Reversible Fragile Watermarking for Multichannel Images with High Redundancy Channels

Reversible Fragile Watermarking for Multichannel Images with High Redundancy Channels

Marco Botta [a], Davide Cavagnino [a], Victor Pomponiu [b]

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

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

[b] victor.pomponiu@gmail.com

Abstract. The paper presents a methodology to protect the integrity of multichannel images, having *some* highly redundant channels, by means of a reversible fragile watermarking algorithm. The watermark embedding phase uses a lossless compression method to compress the high redundancy channels, stores the compressed stream into their most significant bits, then embeds a secret fragile watermark by modifying the least significant bits of the high redundancy channels. In case the watermarked image is not modified, the host image can be perfectly reconstructed; otherwise, the modified area can be detected and located with very high probability and the area that has not been forged can be restored as in the original host image. The embedding of the watermark is performed by a Genetic Algorithm in the Karhunen-Loève Transform (KLT) domain: the use of a secret space defined by the KLT guarantees both security of the method and a high sensitivity in the detection of the forged areas.

Keywords: reversible watermarking; fragile watermarking; multichannel image processing; image authentication; genetic algorithm; Karhunen-Loève Transform.

1 Introduction

Digital data may undergo different kinds of attacks, like malicious modification (i.e. tampering), unauthorized copying and copyright infringement.

In the field of data hiding, digital watermarking (watermarking in the following) is a technology that may be used to protect digital contents, like images, videos and sounds, from the cited security attacks.

Basically watermarking embeds a signal W into a digital object I by modifying some parts of it. Depending on the objective of the protection, different watermarking algorithms are developed.

Before describing the properties of a watermarking algorithm we introduce a simple model of the whole process.

The watermarking process is composed by two distinct phases, sequential and non-overlapping in time. The first one is the *embedding phase*, which inserts a watermark signal W into

an object I possibly using a secret key K: the output is a watermarked object I_W. The second phase is a *extraction stage*: after receiving I'_W (a possibly altered watermarked object due to intentional attacks or to various kinds of transmission or storage errors) it returns the watermark signal present W' and/or a Boolean value stating if the original watermark is present; this phase, in general, needs the secret key K used to embed the watermark and may also require the original image I and watermark W.

Watermarking algorithms may be classified according to some properties and characteristics; we report the main ones in the following list:

- *reversibility*: if the extraction stage may obtain the original host image I from I_W and K (and possibly W) the method is *reversible*, otherwise it is *irreversible*;
- *robustness*: an algorithm designed to embed a watermark that must resist attacks aimed at its removal (like in copyright protection) is said to perform *robust* watermarking; on the converse, if the objective is to detect the minimal modification to the object (e.g. for integrity protection) then the algorithm must be designed to insert a *fragile* watermark;
- *blindness*: a method that does not need the host object I in the extraction phase is said *blind*, otherwise it is said *informed* (or *non-blind*);
- *embedding domain*: if the watermark is embedded directly in the data composing the object (like pixels for images, or samples for sounds) then the algorithm is said to work in the *spatial domain* or *time domain*; on the other hand, if the data is firstly transformed in another domain like the Fourier Transform domain or the Discrete Cosine Transform domain, and the watermark is embedded into the transform coefficients, then the algorithm works in the *frequency domain*; also, other domains are possible, like the *fractal domain*;
- *perceptibility*: while the other properties are objective, perceptibility is subjective because it is referred to a human judgment: if the degradation due to watermark embedding can be seen by an average observer the watermark is called *perceptible* (*visible*), otherwise it is said *imperceptible* (*invisible*); notice that there are algorithms that explicitly degrade the object superimposing a signal that makes the object unsuitable for any application unless it is removed (like the logos on images to be sold), or that insert a logo to identify a property, like ownership of a television transmission.

The proposed algorithm may be applied to multichannel images: it is reversible, fragile and blind; moreover, it embeds the watermark in a secret frequency domain defined by the Karhunen-Loève Transform. The high redundancy channels undergo a visible degradation due to the compression step, but they can be reversibly restored. The main properties of the algorithms are:

- *image tampering detection*: the algorithm verifies the integrity of the image and signals with very high probability any modification it has undergone;
- *localization*: any tampering is localized at block (i.e. subimage) level;
- *reversibility*: any image that has not been tampered may be restored to the original form it had before watermark embedding;
- *security*: due to the use of a secret space for watermark embedding and to the information stored into the watermark, the algorithm is secure against intentional and unintentional attacks.

The paper is structured into seven sections: following this introduction, the next one recalls the scientific papers dealing with the subject of the present research; then, a background section presents the main concepts used in the development of the algorithm, which is presented in the fourth section. A set of experiments proving the feasibility and capabilities of the method are shown

in section 5. Section 6 discusses the security of the algorithm and finally some observations and conclusions are drawn in section 7.

2 Related works

Works on watermarking of multichannel images have been mainly oriented to RGB color images, eventually also considering the α-channel. Algorithms for robust and fragile watermarking have been developed both in the spatial and in the frequency (transformed) domains: in the following of this section, we will firstly review some approaches and then we will describe some reversible watermarking schemes. Generally, we noted that it is somewhat difficult to make comparisons amongst and with these approaches due to the different features and characteristics, like localization capability, reversibility or embedding channels (e.g., RGB channels or α-channel only).

An interesting work on reversible data embedding in the spatial domain was proposed by Tian [23]: pairs of pixels are considered for carrying one bit of information, with the drawback to take into account the possible overflow and thus the need of a location map for storing the modified pairs positions. Alattar [0] improves the embedding capacity and efficiency of [23] by considering vectors of pixels instead of pairs.

[15] develops a data hiding method by considering the difference histogram of neighboring pixels: bits are embedded into pixels whose difference with the contiguous one is the most frequent (difference histogram shifting); overflow is dealt with the use of thresholds, under the assumption of low gradient gray level images.

In [17] a framework for applying reversible data embedding to fragile watermarking is developed. The work divides an image into tamper localization blocks and shows how to embed in these blocks a secure hash (apparently used with a key, thus properly a message authentication code) using any reversible data embedding algorithm; moreover, it uses the idea of merging blocks in case a single block is not able to completely carry a hash.

The algorithm presented in [6] embeds in the pixels integer values in the range $[1, n]$; it classifies pixels in three sets: the embedding ones, those that cannot carry information (called to-correct), and the original ones that allow a correct decoding of the data stream. Embedding and decoding are performed in two passes, and the algorithm is capable, for well-behaved images, to reach 2 bit per pixel payloads.

