characteristic of watermarking algorithms is the ability to restore the host image from the watermarked image: this is called reversibility and the algorithm possessing it is called *reversible*. This feature is typically required by algorithms applied in the medical field.

The present paper presents *a non-reversible algorithm for the fragile blind watermarking of color images, both for lossless compressed images and for lossy compressed (at block level) images*: in particular we demonstrate the application on the JPEG compressed format. Due to the modular structure of the proposed algorithm, and its composition of basic units, we call it Multichannel Image Modular Integrity Checker (MIMIC). Its main characteristics are:

- *detection* of image modifications: the watermark is highly sensitive to even small changes (one intensity level in one color channel) of the image;
- *localization*: the modification is identified at block level (i.e. a group of contiguous pixels);

- *invisibility*: the watermark signal is imperceptible for general applications of the image, in particular it is invisible to humans, due to the high PSNR (>55 dB) of the resulting images. Thus the proposed algorithm is tailored to many application scenarios; but due to the non-reversibility, the host image cannot be restored, so this algorithm cannot be used when the original image is required, like in all medical applications;
- *security*: due to the methodology and the secret key, the watermark is *secure* against various possible attacks; in particular, we will show that it is able to detect transplantation, birthday and cut-and-paste attacks.

The algorithm introduces the following new features with respect to previously published fragile watermarking algorithms:

- it extends our previous algorithm for gray-scale images to color images;
- it may use syndrome coding for embedding the watermark bits, in order to increase the objective quality and to ease the embedding effort;
- it proposes a modular architecture that increases flexibility in improving the functionalities and allows for application customization of the performance;
- it improves the objective quality of the watermarked images w.r.t. other fragile watermarking algorithms.

The following sections will present a number of scientific works related to watermarking that we consider pertinent to our algorithm (Section 2) and the set of mathematical tools we applied in our algorithm (Section 3). The core of our work is the algorithm developed that we detail in Section 4 and whose performance is shown in Section 5 where many experimental results over a large set of (publicly available) images are presented. Our conclusions are drawn in the final section where we also discuss the obtained results. An Appendix contains details about the mathematical tools we used.

2. Related works

In general, watermarking schemes devised for gray-scale images can be extended and adapted to color images although several approaches have used the color information as an independent feature within the watermarking process in order to achieve an imperceptible watermark.

Depending on how the host image is perturbed during the embedding phase, state-of-the-art methods which are used for color image watermarking can be categorized into two broad areas: spatial domain techniques and transform domain techniques [13].

In spatial domain the embedding is performed by directly modifying the pixel values. Van Schyndel et al. [45] proposed one of the first spatial domain watermarking scheme, known as Least Significant Bit (LSB) embedding. Its main idea is to insert a fragile watermark in the pixel low order bits by perturbing them with a pseudo-random noise sequence. Kutter et al. [24] introduced another method which takes into account the color sensitivity of the human visual system (HVS): since the HVS is less sensitive to the changes of the image blue channel of the RGB color space, the method alters its pixels through an additive embedding rule. It is worth mentioning that, despite its limited imperceptibility and non-blindness, this method was the first robust watermarking approach that was explicitly designed for color images. Several works have been proposed to improve the performance of Kutter's technique. For instance, [48] enhances the robustness of the scheme by using a computational intelligence technique to choose the features of the embedded watermark with respect to the watermarked image.

In [11] Basu et al. present an algorithm for fragile watermarking of color images by embedding a secret color image. The pixels bits of the secret image are embedded in place of the 2 LSBs of every host image pixel. The authors claim a PSNR for the three channels ranging from 32.018 dB to 46.685 dB, and a complete correlation between the inserted and extracted watermark image (in case of no attack). We note that, apart from the average quality easily predictable (44.15 dB after randomly modifying the 2 LSBs of an 8 bit-per-channel image), the system does not provide enough security because the watermark signal does not depend on the host image: any host image may be made authentic by simply substituting its 2 LSB planes with the 2 LSB planes of any other authentic color image. Anyway, by making the watermark dependent on the host image one would prevent a number of similar attacks.

The above cited problem is indeed not present in [32], where a secret binary logo is used to authenticate a color image. The host image is split into blocks and the fragile watermark is embedded into the LSB of the pixels. To prevent security attacks, the watermark is a function of the binary logo and the pixel values of the blocks along with other properties of the image and of every single block, avoiding copy-and-paste and transplantation attacks. By embedding the watermark into all the LSBs of the host image, we

expect an average PSNR of 51.14 dB (the authors report the values of 51.9 dB and 52.4 dB for two classical images).

In general, transform domain watermarking schemes use the Discrete Cosine Transform (DCT) as the medium to insert the watermark in digital images [35][2][3][44][27][21]. These algorithms comply with most of the design guidelines which reflect both strengths and limitations of watermarking systems. For example, in [35] a DCT-based scheme which takes into consideration the statistical dependency between the color bands has been proposed. For each color band, a set of coefficients is selected and then perturbed to insert the watermark. The strength of the watermark is adjusted depending on the sensitivity of each color channel. The optimal detection is sought with respect to the Neyman-Pearson criterion, i.e. the minimization of the probability of not detecting the watermark. Subsequently, the authors lowered the probability of missing the watermark during the detection phase by assuming a different distribution for the host signal [2][3]. Vidal et al. [44] insert the watermark (a minimum length sequence) into the DCT coefficients of the color components. In order to assure the imperceptibility and detectability of the watermark, each symbol of the sequence is embedded randomly in middle frequency DCT coefficients using the amplitude of the coefficients as side information. One problem that can occur with the DCT approach is that it may induce noise in images when watermarks with large payload need to be inserted. To avoid this problem, Meng et al., [29] have proposed an embedding method in the DCT domain based on phase-shifting that enhances the imperceptibility of a large concealed watermark.

The Discrete Wavelet Transform (DWT) has extensively been used for watermarking color images [30][16][28]. The main advantage of using DWT for robust watermarking schemes is that it better takes into account the local image characteristics at different resolution levels which can significantly improve the robustness and imperceptibility of the watermark. By adopting a frequency spread of the watermark together with a spatial localization, these schemes are able to better conceal the watermark within the salient components of the image [13]. Each level of the DWT decomposition generates four bands denoted by LL, HL, LH, and HH. The LL sub-band is further decomposed to obtain another level of decomposition. This procedure continues until the desired number of decomposition levels is reached. The LL sub-band represents the information at all coarser scales. Generally, a pseudo-random number (PRN) sequence is embedded in a selected set of the DWT coefficients and the strength of the embedded watermark is adjusted

with the help of scaling factors for each band. For example, in [30] the authors proposed to compute the local entropies of the image wavelet coefficients to control the imperceptibility and to achieve an enhanced detection and localization of the watermark.

Recently, several watermarking methods that combine different domain transforms, like DCT and DWT, have started to appear. For example, Zhao et al. [49] proposes a scheme based on a DCT-DWT dual domain, which is able to jointly compress and authenticate cultural heritage images. The watermarking framework has two key components: the first is a semi-fragile authentication and tamper detection watermark, while the second component is a chrominance watermark (that contains the color information of the image) employed to improve the efficiency of the compression process. The flexible watermark approach is designed by exploiting the orthogonality of the dual domain used for authentication, color decomposition and watermark insertion. In the same vein as [49], Kougianos et al. [25] have investigated several DCT-DWT domain dual (robust-fragile) watermarking methods which are embedded in a hardware processor in order to achieve low power usage, real-time performance, reliability, and ease of integration with existing consumer electronic devices. Other common transforms such as the Fourier transform [41], [42], the fractional Fourier Transform [38], the Hadamard transform [31] or the Schur transform [39] have been used in the cited robust watermarking schemes to lower the computation cost and improve the authentication assessment of the watermark during common image processing attacks.

The Stirling Transform (ST) is used in [17] to embed a fragile watermark into a color image. The watermark (a bit string derived from a digital object like text or an image) is embedded into the LSBs of pairs of ST components. The ST is applied to pairs of pixels and, after embedding, the inverse ST is applied to the watermarked components obtaining the watermarked pixels. The method embeds two bits into every ST component, leading to a payload of 2 bits-per-byte. The average image quality is about 43.5 dB. The method authenticates an image as a whole thus no localization of a tampered area is available.

In the field of data hiding moment invariants have found extensive applications due to their ability to achieve a detailed representation of the host content. For instance, Savelonas et al. [38] make use of certain fractional Fourier (frF) coefficients to insert a noise-resistant watermark. The selection of the suitable frF features is guided by image moments. By adopting the frF transform the proposed scheme achieves simultaneously a better embedding domain, since it exploits the spatial and frequency information of the

image, and a higher security for the watermark, i.e., an enlarged secret space that includes also the frF angles. It is worth pointing out that the algorithm requires a down sampled version of the host image in order to recover the watermark during the verification phase. Experimental results demonstrate that the scheme achieves a good robustness against noise-based alterations while offering acceptable quality for the watermarked image (i.e., 30.7 dB).

It is worth mentioning that some works exploit the possibility to use quaternions for storing the color information of pixels. In [4] the color pixels are interpreted as quaternions and the Quaternion DFT is applied to compute coefficients which are marked with the Quantization Index Modulation; a watermarking scheme is obtained that is judged suitable for data hiding by its authors. In [42] the DFT is applied for the robust, non-blind watermarking of color images: two approaches are presented, both aimed at minimizing the watermark visual impact giving more strength to the yellow-blue component; the first approach is based on the Spatio-Chromatic DFT whilst the second applies the Quaternion Fourier Transform (QFT).

Differently from the above schemes, several researchers have investigated the embedding of robust watermarks into color images through color quantization or histogram modification, which possess interesting signal processing properties [1], [40]. However, the major issues regarding color histograms are the non-linear association between its representation and the pixels, the complexity of the representation, the existence of multiple histograms (one per color band) and the intrinsic correlation between its color components. To solve these problems, Roy and Chang [37] use the Earth Mover Distance to modify an image in order to reach a target histogram. Furthermore, to lower the complexity of the representation, the embedding uses merely the color histogram extracted from the luminance-chrominance (i.e., YC_bC_r) color spaces.

On the other hand, color quantization-based watermarking schemes aim to generate a set of colors such that the observed difference between the original image and the quantized one is as small as possible. Such schemes usually involve two steps: the first consists of selecting a suitable color palette while the second phase consists in reconstructing the image by substituting each original color with the most similar one found in the palette. A plethora of quantization-based color watermarking methods, using different color spaces, have been proposed in the past. For instance, Pei and Chen [36] present an approach which embeds two different types of watermarks in the host image by exploiting the *Lab* color space. A fragile watermark,

generated by modulating the indexes of a color palette, is embedded into the chromatic plane while the robust watermark is embedded into the luminance component. The scheme shows good authentication ability and fine localization of slight modifications applied to the watermarked image.

