FROM IMAGE TO AUDIO WATERMARKING USING SELF-INVERTING PERMUTATIONS

Maria Chroni, Angelos Fylakis, and Stavros D. Nikolopoulos
Department of Computer Science and Engineering, University of Ioannina, GR-45110 Ioannina, Greece
{*mchroni, afylakis, stavros*}*@cs.uoi.gr*

Keywords: Watermarking Techniques; Audio Watermarking Algorithms; Self-inverting Permutations; Representations of Permutations; Frequency Domain; Embedding/Extracting Algorithms; Performance Evaluation.

Abstract: The intellectual property infringement in music due to the proliferation of the internet and the ease of creating and distributing identical digital objects has brought watermarking techniques to the forefront of digital rights protection. Towards this direction, a significant number of watermarking techniques have been proposed in recent years in order to create robust and imperceptible audio watermarks. In this work we propose an audio watermarking technique which efficiently and secretly embeds information, or equivalently watermarks, into an audio digital signal. Our technique is based on the main idea of a recently proposed image watermarking technique expanding thus the digital objects that can be efficiently watermarked through the use of self-inverting permutations. More precisely, our audio watermarking technique uses the 1D representation of self-inverting permutations and utilizes marking at specific areas thanks to partial modifications of the audio's Discrete Fourier Transform (DFT); these modifications are made on the magnitude of specific frequency bands. We have evaluated the embedding and extracting algorithms by testing them on various and different in characteristics audio signals that were in WAV format and we have obtained positive results. The algorithms have been developed and tested using the mathematical software package Matlab.

1 INTRODUCTION

Digital watermarking is a technique for protecting the intellectual property of a digital object; the idea is simple: a unique marker or identifier, which is called *watermark*, is embedded into a digital object which may be used to verify its authenticity or the identity of its owners (Grover, 1997; Collberg and Nagra, 2010).

Audio Watermarking. In a copyright protection framework, an audio watermarking technique aims to embed a unique identifier, i.e., the watermark w, into audio's data through mainly the introduction of errors not detectable by human perception. Within the same framework, audio watermarking can be described as the problem of embedding a watermark w in the host signal S producing thus the watermarked audio signal S_w such that w can be reliably located and extracted from S_w even after S_w has been subjected to transformations such as compression, filtering, noise addition, cropping, etc. It is worth noting that, if a watermarked audio signal S_w is copied or transferred through the internet then the watermark w is also carried with the copy into the audio's new location ensuring thus the maintenance of copyright protection.

Recently, a significant number of watermarking techniques have been proposed in the literature in order to create robust and imperceptible audio watermarks. Initial research on audio watermarking dates back to the mid-nineties where Bender et al. (Bender et al., 1996) presented data hiding techniques for audio signals; the first techniques were directly inspired from previous research on image watermarking. A broad range of audio watermarking techniques goes from simple least significant bit (LSB) scheme to the various spread spectrum methods and can be classified according to the domain where the watermarking takes place in frequency, time, and compressed domain (Sharma et al., 2012; Cox et al., 2008; Alsalami and Al-Akaidi, 2003; Hartung and Kutter, 1999).

Motivation. Nowadays, digital audio is a representative sample of internet data that has been subjected to extensive intellectual property violation. Thus, we consider important the development of methods that deter malicious users from claiming others' ownership, motivating thus internet users to feel more safe to publish their work online.

Audio watermarking, in contrast with other techniques, allows audio signals to be available to third

internet users but simultaneously carry an "id" that is actually the ownership's proof. This way audio watermarking achieves its target of deterring copy and usage without owner's permission.

Watermarking digital objects such as image, audio, video, text and software enables the proof of ownership on copyrighted objects preventing thus the intellectual property infringement.

Contribution. In this work we present an efficient and easily implemented technique for watermarking audio signals. What is important of the proposed technique is the fact that it suggests a way in which an integer number w can be represented first as a self-inverting permutation π^* and then as an one-dimensional array (or, equivalently, 1D representation). The idea comes from our previous work on image watermarking where the integer watermark number w is represented as a two dimensional array.