An elegant solution to fragile watermarking of images in the spatial domain has been developed in [14] and revised in [4]: the method non-reversibly watermarks image blocks and produces images with a high Peak Signal-to-Noise Ratio (PSNR) with some constraints on the block size; it is proposed for grayscale images but can be adapted to multichannel images as done in [5].

Embedding authentication information in the α-channel is applied in [3]: from an RGB image a completely transparent α-channel is added; then, the 2 LSB planes of this channel are zeroed and a Message Authentication Code (MAC) is computed for every image block; the MAC is XOR-ed with a watermark string and the resulting bits are stored in the α-channel LSBs. The use of the LSBs allows for a minimal impact on the Human Visual System.

In the work [12] the authentication information with self-repair capability is embedded into a newly added α-channel of a grayscale image: to minimize the alteration of the transparency the watermark is stored modifying α-values near the maximum of their range. This algorithm is extended and applied in [13] to color images, adding an α-channel to an RGB image embedding therein the authentication information (made of strings computed with the Shamir secret sharing algorithm).

In the frequency domain watermarking, one of the mostly used transform is the Discrete Cosine Transform (DCT). In [20] the DCT is used to compact the energy of a block and to generate part of a watermark: the DCT is applied for integrity protection and recovery of the tampered areas. In [28] the DCT is combined with fractal compression to embed a fragile watermark along with recovery information.

The Singular Value Decomposition is applied in [18] to authenticate image blocks: the singular values of the image blocks are used to compute authentication information which is successively used to modify the LSBs of the pixels along with secret keys to scramble data.

Genetic algorithms have been used in the context of image watermarking for minimizing distortion, as in [24] where the watermark is embedded in the spatial domain, or to cope with and correct the watermark string modification induced by pixel value integer rounding after watermark embedding in the DCT domain [2].

3 Background

This section is devoted to recall the main concepts used in the following sections with the objective of making the paper self-contained. Nonetheless, we report references to give pointers for an in-depth analysis and to perform a detailed study of the tools used by the proposed algorithm.

3.1 Channel redundancy

The algorithm requires that $p \geq 1$ channels of an m-channel ($m \geq p$) image have a certain amount of redundancy. In particular, the objective is to exploit this redundancy to compress the pixel values in these p channels to create an area that may be used to embed the watermark. In the verification phase, after the watermark extraction, the p channels may be restored to their original values, making the method completely reversible.

Every channel is divided into blocks of equal size and compressed at the pixel level, one block independently from the others; block independency allows for the tamper localization and block reversibility of the algorithm: even if a block is tampered and damaged, all other non-tampered blocks can be reversibly reconstructed.

In the present embodiment the embedding algorithm changes the LSB plane of a block in a channel, thus for images having k bit per channel per pixel the redundancy required must allow a compression ratio of at least $\rho_{min} = k/(k-1)$. An image channel for which $\rho \geq \rho_{min}$ holds for every block is called High Redundancy Channel (HRC), otherwise it is called Low Redundancy Channel (LRC).

3.2 The compression algorithm

We designed the compression of the HRCs to work locally for every block: the locality avoids that an attack modifying an area in the HRCs propagates to untouched areas nullifying the localization ability of the algorithm.

The pixels in a block are linearized using a raster scan order, one HRC after the other. Then, using a simple prediction method, the differences between a pixel value and the previous one are encoded with a given Huffman table (in a similar fashion to lossless JPEG or lossless PNG), the latter computed on a set of images and kept fixed and publicly available. The computational complexity of this procedure is linear in the number of pixels.

The first pixel of the first HRC channel of every block is preceded by a *virtual* pixel having value at half the dynamic range, i.e., for k bit per pixel per channel a value 2^{k-1} is considered.

The compressed stream is stored in the MSBs of the HRCs, leaving room in the free LSBs to the modifications for watermark embedding. The compressed stream is stored into the MSBs because the channel LSBs are modified to embed the watermark; moreover, referring to Fig. 1, the stream is embedded starting from the bits labelled 1 in all the pixels, then continuing to the bits labelled 2 and so on: this allows, in case of a highly compressible channel producing a short stream, to leave untouched the high order MSBs, i.e. 7, 6, 5, …, reducing the distortion in the watermarked image HRCs.

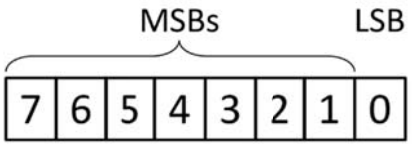

Fig. 1 LSB and MSBs of a pixel byte in a channel.

3.3 The cryptographic hash function

A cryptographic hash function (c.h.f.) is a function H that receives a bit string $\mathcal{S}$ of any length as input and produces a fixed length output $\mathcal{D} = H(\mathcal{S})$ (digest) having the following properties:

- given only a digest $\mathcal{D}$ it is computationally difficult to find a bit string $\mathcal{S}$ such that $H(\mathcal{S}) = \mathcal{D}$ (one-way property);
- given a message $\mathcal{S}$ it is computationally difficult to find a bit string $\mathcal{S}' \neq \mathcal{S}$ such that $H(\mathcal{S}) = H(\mathcal{S}')$ (weak collision resistance);
- it is computationally difficult to find any two bit strings $\mathcal{S} \neq \mathcal{S}'$ such that $H(\mathcal{S}) = H(\mathcal{S}')$ (strong collision resistance).

Quite a large number of such cryptographic functions have been developed in the past and some of them, like SHA-3 [7], may produce outputs of arbitrary length.

3.4 The Karhunen-Loève transform and bit embedding

The Karhunen-Loève transform (KLT) is a linear transformation between two vectors spaces $\boldsymbol{U}$ and $\boldsymbol{V}$. In case of vector spaces with finite dimensions a linear transform may be defined by a matrix: let $\boldsymbol{U}$ and $\boldsymbol{V}$ have dimensions a and b respectively, then a matrix A, called kernel, of size $b \times a$ defines a linear transformation between $\boldsymbol{U}$ and $\boldsymbol{V}$. A (column) vector $\boldsymbol{u} \in \boldsymbol{U}$ is mapped to a vector $\boldsymbol{v} \in \boldsymbol{V}$ with the mapping $\boldsymbol{v} = A\,\boldsymbol{u}$; generally, the elements of $\boldsymbol{v}$ are called coefficients of the transform. Examples of linear transformations are the Discrete Cosine Transform, the Fourier Transform and the Hadamard Transform: all of them have the characteristic of having a fixed kernel. Differently, the KLT (also called Hotelling Transform) has a square matrix kernel computed from a set of vectors: the transform operates a basis transformation to align the new basis along the lines where the vectors have the largest dispersion, therefore it corresponds to a Principal Component Analysis.