The use of symmetric and asymmetric (i.e. signature) techniques has been applied in [14] to detect tampering and to authenticate an image. Starting from a work from Wong [46] the authors propose a technique to thwart Vector Quantization (VQ) attacks. Briefly stated, the image is divided into blocks and the pixels LSBs are used to convey authentication information (Message Authentication Codes, MACs, or signatures) of the block itself and of a hierarchy of larger blocks obtained merging blocks from lower layers: thus the LSBs will contain a complete MAC or signature of the block and part of the authentication information for the larger blocks they belong to. In this way, a large block will be authenticated using data from some of the LSBs of blocks composing it. This method has a very high security, but its disadvantages are the resulting low PSNR and, due to the necessity to have secure MACs and signatures, the large block size, reducing, in some measure, the localization capability. Anyway, we think this method is secure, so we will compare its performances with those of our algorithm.

In this paper we present an extension of the work described in [7], along several directions, that completely revises the workflow and introduces new techniques and algorithms. The previously developed algorithm was limited to gray-scale images and used an embedding method (also available in the present algorithm) that resulted in a lower objective quality (i.e. lower PSNR). Moreover, the possibility to generate JPEG compressed images is an improvement that enlarges the range of image file formats to which our algorithm applies. Finally, the added possibility to combine various methods in the processing steps represents a significant improvement towards the flexibility of the proposed framework.

3. The Watermarking Algorithm Building Blocks

Before describing in details the architecture of MIMIC, we briefly introduce in this section the three basic concepts and motivations on which the proposed watermarking scheme is founded, and forward the reader to the Appendix for a thorough description of these topics.

First of all, in order to make the algorithm more secure, the watermark is inserted into a transformed space computed by applying the Karhunen-Loève Transform [†] (KLT): differently from common transformation, such as DCT or DWT, the nice feature of this transformation is that the obtained space depends on an input secret key and therefore the obtained coefficients cannot be easily computed, making the recovery of the watermark more difficult for an attacker. In the Appendix, we describe several ways in which the transformed space can be derived from a secret input image, either gray-scale or color image.

Secondly, in order to guarantee fine grained tampering localization, the authentication is performed at block (subimage) level by storing q watermark bits into r coefficients (in general $q \neq r$), computed by applying the KLT to a block of pixels. Watermark bits may be stored directly into the coefficients, or using different functions, i.e., syndrome coding and weighted modulo sum. Moreover, to further improve security, each block can be transformed by using a different KLT transformation matrix. These methods are discussed in detail in the Appendix.

Finally, in order to obtain high quality watermarked images, an optimization algorithm, specifically a Genetic Algorithm, is used to find the smallest alteration to the image pixels that results in the least distorted (according to a quality function) watermarked image. We chose Genetic Algorithms because they are easy to implement in this context and are non-deterministic, thus introducing a further degree of unpredictability (e.g., inserting the same watermark bits in two identical blocks will result in two different watermarked blocks with high probability).

4. The Modular Architecture of MIMIC

In this section we describe the components and workflow of the algorithm that from a host image I_o return a watermarked image I_w based on a secret KLT basis defined by a key image I_k. Every component of the algorithm will be called *unit*; in each unit different data may be processed and different procedures may be applied to the data by *modules*: for example, the key generation unit may operate on a gray-scale image or a color image by means of two different modules (this is coherent with [5] in which we had only one module per unit, so only the term module was used).

[†] The Karhunen-Loève Transform is also known as Hotelling Transform, or Principal Component Analysis, or Eigenvector Transform.

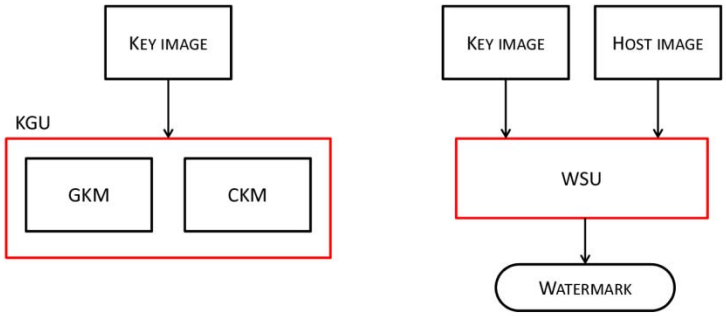

Figure 2: Generation of secret KLT basis and watermark: Key Generation Unit, KGU (GKM: Gray-scale Key Module, CKM: Color Key Module), and Watermark Supply Unit, WSU.

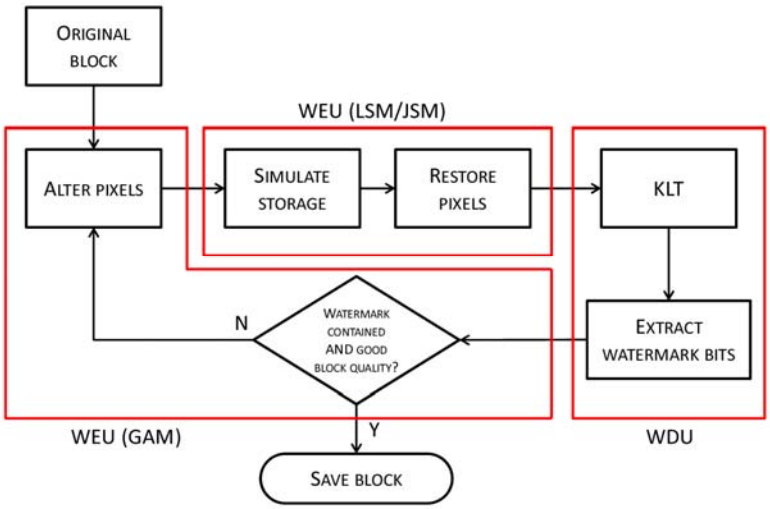

Figure 3: Subimage processing steps (WEU: Watermark Embedding Unit, LSM: Lossless Storing Module, JSM: JPEG Storing Module, WDU: Watermark Distilling Unit, GAM: Genetic Algorithm Module).

In Figure 2 and Figure 3 a high level scheme of the watermarking procedures is presented.

The watermark *embedding phase* is performed by the Watermark Embedding Unit (WEU) which requires a key (generated by the Key Generation Unit, KGU) and a host image to be watermarked. Using the Watermark Supply Unit a per-host-image watermark is created; then, the watermark bit string is split into chunks and each one is inserted by the WEU into the corresponding block. For every subimage, if it does not already contain the watermark, a search through the space of possible pixel modifications is performed. This search may be achieved by any algorithm that looks for an optimum determined by an evaluation function. We isolated this search in a module. Presently, we use a Genetic Algorithm (GA), so we will describe a GA Module (GAM). The use in this phase of the Watermark Distilling Unit is purely instrumental to the GAM, to verify if the watermark has been inserted.

The watermark *extraction phase* is executed by means of the Watermark Distilling Unit (WDU) using the same key (provided by the KGU) employed for embedding (the method is symmetric). The extracted watermark is compared by the Tamper Detection Unit (TDU) with the one that should be present in case of no image alteration, allowing for an identification of the tampered blocks.

The various units are described in the following subsections.

4.1. Key Generation Unit (KGU)

This unit is composed of two modules, one for gray-scale images and one for color images. Its objective is to derive a KLT basis from a secret key image I_k. The results of this unit must be used during both the embedding and the decoding/verification, and must be kept secret because knowledge of the KLT basis allows to derive the secret space in which the watermark is embedded (allowing an attacker to forge the watermarked image without being detected). The computations involved in this unit are required when using a new key image (that will be typically applied to watermark many host images).

4.1.1. Gray-scale Key Module (GKM)

When the key image is gray-scale, it is divided into contiguous non-overlapping blocks of size $n \times n$, and a KLT basis $\boldsymbol{A}$ with average $\boldsymbol{m_A}$ (upper part of Figure A1) can be computed.

4.1.2. Color Key Module (CKM)

A color image is divided into contiguous non-overlapping blocks of size $n \times n$, then two different KLT procedures may be applied:

1. the three RGB channels are considered as three independent gray-scale images, and three KLT bases for $n \times n$ blocks are derived: $\boldsymbol{A_R}$, $\boldsymbol{A_G}$, $\boldsymbol{A_B}$ with average blocks $\boldsymbol{m_R}$, $\boldsymbol{m_G}$, $\boldsymbol{m_B}$ (upper part of Figure A2). This procedure may be also used in the case of watermarking a gray-scale host image: anyone of the three kernels may be employed, e.g., the one from the green channel;
2. the color information is integrated with the spatial information defining blocks of size $n \times n \times 3$ to obtain an average subimage $\boldsymbol{m}$ and a kernel $\boldsymbol{A}$ of $3n^2$ basis images (upper part of Figure A3).

4.2. Watermark Supply Unit (WSU)

This unit has to be called for every host image to be watermarked to create the bit string used for authentication. The watermark for a host image I_o depends on the key I_k and I_o itself. This avoids copy-and-paste attacks, transplantation attacks [10] and VQ attacks.

Our idea is to use the values of a set P_1 of pixels in I_k as pointers to a set of pixels Q of I_o and then, in turn, use the values of the pixels in Q as pointers to other pixels in I_k: let us call this last set P_2. The values of the pixels in P_2 are concatenated with the host image size (width and height) and used to initialize a cryptographic hash function (c.h.f.) like SHA-3 [23], repeatedly called to generate a sufficient number of bits to compose the whole watermark W. The use of the host image sizes allows to detect image cropping: in case of this attack, all the blocks will likely be evidenced as modified because a different watermark will be generated during the verification phase (because a cropped image will have different width and/or height).

Given that this unit is also called by the verification unit (see the following subsection on the Tamper Detection Unit), then the values of the pixels in Q must not be altered by the embedding process of the successively described Watermark Embedding Unit (to compute the same watermark W when extracting and verifying its presence). Moreover, the alteration of one of the pixels in Q (performed by an attacker) will affect the localization ability of the verification module (but the tampering of the image would not go undetected). A reasonable choice is to let the set Q be small, e.g. four pixels, whilst the cardinality of the set P_2 may be adjusted to further increase the randomness of the watermark. The coordinates of the pixels in P_1 are fixed a priori (i.e. public), and the pixel values are multiplied by the dimensions of I_k to increase randomness. In general, to obtain a meaningful pixel coordinate, the resulting numbers are computed modulus the size of the pertaining image.

4.3. Watermark Distilling Unit (WDU)

The watermarked image I_w is divided into contiguous non-overlapping blocks of size $n \times n$. Depending on the degree of security of the method (we discuss this issue later) groups of a pre-defined number q of bits are consecutively extracted from each block (e.g. considering the blocks in raster scan order, as the WEU does). The q bits contained in each block are obtained accordingly to the embedding method used: we consider three different embedding methods, implemented into three modules that we present in the

following. We confined every extraction method in a different module, so any other embedding/extraction may be foreseen and considered as a new module, the only constraint being to work on image blocks.