More precisely, our proposed algorithm embeds a self-inverting permutation π^* over n elements into an audio signal S by first mapping the elements of π^* into an $n \times n$ matrix A^* and then, based on the information stored in A^*, marking specific areas of audio S in the frequency domain resulting thus the watermarked audio S_w. An efficient algorithm extracts the embedded self-inverting permutation π^* from the watermarked audio S_w by locating the positions of the marks in S_w; it enables us to reconstruct the 1D representation of π^* and, then, obtain the watermark w.

At this point we would like to point out that the primary purpose of the paper is not to fill a gap of the existing audio watermarking methods by proposing a new embedding technique, but to expand the idea used on our previous work and show that it can be efficiently applied for audio watermarking depicting thus the high versatility of the whole concept.

Evaluation. We have evaluated the embedding and extracting algorithms by testing them on various and different in characteristics audio signals that were in WAV format and we had positive results as the watermark was successfully extracted. What is more, the method is open to extensions as the same method might be used with a different marking procedure. Note that, all the algorithms have been developed and tested in MATLAB (Ingle and Proakis, 2010).

2 OUR WATERMARKING TOOLS

In this section we present the structural and algorithmic tools we use towards the watermarking of an audio signal. We first briefly discuss a codec system which encodes an integer number w into a self-inverting permutation π, and then we present a transformation of a self-inverting permutation into 2D and 1D representations.

2.1 Self-inverting Permutations

In a formal (i.e., mathematical) way, a permutation of a set of objects S is defined as a bijection from S to itself, that is, a map $S \to S$ for which every element of S occurs exactly once as image value.

Permutations may be represented in many ways, where the most straightforward is simply a rearrangement of the elements of the set $N_n = \{1,2,\ldots,n\}$; for example, the permutation $\pi = (4,7,6,1,5,3,2)$ is a rearrangement of the elements of the set N_7 (Sedgewick and Flajolet, 1996; Golumbic, 1980).

Definition 2.1.1. Let $\pi = (\pi_1,\pi_2,\ldots,\pi_n)$ be a permutation over the set N_n, where $n > 1$. The inverse of the permutation π is the permutation $q = (q_1,q_2,\ldots,q_n)$ with $q_{\pi_i} = \pi_{q_i} = i$. A *self-inverting permutation* (or, for short, SiP) is a permutation that is its own inverse, that is $\pi_{\pi_i} = i$.

There are several systems that correspond integer numbers into permutations (Sedgewick and Flajolet, 1996). Recently, we have proposed algorithms for such a system which efficiently encode an integer w into a self-inverting permutation π and efficiently decode it; our algorithms run in $O(n)$ time, where n is the length of the binary representation of w.

2.2 2D and 1D Representations

In the 2D representation, the elements of the permutation $\pi = (\pi_1,\pi_2,\ldots,\pi_n)$ are mapped in specific cells of an $n \times n$ matrix A as follows:

$$\text{number } \pi_i \longrightarrow \text{entry } A(\pi_{\pi_i}^{-1},\pi_i)$$

or, equivalently, the cell at row i and column π_i is labeled by the number π_i, for each $i = 1,2,\ldots,n$.

Figure 1(a) shows the 2D representation of the self-inverting permutation $\pi = (4,7,6,1,5,3,2)$.

Based on the previously defined 2D representation of a permutation π, we next propose a two-dimensional marked representation (2DM representation) of π which is an efficient tool for watermarking images. In our 2DM representation, a permutation π over the set N_n is represented by an $n \times n$ matrix A^* as follows:

- the cell at row i and column π_i is marked by a specific symbol, for each $i = 1,2,\ldots,n$;

where, in our implementation, the used symbol is the asterisk, i.e., the character "*". Figure 1(b)

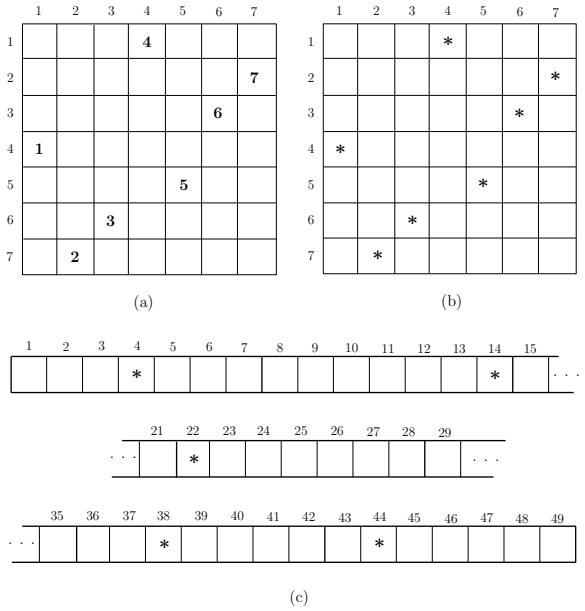

Figure 1: The 2D, 2DM and 1DM representations of the self-inverting permutation $\pi = (4,7,6,1,5,3,2)$.

shows the 2DM representation of the permutation $\pi = (4,7,6,1,5,3,2)$.