Given a set of a-dimensional vectors $\boldsymbol{u}_i \in \boldsymbol{U}$ a kernel A of size $a \times a$ for a KLT is derived with the following steps [9]:

- compute the average vector $\boldsymbol{\mu} = E\{\boldsymbol{u}_i\}$;
- compute the covariance matrix $C = E\{(\boldsymbol{u}_i - \boldsymbol{\mu})(\boldsymbol{u}_i - \boldsymbol{\mu})'\}$;
- compute the a eigenvectors $\boldsymbol{e}_j$ and associated eigenvalues λ_j of $\boldsymbol{C}$;
- (possibly) sort the eigenvectors by decreasing order of their associated eigenvalues;
- arrange the eigenvectors as rows of matrix A.

Given a vector $\boldsymbol{x} \in \boldsymbol{U}$ its KLT is computed as $\boldsymbol{y} = A(\boldsymbol{x} - \boldsymbol{\mu})$; it is possible to perform the inverse transformation KLT^{-1} with $\boldsymbol{x} = A^{-1}\boldsymbol{y} + \boldsymbol{\mu}$. The elements y_i of $\boldsymbol{y}$ are called coefficients of the transform and a coefficient's position i in $\boldsymbol{y}$ is called *order*.

In the proposed algorithm, the watermark bits will be encoded in the binary representation of KLT coefficients, so that a coefficient y_i carries a bit β in position α if and only if:

$$\beta = \mathcal{E}(y_i, \alpha) \quad \#(1)$$

where $\mathcal{E}(\mathrm{y_i}, \alpha) = \lfloor y_i \, 2^{-\alpha} \rfloor \bmod 2$ and $\lfloor \quad \rfloor$ is the floor function.

To embed a bit string $\mathcal{S}$ of length b bits into a vector $\boldsymbol{x}$ the following must be defined:

- a fixed sequence of l coefficients' orders o_1, o_2, ..., o_l, where $l \geq b$;
- a bit position α;
- a function $\mathcal{B} : \{0,1\}^l \rightarrow \{0,1\}^b$.

Then, applying slight modifications to the elements of $\boldsymbol{x}$, a vector $\boldsymbol{x}'$ is sought such that it holds

- $\boldsymbol{y} = A(\boldsymbol{x}' - \boldsymbol{\mu})$, and
- $\mathcal{S} = \mathcal{B}\left(\mathcal{E}\left(y_{o_1}, \alpha\right), \mathcal{E}\left(y_{o_2}, \alpha\right), \ldots, \mathcal{E}\left(y_{o_l}, \alpha\right)\right)$.

Previous studies [5] analyzed the influence of the parameters l, α and $\mathcal{B}$ on the final resulting image; in particular, the different tested $\mathcal{B}$ functions were direct mapping (used here), syndrome coding and modulo sum of coefficients [5].

The KLT is used to define a secret embedding space: in our setting the use of a secret image pixel values as vectors of an a-dimensional space $\boldsymbol{U}$ from which to derive the kernel A is a way to build a compact representation of a secret symmetric key.

3.5 Genetic Algorithms

A genetic algorithm (GA) is a computing paradigm that evolves representations of solutions to an optimization problem with the objective of finding an optimal solution.

When the representation of a problem may be coded as an ordered set of parameters $\omega_1\omega_2 \cdots \omega_g$ and it is possible to express with a function the degree of fitness of a sequence of values to the optimum then a GA may be used to evolve a set of candidates that approximate the desired solution.

A GA starts from a population of individuals, each one encoding a possible solution and initialized with random values.

The population is evolved through *generations* until an individual coding a viable solution is found (or a maximum number of generations is reached): in every generation the individuals are evaluated with a *fitness function* and are reproduced by means of operators and rules to build a new population.

In general, three operators are considered to reproduce individuals:

- *selection*: pairs of individuals are chosen from the actual population P and the one with best fitness in each pair is saved in a set $\tilde{P}$;
- *crossover*: pairs of individuals $\omega_1\omega_2 \cdots \omega_g$ and $\tau_1\tau_2 \cdots \tau_g$ in $\tilde{P}$ are considered for mating; with probability p_c the two individuals exchange their parameters: in one point crossover an integer random number ζ, $1 < \zeta < g$, is generated and the new offsprings $\omega_1 \cdots \omega_\zeta \tau_{\zeta+1} \cdots \tau_g$ and $\tau_1 \cdots \tau_\zeta \omega_{\zeta+1} \cdots \omega_g$ are created, in two point

crossover two random numbers are used to define the subset of parameters to be exchanged;
- *mutation*: to allow a better exploration of the solution space and avoid local minima, all the individuals in $\tilde{P}$ have one of their parameters randomly modified with probability p_m.

Many strategies are possible to build the new population P' starting from $\tilde{P}$; one is to replace P with the newly created individuals, another is to replace the worst individuals in P with the new ones, a third one is to perform tournament selection to produce P'.

For detailed information on GAs we suggest [8, 10] as starting points.

In the proposed algorithm the GA is used because of its ability to find a (quasi) optimal solution to a non-linear problem: as it will be shown in the following, altering the image pixel values to embed the watermark bit string is a problem in which an evolutionary algorithm succeeds producing good results.

4 The proposed algorithm

In this section logical data structures and the main steps of the proposed algorithm are presented.

The input to the algorithm consists of:

- a multichannel host image I_h, with m channels, where $p > 0$ of them have a high redundancy and $q = m - p$ are low redundancy channels, and
- a secret key image I_k.

Fig. 2 shows a diagram of the high level structure of the watermark embedding procedure, while Fig. 3 shows a diagram of the watermark verification procedure.

