Fragile watermarking using Karhunen–Loève transform: the KLT-F approach

Marco Botta; Davide Cavagnino; Victor Pomponiu. Fragile watermarking using Karhunen–Loève transform: the KLT-F approach. SOFT COMPUTING. 19 (7) pp: 1905-1919.
DOI: 10.1007/s00500-014-1373-y

Fragile watermarking using Karhunen-Loève transform: the KLT-F approach

Marco Botta, Davide Cavagnino, Victor Pomponiu

Abstract The paper presents a fragile watermarking technique that may be used to discover image manipulations at block level. The proposed algorithm, based on the Karhunen-Loève linear transform (KLT), modifies some of the KLT coefficients obtained from a secret base (defined by a secret key image) so that they contain a binary pseudo-random message. A genetic algorithm (GA) is used to compensate for rounding errors introduced by inverse transforming in the integer pixel domain. An extensive experimentation has been performed to test the effectiveness of the method and to show the sensitivity of the algorithm to single pixel modifications (also as a function of the number of modified coefficients). A comparison with other fragile watermarking methods is reported. It should be noted that the proposed approach results in both a high PSNR (more than 53 dB on average) and a high subjective quality. The system may be tested online by submitting images to be watermarked and subsequently verifying the presence of modifications in a previously marked image.

Keywords Fragile watermarking • Content authentication • Karhunen-Loève transform • Genetic algorithms • Data security

Marco Botta, Davide Cavagnino
Dipartimento di Informatica,
Università degli Studi di Torino
Corso Svizzera 185, 10149 Torino, Italy
{marco.botta, davide.cavagnino}@unito.it

Victor Pomponiu
Department of Radiology, University of Pittsburgh
3362 Fifth Avenue, Pittsburgh, 15213, PA, USA
vpomponiu@acm.org

1 Introduction

The wide use and diffusion of digital multimedia contents like images and sounds, has given rise to the necessity of developing methodologies and techniques that may guarantee the protection of the digital content. In fact it is possible, if appropriate solutions are not adopted, to copy, modify or in some way misuse a multimedia object taking advantage of the digital nature of the object itself. Depending on the application context, different techniques may be used to protect digital objects. One example are digital signatures. Another possibility are watermarks. The main difference between a signature and a watermark is that the latter is contained into the digital object itself. Moreover, invisible watermarks are hidden by modifying some features of the digital object.

Watermarks have many applications; for example, they may be used for copyright protection, content authentication, track of origin (Cox et al. 2008). Depending on the field of application, the various watermarking algorithms differ in the properties and characteristics they possess. A first classification of watermarking algorithms is between *informed* or *blind* systems, and refers to the detection part of the algorithm: an *informed* (or *non-blind*) system needs the original digital object (or some features derived from it) to perform the detection of the watermark. On the contrary, if only the watermarked object (and possibly a secret key) is used for the detection then the watermarking algorithm is called *blind*.

The application that will use the watermarking system also defines the requirement for the watermark to be *robust* or *fragile*. A *robust* watermark is designed to survive modifications to the digital object aimed at removing the watermark itself and keeping high the quality of the object. On the contrary, a *fragile* watermark is designed to be removed by the minimal modification of the digital object (and possibly specifying the position of the modification). A class of watermarking algorithms are designed to be *semi-fragile*, in the sense that it is possible to detect heavy modifications to the digital object whilst light processing operations (like mild filtering or compression with high quality) still allow the detection of the watermark (Fridrich 2002).

The present work is aimed at presenting a fragile watermarking algorithm, so we detail here the most important characteristics required by this class of algorithms:

- invisibility: the watermark should not degrade the quality of the object it is protecting; in particular, its presence should not be perceived by the application using the object (human or automatic);
- detection and localization: any alteration(s) to the digital object should be revealed by a check, and possibly the position(s) of the modification(s) should be enlightened;
- security against intentional attacks: the watermark should reveal cut-and-paste (Fridrich 2002; Wong and Memon 2001), transplantation and birthday attacks.

Barreto et al. (2002) propose to use a technique called hash block chaining to deal with the mentioned security attacks: the idea is to make the authentication information of each block of the image dependent on one of its neighbors.

In general, the watermark embedding is performed by modifying some features of the digital object; the features may be extracted from many different spaces, leading to many different algorithms with peculiar characteristics and properties. For example, the watermark may be directly inserted into the spatial domain (e.g. pixels of an image) or time domain (e.g. samples of an audio), or both domains (e.g. frames of a video). Other algorithms insert the watermark into a transformed domain, like the Discrete Cosine Transform domain (Cox et al. 1997), or the Fourier Transform domain (Premaratne 1999), or in the vectors and singular values of the Singular Value Decomposition (Pomponiu et al. 2010). A comprehensive description of various aspects of digital watermarking may be found in (Cox et al. 2008).

A first classification of fragile watermarking algorithms may be established considering at which level the localization of tampered areas is performed: block-wise methods allow the identification of tampered blocks (in which at least one pixel is tampered): examples of these algorithms are presented in (Bravo-Solorio and Nandi 2011; Barreto et al. 2002; Liu et al. 2008; Shih and Wu 2005a; Usman et al. 2007); differently, pixel-wise methods allow for the identification of single altered pixels: for example, (Bravo-Solorio and Nandi 2011; Yeung and Mintzer 1997) discuss algorithms of this kind.

Han et al. (2011) devised several fragile watermarking methods for securing RFID data networks, in particular the RFID data streams and databases. Essentially, they adapted several well-known CDMA and chain-based watermarking schemes to cope with the restrictive constraints of wireless environments, such as limited resources, reliability and lack of protection.

The algorithm discussed in this paper is developed to ensure content authentication of digital grey-scale images. Its main characteristics are:

- the use of a secret key image to define a feature space based on the Karhunen-Loève transform (KLT);
- the use of a watermark that is dependent on both the key image and the image to be authenticated; and
- the use of a genetic algorithm to compensate pixel rounding errors.

This paper extends the work presented in (Botta et al. 2011) improving the approach along several directions. Moreover, we performed systematic and extensive tests to yield a deeper insight of the properties of the proposed algorithm.

The paper is organized as follows: Section 2 presents the terminology and recalls the main concepts related to the Karhunen-Loève transform and to the genetic algorithms. Then, Section 3 presents related works. Section 4 details our algorithm and its parameters, and Section 5 presents an analysis of the performance of the system, also comparing it with other fragile watermarking systems. Conclusions and future developments are discussed in the final section.

2 Background

2.1 Notations

Before describing the fragile watermarking algorithm in detail, we present here the two fundamental tools we used to obtain a secure and precise authentication: the Karhunen-Loève transform (KLT), which provides a simple method to define a secret space where the authentication information is hidden, and the genetic algorithms (GAs), that are used to fine tune the image pixels in such a way they store the desired authentication information.

These tools make a heavy use of vectors and matrices, so we recall the notation used to represent them. The convention employed in this work is to denote vectors as boldface lowercase letters (as $\boldsymbol{f}$), matrices as boldface capital letters (as $\boldsymbol{B}$) and matrix transposition with the symbol ' (prime). If $\boldsymbol{f}$ is a vector, then f_i is the i-th element of vector $\boldsymbol{f}$. Similarly, B_{ij} is the element in row i and column j of matrix $\boldsymbol{B}$.

2.2 The Karhunen-Loève Transform (KLT)

In this work the authentication information is inserted into a secret space on which the pixels are linearly projected. That is, the features in which to store the bits used to verify the integrity of the image are obtained by a linear transformation of the form:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \qquad (1)$$

where $\boldsymbol{x}$ is a column vector of pixels, $\boldsymbol{y}$ is a column vector of features (which will be called transform coefficients) and $\boldsymbol{A}$, called kernel, is a square matrix defining the linear transformation. The rows of $\boldsymbol{A}$ are an orthonormal basis for the space in which the vectors $\boldsymbol{x}$ are represented. Examples of linear transformations are the Discrete Cosine Transform, the Fourier Transform, the Hadamard Transform; from this, the space in which the vector $\boldsymbol{y}$ is expressed is called frequency domain.

These transformations are defined by fixed kernels, so their mapping is the same for every image to which they are applied. Nonetheless, there is a linear transformation, the Karhunen-Loève transform (KLT) that does not have a fixed kernel. On the contrary, the kernel is computed from a set of vectors considered as a random field. The resulting basis is made of vectors with directions along the maximum data spread. We recall here the main points of this transform, but a more detailed description of the KLT may be found in (Gonzalez and Woods 1992).

Consider a random field of column vectors $\boldsymbol{f}$, and compute the mean vector $\boldsymbol{m} = E\{\boldsymbol{f}\}$. Then compute the covariance matrix $\boldsymbol{C} = E\{(\boldsymbol{f} - \boldsymbol{m})\ (\boldsymbol{f} - \boldsymbol{m})'\}$ and evaluate its eigenvectors $\boldsymbol{e}_i$ and their associated eigenvalues λ_i.