In our 1D representation, the elements of the permutation π are mapped in specific cells of an array B of size n^2 as follows:

$$\text{number } \pi_i \longrightarrow \text{entry } B((\pi_{\pi_i}^{-1} - 1)n + \pi_i)$$

or, equivalently, the cell at the position $(i-1)n+\pi_i$ is labeled by the number π_i, for each $i = 1,2,\ldots,n$.

We next describe the 1DM representation acquired in a similar manner as the 2DM representation. In our 1DM representation, a permutation π over the set N_n is represented by an n^2 array B^* as follows:

- the cell at the position $(i-1)n+\pi_i$ is marked by a specific symbol, for each $i = 1,2,\ldots,n$;

where, in our implementation, the used symbol is again the asterisk character "*". Figure 1(c) shows the 1DM representation of the same permutation $\pi = (4,7,6,1,5,3,2)$.

Hereafter, we shall denote by π^* a self-inverting permutation and by n^* the number of elements of π^*.

3 PREVIOUS RESULTS ON IMAGE WATERMARKING

In a recent work of ours, we have proposed an image watermarking technique that embeds watermarks into digital images by interfering in the frequency domain of images. Since our audio watermarking technique, that is going to be later described, is mainly based on the idea of image watermarking, we next briefly describe the main steps of our image watermarking technique and state points regarding some of its main characteristics.

The embedding image watermarking algorithm first computes the 2DM representation of the permutation π^*, that is, the $n^* \times n^*$ array A^* (see, Subsection 2.2). Next, it takes the input image I, covers it with an $n^* \times n^*$ imaginary grid C, resulting in $n^* \times n^*$ grid-cells C_{ij}, and takes the Discrete Fourier Transform (DFT) F_{ij} of each C_{ij}. The algorithm goes to each grid-cell C_{ij}, takes the magnitude M_{ij}, and places on it two imaginary ellipsoidal annuli denoted as "Red" and "Blue". It then computes the average of the magnitude values grouped by the "Red" and the "Blue" annuli, say, $AvgR_{ij}$ and $AvgB_{ij}$, respectively, and after that, for each M_{ij} computes the value $D_{ij} = |AvgB_{ij} - AvgR_{ij}|$ if $AvgB_{ij} < AvgR_{ij}$, otherwise $D_{ij} = 0$. Subsequently, the algorithm computes for each row i the maximum value $MaxD_i$. Once again the embedding algorithm goes to each grid cell C_{ij} and if $A_{ij} =$ "$*$" it increases the values of M_{ij} grouped by the "Red" annulus by $AvgB_{ij} - AvgR_{ij} + MaxD_i + c_{opt}$. Finally, it reconstructs each DFT cell F_{ij} using the modified M_{ij} with the trigonometric formula and with the inverse DFT it reconstructs the grid cells C_{ij}. The extracting algorithm works in a similar manner.

Regarding the main characteristics of this technique, we should first mention that it is efficient. As the experimental results showed, watermarks are imperceptible leading also to high fidelity. Moreover, watermarks are robust to distortions as we got positive results testing the watermarked images against JPEG compression and other attacks.

4 THE AUDIO WATERMARKING TECHNIQUE

In this section we present an algorithm for encoding a self-inverting permutation π^* into an audio signal S by marking specific time segments of S in the frequency domain resulting thus the watermarked audio signal S_w. We also present a decoding algorithm which extracts the embedded permutation π^* from S_w by locating the positions of the marks in S_w.