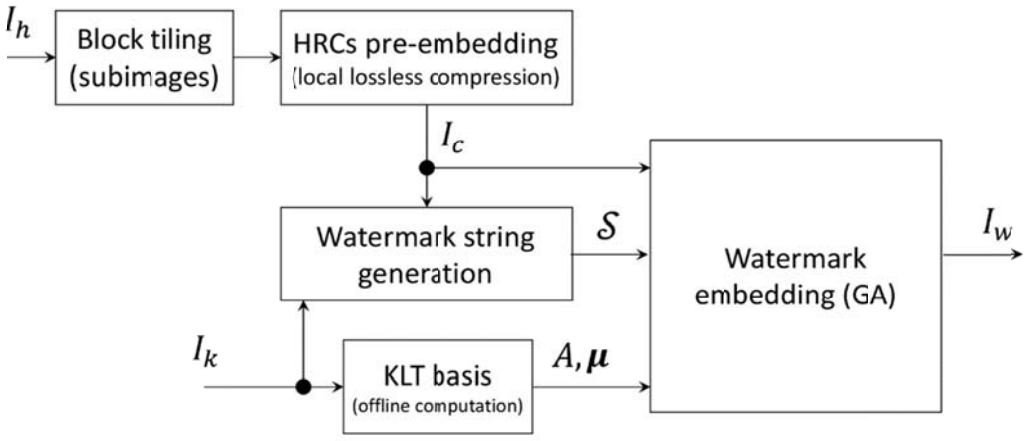

Fig. 2 High level structure of the fragile watermark embedding procedure: I_h is the host image, I_k is the key image, I_c is the host image with the compressed HRCs, I_w is the watermarked image, $\mathcal{S}$ is the secret watermark string.

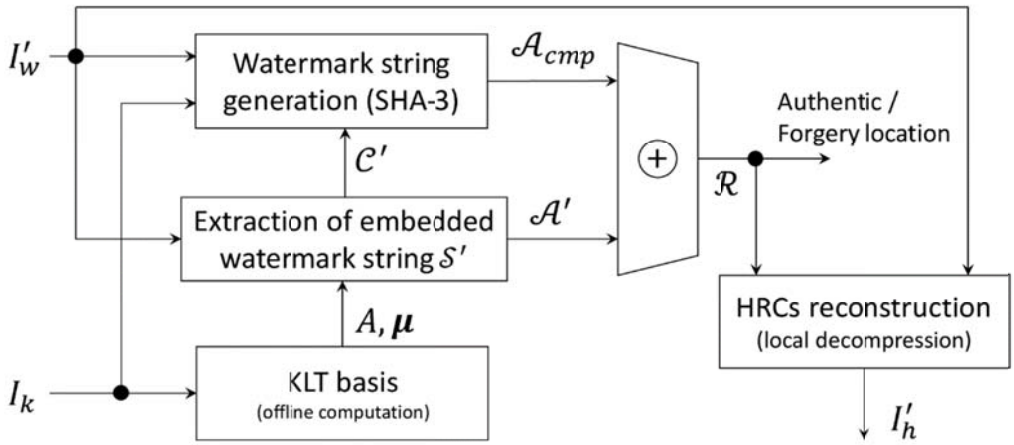

Fig. 3 High level structure of the fragile watermark extraction/detection and image reconstruction procedure: I'_w is the watermarked image to be verified, I_k is the key image, I'_h is the reconstructed host image, $\mathcal{A}_{cmp}$ is the expected watermark string, $\mathcal{A}'$ is the extracted authentication string, $\mathcal{R}$ is the string identifying the forged blocks.

The algorithm is composed of the following steps:

1) tiling of the host and key images into non-overlapping blocks;
2) computation of secret KLT basis (performed only once);
3) compression of the HRCs and inserting (pre-embedding) the compressed stream into the HRCs themselves;
4) generation of the authentication string;
5) generation of the watermark string;
6) watermark embedding;
7) watermark extraction and integrity verification;
8) original host image restoration.

These steps are described in detail in the following subsections.

4.1 Block tiling of I_h

Firstly, the m channel host image of size $H \times W$ pixels is conceptually divided into non-overlapping contiguous subimages of size $n \times n$ pixels[*] (see Fig. 4) onto which the watermarking algorithm operates independently (allowing parallelization): every pixel is composed of m channels, typically m bytes. Given that this is an operation oriented to data management it has no additional computational cost.

[*] For simplicity of the discussion we assume that both H and W are multiples of n.

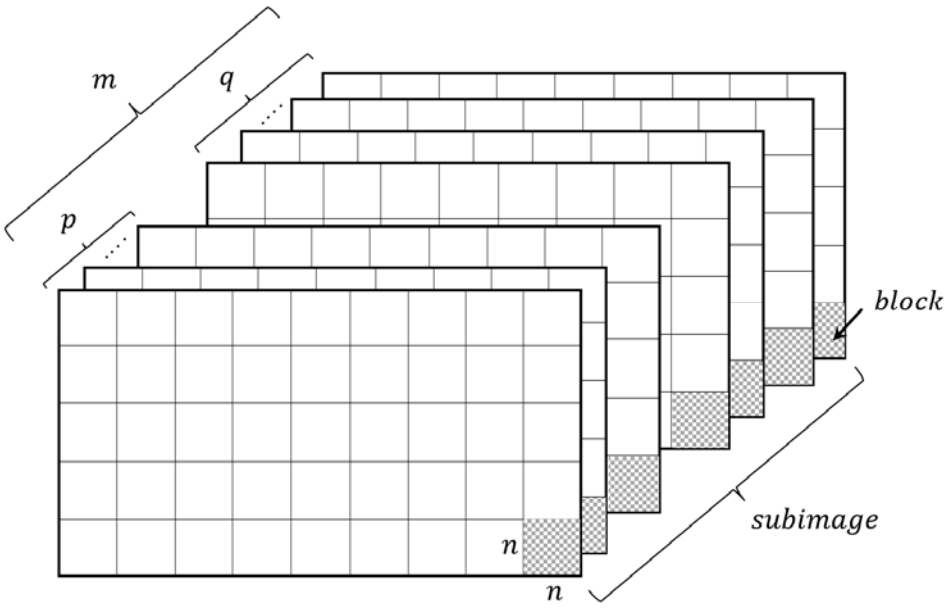

Fig. 4 Block splitting of a multichannel host image: the pixels in the p HRCs channel can be modified to carry the watermark, while the pixels in the q LRCs channels will not be changed by the embedding algorithm.

4.2 Secret KLT basis computation using I_k

The computation of a secret KLT basis may be performed once for every secret key image I_k, as long as the number of channels m of the host image(s) to be watermarked is known. After that, the same KLT basis may be applied to watermark any number of images with m channels. The computational complexity of this step is the one of the algorithm used to compute the eigenvectors and eigenvalues of a symmetric matrix.

The key image may be single channel or multichannel: to generate a set of samples, non-overlapping groups of mn^2 contiguous pixels are built. Then, the method described in the previous subsection 3.4 is used to produce a mean vector $\boldsymbol{\mu}$ and a kernel matrix A composed of mn^2 eigenvectors having mn^2 components.

4.3 High Redundancy Channels pre-embedding

A pre-embedding phase applied to I_h saves the information required to make the method reversible. In particular, the p HRCs are compressed in a lossless way (as presented in subsection 3.2), every block independently from the others, and the resulting compressed stream stored in the MSBs of the p redundant channels. By means of this operation we can save space and allow for the LSBs of the p channels to be used in the fragile watermark embedding.