The three bit extraction modules consider a set of *r* KLT coefficients obtained by applying the KLT to a block and from them provide a sequence of q bits. The orders of the coefficients may be the same for every block, or may change (for example, driven by a key derived from I_k) from one block to another.

4.3.1. KL transform Module (KLM)

This module is called before any of the bit extraction modules. Its objective is to perform the Karhunen-Loève transform using the data (orthonormal basis and average block) provided by the KGU and transform a block of n^2 RGB color pixels into $3n^2$ coefficients.

4.3.2. Smart Reduction Module (SRM)

If a reduction of dimension is required and if the transform basis has size n^2 as in the cases shown in Figure A1 and Figure A2, then it is possible to apply a reduction of dimensionality from $3n^2$ to n^2 on the extracted coefficients according to Equations (A5) or (A6): a linear combination of the coefficients obtained from the three independent R, G, B channels is performed. Calling $\boldsymbol{y}_{\boldsymbol{C}}^{(\boldsymbol{i})}$ the i-th coefficient from channel $C \in \{R, G, B\}$, then the resulting n^2 coefficients are computed as:

$$\boldsymbol{Y}^{(i)} = \boldsymbol{w}_{\boldsymbol{R}}\boldsymbol{y}_{\boldsymbol{R}}^{(\boldsymbol{i})} + \boldsymbol{w}_{\boldsymbol{G}}\boldsymbol{y}_{\boldsymbol{G}}^{(\boldsymbol{i})} + \boldsymbol{w}_{\boldsymbol{B}}\boldsymbol{y}_{\boldsymbol{B}}^{(\boldsymbol{i})}, \qquad i = 1..n^2$$

Note that the use of this module is optional.

4.3.3. Bit Collect Module (BCM)

This module applies Equation (A7) to extract a bit from a coefficient; in this case, parameters of the module are the orders of the coefficients and the bit position *p*: for every block, from a set containing the orders of the r ($= q$) coefficients, r bits are returned.

4.3.4. Syndrome Coding Module (SCM)

This module works in a similar manner to BCM but recovers the q watermark bits per block as the syndrome of a codeword of r bits; the r bits are retrieved from r KLT coefficients as indicated by Equation (A7), where the position p and the coefficients orders are pre-defined. The idea driving the use of this module is to reduce the average number of required changes to the coefficients, dependent on the covering

radius R of the used code. Presently, we experimented the Golay code [24, 12, 8] (obtained from the Golay code [23, 11, 7] by adding a parity bit), the Hamming code [7, 4, 3], the Hadamard code [32, 6, 16], the BCH code [31, 11, 11] and the BCH code [31, 6, 15], where the first two elements of every triple $[r, r-q, d]$ are related to the number of coefficients r to be used to convey q (syndrome) bits of the watermark. In this context it is also possible to use combinations of the previous codes to store in every block a number of bits not directly possible with a single code: for example, in order to store 15 bits, one can use either the Golay code [24, 12, 8] along with the Hamming code [7, 4, 3] applied to 24 + 7 = 31 coefficients, or 5 times the Hamming code [7, 4, 3] applied to 7 × 5 = 35 coefficients.

The covering radius R of each used code is reported in Table 1 along with the number of syndrome (watermark) bits carried and the average number of bit modifications (we computed this value assuming a uniform probability distribution of the syndrome values). Using syndrome coding the maximum number of required modifications is R; for some codes, there are syndromes having more than one coset leader (i.e. words with smallest Hamming weight), so some criteria will be developed to choose one (the most trivial one being random choice).

CODE	NUMBER OF SYNDROME BITS	COVERING RADIUS R	AVERAGE NUMBER OF BIT MODIFICATIONS
HAMMING [7, 4, 3]	3	1	0.875
GOLAY [24, 12, 8]	12	4	3.353
BCH [31, 11, 11]	20	7	6.069
BCH [31, 6, 15]	25	11	8.800
HADAMARD [32, 6, 16]	26	12	9.300

Table 1: characteristics of each code used by the Syndrome Coding Module.

4.3.5. Weighted-Sum Coding Module (WCM)

This module extracts q bits representing a number l between 0 and $2^q - 1$ from $r = 2^{q-1}$ KLT coefficients with orders o_1, ..., o_r performing the modulo sum defined in Equation (A11). The value q and the coefficients orders are pre-defined parameters of the module.

4.4. Watermark Embedding Unit (WEU)

This unit is the part that inserts the watermark bits into the host image I_o. The host image is divided into contiguous blocks of size $n \times n$ and for every block the pixels are modified in such a way that "the KLT coefficients *contain* the pertinent q watermark bits". The term *contain* refers to an extraction method of bits from coefficients among the ones presented in the previous section. Each block is thus processed altering its

pixel values in such a way that the extraction (performed according to one of the modules BCM, SCM or WCM, the same that will be used by the WDU) of q bits from r KLT coefficients of the block returns the corresponding watermark bits. Thus, if I_o has size[‡] $l_x \times l_y$, then the watermark supplied by the WSU must have length equal to $q \times l_x \times l_y / n^2$. In this case the image is said to have a payload of q bits-per-block (bpb).

To evaluate the suitability of a modification to a block, the new pixel values should undergo a simulated storage and retrieval: if the image format is lossless (like bitmap) then no variations to the pixels are expected but if the image format is lossy, like JPEG, then the pixel values could change when retrieved after storage. In order to be consistent with the WDU that has available only the watermarked image, a storage of the block must be simulated and applied before extracting the KLT coefficients and the watermark bits. We achieved this by developing two modules, one for lossless image formats and one for the JPEG (JFIF) format; other modules for lossy compressed image formats working on blocks could be developed and used in this unit.

After the bits have been extracted they are compared with the ones we must embed: if they are equal then the q watermark bits are stored in the block (when saved) otherwise another modification to the pixels has to be performed. In this search many other block characteristics may be taken into account, like the resulting block quality and/or the distortion w.r.t. the original block.

Due to the mode of operation of the algorithm we do not allow interactions between blocks, so the JPEG file for a color image is produced with no subsampling (4:4:4): in this way both the luminance (Y) and the color (C_b, C_r) components involve the same set of pixels for every block (in case one used a 4:2:0 subsampling then a single color component block would span 4 gray-scale blocks requiring a more complex operation of the WEU). Moreover, when the embedding is performed on the color channels *independently*, then the JPEG compressor should also work keeping the channels independent (otherwise the part of the watermark already stored in a channel could be removed by a following channel embedding): we solved this issue by generating RGB JPEG images instead of YC_bC_r JPEG images.

[‡] For simplicity we assume that the image dimensions are a multiple of the block dimensions. Different strategies may be implemented to cope with the case of non-multiple dimensions.

4.4.1. Genetic Algorithm Module (GAM)

To perform a search of the optimal solution in the space of all possible pixel configurations, a steady state GA was used.

This module applies a GA that encodes individuals representing pixel value modifications in a block: thus a GA individual is composed of n^2 (or $3n^2$ if SRM is not used) integer numbers, each one expressing the modification to the corresponding pixel in the original block. In the present implementation, to keep the PSNR high, we limited the possible values to the range of integers $[-3, 3]$; notice that a modification generating a value out of the allowable range (lower than 0 or greater than 2^d) is clipped by returning the range limit.

The fitness function takes into account:

1) the Hamming distance between the stored q bits (in the KLT coefficients) and the desired watermark, which must be 0 (i.e. we want the watermark stored in the block);
2) the PSNR, i.e. a function of the MSE between the host block and the resulting block.

Different strategies have been applied to terminate the search: the first condition, as we stated, requires that the watermark is stored in the block; then, when a solution is found, the GA may be run for a certain number of epochs to improve the quality (in terms of PSNR) of the solution.

4.4.2. Lossless Storing Module (LSM)

This module is an identity module that does not alter the block, but simulates the storing and reading of any lossless image storage format.

4.4.3. JPEG Storing Module (JSM)

To produce a watermarked image in JPEG format, the block resulting from the pixel alteration must be compressed and then decompressed to obtain the same pixels that the WDU will receive: in fact the lossy compression may alter some of the block pixels. In this way pixel alterations performed by the search module (GAM) undergo the compression/decompression process and the KLT coefficients are computed from the decompressed block: if the watermark is no longer present, then the GA modification will not be accepted and other modifications will be tried.

4.5. Tamper Detection Unit (TDU)

The TDU (see Figure 4 for a high level scheme) works by extracting the watermark $\widehat{W}$ from the image I_w, as in the WDU, and generates the expected watermark W with the WSU (using the key image I_k and the set of pixels Q in I_w). By splitting $\widehat{W}$ and W into sections of q bits, and comparing the corresponding sections, a tampering in a block can be easily detected, thus providing localization of the tampering at block level.

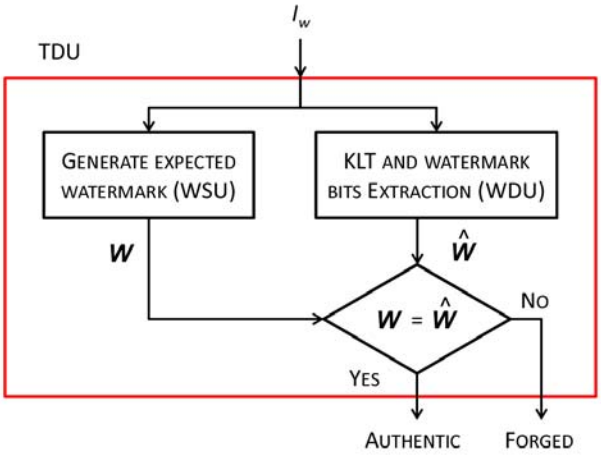

Figure 4: High level scheme of the Tamper Detection Unit (TDU).

5. Experimental results

We will compare the performance of the algorithm with respect to various parameter settings and with two other secure watermarking methods [14][9].

The parameters considered are:

- the execution time (evaluated on Linux workstations, each equipped with 4GB RAM and an Intel(R) Xeon(R) E5410 2.33GHz processor), which gives an idea of the computational complexity of the various module combinations;
- the Peak Signal-to-Noise Ratio (PSNR) of the watermarked image, defined as

$$\text{PSNR} = 10 \log_{10} \frac{\left(2^d - 1\right)^2}{MSE}$$

where MSE (mean squared error) is the average squared pixel difference between the watermarked and the host image, and d is the bit depth of each channel. For an RGB color image we compute the MSE summing the squared differences between pixels in the same position of the three color channels and dividing by three times the number of pixels;

- the Stuctural SIMilarity index (SSIM): this index, introduced in [47], accounts for a value that measures the similarity of two images in a way more similar to a human judgment; it takes into account variances of various areas of the images being compared along with a cross correlation between the same areas. It ranges in the interval $[-1, 1]$, with a value of 1 only if two images are identical.