4.1 Embed Watermark into Audio

The embedding algorithm of our proposed technique encodes a self-inverting permutation (SiP) π^* into a

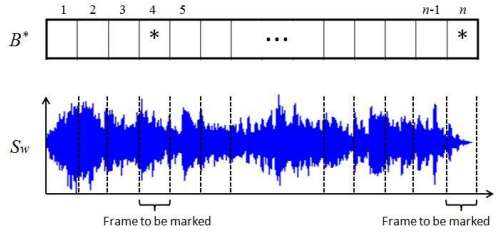

Figure 2: Segmentation of the S's signal into specific frames according to 1DM representation of the permutation π^*.

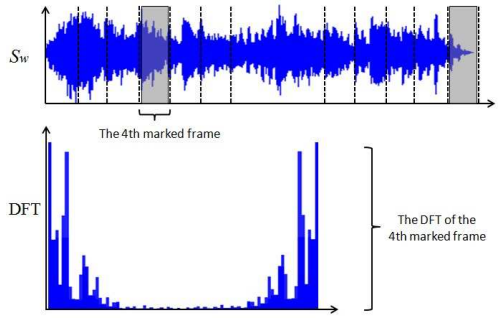

Figure 3: The DFT representation of a marked frame.

digital audio signal S. Recall that, the permutation π^* is obtained over the set N_{n^*}, where $n^* = 2n+1$ and n is the length of the binary representation of an integer w which actually is the audio's watermark (author's technique).

The main idea of embedding. The watermark w, or equivalently the corresponding self-inverting permutation π^*, is imperceptibly inserted in the frequency domain of specific frames on the audio track signals S; see, Figure 2. More precisely, we mark certain frames getting the DFT and do alterations at the magnitude values of high frequencies for each audio frame to be marked; see, Figure 3. This is achieved by choosing two groups of magnitude values specified with two segments of the magnitude vector namely "Red" and "Blue" and the alterations are actually on their difference; see, Figure 4. In our implementation we use fixed segments' widths and distances from the center of symmetry of the DFT's magnitude vector. The added value is specified by the maximum value in the defined area.

The embedding algorithm. Our embedding algorithm takes as input a SiP π^* and an audio signal S and returns the watermarked audio signal S_w; it performs the following main processes:

i. construct the 1DM representation of the watermark number w;

ii. transform the input audio signal S and acquire the frequency representation of it;

iii. modify signals' frequency representation according to the 1DM representation of the signal S;

iv. returns the watermarked audio signal S_w;

We describe below in detail the embedding algorithm in steps.

Algorithm Embed_SiP-to-Audio
Input: the watermark $\pi^* \equiv w$ and the original audio signal S;
Output: the watermarked audio signal S_w;

Step 1: Compute first the 1DM representation of the permutation π^*, i.e., construct the array B^* of size $n = n^* \times n^*$; recall that the entry $B^*((i-1)n^* + \pi_i^*)$ contains the symbol "*", $1 \le i \le n^*$.

Step 2: Segment the audio signal S into n non-overlapping frames f_i of size $f_i[a,b] = \lfloor \frac{N-1}{n} \rfloor$, $1 \le i \le n$, where N is the length of the audio signal.

Step 3: For each frame f_i, compute the Discrete Fourier Transform (DFT) using the Fast Fourier Transform (FFT) algorithm, resulting in n DFT frames F_i of size $F_i[a,b] = \lfloor \frac{N-1}{n} \rfloor$, $1 \le i \le n$, that is, $F_i = \mathrm{FFT}(f_i)$.

Step 4: For each DFT frame F_i, compute its magnitude M_i and phase P_i vectors (or, arrays) which are both of size $M_i[a,b] = P_i[a,b] = \lfloor \frac{N-1}{n} \rfloor$, $1 \le i \le n$.

Step 5: Then, the algorithm takes each of the n magnitude vectors M_i and determines two segments in M_i, $1 \le i \le n$, denoted as "Red" and "Blue" (see, Figure 4). In our implementation,

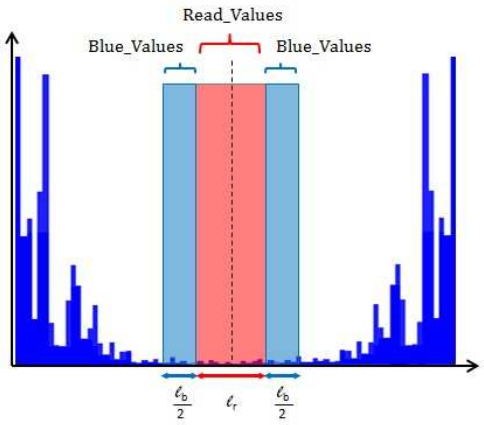

Figure 4: The "Red" and "Blue" segments on DFT.