As previously stated, the compressed stream is stored starting from the less significant MSBs leaving untouched the high order MSBs, i.e. first filling bit 1, then bit 2, then 3, and so on till bit 7 of the MSBs. Should the compressed stream be much shorter than the available space, this constraint may be relaxed, and one can start from bit 2 or bit 3 of the MSBs, so leaving more space for the watermark embedding.

The decoding of the compressed stream may be performed unambiguously, provided that the stream starts at a fixed position and the decoding table is known, as it is in our case.

The computational complexity of this step is linear in the number of blocks.

After the compressed stream has been stored in the MSBs of the p channels blocks, a watermark embedding phase similar to the one proposed in [5] is performed, using the image I_c resulting from the pre-embedding process as host image.

4.4 Authentication string generation

The proposed algorithm embeds an authentication bit string $\mathcal{A}$ computed from I_k and I_c. In this way different images will have different watermarks embedded (with the limits discussed below) with the aim of increasing security and protecting against transplantation and copy-and-paste attacks. The string $\mathcal{A}$ is generated by using SHA-3[†] (in its version for arbitrary output length SHAKE [7]) applied to a key obtained concatenating the following data:

- dimensions of the image (height H, width W, number of channels m), to protect also against image cropping and channel removal;
- a sequence T_k of pixel values in I_k whose coordinates are determined as follows: we use the values of a sequence of pixels in fixed (and publicly known) positions in I_k as indexes to select a sequence of pixels values T_c of I_c (in the LRCs, not modified nor compressed by our algorithm) which in turn are used as indexes to determine the sequence T_k.

Given the secrecy of I_k it is unfeasible for an attacker to derive $\mathcal{A}$.

4.5 Watermark string generation

The watermark string $\mathcal{S}$ carries two types of information, namely an authentication bit string $\mathcal{A}$ and a classification bit string $\mathcal{C}$.

1) The authentication bit string $\mathcal{A}$ is generated according to the method presented in subsection 4.4.
2) The classification bit string $\mathcal{C}$ specifying which channels are HRCs (1s) and which are LRCs (0s): these data consist of m bits (where m is the number of channels). For example, an image with 4 channels (RGBα) that has a highly redundant α-channel will have the bit string 0001 as classification bit string $\mathcal{C}$. These classification bits will allow to determine which are the HRCs and LRCs channels in the restoration phase.

If the payload is b bits-per-subimage (bps) then the length of $\mathcal{S}$ will be bHW/n^2; for every block, $b-1$ bits are consecutively taken from $\mathcal{A}$ and one bit is derived from $\mathcal{C}$: this means that for an $H \times W$ image with blocks of size $n \times n$ the required length of $\mathcal{A}$ will be $(b-1)HW/n^2$ bits and the m classification bits in $\mathcal{C}$ will be completely repeated $\left\lfloor \frac{HW}{mn^2} \right\rfloor$ times and correspondingly embedded in the image (for block size $n = 10$ and images of size $512 \times 512 \times 4$, we have $\left\lfloor \frac{HW}{mn^2} \right\rfloor = 655$).

For example, suppose a payload of 6 bps, an authentication bit string

$$\mathcal{A} = 00110\ 11001\ 11000\ 00010\ 10000\ 01100\ 11111\ 10101\ \ldots.$$

and a classification bit string $\mathcal{C} = \mathbf{0001}$. Then, the string $\mathcal{S}$ will be

$$00110\mathbf{0}\ 11001\mathbf{0}\ 11000\mathbf{0}\ 00010\mathbf{1}\ 10000\mathbf{0}\ 01100\mathbf{0}\ 11111\mathbf{0}\ 10101\mathbf{1}\ \ldots.$$

[†] The computational complexity of this step depends on the complexity of the SHA-3 algorithm.

4.6 **Watermark embedding**

After a KLT basis of mn^2 vectors is derived from the secret key image I_k and a watermark string $\mathcal{S}$ is computed, consecutive parts of $\mathcal{S}$, each one of b bits, are embedded into the subimages $n \times n$ (examined in raster scan order) by the GA.

For every subimage of I_c, the GA embeds the b bits by modifying the subimage HRCs LSBs. More precisely, the GA evolves a population where every individual represents a bit string used to substitute the subimage available LSBs (after the HRCs compression). The new subimage is KLT transformed and from l (fixed) coefficients b bits are extracted: if they coincide with those to be embedded, then the new subimage substitutes the original one in I_c to obtain, when all the subimages have been examined, the final watermarked image I_w. The evolution of the population is concluded when the b bits are embedded into the subimage or a maximum number of generations has been reached: in the latter case the image is declared non-watermarkable or a new population is created for a new attempt to embed the bits in the subimage.

The GA fitness function measures the Hamming distance between the bit string stored in the KLT coefficients and the bits of $\mathcal{S}$ pertaining to the subimage under examination, and the GA requires this distance to be equal to 0 for a viable block. No other quality measures are considered, like Peak Signal-to-Noise Ratio (PSNR), because the reversibility of the method allows to obtain the original channels: in fact, the GA may modify only the LSBs of the HRCs to embed the relevant part of $\mathcal{S}$ into the KLT coefficients of the subimage transform. In this way, in case a subimage is not modified by an attack it can be completely restored to its original values, making the method *reversible*.

When the GA has embedded the secret watermark string $\mathcal{S}$ into all the image subimages, the watermarked image I_w may be released in the public: its low redundancy channels will result untouched by the algorithm, whilst the HRCs will be altered with the property of being possibly restored by *anyone* to their original status if no attacks have been performed. Nonetheless, only the *holder* of I_k can check the integrity of the various subimages.