Each eigenvalue is a measure of the data spread along the direction defined by its eigenvector. Sorting the eigenvectors by non-increasing order of their eigenvalues moves into the first positions the eigenvectors having, on average, a large contribution in the construction of the vectors $\boldsymbol{f}$. If the eigenvectors are combined as the rows of a matrix $\boldsymbol{B}$, then $\boldsymbol{y} = \boldsymbol{B}\ (\boldsymbol{x} - \boldsymbol{m})$ is the KLT of the vector $\boldsymbol{x}$. The components of $\boldsymbol{y}$ are called the transform coefficients. The vector $\boldsymbol{x}$ may be perfectly recovered from $\boldsymbol{y}$ as $\boldsymbol{x} = \boldsymbol{B}^{-1}\ \boldsymbol{y} + \boldsymbol{m}$. For each component y_i of $\boldsymbol{y}$, i is called the order of the coefficient.

In general, the KLT may be used to reduce the number of features by exploiting the correlation among vector components: this is useful, for example, in compression algorithms. In our algorithm, the KLT is used to create a hidden space in which the bits used for the authentication are stored: this increases the security of the method because an attacker is not able to determine the values of the secret bits. And even if it could be possible to generate an orthonormal basis with a method based on a key, the KLT allows to immediately generate a basis from a real (secret) image: this has the advantage that the coefficients in which the bits will be inserted are related to frequencies of images and are not completely random reducing, with a wise choice of the order of the coefficients, the visual distortion.

2.3 Genetic Algorithms

Genetic Algorithms (GAs) are a computational model inspired by the evolution of biological beings, that has proved to be quite effective at solving optimization problems. Mimicking nature, a GA represents problem solutions as individuals, and by implementing a natural selection schema, it makes the best individuals survive and reproduce through the use of genetic operators. In this way, GAs may find (quasi) optimal solutions to optimization problems. A problem solution is encoded, as a sequence of parameters, into an individual just like a sequence of genes composes a chromosome.

A GA starts its computation with a population of individuals that represent initial approximate solutions to the problem at hand, and is generally randomly generated; nonetheless, individuals representing an approximate solution, if known, may be included in this initial population, with the objective of easing and speeding up the convergence to an optimal solution. Then, the initial population evolves for a number of generations, or epochs; in each epoch the GA repeats the following steps until a termination condition is reached (usually, a maximum number of generations is performed or some threshold is exceeded):

- evaluation of each individual of the population;
- selection of individuals for reproduction;
- reproduction;
- update of the population for the next generation.

Individuals in the population are evaluated by computing a so-called fitness function F. This function, applied to an individual, returns a quantity expressing how much the individual is close to the optimal solution. In the following we will use the convention that the smaller the fitness value, the better the solution (but the opposite convention could be used, e.g. using the reciprocal of the fitness function).

Individuals are selected for reproduction according to their fitness value. Two strategies are commonly used to this purpose: roulette wheel and tournament selection. We only describe the latter strategy here, being the one used in our algorithm, and refer the interested reader to the literature for the former (Goldberg 1989). In the tournament selection strategy, a pair of individuals is uniformly randomly selected from the population to play a tournament: the one with smaller fitness value wins the tournament and is selected for reproduction. This process is repeated until a sufficient number of individuals is chosen.

Afterwards, the selected individuals are mated two by two and new offsprings are generated by applying a crossover operator with probability p_c. In one point crossover, the mating individuals exchange their genetic material from the beginning to a randomly chosen cutting point; in two points crossover, the material exchanged in the individuals is the one included in between two randomly selected points.

The reproduction step terminates by applying a mutation operator to each offspring: each individual has probability p_m to have its genes randomly modified. Just like in biological species, mutation has the objective to explore new and probably unusual possibilities that may lead to better solutions than those obtainable from crossover alone.

Finally, the population is updated with the newly generated individuals. Several strategies can be used to perform this step. In our case, we implemented an elitist strategy, in which the p_s% best individuals from a generation become part of the next one, while the rest is replaced by the newly generated offsprings. For a more detailed discussion on GAs the reader is invited to consult (Goldberg 1989; Hassanien et al. 2008).

The termination condition is either a maximum number of allowed generations reached or the fitness value of the best individual did not change in the last 10 generations and is below a predefined threshold. In both cases, the best individual is returned. Note that, in some cases, the algorithm can terminate without actually finding a viable solution.

3 Related works

Genetic Algorithms are used in many watermarking algorithms for images. The editorial by Pan and Abraham (2009) presents various fields of research in information hiding inspired to biological systems, and introduces some works in the field published in the same journal issue.

A GA is used in the work by Lee and Ho (2002) to insert a random fragile watermark into the LSB of the image pixels; in the embedding phase, the GA fitness function takes into account the information on the edges contained in a block using as a measure the first two AC coefficients of the block's DCT.

Usman et al. (2007) present a technique for image authentication aimed at resisting to various attacks like vector quantization, cover-up and transplantation; the algorithm watermarks every image block (of size 8×8) by modifying five of its DCT coefficients and creating a dependency with its neighboring blocks: to improve the quality of the resulting watermarked image, a GA is used in the selection of the five coefficients. Due to the dependency of the watermark in a block to its neighbors, the authors have to develop a rule to avoid false positives, sometimes decreasing the localization capability.

In (Shih and Wu 2005a,b; Aslantas et al. 2009) a GA is used to deal with the rounding error introduced when transforming from the frequency domain (i.e. DCT) to the integer pixel domain: analogously to us, the objective is to retrieve a watermark with possibly no errors w.r.t. the embedded one in case of absence of attack. In (Aslantas et al. 2009) the performance of the GA is also compared with other Intelligent Optimization Algorithms.

In the field of image processing, the Karhunen Loève Transform (KLT) has been applied in many areas due to its ability in compacting the pixel's information in a reduced number of features. For example, image compression algorithms have been developed, and also feature extraction algorithms for pattern recognition or search in databases have been devised. In image watermarking, in general the DFT, the DCT and the DWT have been widely used, but also the KLT has been applied at various levels.

In (Barni et al. 2002) the KLT is used for developing a robust watermarking method. The transform is used to find three uncorrelated bands from the original RGB channels. The new bands are then DFT transformed and the watermark is redundantly embedded into some of the coefficients of the three bands, only changing the strength depending on the importance defined by the KLT eigenvalues.

Dafas and Stathaki (2003) insert a watermark by slightly rotating the eigenvectors computed from blocks of the host image: the eigenvectors to be rotated and the amount of rotation are specified by the watermark bits. The verification requires both the original image and the watermark.

Stanescu et al. (2007) propose a method for robust watermarking a color image by embedding a gray-scale image. The blocks of the host image are linearly transformed with KLT; then the second, and possibly third band of each block are replaced by the block of the watermark that has a similar energy content (and has not yet been embedded), also rotating the second and third eigenvectors. The watermark extraction needs the host image along with some other side information (like the order of the blocks during embedding).

In (Yeung and Mintzer 1997) a method for image authentication is proposed; the watermark is a binary image, and the algorithm inserts one watermark bit per pixel. For each pixel in the host image, the bit extraction is applied using a function based on one or more secret Look Up Tables: then, the pixel value is modified until the extraction returns the corresponding watermark bit, with a process that applies the minimum necessary distortion. The error introduced is propagated by distributing it to the contiguous pixels in such a way to keep unchanged the mean value of each color channel. The watermarked image has been produced after all the pixels have been processed. When an image has to be authenticated, the same LUTs are used to extract the bit sequence and the result is compared with the watermark: differing values indicate areas of tampering.

The z-transform is used in (Ho et al. 2008) to insert the bits of a fragile watermark for authentication purposes. The algorithm described in the paper performs the z-transform of rows of size 1×8 of contiguous pixels (and makes sub-images composed by 8 vertically contiguous rows): the pixels of each row are considered samples of a mono-dimensional signal which are z-transformed. Then, the only negative real root is modified according to the bit value of the watermark to be inserted. The obtained z-transform zeroes are then transformed back into the pixel domain (in general requiring a rounding) to obtain the watermarked image. In the verification phase, the rows of each block are z-transformed and the bits extracted according to the value of the real root: an 8×8 block is considered tampered if all the 8 rows return a bit that does not match the corresponding watermark bit. The authors did not explain how to deal with the pixel rounding error.

The scheme introduced in (Oktavia and Lee 2005) uses the singular values (SVs) of the original image as the authentication signature. After splitting the host image and the watermark into non-overlapping blocks, the watermark insertion is done in the following steps: first, the LSBs of *all* the block's pixels are replaced by a random sequence generated through a secret key. Secondly, for each block the LSBs of the SVs are computed and XOR-ed with the watermark. Finally, the image is reconstructed from the so modified SVs. The whole security of the scheme relies on the secret key used during the embedding process.

In (Rawat and Raman 2011) a chaos based fragile watermarking scheme is proposed. To improve the security of the scheme two chaotic maps are used to scramble the host image and the watermark bits (i.e., a logo image) prior the embedding process. The authentication bits are inserted into the LSBs of the scrambled version of the host image. Experimental results demonstrate that the proposed scheme achieves superior tamper detection and localization accuracy under different attacks such as copy-and-paste and collage.