MIMIC represents a family of procedures that may be used to verify the integrity of a color (or multichannel) image. Given that the various modules may be combined in many ways, the first objective was to find the configurations that obtain the best performance in terms of PSNR, SSIM or execution time. Moreover, according to previous analysis [8] aimed at comparing the performances of the GA with different settings, we chose to set GA parameters to the following values in all experiments: population size set to 100 individuals, crossover probability set to 0.9, mutation probability set to 0.04, termination condition set to best solution does not change for 10 generations or 2000 generations total.

Secondly, we wanted to test the sensitivity of the various combinations. The sensitivity of level D is defined as the percentage of image blocks detected as altered when only one pixel is modified by $+D$ or $-D$ intensity levels in one channel (e.g. red, or green, or blue). We performed experiments for sensitivity levels 1 and 2. To test the sensitivity of MIMIC when the JSM was employed we could pursue two lines: the first one is to alter single JPEG DCT coefficients and verify the ability of detection of our algorithm; we found this method unfair because a modification of a single unit typically results in large pixel changes that are easily detectable. The second, and fairer, method is to suppose the attacker could have access to an oracle capable of altering a JPEG encoding of an image to produce a single pixel alteration of one level in one of the three channels: this is a very powerful assumption for an attacker, sometimes even impossible, but we show later that MIMIC is capable of detecting these changes with a very high probability; the attack is thus performed in an analogous way to those for bitmap images and lossless compressed images.

Then we compared the MIMIC combination of modules giving the best performance with the algorithm [14], which is a secure algorithm that, by using MACs or digital signatures, has a sensitivity of 100% at all levels, and with [9], which produces high quality images.

5.1. Performance analysis

The evaluation of the performance of the proposed algorithm are grouped into tables, depending on two factors. The first one is the type of the resulting image, uncompressed (or lossless compressed) format and JPEG compressed format. Furthermore, we make a further distinction on the localization ability: if the embedding method allows for identifying which channel has been tampered (i.e. no SRM module has been used), then we call this kind *single channel block authentication*, otherwise the method is called *color block authentication*, because it can only detect a block tampering but cannot further identify the channel.

Furthermore, even though we performed experiments on every combination of modules, we reported only those module combinations for which the GAM module always converged to a solution, i.e. if for all the blocks of an image the GA was able to embed the watermark it was considered a success, whilst if for just a single block it was not possible to embed the pertaining part of the watermark it was considered a failure (because the TDU would detect the block as tampered).

We do not report the SSIM because for all the images it was greater than 0.998.

Table 2 reports PSNR, execution times, sensitivity of MIMIC for different settings of payload, embedding method, embedding position and syndrome coding for single channel block authentication, averaged over 500 color images (we feel that in the literature some algorithms even if comparing with others lack experimentation on a large number of cases) taken from [33][34] cropped to 256×256 pixels (to reduce overall computation time).

Payload (bpb)	Module Combination	PSNR (dB)	Time (s)	Sensitivity ±1 (%)	Sensitivity ±2 (%)	Insertion position *p*
9	SCM (Hamming [7,4,3])	**66.73±0.03**	**30.18±0.53**	79.67±0.31	87.62±0.14	–1
9	SCM (Hamming [7,4,3])	64.54±0.1	61.74±1.38	61.82±0.39	81.06±0.17	0
12	BCM (4 bits/channel)	61.67±0.16	57.96±1.03	64.61±0.58	90.95±0.12	–1
12	WCM (4 bits/channel)	63.16±0.49	64.12±2.46	67.67±0.34	86.88±0.13	0
12	WCM (4 bits/channel)	61.75±0.17	57.34±1.01	84.15±0.27	93.98±0.1	–1
12	BCM (4 bits/channel)	61.44±0.43	56.8±6.66	51.15±0.45	70.3±0.32	0
18	SCM (2×Hamming [7,4,3])	62.2±0.08	114.14±1.34	82.2±0.26	95.98±0.06	0
24	BCM (8 bits/channel)	59.28±0.14	86.92±1.01	85.02±0.33	99.2±0.04	–1
24	WCM (8 bits/channel)	61.12±0.11	64.11±1.16	99.48±0.02	**99.61±0.01**	0
24	WCM (8 bits/channel)	61.94±0.07	54.62±0.95	**99.61±0.03**	99.61±0.02	–1
24	BCM (8 bits/channel)	59.06±0.2	86.92±2.45	66.37±0.4	87.74±0.17	0

Table 2: Results for single channel block authentication.

Table 3 reports the same measures for the same set of images when performing color block authentication.

Payload (bpb)	Module Combination	PSNR (dB)	Time (s)	Sensitivity ±1 (%)	Sensitivity ±2 (%)	Insertion position *p*
8	BCM & SRM	**66.42±0.18**	**25.99±0.5**	85.93±0.52	99.21±0.06	–1
8	BCM & SRM	64.33±0.18	28.39±1.07	64.89±1.01	87.08±0.42	0
12	BCM	58.47±0.12	72.97±0.83	76.8±0.92	96.49±0.17	–1
12	SCM (4×Hamming [7,4,3])	58.52±0.14	179.2±2.65	97.05±0.31	99.89±0.04	–1
12	BCM & SRM	64.07±0.25	41.32±0.71	93.89±0.3	99.93±0.02	–1
12	BCM	57.17±0.1	132.93±2.71	48.68±0.68	74.22±0.45	0
12	SCM (Golay [24,12,8])	58.16±0.12	188.78±4.21	68.75±0.56	90.71±0.21	0
12	SCM (Golay [24,12,8])	58.56±0.13	167.84±2.97	95.59±0.27	99.89±0.02	–1
12	SCM (4×Hamming [7,4,3])	58.21±0.12	158.68±3.19	75.5±0.46	93.94±0.14	0
12	WCM	54.34±0.09	280.29±9.4	99.58±0.07	99.71±0.06	–1
12	WCM	57.68±0.13	343.94±5.42	**99.89±0.04**	**99.97±0.01**	0
12	BCM & SRM	62.7±0.17	37.32±1.97	72.43±1.02	93.43±0.29	0
20	SCM (BCH [31,11,11])	55.45±0.13	439.73±15.69	70.71±0.53	93.01±0.16	0
24	BCM	54.62±0.11	255.89±4.1	63.82±0.6	88.65±0.24	0
25	SCM (BCH [31,6,15])	54.69±0.13	402.99±11.92	70.49±0.56	92.94±0.17	0
26	SCM (Hadamard [32,6,16])	54.55±0.13	433.55±13.59	71.33±0.52	93.38±0.15	0

Table 3: Results for color block authentication.

Table 4 and Table 5 report the same measures of Table 2 averaged over a set of 100 color images taken from [33][34] cropped to 256×256 pixels: the former for JPEG compressed images with quality 100% and the latter for JPEG compressed images with quality 95%.

Payload (bpb)	Module Combination	PSNR (dB)	Time (s)	Sensitivity ±1 (%)	Sensitivity ±2 (%)	Insertion position *p*
8	BCM & SRM & JSM	57.65±0.25	**269.86±66.29**	85.42±0.26	99.21±0.04	–1
8	BCM & SRM & JSM	57.04±0.22	288.74±9.23	63.5±14.58	86.95±5.11	0
12	SCM (Golay [24,12,8]) & JSM	52.15±1.67	712.87±34.67	77.71±0.86	95.63±0.28	0
12*	BCM (4 bits/channel) & JSM	55.21±0.13	561.33±37.38	79.03±0.59	96.8±0.16	–1
12	BCM & JSM	54.7±0.12	337.63±5.48	50.73±1.07	77.67±0.69	0
12*	BCM (4 bits/channel) & JSM	**58.7±0.13**	371.46±15.23	47.15±0.68	69.99±0.43	0
12	SCM (Golay [24,12,8]) & JSM	55.24±0.13	701.02±42.25	**95.46±0.2**	**99.9±0.02**	–1
12	BCM & SRM & JSM	56.82±0.16	334.49±12.17	74.98±9.85	94.68±2.09	0
24*	BCM (8 bits/channel) & JSM	57.59±0.13	488.86±57.07	85.79±0.18	99.19±0.03	–1
24*	BCM (8 bits/channel) & JSM	56.89±0.06	520.5±6.49	63.61±0.63	87.41±0.27	0

Table 4: Results for JPEG compressed output images with quality 100%. * identifies single channel block authentication.

Payload (bpb)	Module Combination	PSNR (dB)	Time (s)	Sensitivity ±1 (%)	Sensitivity ±2 (%)	Insertion position *p*
8	BCM & SRM & JSM	43.04±0.83	188.66±60.54	89.76±6.43	99.33±0.18	–1
8	BCM & SRM & JSM	43.04±0.83	**169.34±11.43**	81.42±19.69	93.33±6.8	0
12	BCM & JSM	42.89±0.82	368.99±11.66	50.8±1.64	77.68±1.07	0
12*	BCM (4 bits/channel) & JSM	**43.08±0.88**	334.44±25.65	43.88±7.13	67.26±3.82	0
12	SCM (Golay [24,12,8]) & JSM	42.87±0.82	678.09±62.24	**95.54±0.31**	**99.89±0.02**	–1
12*	BCM (4 bits/channel) & JSM	42.89±0.83	583.92±16.83	79.41±0.84	96.92±0.19	–1
12	SCM (Golay [24,12,8]) & JSM	42.89±0.83	660.49±17.59	77.73±0.85	95.64±0.28	0
24*	BCM (8 bits/channel) & JSM	42.93±0.75	428.09±24.72	60.42±1.41	86.08±0.52	0
24*	BCM (8 bits/channel) & JSM	42.93±0.75	427.05±30.85	85.52±0.26	99.2±0.03	–1

Table 5: Results for JPEG compressed output images with quality 95%.* identifies single channel block authentication.

From these tables, depending on the requirement of the application, one can choose which module combination results in the best performance for a given evaluation criteria: we highlighted the best performance for each criteria in boldface. As a suggestion, we highlighted in green rows that we think are a good compromise among all parameters. Otherwise, if time is not a concern, WCM with 12 bits payload has very good quality and practically total sensitivity (see Table 3) that makes it a good candidate when protecting images in offline systems (i.e. no real time response required in embedding).

Anyway, as reported in [7], the complexity of both watermark insertion and verification scales linearly with the number of blocks, while grows almost exponentially in the number of payload bits.