- each "Red" segment $[x_r, y_r]$ has length ℓ_r (even), where $x_r = \lfloor \frac{N-1}{2n} \rfloor - \frac{\ell_r}{2}$ and $y_r = \lfloor \frac{N-1}{2n} \rfloor + \frac{\ell_r}{2}$;
- each "Blue" segment $[x_b, y_b]$ has length ℓ_b (even), where $x_b = x_r - \frac{\ell_b}{2}$ and $y_b = y_r + \frac{\ell_b}{2}$

The "Red" and the "Blue" segments determine two groups of magnitude values on M_i; the Red_Values and the Blue_Values (see, Figure 4).

Step 6: For each magnitude vector M_i, $1 \leq i \leq n$, compute the average value $AvgR_i$ of the Red_Values and the average value $AvgB_i$ of the Blue_Values of M_i.

Step 7: For each magnitude vector M_i, $1 \leq i \leq n$, compute first the variable D_i as follows:

- $D_i = |AvgB_i - AvgR_i|$, if $AvgB_i \geq AvgR_i$
- $D_i = 0$, otherwise.

Step 8: Partition the n values D_1, D_2, ..., D_n into n^* sets $E_1, E_2, \ldots, E_{n^*}$, each of size n^* (recall that $n = n^* \times n^*$); let $\{D_{i1}, D_{i2}, \ldots, D_{in^*}\}$ be the elements of the i-th set E_i, $1 \leq i \leq n^*$. Then, compute the values

- $MaxD_1, MaxD_2, \ldots, MaxD_{n^*}$

where $MaxD_i$ is the maximum value of the i-th set $E_i = \{D_{i1}, D_{i2}, \ldots, D_{in^*}\}$, $1 \leq i \leq n^*$.

Step 9: For each marked cell $B^*(i)$ of the 1DM representation matrix B^* of the permutation π^* (i.e., the call which contains the symbol "*"), mark the corresponding frame F_i, $1 \leq i \leq n$; the marking is performed by increasing all the Red_Values in M_i by the value

$$AvgB_i - AvgR_i + MaxD_k + c, \qquad (1)$$

where $k = \lceil \frac{i}{n^*} \rceil$ and $c = c_{opt}$. The additive value of c_{opt} is a predefined value which enables successful extracting.

Step 10: Reconstruct the DFT of the corresponding modified magnitude vector M_i, using the trigonometric form formula (Gonzalez and Woods, 2007), and then perform the Inverse Fast Fourier Transform (IFFT) for each frame F_i, $1 \leq i \leq n$, in order to obtain the audio signal S_w.

Step 11: Return the watermarked audio signal S_w.

Note that concerning the placement of the "Red" and "Blue" segments, their position can vary according to the frequency band in which we want to mark a frame. At the above illustration we mark it in the high frequencies but there can be a different approach. Specifically, we can mark instead lower frequencies and that is performed by moving the segments from the center to the right and left edges of the magnitude array of the Discrete Fourier Transform (DFT) representation.

4.2 Extract Watermark from Audio

In this section we describe the decoding algorithm of our proposed technique. The algorithm extracts the SiP π^* from a watermarked digital audio signal S_w, which can be later represented as an integer w.

The main idea of extracting. The main idea behind the extracting algorithm is that the self-inverting permutation π^* is obtained from the frequency domain of specific frames of the watermarked audio signal S_w. More precisely, using the same two "Red" and "Blue" segments, we detect certain areas of the watermarked audio signal S_w so that the difference between the average values of the "Red" segment have the maximum positive difference over the average values of the "Blue" segments. In this way we can detect marked frames that enable us to obtain the 1DM representation of the permutation π^*.

The extracting algorithm. We next describe the extracting algorithm which consists of the following steps.

Algorithm Extract_SiP-from-Audio
Input: the watermarked audio S_w marked with π^*;
Output: the watermark $\pi^* = w$;

Step 1: Take the input watermarked audio S_w and compute its size N. Then, segment S_w into n non-overlapping frames f_i of size $f_i[a,b] = \lfloor \frac{N-1}{n} \rfloor$, $1 \leq i \leq n$.