Lin et al. (2011) introduce a high quality image authentication algorithm which uses the weighted-sum of the pixels of image blocks to insert the authentication bits. The scheme makes use of a parameter that controls the size of the block to be watermarked and the authentication payload. In addition, due to the use of the weighted-sum, the scheme embeds the payload bits by modifying merely one pixel per block by a value of ±1. To improve security, the scheme uses a key-lock pair access control mechanism, i.e., the authentication bits are treated as the lock and the watermarking secret key, assigned to each image, is considered as the key information. In (Botta et al. 2014) we proposed an improvement to their method.

Recently, there has been a trend to extend the fragile watermarks with a new requirement, that is self-recovery: put it simply, it implies the capacity of the watermark to recover the damaged areas of the digital object to its original state. A recent advancement in this area is the work proposed by He et al. (2012) which assesses the integrity (validity) of a block by comparing neighboring blocks. The embedding process is quite straightforward and consists in the partitioning of the image into non-overlapping blocks, a secure block mapping and the watermark generation and insertion. The watermark generated for each block has two components: i) the recovery information which is computed from the six most significant bit planes of each pixel in the block, and ii) a pseudo-random sequence of two bits used to overcome the constant-average attack (Chang et al. 2008). In order to recover the block in case of tampering its watermark sequence is inserted into the two least significant bits of the pixels of another block obtained through a secure mapping. Experimental results prove that the proposed method outperforms conventional self-recovery fragile watermarking algorithms in terms of detection and recovery of the tampered areas. Furthermore, the proposed scheme is secure against targeted attacks such as collage attack and four-scanning attack. The only drawback of the scheme is its low quality of the watermarked image, i.e., PSNR = 44.15 dB.

We conclude this section citing a generic algorithm that may be applied to many watermarking methods: Koval et al. (2011) propose a procedure based on Gaussian integers that may be used to shuffle (i.e. rearrange) the pixels of an image before the watermark embedding, and to restore them in their proper position after the insertion of the watermark. It is worth noting that for many watermarking algorithms this step can be a strong improvement w.r.t. security.

4 Proposed scheme

The algorithm developed has two key features: the first one is that the signal inserted into the image to be authenticated is dependent on both the image itself and on a symmetric secret key (an image shared among those involved in the authentication process). The second one is that the signal is embedded into a hidden space, thus the features that are modified are secret ensuring the fact that the embedded signal cannot be extracted to perform attacks to the integrity of the protected object.

The basic idea of the developed algorithm is to embed a secret watermark binary sequence W into the KLT coefficients of an input image $\boldsymbol{I_h}$ of size $N\times N$. The KLT basis is derived from a secret key image $\boldsymbol{I_k}$ of size $M\times M$. Moreover, the watermark W is made dependent on both the secret key image and the image to be watermarked, as explained in the following subsections.

The whole fragile watermarking scheme KLT-F consists of four algorithms, namely the KLT basis generation, the watermark generation, the watermark embedding and the watermark verification, implemented in two software applications: KLT-FW is used to watermark an image, while KLT-FV is used to check authenticity.

4.1 KLT basis generation

The key image $\boldsymbol{I_k}$ is divided into sub-blocks (sub-images) that are contiguous, non-overlapping and of size $n\times n$. Without loss of generality we assume that M is a multiple of n (if not, some image rows and columns will not be considered). The complete set of sub-images is considered as a random field, and from it a KLT basis is derived. Thus, n^2 basis sub-images are obtained, each one having n^2 components.

4.2 Watermark generation

The embedded watermark is generated as a function of the image to be watermarked and of the key image. This choice is made for two reasons: first, by generating a bit sequence that depends on (features of) the host image we prevent cut-and-paste attacks, birthday attacks and transplantation attacks (Barreto et al. 2002); secondly, by using the secret key image as the source of information for selecting features of the host image, no other information for defining the watermark sequence is necessary, thus allowing the image integrity verifier to use the secret key image only.

In the present implementation, we generate the bit sequence W to be embedded in the following way:

- the values of a set of pixel of the secret key image in predefined positions are used as pointers to address a set $\mathrm{P_h}$ of pixels in the host image;
- the values of the pixels in $\mathrm{P_h}$ are in turn used as pointers to a sequence of pixels $\mathrm{P_k}$ in the secret key image;
- the values of the pixels in $\mathrm{P_k}$ are used to initialize a cryptographic hash function (in our case we used SHA-256) that is called a sufficient number of times to produce the requested amount of watermark bits.

4.3 Watermark embedding

To insert the watermark W used to control the image authenticity, $\boldsymbol{I_h}$ is divided into non-overlapping, contiguous sub-images of size $n\times n$ which are transformed in column vectors of size n^2 (we assume, for simplicity, that N is a multiple of n). Let's call these sub-images $\boldsymbol{b}_m^h$, $1\leq m\leq(N/n)^2$. Then, each sub-image is treated separately, and its KLT coefficients $\boldsymbol{c_m}$ are computed using the previously generated secret KLT basis.

The algorithm inserts the watermark bits into a set of s coefficients $\{c_{t_1}, c_{t_2}, ..., c_{t_s}\}$, $t_i \in [1, n^2]$, $1 \le i \le s$, chosen from $\boldsymbol{c}_m$. The quantity s is representative of the payload of the algorithm and is measured in bits per block (bpb). Given that this subset (and its cardinality s) may influence the performance of the algorithm, an analysis on it will be performed in a following section. To embed the integrity watermark bits, each one of the selected KLT coefficients c_{t_i} is modified according to the following rule

$$c_{t_i}^{w} = \begin{cases} 2^p\left(2^{-p}c_{t_i}+1\right), if\ \left[2^{-p}c_{t_i}\right] \bmod 2\ \neq W_{(m-1)s+i}, c_{t_i} > 0 \\ 2^p\left(2^{-p}c_{t_i}-1\right), if\ \left[2^{-p}c_{t_i}\right] \bmod 2\ \neq W_{(m-1)s+i}, c_{t_i} \le 0 \\ c_{t_i}, \qquad otherwise \end{cases}$$

$$1 \le i \le s \qquad (2)$$

where $c_{t_i}^{w}$ represents the watermarked coefficient, [.] denotes the rounding towards $-\infty$ and $W_{(m-1)s+i}$ is the $((m-1)s+i)$-th watermark bit. With this embedding rule each watermark bit is inserted into the bit p of the binary representation of each selected coefficient.

After embedding the s bits into the KLT coefficients of the block, the inverse KLT is applied to obtain the modified pixels. The new pixel values will have, in general, real values, so they are rounded and eventually clipped to the original range (e.g. for 8-bit pixels, from 0 to 255). This non-linear operation may change the value of the embedded bit of the modified coefficients.

To restore the bits of the watermark into the coefficients we used a GA. An individual of the GA population is a vector $\boldsymbol{v}$ of n^2 integer values in the range [–3, +3], and represents pixel modifications of a sub-image, that once added to the actual sub-image pixel values, should allow to recover the correct watermark bits from the KLT coefficients.

The GA fitness function F takes into account two terms: the Bit Error Rate (BER), i.e. the correct extraction of the watermark bits from KLT coefficients, and the Mean Square Error (MSE), i.e. the distortion w.r.t. the host image:

$$F = \alpha \cdot BER_m + \min(\beta, MSE_m) \qquad (3)$$

where the BER_m and the MSE_m of sub-image m are computed as:

$$BER_m = \sum_{i=1}^{s}\left|W_{(m-1)s+i} - W_{(m-1)s+i}^{r}\right| \qquad (4)$$

$$MSE_m = \{(\boldsymbol{b}_m^h - \boldsymbol{b}_m^{GA})' \, (\boldsymbol{b}_m^h - \boldsymbol{b}_m^{GA})\}/n^2 \qquad (5)$$

where $W_{(m-1)s+i}^{r}$ is the recovered $((m-1)s+i)$-th watermark bit, $\boldsymbol{b}_m^{GA} = \boldsymbol{b}_m + \boldsymbol{v}$ is the corrected sub-image using individual $\boldsymbol{v}$ from the GA, and $\boldsymbol{b}_m$ is the vector sub-image resulting from the inverse KLT and rounding (intermediate sub-image). The aim of incorporating these two terms into F is to assure the correct extraction of the watermark's bits after the distortion caused by the pixel rounding operation and to preserve the quality of the marked image. Therefore, the fitness function F of an individual is a sum of these two parts, with the convention that the smaller F is, the better the individual. In particular, by choosing $\alpha \ge \beta$, if $F \le \beta$ then BER_m=0; in this way, we can always verify if the GA accomplished its task and found a solution.