A high level scheme of the algorithm is reported in Algorithm 1 depicted in Fig. 5. The computational complexity of this step depends on the compression algorithm, as previously discussed, and the embedding has a complexity that is linear in the number of subimages; moreover, the computational load for every subimage depends on the parameter configuration of the GA algorithm used. In the current implementation, the GA runs for a maximum number of generations (2000), but it usually finds a solution in $25 - 35$ generations on average. Anyway, the total running time is upper bounded by tHW/n^2,where t is the maximum running time of the GA.

```
Algorithm 1:

Input: I_h image to watermark
       I_k secret key image
Output: I_w watermarked image
compute KLT basis from I_k
split I_h into subimages according to block size
select HRCs and LRCs: for every channel
                            compress all subimages
                            if (all subimages of a channel satisfy
                                      the compression ratio condition)
                                then mark it as an HRC
compress and pre-embed the pixel values of the HRCs and obtain I_c
compute the authentication string 𝒜 using I_k and I_c LRCs
generate the classification string 𝒞
interleave 𝒜 and 𝒞 to build the watermark 𝒮
for every subimage
      embed corresponding 𝒮 part into subimage using the GA (GA_embedding)
      if (embedding fails) then return Fail
return I_w

function GA_embedding
    generate population of random individuals coding HRCs LSBs
    for maxnumgenerations
      evaluate all individuals: if (one individual has fitness=0)
                                then return success and
                                     subimage having the
                                        individual in HRCs LSBs
      evolve population
    return embedding_failed

function fitness
    compute and return Hamming distance between
                            𝒮 part and ℬ(ℰ(y_{o_1}, α), ℰ(y_{o_2}, α), …, ℰ(y_{o_l}, α))
```

Fig. 5 Pseudo code of the fragile watermark embedding procedure.

4.7 Watermark extraction, verification and image restoration

The verification/restoration phase works as follows (see Fig. 3).

Firstly, every subimage in I'_w is transformed with the KLT (using the basis derived from I_k) and the embedded watermark $\mathcal{S}'$ is extracted. From $\mathcal{S}'$ the authentication string $\mathcal{A}'$ and the classification string $\mathcal{C}'$ are demultiplexed. In case of no attacks the $\left\lfloor \frac{HW}{mn^2} \right\rfloor$ copies of the m bit string classifying the channels will be identical, but if any subimage has been tampered there will be some differing string(s). To determine the HRC-LRC distribution the most frequent string will be chosen as the correct classification (as discussed in section 6, a wrongly decoded string will not expose any security breach).

After that, the watermark string $\mathcal{A}_{cmp}$ that should be found embedded is generated using the HRCs of the watermarked image I'_w and the secret key I_k.

Thirdly, a comparison performing the XOR operation $\mathcal{R} = \mathcal{A}' \oplus \mathcal{A}_{cmp}$ leads a resulting string that has ones ('1') corresponding to altered subimages.

If the part (having length b bits) of $\mathcal{R}$ corresponding to a subimage is all 0s the subimage is considered intact and the HRCs are restored decompressing the stream in the MSBs, otherwise the

subimage is marked as forged. So, the subimage is the minimum forge localization area, i.e. image modifications are signaled to the user by marking the whole subimage containing the forged pixels.

A high level scheme of the verification and restoration Algorithm 2 is reported in Fig. 6. The computational complexity of the verification and reversible phase is linear in the number of subimages.

Algorithm 2:

Input: I'_w possibly watermarked image
I_k secret key image
Output: I'_h verified and restored image
compute KLT basis from I_k
split I'_h into subimages according to block size
for every subimage X
extract $\mathcal{S}'(X)$ with KLT^{-1} and function $\mathcal{B}$
select HRCs and LRCs from $\mathcal{C}' \subset \mathcal{S}'$
compute watermark string $\mathcal{A}_{cmp}$ using I_k and I'_w LRCs
for every subimage X
if $(\mathcal{A}'(X) = \mathcal{A}_{cmp}(X))$
then decompress subimage HRCs and restore subimage in I'_h
else mark subimage as forged in I'_h
return I'_h

Fig. 6 Pseudo code of the fragile watermark verification and restoration procedure.

5 Experimental results

This section summarizes the results of a large number of experiments we executed to test the performance of the proposed algorithm: we applied the reversible fragile watermarking algorithm to a set of images with at least one HRC.

We considered three objective parameters, namely the sensitivity, the Mean Absolute Error (MAE) and the Peak Signal-to-Noise Ratio (PSNR). Among them, we consider the sensitivity the most important, given that the reversibility property of the algorithm allows the authorized user to obtain the original host image as final result. Indeed, the sensitivity[‡] gives a measure of the capability of the algorithm to detect tampering attacks.

Thus, the meanings of the parameters used to measure the performance of the proposed algorithm are:

- Sensitivity (as in [5]): the sensitivity of level D is defined as the fraction of blocks that are detected as tampered in the verification phase when just one pixel per block is modified by D levels;
- Mean Absolute Error (MAE): measures the average absolute difference between pixels of the host and the watermarked images;
- Peak Signal-to-Noise Ratio (PSNR): this classical objective quality measure is defined as

[‡] We designed an experimental setting in which we test how many times the verifier misses a tampered subimage when a single pixel of that subimage is changed by just 1 or 2 grey levels. This is actually the smallest tampering an attacker can perform: any other attack, such as filtering, blurring, noise addition, copy-and-paste, cropping, etc., would modify more pixels and, very likely, of more than just 1 grey level.

$$\text{PSNR} = 10 \log_{10} \frac{MaxLevel^2}{MSE}$$

where $MaxLevel$ is the maximum intensity level of each channel (i.e., for 8 bit images it is $2^8 - 1 = 255$) and MSE represents the mean squared error between the host and the watermarked images.

We chose a setting for the genetic algorithm to be used in all experiments; in particular, we derived the setting values from many experiments devised to have fast convergence to a solution resulting in a high quality image. The settings were population size $= 100$, $p_c = 0.9$, $p_m = 0.06$, maximum generations $= 2000$.

The first set of tests were performed on a public collection of images (from the Telegram application [21, 22]); the 29 images are in PNG RGBα (RGB with alpha channel) format: the alpha channel is not trivial (in the sense that it varies on the whole image to follow the sticker shape) and we expected it to be a HRC. Table 1 reports the watermarking results for two different values of payload; the PSNR is influenced only by the differences induced by the HRC, as the process does not alter any LRC pixel.

In the presented tests the block size was set to $n = 11$ (i.e. the subimages were composed of 11×11 four channel pixels). We also conducted tests with smaller values for n, but for some images no channel was an HRC, so we selected the smallest value for which we could compress enough every block.

Table 1 Performance of the proposed algorithm on a set of RGBα images publicly available [21, 22].

Payload (bps)	Sensitivity ±1 (%)	Sensitivity ±2 (%)	MAE	PSNR (dB)	Time (s)	GA Avg Generations
8	86.79±2.83	96.61±0.98	0.6056±0.1544	36.54±3.54	46.77±8.18	26
10	89.23±1.07	96.81±1.14	0.6171±0.1541	36.52±3.52	59.78±11.39	32

We compared our algorithm with the ones proposed in [13] and [20], that have been showed superior against other approaches ([11], [16] and [19], [25], [26], [27] respectively). One of the settings of [13] is RGB image authentication, which is performed by *adding* an alpha channel where every pixel stores three authentication bits derived from the corresponding RGB channel values (scrambled with a key) using a secret sharing method: the effect is that every alpha channel pixel contains the sum of 248 and the authenticating value; moreover, the authentication data is distributed among groups of three pixels each, making each group the smallest forge localization unit.