We have also compared the results of our algorithm with those of [14]. For the sake of fair comparison, we built an implementation for color images that allows to authenticate color blocks of size 16×16 or single-channel (e.g. R, G or B) blocks of the same size: in the latter case the localization capability is improved, allowing the detection not only of the block but also of the channel(s) in which the tampering has been performed. [14] requires to choose an integrity/authentication function, so we opted for an HMAC [22] based on SHA-1 or RIPEMD-160: we consider the choice of a 160-bit MAC as a tradeoff between the necessity of keeping the authentication information small (that's why we did not consider SHA-2 and SHA-3), to allow for a more precise localization of possibly tampered regions, and the use of a secure cryptographic hash function. For a 256×256 pixels color image we made blocks of size 16×16, so we have a hierarchy of levels as follows:

- level 1: $16 \times 16 = 256$ blocks of size 16×16,
- level 2: $8 \times 8 = 64$ blocks of size 32×32,
- level 3: $4 \times 4 = 16$ blocks of size 64×64,
- level 4: $2 \times 2 = 4$ blocks of size 128×128,
- level 5: $1 \times 1 = 1$ block of size 256×256.

Every block at each level requires 160 bits of authentication, distributed in the LSBs of the block itself: that makes a total of 54560 bits embedded in every channel when single channel localization is desired, and 54560 bits distributed among the channels when color block localization is sufficient. Given that the authentication bits may be considered as having each 1 bit entropy (thanks to the cryptographic hash function), an estimation of the average PSNR of the resulting watermarked images can be easily computed: on average, half of the LSBs used to store the MACs will be modified, the modification contributing as 1 in the squared error. With these chosen considerations, the average PSNR when authenticating single channel blocks is 51.937 dB, and 56.708 dB when authenticating color blocks. We also verified these values by running our implementation of [14] on the same set of images we used to test MIMIC.

The advantages of [14] are a sensitivity of 100% and a computing time that may be predicted due to the use of MACs and signatures. Its main disadvantages are the need for space to keep the authentication

information leading to large blocks and thus a reduced localization capability, and lower PSNR than MIMIC: even if it is possible to reduce the block size to improve localization, this may be done at the expense of a further reduction in PSNR, for example using 2 LSBs per pixel instead of 1.

For sake of completeness of our tests, we run the MIMIC algorithms on classical color images (Lena, Baboon, Peppers, etc.) taken from [43]. The average PSNR values are consistent with the ones shown in Table 2 to Table 5 and ranged from 44.11 dB for JPEG compressed images to 66.75 dB for SCM (Hamming [7,4,3]).

We performed another comparison with an algorithm we corrected and improved in [9]. This method is the best (to our knowledge) for what concerns quality of the watermarked images, because it modifies at most 2 pixels per block of ± 1 level (in 98.75% of the cases only 1 pixel). That algorithm, called RLLC, works in the spatial domain and embeds sequences of $k + 1$ bits in blocks of 2^k pixels. The resulting images have a very high objective quality, but the main disadvantage is the connection between the number of bits embedded and the block size: to increase the localization capability the block size should be reduced, but this requires reducing the number of bits embedded, increasing the false negative probability. In any case, we run RLLC on the test images present in [43] and the average resulting quality is 66.17 dB when embedding 7 bits in blocks of 8×8 pixels (in every single channel). This value is even slightly worse than the best quality obtained by MIMIC on the same set of images when embedding 8 bits per block (first row in Table 3).

In order to further prove the high quality of the watermarked results, Figure **5** shows an example of watermarking a color Lena image (512×512 24bpp) (Figure **5**a), by inserting 8 bits per block using BCM & SRM, resulting in a watermarked image (Figure **5**b) with PSNR = 64.59 dB. Moreover, Figure **5**c shows the amplified difference image to make the pixels modified by MIMIC visible: in particular, there are a total of 8924 pixels altered by 1 level in a single color channel. Note that the highlighted pixels are not the ones containing the watermark, which is instead inserted into the KLT coefficients derived from all the pixels in the block.

(a) (b) (c)

Figure 5: (a) Original, (b) watermarked, (c) difference Lena images.

The advantages of MIMIC are a very high PSNR, a good localization capability, that can be further improved by choosing smaller blocks (e.g., 4×4 pixels) and a sensitivity that may be chosen to meet various requirements. The main disadvantage of our approach is due to the stochastic approach taken (using a GA to solve a non-linear optimization problem), that does not ensure convergence for some demanding configurations used to watermark some images: one way to overcome this problem is running many times the GA on the blocks failing the embedding of the watermark (relaxing the constraint to keep the PSNR low for that block, allowing pixel modifications progressively larger until a solution is found) at the expense of longer computation times.

5.2. Security analysis

The security of the system is based on the secrecy of the key image and on the use of the modules previously described.

The first bulwark of MIMIC is the hidden space of embedding defined by a linear transform (the KLT) derived from a secret (key) image: an attacker cannot figure out the secret kernel base, which changes according to the key image, and consequently the coefficients cannot be determined, in particular the bit values in the less significant positions (which we use to embed the watermark).

The second bastion is the dependency of the watermark string on both the host image and the key image: thus, every block receives a part of the watermark string that is a function of the host image and of its position. This characteristic has as consequences that:

- copy-and-paste and transplantation attacks do not have success;

- VQ attacks are meaningless because the same block, in two different images, will receive a different watermark string, even if the same key image is used.

A possible trivial (and naive) attack is to substitute a block in the image with a random block (or, equivalently, randomly change one or more pixels of a block): apart from the sensitivity arguments previously discussed, the probability that a random modification of a block has success for an attacker is that the new pixel configuration contains the correct watermark string of q bits: thus the attack goes undetected, for one block, in 1 every 2^q attempts. This value is small even for $q = 8$ (as used in some experiments), and the probability that none of T attacked blocks is detected is $1/2^{Tq}$, which decreases dramatically, even for T small.

Cropping or embedding (i.e. inserting the host image into a larger one) are dealt with the solution provided in the WSU: generating a watermark dependent on the host image size makes the detection of the watermark fail in almost all (see previous paragraph) blocks, because the watermark generated by the TDU is different (uncorrelated) from the embedded one. We agree with the fact that the localization property is lost in this case: anyway, we think that a cropped image may have lost important information, and an embedded image may add fake information, so the available data are meaningless.

Strictly related to the cropping issue is the modification of the pixels of the host image used to generate the watermark (named Q in the WSU section): this will obviously result in a different watermark and the TDU will detect the attack, but the localization property is lost again. Anyway, we suggest to keep the set Q small to reduce the possibility of localization loss.

6. Discussion and conclusions

This paper presented MIMIC, a modular algorithm for the fragile watermarking of color (or multichannel) images. Its main characteristics are the possibility to customize the performance depending on the requirement of the application domain. The main advantages of MIMIC are the very high PSNR of the watermarked images, which makes them indistinguishable from the original by a human being, and the security of the whole system. The main disadvantage is, for some configurations, the long running time, that may be unacceptable in some application domains.

One interesting observation about the JSM module is that neither the image quality nor the file size are significantly affected by the insertion of the watermark, and almost all degradation is mainly due to the lossy

JPEG compression. In particular, we found that average file size of watermarked images is only 1.5% larger than the corresponding lossy JPEG compressed host image. Moreover, the average PSNR of lossy JPEG images is 58.6 dB for quality 100% and 43.23 dB for quality 95%, values that are very similar to the PSNR of watermarked images generated by the best MIMIC module combination with JSM (58.7 dB and 43.08 dB, respectively).

We are planning to perform some tests using optimization algorithms alternative to GAs (namely, substituting the GAM), such as quadratic mixed integer programming and local search algorithms, to verify if the running times may be reduced, at least for some MIMIC configurations.

Localization and security can be further enhanced should the embedding algorithm be reversible: in this case, one could embed authentication information computed from the original pixels along with the watermark and make tamper detection always 100%; moreover this would widen the applicability of our method to more demanding fields in which the original host image is mandatory after a positive authentication. Our studies are presently in this direction and we are developing an extra MIMIC module to achieve this.

7. Acknowledgments

The authors thank Prof. L. Roversi and Prof. A. E. Werbrouck for their careful reading of this paper and their useful suggestions which greatly improved the content of this work.

This work was supported by MIUR (Ministero dell'Istruzione, dell'Università e della Ricerca).

8. References

[1] Bevilacqua, A., Azzari, P. *A High Performance Exact Histogram Specification Algorithm*. In 14th International Conference on Image Analysis and Processing, pages 623–628 (2007).

[2] Barni, M., Bartolini, F., Cappellini, V., Piva, A. A DCT-domain system for robust image watermarking. Signal Processing 66(3), 357–372 (1998).

[3] Barni, M., Bartolini, F., Rosa, A. D., Piva, A. *Color image watermarking in the Karhunen-Loeve transform domain*. Journal of Electronic Imaging 11(1), 87–95 (2002).

[4] Patrick, B., Le Bihan, N., Chassery, J.-M. *Color image watermarking using quaternion Fourier transform*. Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing 2003 (ICASSP'03), vol. 3, pages 521–524 (2003).

[5] Botta, M., Cavagnino, D., Pomponiu, V. *KL-F: Karhunen-Loève Based Fragile Watermarking*. 5th International Conference on Network and System Security NSS 2011, pages 65–72, ISBN: 978-1-4577-0459-8, September 6-8, 2011, Milan, Italy.

[6] Botta, M., Cavagnino, D., Pomponiu, V. *Weighted-Sum Fragile Watermarking in the Karhunen-Loève Domain*. 9th International Workshop Security and Trust Management, STM 2013, LNCS 8203, pages 223–234, ISBN 978-3-642-41097-0, Springer-Verlag Berlin Heidelberg, 2013.

[7] Botta, M., Cavagnino, D., Pomponiu, V. *Fragile watermarking using Karhunen–Loève transform: the KLT-F approach*. Soft Computing, ISSN 1432-7643, Springer-Verlag Berlin Heidelberg, 2014.

[8] Botta, M., Cavagnino, D., Pomponiu, V. *Automatic Selection of GA Parameters for Fragile Watermarking*. 17th European Conference EvoApplications, LNCS 8602, pages 525-537, ISBN 978-3-662-45522-7, Springer Berlin Heidelberg, 2014.

[9] Botta, M., Cavagnino, D., Pomponiu, V. *Protecting the Content Integrity of Digital Imagery with Fidelity Preservation: An Improved Version*. ACM Trans. Multimedia Comput. Commun. Appl. vol 10, no. 3, pp. 29:1-29:5, April 2014, ACM, New York, NY, USA, doi: 10.1145/2568224.

[10] Barreto, P. S. L. M., Kim, H. Y., Rijmen, V. *Toward secure publickey blockwise fragile authentication watermarking*. In: IEE Proceedings - Vision, Image and Signal Processing 2002, vol. 148(2), pages 57–62 (2002).

[11] Basu, D., Sinharay, A., Barat, S., *Bit Plane Index Based Fragile Watermarking Scheme for Authenticating Color Image*. Integrated Intelligent Computing (ICIIC), 2010 First International Conference on, pages 136–139, 5-7 Aug. 2010.