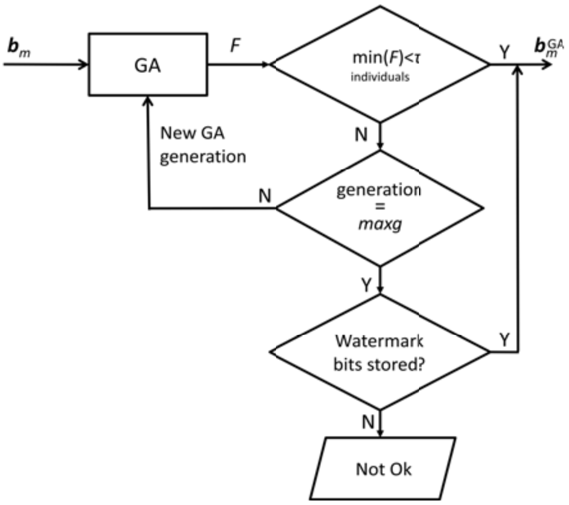


Fig. 1 Block diagram of the GA-based watermarking technique: $\boldsymbol{b}_m$ denotes the intermediate sub-image, F is the fitness function, τ is a threshold that is considered acceptable for the fitness value of an individual and *maxg* is the maximum number of generations

Figure 1 shows the block diagram of the operations performed by the GA for every sub-image. The constant τ ($\tau \le \beta$) is a threshold that is considered acceptable for the fitness value of an individual, and *maxg* is the maximum allowed number of generations. If a viable solution is not found in *maxg* generations then the GA is not able to converge starting from the initial population.

As suggested in (Goldberg 1989), for the GA parameters we used the following values: *population size* = 100, *maxg* = 500, p_s = 10, p_c = 0.8, p_m = 0.05 and α=β=150. From a multitude of crossover variants, we chose to implement the double-point crossover with tournament selection.

The final watermarked image $\boldsymbol{I_w}$ is obtained by composing all the sub-images resulting from the GA computation.

4.4 Watermark extraction and verification

We note that the selected pixels in P_h are also available to the image verifier given that the secret key image does not change. Thus, from the secret key image $\boldsymbol{I_k}$, the verifier extracts the KLT basis and then the expected watermark W^e can be derived. Afterwards, the watermarked image $\boldsymbol{I_w}$ is divided into sub-images of size $n\times n$. On the m-th sub-image, the KLT is applied and the coefficients $c_{t_1}, c_{t_2}, \dots, c_{t_s}$ chosen to store the bits are extracted, and from them each embedded bit is restored according to the following formula:

$$W^r_{(m-1)s+i} = \text{round}\left(2^{-p} c_{t_i}\right) \bmod 2\,,\ 1 \le i \le s \qquad (6)$$

By comparing the expected watermark W^e and the recovered watermark W^r, it is possible to establish if the image is genuine (the watermarks are equal) or if it has been modified (the positions of the differing bits indicate which sub-images are altered). Thus, the proposed method has a tamper detection resolution at a sub-image level.

It is important that the pixel values of the host image in the set P_h be left unchanged by the watermarking algorithm, otherwise the verifier would not be able to reconstruct the correct W, leading to an unsuccessful authentication even if the watermarked image did not undergo any modification. At the same time, it is worth noting that an attack modifying one of the selected pixels in P_h will lead to a completely different expected watermark W^e and thus to a highly likely failed authentication. Anyway, should this happen, the verifier is still able to detect the tampering, but loses its localization property. To reduce this possibility we kept the set P_h of these pixels very small (in our experiments we used just 2 pixels).

4.5 Watermarking example

We report a brief example of how the algorithm works on a block of 8×8 pixels. Let us suppose that the KLT basis has already been computed from a secret image (*pentagon*), and that 8 watermark bits are to be inserted in position p = 0 (i.e. the LSB of the integer part of the coefficient) into 8 consecutive KLT coefficients starting from the third. For the sake of clarity, only these 8 KLT coefficients (instead of all 64) will be shown.

Moreover, let us suppose the watermark 11100000 is to be inserted into the following host image block:

6	13	154	118	86	101	102	96
5	5	131	139	101	101	101	94
2	2	32	148	109	99	95	95
31	3	2	147	121	95	95	93
87	0	4	79	153	93	9	3
91	85	11	1	2	149	119	89
91	86	60	5	7	129	133	88
86	83	85	2	0	23	145	91

The 8 KLT coefficients that should carry the watermark computed from this block are: 32.431, –55.942, –35.078, 41.47, 79.788, –127.44, –196.31, 39.725; by applying formula (6), it is easy to see that the watermark bits are not stored into the block, so the correction (2) is applied to each coefficient, resulting in 31.431, –55.942, –35.078, 40.47, 78.788, –128.44, –196.31, 38.725 which inverse-transformed lead to the following image block (with pixel values rounded to integers):

5	13	153	118	86	100	102	96
5	5	131	139	101	101	101	94
2	3	32	148	109	99	95	95
32	3	1	147	121	95	95	93
88	0	4	79	153	93	9	3
91	85	11	1	2	149	119	89
91	86	60	5	7	129	133	88
86	83	85	2	0	23	145	91

where the shaded pixels had their values changed w.r.t. the original block.

The KLT coefficients computed from this block are 31.391, –56.255, –35.258, 40.573, 78.854, –127.97, –196.55, 39.178 and bear the watermark sequence 10111011 that is not the intended one; so the GA is applied to the image block and finds the following individual

0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0
0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	–1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

which summed to the intermediate block produces (in boldface the pixels changed w.r.t. the host block):

5	**14**	**153**	118	86	**100**	102	96
6	5	**132**	139	101	101	**102**	94
2	**3**	32	148	109	99	95	95
32	**4**	**1**	147	121	95	95	93
88	0	4	79	153	93	9	3
91	85	**10**	1	2	149	119	89
91	86	60	5	7	129	133	88
86	83	85	2	0	23	145	91

The KLT coefficients computed from the resulting block are 31.345, –56.845, –35.399, 40.272, 78.497, –127.61, –196.3, 39.688, and from (6) the watermark contained is 11100000 as desired.

5 Experimental analysis

The developed fragile watermarking algorithm[*] was tested by experiments aimed at showing its characteristics, performance, security and reliability. Obviously, given the flexibility of the proposed system, many other tests varying some of the parameters could be performed, but we feel we made a reasonable compromise among completeness, meaningfulness and complexity. The tests we performed were aimed at:

- testing which subset (of fixed size) of contiguous KLT coefficients produces the best PSNR; studying the effect of subsets of different sizes, and potentially non-contiguous coefficients, is another option, but has a high computational complexity;
- comparing the sensitivity of the algorithm to image modifications w.r.t. the size of the watermark message;
- comparing the performance of our algorithm to the performance of other algorithms (Yeung and Mintzer 1997; Ho et al. 2008; Oktavia and Lee 2005; Rawat and Raman 2011; Lin et al. 2011).

In all the following experiments, we used a set of 1000 images taken from various image collections (Li et al. 2008). These images are 8 bpp gray-scale images of size 256×256 pixels. The key images were also 8 bpp gray-scale images of the same size (but the latter characteristic was purely coincidental as the size of the key image has no relationship with the size of the image to be watermarked). Figure 2 shows the three key images used throughout the experiments. Moreover, the size of each sub-image block is fixed to 8×8 pixels.

5.1 Quality assessment

Apart from the visual assessment made by the authors and some of their colleagues, two other metrics were used to measure the quality of the watermarked images. The first one was the Peak-Signal-to-Noise ratio (PSNR) defined as

[*] A demo of our watermarking scheme is available at http://kdd.di.unito.it/~botta/KLTWatermark/watermark.html

$$PSNR = 10\log_{10}\frac{255^2}{MSE} \quad (7)$$

for 8 bpp images. The second one was the Structural Similarity index (SSIM) (Wang and Bovik 2009) that should give a more precise measure of the distortion and of the quality perceived by a human. The closer to 1 the SSIM, the less distortion is induced by the algorithm.

The first test we performed was the selection of the feature set. To this aim, we analysed the effect on the image quality of inserting the watermark in different coefficients subsets. We kept fixed the number of coefficients involved (we chose to use 8 *contiguous* coefficients) starting from the second one (i.e. the coefficients from the second to the ninth were marked) and arriving to mark the coefficients from the 20th to the 27th. Figure 3 reports the average PSNR along with the corresponding average SSIM, obtained by using the 3 different key images. We performed an ANOVA test to check if the reported values are statistically significant. From a statistical point of view, due to the large number of experiments performed, the F-value is > 6.3, so there is at least one configuration statistically different from the other. However, from a domain application point of view, as the standard deviation computed over all experiments is only 0.13, we can claim that the subset of coefficients involved has little influence on the image quality.

