Karhunen-Loève transform for Fragile Watermarking

Karhunen-Loève transform for Fragile Watermarking

Marco Botta, Davide Cavagnino, Victor Pomponiu
Department of Computer Science
Università degli Studi di Torino
10149, Torino, Italy
marco.botta@unito.it, davide.cavagnino@unito.it, victor.pomponiu@ieee.org

***Abstract*— This technical report presents a fragile watermarking technique useful to discover image manipulations at block level. The proposed algorithm is based on the Karhunen-Loève linear transform (KLT): it modifies some of the KLT coefficients obtained from a secret base defined by a secret key image so that these coefficients 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.**

Keywords: fragile watermarking, content authentication, Karhunen-Loève transform, genetic algorithms, data security.

INTRODUCTION

The diffusion of digital multimedia content like images, videos 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. Digital watermarking is a method that may be used to protect digital objects. Watermarks have many applications; for example, they may be used for copyright protection, content authentication, track of origin. 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.

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 enlighted;
- security against intentional attacks: the watermark should reveal attacks performed by cut-and-paste [1][2], transplantation and birthday attacks [3].

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.

The algorithm discussed in this technical report 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.

BACKGROUND

A. 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) is used to provide a simple method to define a secret key used to hide the authentication information, 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 represent 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 $\boldsymbol{B}$.

B. 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:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \tag{1}$$

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 [5].

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 $\boldsymbol{f}$ of the random field. 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 hidden: 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 coefficients, the visual distortion.

C. Genetic Algorithms

Genetic Algorithms (GA) 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. In this way, GAs may find (quasi) optimal solutions to optimization problems.

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. Then, the initial population evolves for a number of generations in which 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 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 [6]. 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. In one point crossover, the mating individuals exchange their genetic material from the beginning of their chromosome to a randomly chosen cutting point; in two points crossover, the material exchanged in the chromosomes is the one included in between two randomly selected points.

The reproduction step terminates by applying a mutation operator to each offspring: each chromosome 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 GA the reader is invited to consult [6, 7].

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.

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 also on a symmetric secret key (also 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 $\boldsymbol{I}_o$ 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: KLT-FW is used to watermark an image, while KLT-FV is used to check authenticity.

A. 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 are obtained, each one having n^2 components.

B. 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 original image we prevent cut-and-paste attacks, birthday attacks and transplantation attacks [3]; secondly, by using the secret key image as the source of information for selecting features of the original 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 P_o of pixels in the original image;
- the values of the pixels in P_o are in turn used as pointers to a sequence of pixels P_s in the secret key image;
- the sequence of pixels in P_s are used to initialize a cryptographic hash function (in our case we used SHA-256) that is called a number of times to produce a sufficient amount of watermark bits.

C. Watermark Embedding

To insert the watermark W used to control the image authenticity, $\boldsymbol{I}_o$ is divided into non-overlapping, contiguous sub-images of size $n \times n$ which are transformed in column vectors of size n^2. Let's call these sub-images $\boldsymbol{b}_m^o$, $1 \le m \le (N/n)^2$. Then each sub-image is treated separately, and its KLT is computed using the previously computed secret basis images obtaining a vector of coefficients $\boldsymbol{c}_m$.

The algorithm inserts the watermark bits into a set of s coefficients whose order is $\{t_1, t_2, ..., t_s\}$. 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:

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

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. To measure the number of watermark bits inserted in an image block w.r.t. the total number of blocks N^2, the bits-per-block (bpb) quantity is defined as $l = s/n^2$.

After embedding the *s* bits into the KLT coefficients, 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 takes into account two factors: 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 original image:

$$F = \alpha \cdot BER_m + \beta \cdot MSE_m \tag{3}$$

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

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

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

where $W^e_{(m-1)s+i}$ is the recovered *((m-1)s+i)*-th watermark bit, $\boldsymbol{b}^{GA}_m$ is the corrected sub-image using individual $\boldsymbol{v}$ from the GA, i.e., $\boldsymbol{b}^{GA}_m = \boldsymbol{b}_m + \boldsymbol{v}$, where $\boldsymbol{b}_m$ is the vector sub-image resulting from the inverse KLT and rounding. Therefore, the fitness function *F* of an individual is a weighted sum of these two parts, with the convention that the smaller *F* is, the better the individual. In particular, the BER_m weighting factor α is chosen to require BER_m=0 for the best chromosomes. Figure 1 shows the block diagram of the operations performed by the GA for every sub-image. The constant τ is a threshold that is considered acceptable for the fitness value of a chromosome, 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 find a solution, at least from the initial population used).

For the GA parameters we used the following values: *popsize* = 100, *maxg* = 500, p_s = 10%, and p_m = 0.05. 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.

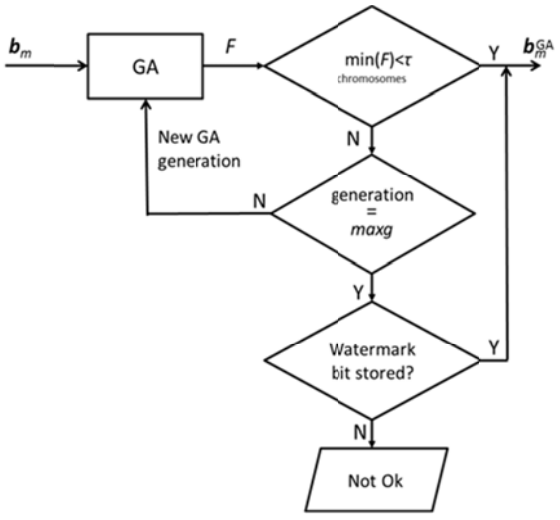

Figure 1: Block diagram of the GA-based watermarking technique: $\boldsymbol{b}_m$ denotes the intermediate sub-image, *F* is the fitness function and *maxg* is the maximum number of generations.

D. Watermark Extraction and Verification

We note that the selected pixels in P_s 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, as explained in the subsection C. *Watermark Generation*. 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:

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

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 pixels of the original image in the set P_o 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 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_o of these pixels very small (in our experiments we used just 2 pixels).

An advantage of our approach is that the extraction step is very light and requires a reduced computational power, and it can be performed in real-time.

REFERENCES

[1] J. Fridrich, "Security of fragile authentication watermarks with localization," Proc. of SPIE - The International Society for Opt. Engineering, vol. 4675, 2002, pp. 691-700.

[2] P. Wong, and N. Memon, "Secret and public key image watermarking schemes for image authentication and ownership verification," IEEE Transactions on Image Processing, 10(10), 2001, pp. 1593 1601.

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

[4] M. Botta, D. Cavagnino, and V. Pomponiu, "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.

[5] R. C. Gonzalez, R. E. Woods, Digital Image Processing, Addison-Wesley, 1992.

[6] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Kluwer Academic Publishers, Boston, MA 1989.

[7] A.-E. Hassanien, A. Abraham, J. Kacprzyk, and J. F. Peters, "Computational Intelligence: Foundation and Trends," Studies in Computational Intelligence, volume 96, Springer , 2008,.pp. 3-49.