[12] Cox, I. J., Miller, M. L., Bloom, J. A., Fridrich, J., Kalker, T. *Digital Watermarking and Steganography, 2nd ed*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, ISBN: 978-0-12-372585-1 (2008).

[13] Cox, I., Miller, J., Bloom, J., Miller, M. *Digital Watermarking*. Morgan Kaufmann (2001).

[14] Celik, M. U., Sharma, G., Saber, E., Tekalp, A. M. *Hierarchical watermarking for secure image authentication with localization*. Image Processing, IEEE Transactions on, vol. 11, no. 6, pages 585–595 (2002).

[15] Cavagnino, D., Werbrouck, A. E. *Color Image Compression by means of Separable Karhunen Loève transform and Vector Quantization*. Proc. of "5th International Workshop on Systems, Signals and Image Processing – IWSSIP'98". Edited by B. Zovko-Cihlar, S. Grgić, M. Grgić, Faculty of Electrical Engineering and Computing, Zagreb (Croatia), pages 235–238 (1998).

[16] Elbasi, E., Eskicioglu, A. *A Semi-Blind Watermarking Scheme for Color Images Using a Tree Structure*. In Western New York Image Processing Workshop, pages 1–8, 2006.

[17] Ghosal, S. K., Mandal, J. K. *Stirling Transform based Color Image Authentication*, Procedia Technology, vol. 10, 2013, pages 95–104, ISSN 2212-0173.

[18] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA (1989).

[19] Gonzalez, R. C., Wintz, P. *Digital Image Processing, 2nd ed.* Addison-Wesley Publishing Company (1987).

[20] Hassanien, A.-E., Abraham, A., Kacprzyk, J., Peters, J. F. *Computational Intelligence in Multimedia Processing: Foundation and Trends*. Studies in Computational Intelligence, 96:3–49 (2008).

[21] Huang, H., Chu, C., Pan, J. The optimized copyright protection system with genetic watermarking. *Soft Computing* 13(4), 333–343 (2008).

[22] Krawczyk, H., Bellare, M., Canetti, R. "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997

[23] *The Keccak sponge function family* (2012), http://keccak.noekeon.org/ (web page last accessed on December, 2013).

[24] Kutter, M., Jordan, F., Bossen, F. *Digital signature of color images using amplitude modulation*. In SPIE, Storage and retrieval for image and video databases vol. 3022, pages 518–526 (1997).

[25] Kougianos, E., Mohanty, P., Mahapatra, R. N. *Hardware assisted watermarking for multimedia*. Computers & Electrical Engineering 35(2), 339–358 (2009).

[26] Lin, P.-Y., Lee, J.-S., Chang, C.-C. Protecting the content integrity of digital imagery with fidelity preservation. *ACM Trans. on Multimedia Comp. Comm. and Appl.* 7(3), 15:1–15:20, Article 15 (August 2011).

[27] Lo-Varco, G., Puech, W., Dumas, W. *Content Based Watermarking for Securing Color Images*. The Journal of imaging science and technology 49(5), 464–473 (2005).

[28] Liu, T., Zheng-Ding Q. *A DWT-based color image steganography scheme*. In 6th International Conference on Image Processing, vol. 2, pages 1568–1571 (2002).

[29] Meng, X., Cai, L., Yang, X., Xu, X., Dong, G., Shen, X. *Digital color image watermarking based on phase-shifting interferometry and neighboring pixel value subtraction algorithm in the discrete-cosine-transform domain*. Applied Optics 46(21), 4694–4701 (2007).

[30] Ming, S. H., Din-Chang, T. *Wavelet-based Color Image Watermarking using Adaptive Entropy Casting*. In Multimedia and Expo, 2006 IEEE International Conference on Multimedia, pages 1593–1596 (2006).

[31] Maity, S. P., Kundu, M. K. *DHT domain digital watermarking with low loss in image informations*. AEU - International Journal of Electronics and Communications 64(3), 243–257 (2010).
[32] Monzoy-Villuendas, M., Salinas-Rosales, M., Nakano-Miyatake, M., Perez-Meana, H.M. *Fragile Watermarking for Color Image Authentication*. Electrical and Electronics Engineering, 2007. ICEEE 2007. 4th International Conference on, pages 157–160, 5-7 Sept. 2007.
[33] Olmos, A., Kingdom, F. A. A. *A biologically inspired algorithm for the recovery of shading and reflectance images*, Perception, 33, 1463 - 1473, 2004.
[34] Fred Kingdom's Laboratory, McGill Vision Research. Website http://tabby.vision.mcgill.ca/ (last accessed on 18 December 2014).
[35] Piva, A., Bartolini, F., Cappellini, V., Barni, M. *Exploiting the cross-correlation of RGB-channels for robust watermarking*. Proceedings of International Conference on Image Processing, ICIP 99, vol. 1, pages 306–310 (1999).
[36] Pei, S. C., Chen. J. H. *Color image watermarking by Fibonacci lattice index modulation*. In Proceedings of the 3rd European Conference on Color in Graphics, Imaging, and Vision, pages 211–215 (2006).
[37] Roy, S., Chang, E. *Watermarking color histograms*. In International Conference on Image Processing, vol. 4, pages 2191–2194 (2004).
[38] Savelonas, M., Chountasis, S. *Noise-resistant watermarking in the fractional Fourier domain utilizing moment-based image representation*. Signal Processing 90(8), pages 2521–2528 (2010).
[39] Su, Q., Niu, Y., Liu, X., Zhu. Y. *Embedding color watermarks in color images based on Schur decomposition*. Optics Communications 285(7), pages 1792–1802 (2012).
[40] Tsai, P., Hu, Y.-C., Chang, C.-C. *A color image watermarking scheme based on color quantization*. Signal Processing 84(1), pages 95–106 (2004).
[41] Tsui, T. K., Zhang, X.-P., Androutsos, D. *Color Image Watermarking Using the Spatio-Chromatic Fourier Transform*. In Proceedings of the 2006 IEEE International Conference on Acoustics, Speech, vol. 2, pages 305–308 (2006).
[42] Tsui, T. K., Zhang, X.-P., Androutsos, D. *Color Image Watermarking Using Multidimensional Fourier Transforms*. IEEE Transactions on Information Forensics and Security, vol. 3, no. 1, pages 16–28 (2008).
[43] University of Southern California, Signal and Image Processing Institute, Ming Hsieh Department of Electrical Engineering, http://sipi.usc.edu/database/database.php?volume=misc , downloaded on March 2015.
[44] Vidal, J., Madueno, M., Sayrol, E. *Color image watermarking using channel-state knowledge*. In Proc. of SPIE, Security and watermarking of multimedia contents IV, vol. 4675, pages 214–221 (2002).
[45] van Schyndel, R., Tirkel, A., Osborne, C. *A digital watermark*. In Proc. of IEEE International Conference on Image Processing, vol. 2, pages 86–90 (1994).
[46] Wong, P. W. *A public key watermark for image verification and authentication*. Proceedings of 1998 International Conference on Image Processing, pages 455–459 (1998).
[47] Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P. *Image quality assessment: From error visibility to structural similarity*. IEEE Transactions on Image Processing, vol. 13, no. 4, pages 600–612 (2004).
[48] Yu, P., Tsai, H., Lin, J. Digital watermarking based on neural networks for color images. *Signal Processing*, 81(3), 663–671 (2001).
[49] Zhao, Y., Campisi, P., Kundur, D. Dual domain watermarking for authentication and compression of cultural heritage images. *IEEE Transactions on Image Processing* 13(3), 430–448 (2004).

A Modular Framework for Color Image Watermarking

Marco Botta[1], Davide Cavagnino[1], Victor Pomponiu[2]

[1] Dipartimento di Informatica, Università degli Studi di Torino
Corso Svizzera 185, 10149 Torino, Italy

{marco.botta, davide.cavagnino}@unito.it

[2] Information Systems Technology and Design, Singapore University of Technology and Design, 138682, Singapore

victor.pomponiu@ieee.org, victor.pomponiu@gmail.com

Appendix

This Appendix is devoted to describe the main mathematical and computational tools that are used by the presented watermarking algorithm. The goal is to provide background and nomenclature to be cited in the paper.

8.1. The Karhunen-Loève Transform

The Karhunen-Loève Transform[§] (KLT) is a linear transform that maps t-dimensional vectors into a different basis for a t-dimensional vector space. In general, a linear transform maps vectors between vector spaces, and is defined by a matrix $\boldsymbol{A}$ called kernel of the transform. If $\boldsymbol{x}$ is a vector then $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ is the forward transform that maps $\boldsymbol{x}$ to $\boldsymbol{y}$. The vector $\boldsymbol{x}$ can be recovered from $\boldsymbol{y}$ using the inverse transform $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}$. Some well known linear transforms are the Fourier Transform, the Discrete Cosine Transform (DCT), the Haar Transform. In the present context we use the discrete Karhunen-Loève Transform which is also called Hotelling Transform.

The kernel of the KLT is defined by means of a set $\boldsymbol{V}$ of vectors: the KLT basis vectors computed from $\boldsymbol{V}$ are oriented in the directions for which the distribution of the vectors in $\boldsymbol{V}$ is the largest.

Consider the column vectors $\boldsymbol{v}_i \in \boldsymbol{V}$ as a random field, and compute the following quantities:

[§] The Karhunen-Loève Transform is also known as Hotelling Transform, or Principal Component Analysis, or Eigenvector Transform.

$$\boldsymbol{m_V} = E\{\boldsymbol{v_i}\} \quad \textbf{(A1)}$$

$$\boldsymbol{C_V} = E\{(\boldsymbol{v_i} - \boldsymbol{m_V})(\boldsymbol{v_i} - \boldsymbol{m_V})'\} \quad \textbf{(A2)}$$

where $E\{.\}$ is the expected value operator, thus defining the mean of the random field and the covariance of the vectors.

Computing the eigenvectors $\boldsymbol{e_i}$ of $\boldsymbol{C_V}$ and their associated eigenvalues $\boldsymbol{\lambda_i}$, $1 \leq i \leq t$, sorting them by non-increasing magnitude of $\boldsymbol{\lambda_i}$, and arranging them as rows of a matrix $\boldsymbol{A}$, it is possible to write the KLT of a vector $\boldsymbol{x}$ as:

$$\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{m_V}) \quad \textbf{(A3)}$$

The KLT operates by centering the $\boldsymbol{x}$ vectors. The components of $\boldsymbol{y}$ are called coefficients of the transform, and their position in this vector is called *order* of the coefficient. Informally, the basis of row vectors in $\boldsymbol{A}$ performs a frequency analysis on $\boldsymbol{x}$, where the frequencies are not simple oscillating functions (like sines or cosines) but are derived from the vectors in $\boldsymbol{V}$.