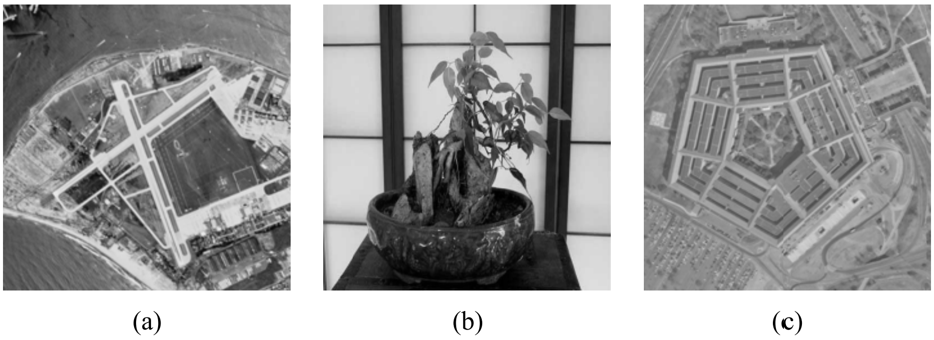


Fig. 2 The three images used as secret keys: (a) *base*, (b) *bonsai*, (c) *pentagon*

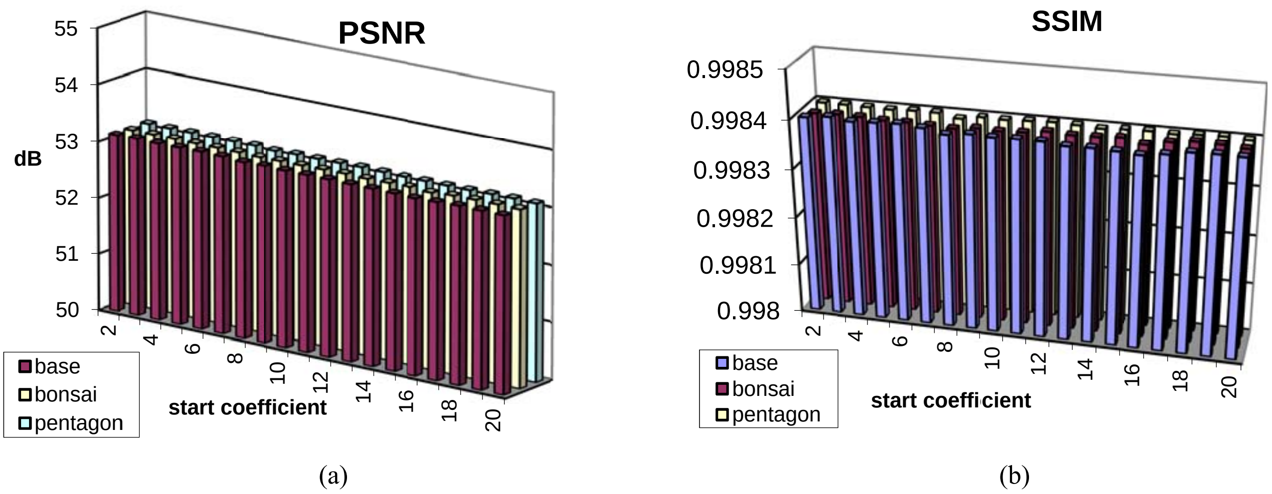
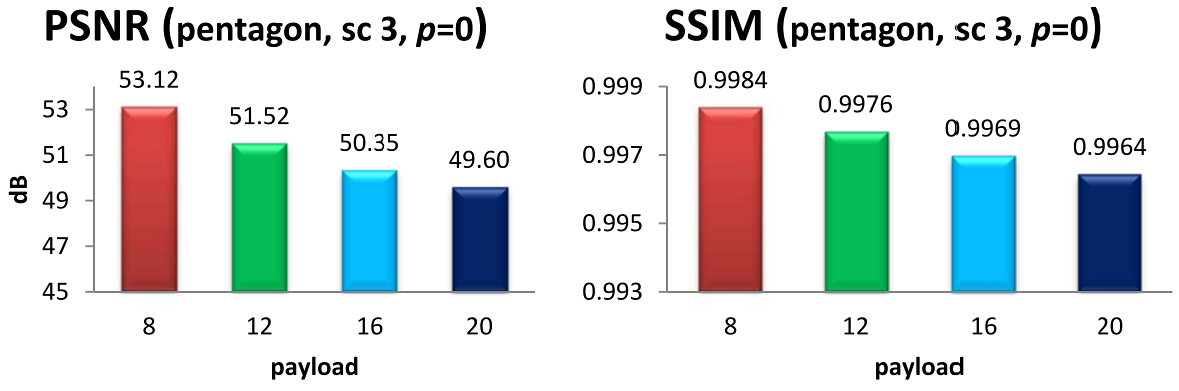


Fig. 3 Average image quality w.r.t. marked coefficients: (a) PSNR, (b) SSIM

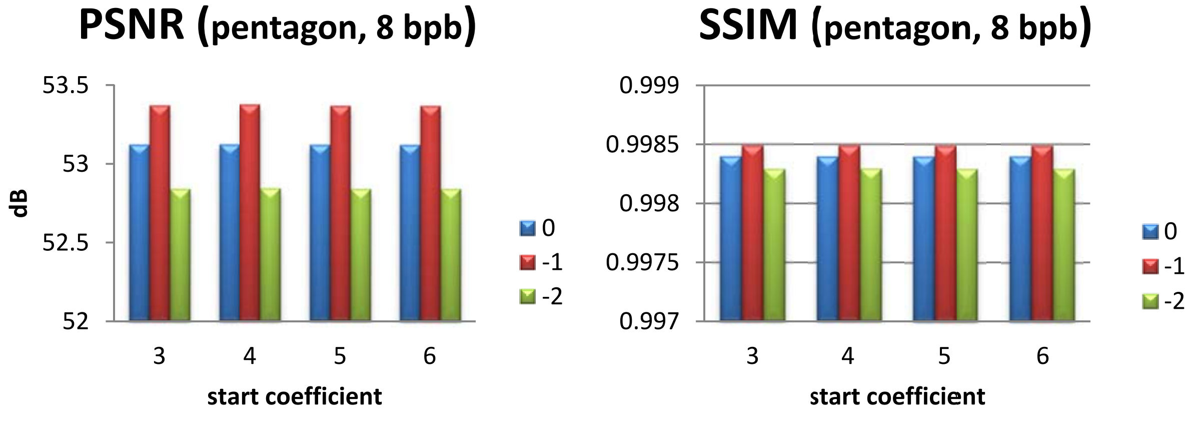


(a) (b)

Fig. 4 Average image quality w.r.t. payload: (a) PSNR, (b) SSIM

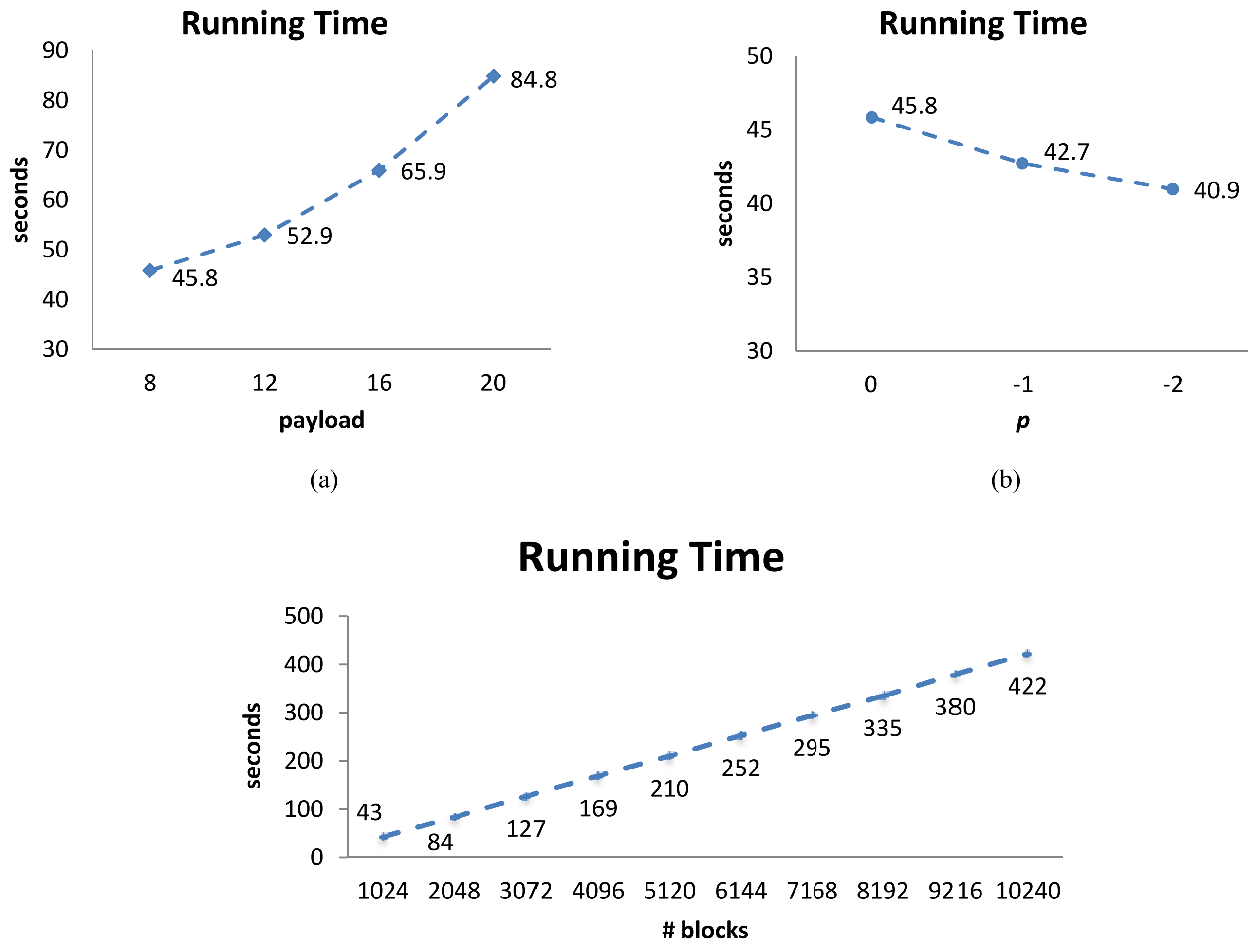
In a second experiment, we evaluated the image quality as a function of the payload, by varying the number of inserted bits per block from 8 to 20. Figure 4 reports the results obtained by inserting the watermark in the bit in position $p = 0$ of the KLT coefficients starting from the 3rd coefficient and by using *pentagon* as key image. Both PSNR and SSIM decrease as payload increases, as expected. Anyway, it should be pointed out that even with payload 20 bpb, the average PSNR is almost 50 dB, a value that is quite acceptable in many practical situations.