Our algorithm, instead, works on any existing HRC channels, thus having a wider range of applications to images of any kind; the disadvantage w.r.t. [13] is the bigger authentication block size.

In [20], Singh et al. described a fragile watermarking algorithm for RGB color images with restoration capabilities of tampered areas, whereas our proposed algorithm only restore untampered blocks but works with any multichannel image.

The images for the comparison reported in Table 2 were obtained by the classical ones (as the names recall) of size 512×512 pixels adding an extra HRC channel completely opaque (i.e. having pixel values equal to 255), as done in [13].

From Table 2 it may be observed that the MAEs computed on the HRC (i.e. the alpha channel) computed by our algorithm are an order of magnitude lower than those in [13]. Moreover, the PSNR is much greater than that reported in [20] on the same set of images.

Table 2 Comparison of MAE and PSNR values for some images.

Image	MAE		PSNR	
	[13]	Proposed algorithm	[20]	Proposed algorithm
Baboon	0.723	0.0192	39.55	65.26
Lena	0.720	0.0190	39.79	65.29
F-16	0.721	0.0191	-	65.25
Tiffany	0.721	0.0194	-	65.30
Boat	-	0.0191	39.93	65.33
House	-	0.0190	39.16	65.33
Pepper	-	0.0194	40.19	65.26
Woman	-	0.0194	39.73	65.25

Extending the test in Table 2 to a larger set of images (500 color images) we obtained an average MAE value of 0.0336 ± 0.0003 and an average PSNR value of 62.86 ± 0.037 confirming the high quality of the watermarked images; moreover, the standard deviation of the resulting MAEs and PSNRs is very low and confirms that the algorithm is stable and is not influenced by the image content.

As an example of the resulting whole authentication process, we present in Fig. 7 an image (a) to show the result from the watermarking process (b) and how the proposed algorithm detects (d) the modifications made in (c): as the HRCs are the blue and alpha channels of this image, it is impossible to notice any difference between host and watermarked images; on the other hand, the image is restored to its original state (d) apart from the tampered blocks in the upper-right part of the image where a counterfeit flame was added.

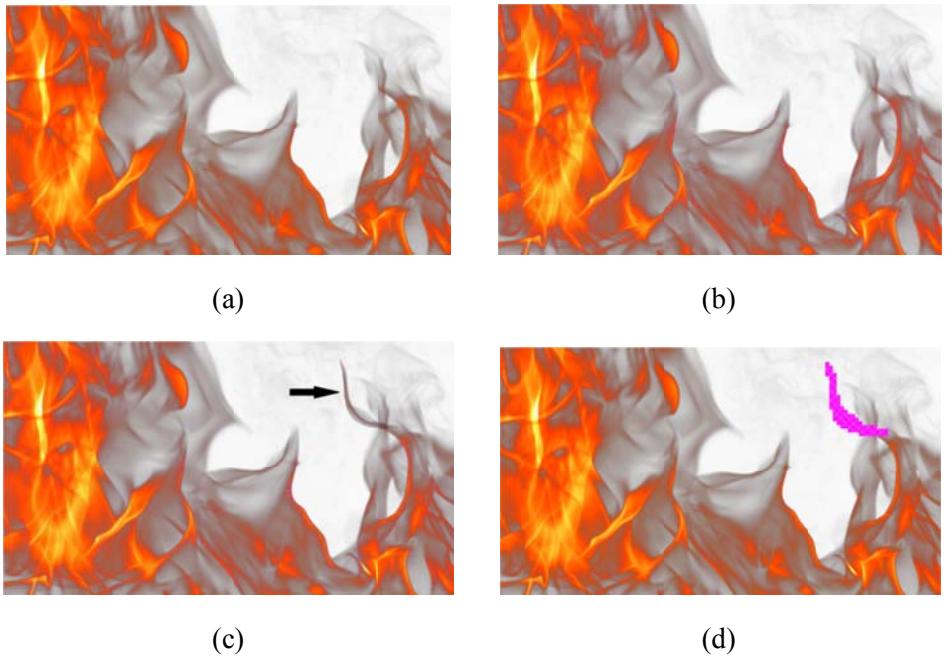

Fig. 7 An example of watermarking and tampering detection: (a) original four channel image (RGB with alpha channel, courtesy of and © Footage Firm, http://www.stockphotosforfree.com); (b) watermarked image, red and green channels untouched, PSNR 30.80 dB; (c) forged image, the added flame is indicated by the black arrow; (d) restored image and tampered area, in magenta color, detected by the algorithm.

Verification time is linear in the number of subimages and the time required to verify a single subimage is of the order of 0.5 msec.

6 Security of the method

The security of the fragile watermarking method is based on the evidence that the embedding space is secret. Indeed, the KLT basis is secret because the sample from which it is computed is secret as is the key image I_k. Thus, the watermark cannot be extracted (nor removed) without the possession of the secret key image.

As shown in section 5, the method is highly sensitive even to small modifications, also having a good ability at tampering localization.

The only observable effects of the watermark embedding are the compressed pixels of the HRCs and the string in the HRCs LSBs: the latter is strictly tightened to the pixel values (and also to the watermark) and cannot be replaced with other pixel values without exposing a forgery detection.

If the watermarked image goes through heavy and extended modifications it is possible that the redundantly embedded classification bit strings have not a frequency distribution with an evident mode. In case the HRCs-LRCs are not correctly identified in the verification phase then the wrong channels of the T_c pixels will be used to generate $\mathcal{A}'$ producing a string uncorrelated with the correct one: almost all subimages will be flagged as forged (even if possibly some of them will not); indeed, many altered subimages have led to a wrong $\mathcal{C}'$ string. The localization ability is lost but the image modification is detected.

In case of cropping or modification attack involving at least one of the pixels (in the Low Redundancy Channels) used to generate the watermark, almost all subimages will be flagged as altered because $\mathcal{A}_{cmp}$ will be completely uncorrelated with $\mathcal{A}'$; in this case no forged subimage

localization is possible (anyway the modification is detected and the image declared altered): for this reason we suggest to keep small the set T_c of pixels used to create the watermark (typically four pixels).