The vector $\boldsymbol{x}$ can be reconstructed from the vector $\boldsymbol{y}$ (i.e. the t coefficients) with:

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y} + \boldsymbol{m_V} \quad \textbf{(A4)}$$

A more detailed description of the KLT can be found in [5].

8.1.1. Computation and use of a KLT basis

As previously seen, a KLT basis may be computed from a set of vectors considered as a random field.

A gray-scale image of size $N \times M$ may be divided into contiguous non-overlapping subimages (also called blocks) of size $n \times n$ (for simplicity, without losing generality, we assume that N and M are multiples of n). Each subimage may be mapped to a column vector by arranging its pixels, taken in raster scan order, as successive elements of the resulting vector of size n^2. In this way, from a gray-scale image a random field of vectors may be derived, and then a KLT basis: this basis is strictly related to the image in the sense that a different image will produce, in general, a different basis. Intuitively, this procedure generates a set of gray-scale basis subimages.

A color image represented in the RGB space[**] may be viewed as three gray-scale images, so different approaches may be considered. If the KLT basis required is for single channel images (e.g. gray-scale) then

[**] In this paper we deal with the most common case of multichannel images, namely RGB images. Anyway, the proposed method may be applied to images comprising many components of the same size.

the subimages from the individual RGB channels may be joined into a single random field, to obtain a KLT basis of n^2 vectors having n^2 components. Otherwise, if the basis is needed to transform color images, then two simple approaches may be followed:

- consider each of the three channels RGB as a single gray scale image, derive a basis from every channel which will be composed of n^2 vectors (having n^2 components) and use each basis to transform the corresponding channel;
- consider the color subimages of size $n \times n \times 3$, reordering their pixels as column vectors of size $3n^2$ (using a raster scan order and considering the RGB channels in sequence) and computing a KLT basis of $3n^2$ vectors (each having $3n^2$ components); intuitively, this creates a set of color basis subimages.

Other methods may be used to create a KLT basis: see, for example, [3] where a Separable KLT for color images is presented.

When the KLT kernel is computed from an image, its vectors are also called basis images recalling the fact that every subimage may be computed as a weighted sum of these vectors.

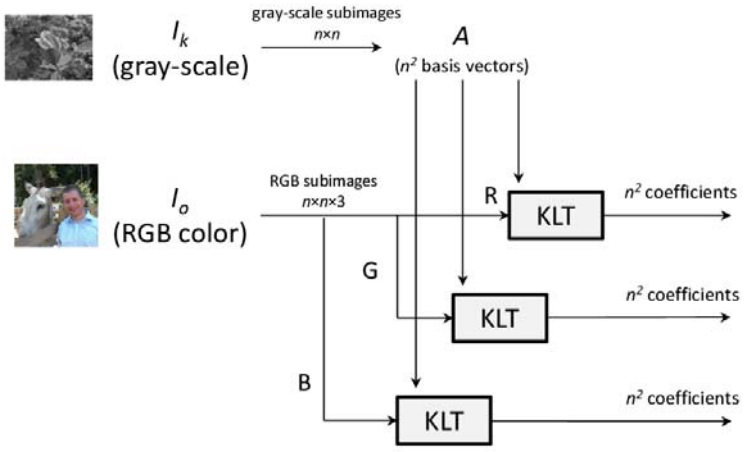

Figure A6: Applying gray-scale basis images from a gray scale image to perform the KLT of a color image.

A KLT basis is computed from an image I_k, and may be used to transform the subimages of size $n \times n$ of a color image I_o: Figure A6 shows a high level scheme of how to perform the KLT of the subimages of an RGB color image using the basis images of a gray-scale image. Figure A7 shows how the three RGB components may be used to compute three kernels to be used one per channel. Finally, in Figure A8 the basis images take into account the color information along with the spatial information integrating them in a kernel of $3n^2$ vectors.

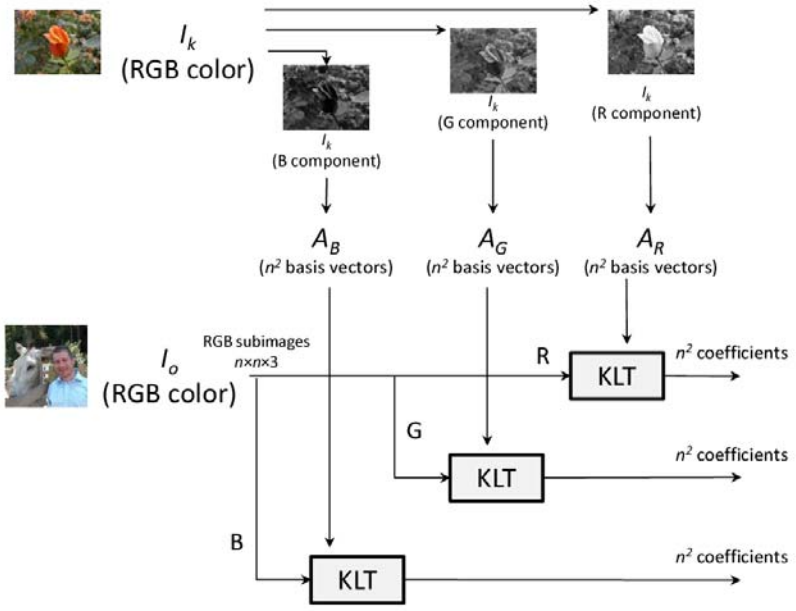

Figure A7: Applying gray-scale basis images from the components of a color image to perform the KLT of a color image.

In some cases it may be necessary a reduction of the dimension of the problem, e.g. for efficiency reasons of a Genetic Algorithm computation. We consider the case of creating a linear combination of the color channels of each pixel: if a pixel P is represented with its three color components (P_R, P_G, P_B), then it is possible to produce a single value $P_C = w_R P_R + w_G P_G + w_B P_B$ (using the same three weights w_R, w_G, w_B for all pixels), use this value to compute the KLT and produce only n^2 coefficients instead of $3n^2$. Calling $\boldsymbol{x_R, x_G, x_B}$ the three components of each block of size n^2 pixels and $\boldsymbol{m_R, m_G, m_B}$ the three channels average blocks, we may express the KLT in Figure A6 for the weighted channels as

$$\boldsymbol{A[w_R(x_R - m_R) + w_G(x_G - m_G) + w_B(x_B - m_B)] = w_R A(x_R - m_R) + w_G A(x_G - m_G) + w_B A(x_B - m_B) = w_R y_R + w_G y_G + w_B y_B} \quad \textbf{(A5)}$$

and for Figure A7

$$\boldsymbol{A_R w_R(x_R - m_R) + A_G w_G(x_G - m_G) + A_B w_B(x_B - m_B) = w_R A_R(x_R - m_R) + w_G A_G(x_G - m_G) + w_B A_B(x_B - m_B) = w_R y_R + w_G y_G + w_B y_B} \quad \textbf{(A6)}$$

From these two equations it is possible to see that, thanks to the linearity of the transform, we may indifferently weight the pixels or the transform coefficients.

For the purpose of our algorithm, any of the presented schemes may be used (more details on possible differing performances are presented in the Experimental results section); the most important aspect is that keeping the image I_k secret, thus unavailable to an attacker, it is impossible to predict the values of the

resulting coefficients (in particular, the less significant bits of the coefficients). This makes the KLT different from other transforms like the Fourier Transform or the DCT which have fixed kernels.

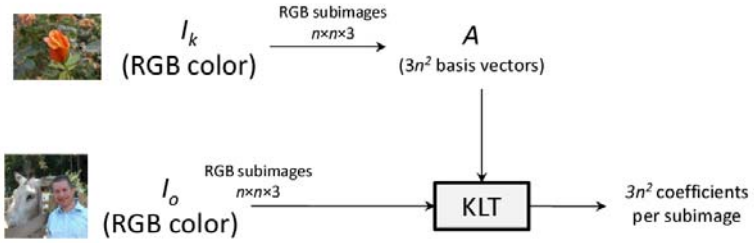

Figure A8: Applying color basis images to perform the KLT of a color image.

8.2. The watermark bit storing function

The authentication performed at block (subimage) level is based on storing q watermark bits into r KLT coefficients (in general $q \neq r$).

We say that a bit value b is stored into a coefficient c when the following extraction rule holds:

$$b = [2^{-p}c] \bmod 2 \qquad \textbf{(A7)}$$

where [.] denotes rounding and p is a bit position in the binary representation of the coefficient c (e.g., positive values of p correspond to the integer part, while negative values of p correspond to fractional bits). For example,

if $p = 0$ then $c = 1011.011_{(2)}$ stores $b = [1011.011_{(2)}] \bmod 2 = 1011_{(2)} \bmod 2 = 1$, whereas $c = 1011.100_{(2)}$ stores $b = [1011.100_{(2)}] \bmod 2 = 1100_{(2)} \bmod 2 = 0$;

if $p = -1$ then $c = 1011.011_{(2)}$ stores

$b = \left[10_{(2)} \cdot 1011.011_{(2)}\right] \bmod 2 = [10110.11_{(2)}] \bmod 2 = 10111_{(2)} \bmod 2 = 1$,

and $c = 1011.100_{(2)}$ stores

$b = [10_{(2)} \cdot 1011.100_{(2)}] \bmod 2 = [10111.00_{(2)}] \bmod 2 = 10111_{(2)} \bmod 2 = 1$.

After defining the value stored in a coefficient (Equation (A7)) we present three different methods for storing q watermark bits into a subimage coefficients. Parameters common to all the methods are the position p and the r coefficients' orders.

Bit insertion

The first method stores one watermark bit into one transform coefficient (in this case $r = q$): a set of r coefficients is chosen and every bit is recovered applying Equation (A7) to each coefficient.

Syndrome coding

The second method uses matrix embedding which implements syndrome coding [2]. The objective is to limit the pixel distortion introduced by the embedding because several coefficient values encode the same watermark and the best ones may be chosen. Syndrome coding is applied in Error Correcting Codes (ECC) in which r bits are used to transmit v bits along with $q = r - v$ redundancy bits. The properties of an ECC assert that if the minimum Hamming distance of the r-bit codewords is d then the code is capable of correcting up to $e = \left\lfloor \frac{d-1}{2} \right\rfloor$ errors.

When an ECC may be expressed in a linear form, then a v-bit data word $\boldsymbol{c}$ (represented as a column vector) may be encoded in an r-bit word $\boldsymbol{w}$ with a matrix multiplication (all the sum and subtraction operations are performed in modulo-2 arithmetic)

$$\boldsymbol{w} = \boldsymbol{G}\boldsymbol{c} \tag{A8}$$

where the $r \times v$ matrix $\boldsymbol{G}$ is the generator of the code. A linear code with these characteristics is also described with the triple in square brackets $[r, v, d]$. An attribute of the code is the *covering radius* defined as the minimum distance R from codewords that allows to cover completely the space of all the 2^r bit sequences of length r. When $R = e$ then the code is said perfect, like the Golay code [23, 11, 7].