A third experiment aimed at evaluating the image quality w.r.t. the insertion bit position p. Again, Fig. 5 reports the results obtained by inserting 8 bpb into the KLT coefficients starting from the 3rd coefficient and by using *pentagon* as key image. Again, the difference is statistically significant according to a t-test (p-value < 0.00001), but in this application domain the obtained results can be considered equivalent, even though both PSNR and SSIM are slightly better for $p = -1$.



(a) (b)

Fig. 5 Average image quality w.r.t. insert bit position p: (a) PSNR, (b) SSIM



(a)

(b)

(c)

Fig. 6 Average running times in seconds using *pentagon* as key image, and inserting into KLT coefficients starting from the 3rd w.r.t.: (a) payload with p=0, (b) insert bit position with payload s=8 bpb, (c) number of image blocks with payload s=8 bpb and p=0

5.2 Performance assessment

For what concerns the performances, we computed the average running times (on an Intel(R) Xeon(R) CPU E5410 2.33GHz, 4GB RAM) of KLT-FW with respect to the payload and the insertion position (see Fig. 6(a) and 6(b)) using *pentagon* as key image and inserting from the 3rd KLT coefficient. Two observations are in order: first of all, the running time scales really well w.r.t. the payload; as shown in Fig. 6(a), doubling the payload increases the running time less than 50%.

Secondly, inserting into less significant bits of the KLT coefficients favorably affects running times, allowing the GA to converge faster on average (Fig. 6(b)). As expected, the running time scales linearly with the number of image blocks (Fig. 6(c)): thus the computational load is directly proportional to the image size.

Moreover, as shown in Table 1, the watermark verification process (KLT-FV) takes 0.065 seconds on average and allows for real time processing of the watermarked images. The verification process time scales linearly with the image size, as well.

Table 1 Average running times (seconds) on pentagon (start coefficient 3) w.r.t. insertion position

Position **Marking phase**	0	–1	–2
KLT-FW (s)	45.8 ± 1.2	42.7 ± 1.0	40.9 ± 1.8
KLT-FV (s)	0.0645	0.0648	0.0648

5.3 Classic tampering

To illustrate the output of the detection algorithm KLT-FV, we show how a modified region of an image is identified. The watermarked image "Aircraft" (embedding 8 bits per block) was used to generate counterfeits by modifying the number "16" on the tail and making it "61" by simply swapping different areas of the image. The fragment of the “Aircraft” watermarked image together with the corresponding counterfeit fragment is shown in Fig. 7. The identified tampered blocks are shown as crossed white blocks in Fig. 7(d).

Note that even if we change a small region of the authenticated image the proposed scheme succeeds in reliably detecting and localizing the tampered blocks with zero false-positives.

`

5.4 Tampering verification ability

Most of the discussed fragile watermarking schemes do not perform a rigorous analysis of their ability to verify tampering of the watermarked image. As shown in the previous section, KLT-FV is able to locate which blocks underwent a tampering manipulation, in case portions of the image swapped position.

The question we would like to answer now is the following: how sensitive is the algorithm to the alteration of a single pixel in a block? In order to assess the KLT-FV detection ability, the algorithm was tested against possible alterations of the watermarked image, by modifying in every block only one pixel at a time by a quantity δ, and computing the percentage of detected modified blocks. The detection ability against such an alteration is measured by the sensitivity rate, *SensR*, defined as follows:

$$SensR = \frac{no.\ of\ detected\ modified\ blocks}{total\ no.\ of\ modified\ blocks} \qquad (8)$$

.

In Fig. 8 we report *SensR* as a percentage of recognized modified blocks w.r.t. the payload, by changing the watermarked image pixels one at a time by a quantity $\delta = \{-2, -1, 1, 2\}$ (watermark inserted in position p=0 using *pentagon* as key image). As one would expect, *SensR* increases as the payload increases, as it is more likely that at least one of the KLT coefficients containing the watermark changes its carried bit value when a single pixel is tampered. Moreover, a larger distortion to an image pixel (δ=±2), results in a better performance of the verification algorithm, reaching 98% with a payload of 20 bpb. This is a very good result, as this means that even though an attacker only modifies a single pixel of δ=±2 levels in the entire image, KLT-FV still has 98% of chances to detect such tampering.

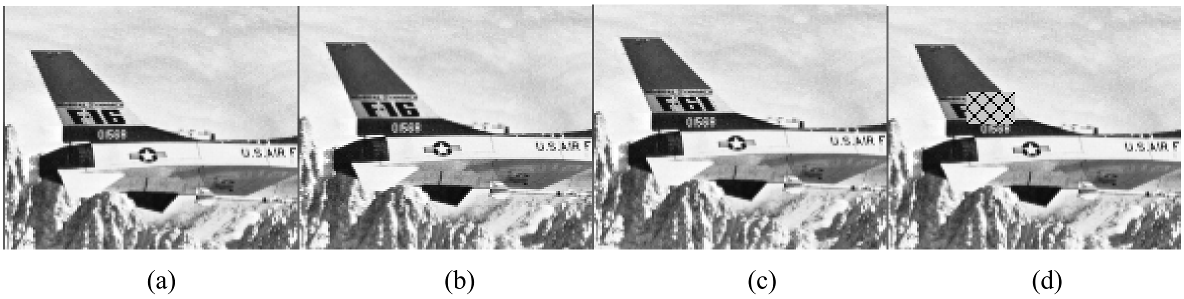

(a) (b) (c) (d)

Fig. 7 Tampering attack. From left to right: (a) the host image (fragment), (b) the watermarked image (SSIM = 0.998, PSNR = 53.04 dB), (c) the tampered watermarked image, and (d) the tampering map (crossed blocks denote the tampered areas)

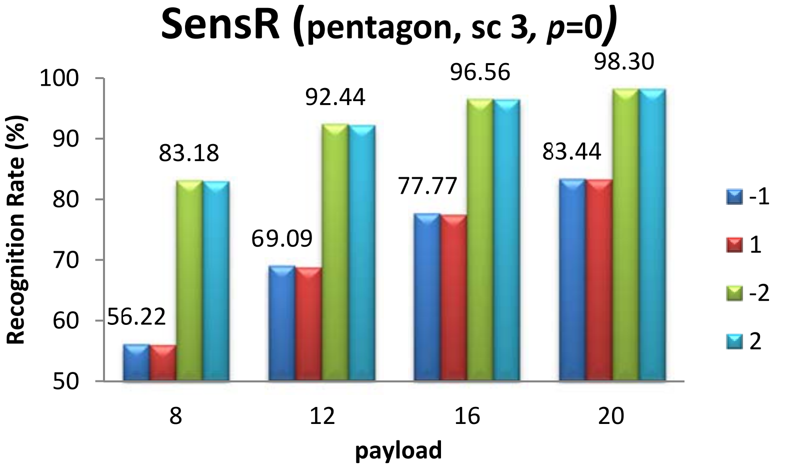


Fig. 8 Average percentage of detected modified blocks w.r.t. payload using *pentagon* as key image, inserting into position p=0 starting from the 3rd KLT coefficient

Figure 9 reports *SensR* as a percentage of recognized modified blocks w.r.t. the insertion position p, and changing the watermarked image pixels one at a time by a quantity $\delta = \{-1, -2, 1, 2\}$ (using *pentagon* as key image and payload of 8 bpb). It is worth noting that the percentage of recognized modified blocks reaches a value greater than 97% for $p = -2$ even when only one pixel per block is modified by just 1 gray level (statistically significant according to a t-test with p-value < 0.00001). This means that KLT-FV can reach high sensitivity either by increasing the payload, or changing the insertion position of the watermark. Note that altering more pixels in each sub-image increases the chances that the KLT coefficients carrying the watermark bits be altered, making it easier for KLT-FV to detect a tampered block. To prove the latter we made an extensive experimentation by changing two pixels at a time in every block of a quantity $\delta = \{-1, -2, 1, 2\}$, on the images watermarked using *pentagon* as key image, inserting into position $p = 0$ of the KLT