A feature of the method is its public reversibility: anyone may reconstruct an image from I_w because the compression/decompression algorithm and Huffman table for the HRCs are public. Nonetheless, only the watermark embedder has the capability to know if any modification to I_w has been performed and which subimages have been forged. Thus, the possibility to reversibly reconstruct the original host image left to anyone is only apparent and poses no security problems, in fact:

- without the key image I_k it is not possible to test if the watermarked image is intact or has been forged, so the reconstruction of I_h is possible only for the watermark embedder;
- anyway, knowledge of I_h does not disclose any critical information because the objective of a fragile watermarking algorithm is to verify the integrity of objects, not hide their content.

Another consideration that is important is that when embedding b watermark bps there is a probability of $1/2^b$ that a random subimage substitution goes undetected. Obviously, the probability that Z different forged subimages are all undetected is $1/2^{bZ}$, which drops dramatically even for small values of b and Z.

As shown in section 5, the main factor to take into account for the applicability of the method is the sensitivity to *single pixel-single intensity level* modifications: as shown, the proposed algorithm is very sensible to these modifications with a high probability of detecting this kind of little tampering, which translates in a strong sensitivity to forgery and a high capability of localization.

7 Discussion and conclusions

In this paper a reversible method for fragile watermarking multichannel images has been presented and discussed: the requirement is for the image to possess one, or more than one, high redundancy channels.

The resulting watermarked image has the low redundancy channels untouched, and contains a compressed version of its HRCs allowing for a complete restoration of the original host image.

The owner of the secret key image can verify the integrity of the image. Everyone is able to restore an image, but only the secret key owner has the certainty of its integrity.

The test performed on a large set of images has shown the high sensitivity of the algorithm to attacks and the ability to localize them.

To summarize, the method has the following properties:

- reversibility;
- low redundancy channels do not need restoration, as they are untouched (nonetheless, they may obviously be forged by an attacker);
- high sensitivity to modifications;
- fast image integrity verification and tamper localization ability;
- applicable to any uncompressed, or lossless compressed, image file formats.

The proposed algorithm fits in the MIMIC framework [5], adding a pre-embedding and a decompression module, along with slight modifications for the management of the HRC-LRC classification.

Acknowledgements

We would like to thank Dr. Roberto Esposito who helped in revising the manuscript according to the reviewers' comments and making new experiments, pointing out and resolving issues in the first submission.

References

1. Alattar AM (2004) Reversible watermark using the difference expansion of a generalized integer transform. IEEE Transactions on Image Processing 13(8):1147–1156
2. Aslantas V, Ozer S, Ozturk S (2009) Improving the performance of DCT-based fragile watermarking using intelligent optimization algorithms. Optics Communications 282(14):2806–2817
3. Bandyopadhyay P, Das S, Chaudhuri A, Banerjee M (2012) A New Invisible Color Image Watermarking Framework through Alpha Channel. International Conference on Advance in Engineering, Science and Management p 302–308
4. 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
5. Botta M, Cavagnino D, Pomponiu V (2016) A modular framework for color image watermarking. Signal Processing 119:102–114
6. Chaumont M, Puech W (2009) A High Capacity Reversible Watermarking Scheme. Proceedings of SPIE 7257
7. Dworkin MJ (2015) SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. NIST FIPS – 202
8. Goldberg DE (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co.
9. Gonzalez RC, Wintz P (1987) Digital Image Processing 2nd ed. Addison-Wesley Publishing Company
10. 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
11. Kim KS, Lee MJ, Lee HY, Lee HK (2009) Reversible data hiding exploiting spatial correlation between sub-sampled images Pattern Recognition, 42(11), p 3083–3096
12. Lee C-H, Tsai W-H (2012) A Secret-Sharing-Based Method for Authentication of Grayscale Document Images via the Use of the PNG Image With a Data Repair Capability. IEEE Trans. on Image Processing 21(1):207–218
13. Lee C-H, Tsai W-H (2013) A data hiding method based on information sharing via PNG images for applications of color image authentication and metadata embedding. Signal Processing 93(7):2010–2025
14. Lin P-Y, Lee J-S, Chang C-C (2011) Protecting the Content Integrity of Digital Imagery with Fidelity Preservation. ACM Trans Multimedia Comput Commun Appl 7(3):15:1–15:20
15. Lin C-C, Tai W-L, Chang C-C (2008) Multilevel reversible data hiding based on histogram modification of difference images. Pattern Recognition 41(12):3582–3591

16. Luo L, Chen Z, Chen M, Zeng X, Xiong Z (2010) Reversible Image Watermarking Using Interpolation Technique. IEEE Transactions on Information Forensics and Security, 5 (1), p 16–21

17. Naskar R, Chakraborty RS (2013) A Generalized Tamper Localization Approach for Reversible Watermarking Algorithms. ACM Trans Multimedia Comput Commun Appl 9(3):19:1–19:22

18. Oktavia V, Lee W-H (2004) A Fragile Watermarking Technique for Image Authentication Using Singular Value Decomposition. Advances in Multimedia Information Processing LNCS 3332, p 42–49

19. Qian Z, Feng G, Zhang X, Wang S (2011) Image self-embedding with high-quality restoration capability. Digital Signal Process 21(2):278–286

20. Singh D, Singh SK (2017) DCT based efficient fragile watermarking scheme for image authentication and restoration. Multimedia Tools and Applications 76(1):953–977

21. Telegram.org Available online: https://telegram.org/blog/stickers-meet-art-and-history (accessed on September 2018)

22. Telegram.org Available online: https://telegram.org/blog/moar-stickers (accessed on September 2018)

23. Tian J (2003) Reversible data embedding using a difference expansion. IEEE Transactions on Circuits and Systems for Video Technology 13(8):890–896

24. Wang R-Z, Lin C-F, Lin J-C (2001) Image hiding by optimal LSB substitution and genetic algorithm. Pattern recognition 34(3):671–683

25. Zhang X, Qian Z, Ren Y, Feng G (2011) Watermarking with flexible self-recovery quality based on compressive sensing and compositive reconstruction. IEEE Trans Inf Forensics Secur 6(4):1223–1232

26. Zhang X, Wang S, Qian Z, Feng G (2011) Self-embedding watermark with flexible restoration quality. Multimedia Tools and Applications 54(2):385–395

27. Zhang X, Wang S, Qian Z, Feng G (2011) Reference sharing mechanism for watermark self-embedding. IEEE Trans Image Process 20(2):485–495

28. Zhang XP, Xiao YY, Zhao ZM (2015) Self-embedding fragile watermarking based on DCT and fast fractal coding. Multimedia Tools and Applications 74(15):5767–5786