The recipient may decode the received word $\boldsymbol{w'}$ using a $q \times r$ matrix $\boldsymbol{H}$ (derived from $\boldsymbol{G}$) computing the *syndrome* $\boldsymbol{s}$ of size q bits:

$$\boldsymbol{s} = \boldsymbol{H}\boldsymbol{w'} \tag{A9}$$

The set $C(\boldsymbol{s}) = \{\boldsymbol{x} \in \{0,1\}^r \mid \boldsymbol{H}\boldsymbol{x} = \boldsymbol{s}\}$ is called coset of $\boldsymbol{s}$. If $\boldsymbol{s} = \boldsymbol{0}$ then the word $\boldsymbol{w} = \boldsymbol{w'}$ was not modified since computed from $\boldsymbol{G}$ (i.e. it is a codeword). Otherwise $\boldsymbol{w'} = \boldsymbol{w} + \boldsymbol{e}$ where $\boldsymbol{e}$ is a vector of changes to $\boldsymbol{w}$; then

$$\begin{gathered} \boldsymbol{s} = \boldsymbol{H}\boldsymbol{w'} = \boldsymbol{H}(\boldsymbol{w} + \boldsymbol{e}) = \boldsymbol{H}\boldsymbol{w} + \boldsymbol{H}\boldsymbol{e} \\ \boldsymbol{s} - \boldsymbol{H}\boldsymbol{w} = \boldsymbol{H}\boldsymbol{e} \end{gathered} \tag{A10}$$

In the context of information hiding the last formula is interesting because it implies that it is possible to alter a syndrome $\boldsymbol{H}\boldsymbol{w}$ to make it equal to a value $\boldsymbol{s}$ by adding a word $\boldsymbol{e}$ to $\boldsymbol{w}$. The word $\boldsymbol{e}$ is any element of the coset $C(\boldsymbol{s} - \boldsymbol{H}\boldsymbol{w})$: $\boldsymbol{w}$ added to any word of this coset will become a word $\boldsymbol{w'}$ having syndrome $\boldsymbol{s}$. To minimize the changes to $\boldsymbol{w}$ it is possible to choose a word in $C(\boldsymbol{s} - \boldsymbol{H}\boldsymbol{w})$ having minimum Hamming weight: any word with this property is called coset leader. It may be shown that the covering radius R of the code is

an upper limit to the Hamming weight of the coset leaders of all the syndromes, thus using a linear code defined by $\boldsymbol{G}$ (or $\boldsymbol{H}$) $[r, r-q, d]$ it is possible to encode q bits in a word of r bits with a maximum of R changes. In our context, a set of r coefficients is chosen and every bit is recovered applying Equation (A7) to each coefficient obtaining $\boldsymbol{w'}$; then, with Equation (A9) the q bits are computed.

Weighted modulo sum

The third method derives an idea from [7] in which a weighted sum of pixel values is used to convey a watermark value: the advantage is the very low modification to the pixel values needed to correct for the watermark insertion. We applied the weighted sum idea to the KLT transformed coefficients, instead of the pixel values: that is, we extract q bits representing a number l between 0 and $2^q - 1$ from $r = 2^{q-1}$ KLT coefficients c_{o_i} with orders o_1, ..., o_r performing the following weighted modulo sum:

$$l = \left(\sum_{i=1}^{r} i \cdot c_{o_i}\right) \bmod 2^q \qquad \textbf{(A11)}$$

From Expression (A11) one can see that it is possible to modify only one coefficient by $+1$ or -1 to represent a desired number l' when the represented number is l (namely, it is sufficient to modify the coefficient multiplied by the weight equal to $l - l'$). This approach, by minimizing the amplitude change of one coefficient only, tries to minimize the (possibly many) pixel modifications.

To increase the flexibility of this last embedding method, we proposed in [1] a slight variation that allows to insert $y \leq 2^{q-1}$ bits into a block of 2^{q-1} pixels (and consequently KLT coefficients). If $y \leq q$ it is sufficient to use 2^{y-1} coefficients; otherwise ($y > q$), the method consists in successively halving the set of coefficients: at the first splitting we obtain two sets of 2^{q-2} coefficients, each capable of storing, with modulo sum, $q - 1$ bits, that is $2q - 2$ bits total. If $2q - 2 < y$ then the halving process may be repeated until the desired number of blocks is obtained (with the limit 2^{q-1}): when the halving has produced 2^z groups (i.e. it has been applied z times) we may store $2^z(q - z)$ bits, that for $z = q - 1$ is 2^{q-1}. (Obviously nothing comes for free: the hidden danger is that the module that alters the pixels to embed the watermark is faced with more complicated problems as the number of bits increases, and in some cases it may not be able to converge to a viable solution).

8.3. Genetic Algorithms

Biological species are known to evolve, through a process of Darwinian natural selection over long times, to individuals that are better (in some sense) than their predecessors. This idea may be mimicked by a computing paradigm to search for a solution to a problem, if some constraints are satisfied: a Genetic Algorithm simulates the evolution of individuals representing possible solutions to a problem and measures the suitability of every individual to an optimal solution, allowing a high probability of reproduction to the best ones (but without completely discarding less suitable individuals, as happens in nature). A GA encodes the problem parameters in a sequence building an individual. Different individuals, made of the same parameters but with different values, compose a *population* that may evolve by simulating biological reproduction. Obviously natural evolution prefers stronger (in some sense) individuals, and the GA measures this strength with a mathematical function (called *fitness function*) applied to the features composing an individual; the function may return a value that is proportional to the quality of the solution represented by the individual (the larger the value the better the individual) or that is inversely proportional (the smaller the value the better the individual): note that in the following we will refer to this second kind. The evolution of individuals is performed by a GA through *genetic operators* applied to individuals of a population to obtain a new population that replaces the older: the time at which a population has been completely evolved is called *epoch*.

The genetic operators that are typically considered are *selection*, *reproduction* and *mutation*; a *create individual* operator also exists, but only for initialization. In the following discussion we assume that an individual i_a is composed by a sequence of t features $i_{a1}\ i_{a2}\ \ldots\ i_{at}$.

With the *create individual* operator these t features are randomly initialized in the new individual: constraints on the range of validity may be imposed, or particular initializations may be used if known to help convergence to a solution. When this operator has created a pre-defined amount of individuals, this population goes through a sequence of epochs and in each one new individuals are generated.

To *select* individuals for reproduction with the tournament selection strategy (other strategies are possible), pairs of individuals are *selected* and the one with best *fitness* in each pair is saved in a set. Individuals from this set are then selected in pairs i_a and i_b and with probability p_c are mated for *reproduction* with one-point or two-point crossover. In one-point crossover a random number $1 < \lambda < t$ is generated and the

two created offsprings are $i_{a1} \ldots i_{a\lambda}\ i_{b(\lambda+1)} \ldots i_{bt}$ and $i_{b1} \ldots i_{b\lambda}\ i_{a(\lambda+1)} \ldots i_{at}$ (the features are exchanged from the random position to the last) while in two-point crossover two random numbers $1 < \lambda < v < t$ are used to create the two individuals $i_{a1} \ldots i_{a\lambda}\ i_{b(\lambda+1)} \ldots i_{bv}\ i_{a(v+1)} \ldots i_{at}$ and $i_{b1} \ldots i_{b\lambda}\ i_{a(\lambda+1)} \ldots i_{av}\ i_{b(v+1)} \ldots i_{bt}$ (i.e. an intermediate portion of the features is exchanged between the individuals). After all individuals have been selected, every resulting offspring has a probability p_m to have one of more features *mutated*, with the objective to better explore the space of possible solutions and avoid to fall in local minima.

Having generated the offsprings leads to a new population, under many possible strategies: for example, the newly generated individuals replace the whole old population, or the new individuals replace only the worst individuals in the old population, or tournament selection is applied to extract the next generation individuals.

In each epoch a stop criterion must be considered to verify if a viable solution has been found (namely, an individual having a fitness value below a pre-defined threshold) or the execution has not yet obtained an acceptable individual after a pre-defined maximum number of epochs: in both cases the current population best individual w.r.t. *fitness value* is returned, signaling the condition of termination.

We may give a high level description of a GA with the following pseudo-code (note that in the present implementation the smaller the fitness value the better the individual):

generate random individuals to build a population
evaluate fitness of every individual in population
while NOT(*fitness of best individual below threshold* OR *reached maximum number of epochs*)
    create new individuals through **selection** and **crossover**
    mutate new individuals
    build new population
    evaluate fitness of every individual in population
end while
return best individual and termination condition

For a more complete discussion and details on GAs we suggest [4, 6] as starting references.

In this context we want to mention the fact that due to the intrinsic randomness of the involved processes, different runs may lead to different optimal solutions. In some scenarios (like debugging, legal issues or reproducibility required by the application) it may be required to have the same solution computed in different executions. Given that GA software uses a pseudo-random number generator, we may reproduce the same watermarked image from a previous GA run with a reproducible seeding of the random number generator.

References

[1] Botta, M., Cavagnino, D., Pomponiu, V. *Weighted-Sum Fragile Watermarking in the Karhunen-Loève Domain*. 9th International Workshop Security and Trust Management, STM 2013, LNCS 8203, pages 223–234, ISBN 978-3-642-41097-0, Springer-Verlag Berlin Heidelberg, 2013.

[2] Cox, I. J., Miller, M. L., Bloom, J. A., Fridrich, J., Kalker, T. *Digital Watermarking and Steganography, 2nd ed*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, ISBN: 978-0-12-372585-1 (2008).

[3] Cavagnino, D., Werbrouck, A. E. *Color Image Compression by means of Separable Karhunen Loève transform and Vector Quantization*. Proc. of "5th International Workshop on Systems, Signals and Image Processing – IWSSIP'98". Edited by B. Zovko-Cihlar, S. Grgić, M. Grgić, Faculty of Electrical Engineering and Computing, Zagreb (Croatia), pages 235–238 (1998).

[4] Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA (1989).

[5] Gonzalez, R. C., Wintz, P. *Digital Image Processing, 2nd ed.* Addison-Wesley Publishing Company (1987).

[6] Hassanien, A.-E., Abraham, A., Kacprzyk, J., Peters, J. F. *Computational Intelligence in Multimedia Processing: Foundation and Trends*. Studies in Computational Intelligence, 96:3–49 (2008).

[7] Lin, P.-Y., Lee, J.-S., Chang, C.-C. Protecting the content integrity of digital imagery with fidelity preservation. *ACM Trans. on Multimedia Comp. Comm. and Appl.* 7(3), 15:1–15:20, Article 15 (August 2011).