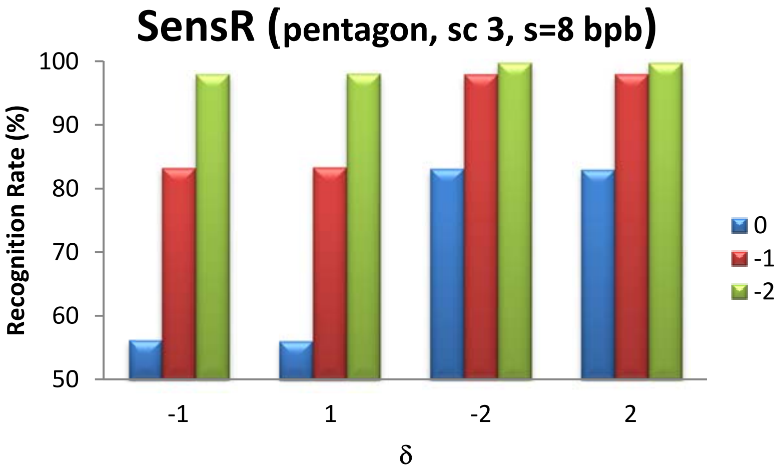


Fig. 9 Average percentage of detected modified blocks w.r.t. insertion position *p*, using *pentagon* and starting from the 3rd KLT coefficient

Table 2 Average block tamper detection percentages in case of JPEG/JPEG2000 compression

quality	**base**		**bonsai**		**pentagon**	
	JPEG	**JPEG2000**	**JPEG**	**JPEG2000**	**JPEG**	**JPEG2000**
100	89.8 ± 4.79	29.52 ± 28.39	89.58 ± 4.65	23.69 ± 20	89.57 ± 4.56	23.68 ± 19.51
95	99.6 ± 0.2	99.6 ± 0.19	99.62 ± 0.19	99.61 ± 0.18	99.59 ± 0.2	99.6 ± 0.2
90	99.61 ± 0.2	99.6 ± 0.19	99.61 ± 0.19	99.61 ± 0.19	99.61 ± 0.19	99.62 ± 0.19
85	99.6 ± 0.2	99.62 ± 0.19	99.61 ± 0.19	99.61 ± 0.2	99.6 ± 0.2	99.61 ± 0.19
80	99.62 ± 0.19	99.61 ± 0.2	99.6 ± 0.2	99.61 ± 0.2	99.61 ± 0.19	99.61 ± 0.2
75	99.61 ± 0.19	99.62 ± 0.19	99.62 ± 0.19	99.62 ± 0.19	99.61 ± 0.19	99.61 ± 0.2

coefficients starting from the 3rd, and computing for every combination the sensitivity *SensR*. It results that *SensR* is ~69% for *δ* = (±1, ±1), ~87% for *δ* = (±1, ±2), and ~92% for *δ* = (±2, ±2), respectively, figures that are much larger (between 10% and 20%) than in the case of altering just 1 pixel at a time, as expected.

5.5 Detection of JPEG and JPEG2000 compression

Given that the JPEG lossy compression is one of the most common image processing tasks, and that its characteristic modifications made to the image are typically difficult to find for a human, we tested the capability of the developed algorithm in detecting changes that this compression, at various quality levels, applies to an image. In our tests we considered both the older lossy standard JPEG and the most recent one JPEG2000.

Table 2 reports the percentage of recognized tampered blocks when the watermarked images (payload 8 bpb, start coefficient *sc* = 3, *p* = 0) undergo JPEG compression with a varying quality factor (from 100 to 75). As the quality factor of JPEG compression is less than 100, KLT-FV found almost all blocks as tampered. It should be pointed out that in case of JPEG2000 compression and quality factor 100, the percentage of tampered blocks is very small, and in 1 case (out of 3000 reported in Table 2) KLT-FV tagged the image as genuine. As a matter of fact, for such image JPEG2000 compression did not alter any pixel of the image.

5.6 Tamper detection in case of cropping

Image cropping is a possible attack aimed at removing parts of an image that are contiguous to a border. Some considerations are in order, but in all the cases the proposed algorithm will detect a cropped image as tampered.

First of all, if the starting position or the size of the cropped region are not a multiple of *n* then many blocks will be detected as modified because the watermark extraction algorithm will compute a different watermark w.r.t. the original bit sequence inserted (that is, the KLT coefficients are computed for blocks that do not coincide with the original ones).

Moreover, if the cropping has the effect to alter the set P_h, then also in this case the verifier will not be able to correctly reconstruct the embedded watermark, so many blocks will be detected as tampered.

If none of the previous cases happens, then the detection of a cropped region whose starting position and size are a multiple of the block size n can be performed by string alignment between the extracted watermark and the inserted one: obviously from a cropped image a shorter (w.r.t. the inserted one) watermark will be extracted, and missing parts may indicate the size and the position of the cropped area.

Finally, let us consider the case in which the cropped area was marked with the suffix of the watermark: tampering may go undetected in this case (even if the remaining part is correctly authenticated by the watermark). This problem may be solved by padding the watermark with a CRC code or a cryptographic hash of the previous part: a wrong CRC (or hash) will indicate at least one of the following facts:

- the watermark has been modified (parts of the image modified);
- the image has been cropped;
- at least one of the blocks bearing the CRC bits has been altered.

In any case, with this solution, the cropping of the ending part of the image does not go undetected, but the localization property may be partially lost.

5.7 Comparison with other algorithms

The PSNR and SSIM index values of other watermarking schemes, compared to KLT-FW, are given in Table 3. Note that the reported values of PSNR/SSIM were computed by running on the same set of 1000 images an implementation of these watermarking schemes (we assume that the given values are representative of the performances of these schemes). The comparison carried out in Table 3 shows that KLT-FW outperforms all of these schemes except that by Lin et al. (2011) in terms of quality, both PSNR and SSIM. However, this scheme is not as flexible as KLT-FW. Indeed the length of the watermark is strongly related to the block size (Lin et al. 2011), while KLT-FW, in principle, can embed any number of bits in a block, provided that the transformed space has enough dimensions. Moreover, the Lin et al. (2001) approach shows a counter-intuitive behavior when the bits per block increase; indeed, the probability of not detecting a tampered block depends on the number s of bits inserted ($1/2^s$), and then the probability of not calling an image tampered (false negative) is $\left(\frac{1}{2^s}\right)^{no.of\ blocks}$. As the number of blocks decreases with increasing values of s, the probability of false negatives exponentially increases. Furthermore, even though larger values of s imply better PSNRs, they result in bigger blocks and thus the localization ability of Lin et al. method is reduced. Instead, in KLT-FW, besides having a 0 false positive (i.e., calling tampered an image that was not altered) rate, the probability of false negatives decreases with increasing values of s, as expected, because the number of blocks does not change (leaving unaltered the localization ability of our method).

5.8 On the characteristics of the key image

One of the ideas that drove the development of this watermarking algorithm was the use of an image as the key defining the space of insertion. An obvious choice was the use of real world images, as shown by the tests. In this subsection we stress the idea that the key image can also be a random image. Namely, we wanted to test the behavior of our watermarking algorithm when the pixels of the key image are generated through a random process. For example, starting from a secret seed of 512 bits it is possible to generate pixel values through the iteration of a cryptographic hash function.

Table 3 Quality assessment of different fragile watermarking schemes

	PSNR (dB)	SSIM	Payload (bits)
Yeung et al. (1997)	46.06 ± 0.30	0.992	$N{\times}N$
Ho et al. (2008)	35.64 ± 1.84	0.898	$N{\times}N/8$
Rawat et al. (2011)	51.14 ± 0.01	0.997	$N{\times}N$
Oktavia et al. (2005)	51.14 ± 0.01	0.997	$N{\times}N$
KLT-FW	**53.12 ± 0.14**	**0.998**	$\boldsymbol{s\,N{\times}N/n^2}$
Lin et al. (2011)	58.06 ± 13.96	0.999	$(s+1)\,N{\times}N/2^s$

Table 4 Average PSNR using a random image as secret key w.r.t. start coefficient *sc* and insert postion *p*, payload *s*=8 bpb

Start coefficient \ Position	0	–1	–2
3	53.146±0.14	53.3867±0.15	52.8577±0.15
4	53.1288±0.13	53.3808±0.15	52.8574±0.14
5	53.1308±0.13	53.3881±0.15	52.8572±0.15
6	53.1385±0.14	53.389±0.15	52.8564±0.15

Table 4 reports the results obtained using a randomly generated key image. As it can be seen, the use of a such key image has little or no influence on the performances of the algorithm, being the average PSNR value equivalent to that obtained with real images. The use of a secret seed to generate the key image (in place of a real image) may be of value when the space for the key storage is an issue (e.g. smartcard applications), or when the key shared for the authentication should be transmitted with a low bandwidth requirement for real time applications.

6 Conclusions

This paper presents a fragile watermarking algorithm in the KLT domain based on genetic algorithms. The proposed algorithm uses the KLT to project the image pixels onto a hidden space in which the authentication information is inserted. Due to the characteristics of the KLT, a refinement step is necessary to ensure that the watermark is correctly present in the marked image: this step is performed with the aid of a genetic algorithm.

The security of the method is obtained through the use of a secret space defined by a key image. The key image is used to compute a KLT basis on which the pixels of the image to be authenticated will be projected. If the key image is kept secret, then it is inconceivably hard for an adversary to reveal the hidden message used for authenticating the image blocks. A high level of sensitivity is obtained from the insertion of the watermark bits into fractional parts of the transform coefficients. To avoid attacks aimed at copying and pasting parts of different authenticated images, the watermark is made dependent on both the key image and on a very small part of the authenticated image (these data are available also to the verifier).

We may recall the main characteristics in the following list:

- watermark inserted into a secret space defined by KLT;
- zero false positives thanks to a GA that corrects problems due to integer rounding of pixels;
- watermark dependent to key and host image to prevent some classes of attacks;
- high sensitivity to small modifications.

The first analysis performed on the algorithm was aimed at determining if the set of features (coefficients' set) influences the performances in terms of image quality. We found that, for the coefficients' sets examined, the effect in terms of PSNR and SSIM was not significant.

The paper also presents an extensive analysis on the detection capabilities of the algorithm, testing its sensitivity to small changes (a single pixel in a block modified by one or two gray levels); we found that with the improvement of inserting the watermark bits into the binary fractional part of the coefficients, the detection capability rapidly increases. An immediate consequence of this fact is that even with a small payload (namely 8 bpb) we already have a high sensitivity. We point out that the only problem due to a small payload of s bpb, is that the probability that a randomly chosen block substituted to the correct one is accepted as authentic is $1/2^s$.

Last but not least, we remark that the resources requested for the watermark insertion are limited, and that the image authentication is very light as it only requires simple multiplications, sums and bit comparisons. From the timing tests reported it can be seen that the complexity of the watermark insertion scales very well w.r.t. the payload. Moreover changing the bit insertion position to one that increases the sensitivity (namely second binary fractional bit) does not increases the insertion times (on the contrary, in our tests the required time decreases). The entire approach described in this paper applies to digital images, however the idea proposed could be easily adapted to other media than digital images, e.g. digital audio.

References

Aslantas V, Ozer S, Ozturk S (2009) Improving the performance of DCT-based fragile watermarking using intelligent optimization algorithms. Optics Communication 282(14):2806–2817

Barni M, Bartolini F, De Rosa A, Piva A (2002) Color image watermarking in the Karhunen-Loeve transform domain. Journal of Electronic Imaging 11:87–95

Barreto PSLM, Kim HY, Rijmen V (2002) Toward secure publickey blockwise fragile authentication watermarking. IEE Proceedings - Vision, Image and Signal Processing, 148(2):57–62

Botta M, Cavagnino D, Pomponiu V (2011) KL-F: Karhunen-Loève Based Fragile Watermarking. In: NSS 2011 5th International Conference on Network and System Security. Milan, Italy, pp 65–72

Botta M, Cavagnino D, Pomponiu V (2014) ‘Protecting the Content Integrity of Digital Imagery with Fidelity Preservation’: an improved version. ACM Trans. Multimedia Comput. Commun. Appl. 10(3), 29:1-29:5, April 2014, ACM, New York, NY, USA, doi: 10.1145/2568224

Bravo-Solorio S, Nandi AK (2011) Secure fragile watermarking method for image authentication with improved tampering localisation and self-recovery capabilities. Signal Processing, 91(4):728–739

Chang C-C, Fan Y-H, Tai W-L (2008) Four-scanning attack on hierarchical digital watermarking method for image tamper detection and recovery. Pattern Recognition, 41(2):654–661

Cox IJ, Kilian J, Leighton TT, Shamoon TG (1997) Secure spread spectrum watermarking for multimedia. IEEE Transactions on Image Processing, 6(12):1673–1687

Cox IJ, Miller ML, Bloom JA, Fridrich J, Kalker T (2008) Digital Watermarking and Steganography, 2nd edition. Morgan Kaufmann Publisher, San Francisco

Dafas P, Stathaki T (2003) Digital image watermarking using block-based Karhunen-Loeve transform. In: Proceedings of the 3rd IEEE International Symposium on Image and Signal Processing and Analysis, vol. 2. pp 1072–1075

Fridrich J, (2002) Security of fragile authentication watermarks with localization. In: Proc. SPIE 4675, Security and Watermarking of Multimedia Contents IV, 691. pp 691–700

Gonzalez RC, Woods RE (1992) Digital Image Processing. Addison-Wesley Publisher

Goldberg DE (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA

Han S, Chu C-H, Luo Z (2011) Tamper Detection in the EPC Network Using Digital Watermarking. IEEE Security & Privacy, 9(5):62–69

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

He H, Chen F, Tai H-M, Kalker T, Zhang J (2012) Performance Analysis of a Block-Neighborhood-Based Self-Recovery Fragile Watermarking Scheme. IEEE Transactions on Information Forensics and Security, 7(1):185–196

Ho ATS, Zhu X, Shen J, Marziliano P (2008) Fragile watermarking based on encoding of the zeroes of the z-transform. IEEE Transactions on Information Forensics and Security, 3(3):567–569

Koval A, Shih FY, Verkhovsky BS (2011) A Pseudo-Random Pixel Rearrangement Algorithm Based on Gaussian Integers for Image Watermarking. Journal of Information Hiding and Multimedia Signal Processing, 2(1):60–70

Lee S-K, Ho Y-S (2002) Fragile watermarking scheme using a simple genetic algorithm. In: Proc. of the International Conference on Consumer Electronics, pp 190–191

Li L-J, Wang G, Li F-F (2008) OPTIMOL: automatic Object Picture collecTion via Incremental MOdel Learning. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, pp 1–8

Lin P-Y, Lee J-S, Chang C-C (2011) Protecting the content integrity of digital imagery with fidelity preservation. ACM Trans. on Multimedia Computing, Communications and Applications, 7(3), Article 15:1–20

Liu P-P, Zhu Z-L, Wang H-X, Yan T-Y (2008) A Novel Image Fragile Watermarking Algorithm Based on Chaotic Map. In: Proc. of Congress on Image and Signal Processing 2008, vol. 5, pp 631–634

Oktavia V, Lee W-H (2005) A Fragile Watermarking Technique for Image Authentication Using Singular Value Decomposition. In: Proc. of Advances in Multimedia Information Processing, LNCS vol. 3332, pp 42–49.

Pan J-S, Abraham A (2009) Guest Editorial: bio-inspired information hiding. Soft Computing, 13(4):319–320

Pomponiu V, Cavagnino D, Basso A, Vernone A (2010) Data hiding schemes based on Singular Value Decomposition. In: Al-Haj A M (ed) Advanced Techniques in Multimedia Watermarking: Image, Video and Audio Applications, IGI Global, pp 254–288

Premaratne P, Ko C-C (1999) A Novel Watermark Embedding and Detection Scheme for Images in DFT Domain. In: Proc. of 7th Int. Conf. on Image Processing and Its Applications, vol 2. pp 780–783

Rawat S, Raman B (2011) A chaotic system based fragile watermarking scheme for image tamper detection. AEU-International Journal of Electronics and Communications, 65:840–847

Shih FY, Wu Y-T (2005) Enhancement of image watermark retrieval based on genetic algorithms. Journal of Visual Communication and Image Representation, 16(2):115–133

Shih FY, Wu Y-T (2005) Robust watermarking and compression for medical images based on genetic algorithms. Information Sciences, 175(3):200–216

Stanescu D, Stratulat M, Ciubotaru B, Chiciudean D, Cioarga R-D, Borca D (2007) Digital Watermarking using Karhunen-Loeve transform. In: Proc. of 4th International Symposium on Applied Computational Intelligence and Informatics, 2007, pp 187–190

Usman I, Khan A, Chamlawi R, Majid A (2007) Image Authenticity and Perceptual Optimization via Genetic Algorithm and a Dependence Neighborhood. Intl. Journal of Applied Mathematics and Computer Sciences, 4(1):37-42

Wang Z, Bovik AC (2009) Mean squared error: love it or leave it? A new look at signal fidelity measures. IEEE Signal Processing Magazine, 26(1):98–117

Wong PW, Memon ND (2001) Secret and public key image watermarking schemes for image authentication and ownership verification. IEEE Transactions on Image Processing, 10(10):1593–1601

Yeung MM, Mintzer F (1997) An invisible watermarking technique for image verification. In: Proc. of International Conference on Image Processing, vol. 2, pp 680–683

APPENDIX

Supplementary material is available on-line at

http://kdd.di.unito.it/~botta/KLTWatermark/watermark.html