Step 2: Then, using the Fast Fourier Transform (FFT), get the Discrete Fourier Transform (DFT) for each frame f_i, resulting in n DFT frames F_i, $1 \leq i \leq n$.

Step 3: For each DFT frame F_i, compute its magnitude M_i and phase P_i vectors, which are both of size $M_i[a,b]$=$P_i[a,b]$=$\lfloor \frac{N-1}{n} \rfloor$, $1 \leq i \leq n$.

Step 4: For each magnitude vector M_i, compute the average values $AvgR_i$ and $AvgB_i$ of the Red_Values and Blue_Values of M_i, respectively, as described in the embedding algorithm.

Step 5: Partition the n vectors M_i, $1 \leq i \leq n$, into n^* sets $L_1, L_2, \ldots, L_{n^*}$, each of size n^*; let $\{M_{i1}, M_{i2}, \ldots, M_{in^*}\}$ be the elements of the i-th set L_i and let $AvgR_{ij}$ and $AvgB_{ij}$ be the average values of the Red_Values and Blue_Values, respectively, of the vector M_{ij}, $1 \leq i, j \leq n^*$.

Step 6: For each set $L_i = \{M_{i1}, M_{i2}, \ldots, M_{in^*}\}$ find the k_{th} vector M_{ij} such that $AvgB_{ik} - AvgR_{ik}$ is minimum and set $\pi_i^* = k$, $1 \leq k \leq n^*$.

Step 7: Return the self-inverting permutation π^*.

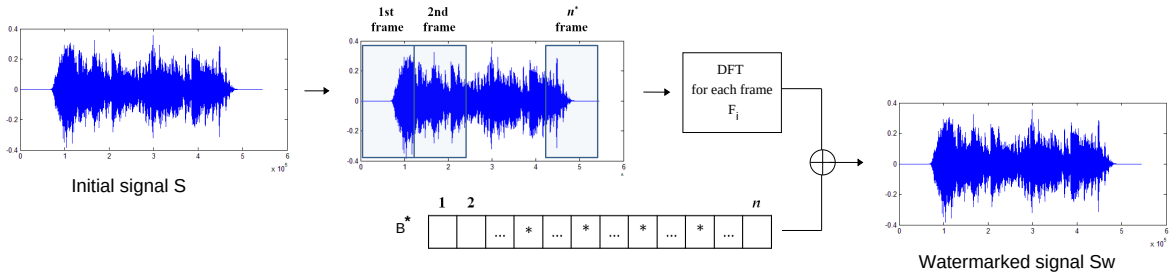

Figure 5: The encoding process of audio signal watermarking.

Having presented the embedding and extracting algorithms, we next briefly comment on the purpose of the additive value $c = c_{opt}$ (see, Step 9 of the embedding algorithm). Similar to image watermarking, we add at the corresponding embedding marking step the additive value c_{opt} which by getting greater increases the robustness of the marks; in our audio watermarking case, we just used a very small value for it.

5 EXPERIMENTAL RESULTS

This section summarizes the experimental results of the proposed audio watermarking codec algorithms; we implemented our algorithms and carried out tests using the general-purpose mathematical software package Matlab (Version 7.7.0) (Ingle and Proakis, 2010).

Testing of our embedding and extracting algorithms has been made by the use of various 16-bit digital audio tracks in *wav* format with 44.1 KHz sampling frequency. Concerning the audio samples used, they where relatively short abstracts with different characteristics. For instance there were tracks containing speech which have many silent segments as well as music track samples and tracks with extreme features such as low and high frequency sounds. Many of the audio tracks that we used for testing were acquired from a web audio repository called wavsource and enriched by some other audio tracks from various sources.

It is well known in the field of watermarking that there are three main characteristics to take into account describing and evaluating a digital watermarking system: *Fidelity*, *Robustness*, and *Capacity* (Cox et al., 2008).

Concerning our watermarking system, it seems to be of high fidelity as watermarked tracks were not distinguished over the original ones and the results using the PSNR metric were interestingly positive.

Concerting the marking procedure of our implementation, we set both lengths ℓ_r and ℓ_b of the "Red" and "Blue" segments respectively, equal to 20% of half the length of magnitude vector as it is mirrored (see, Section 4.1). Recall that, the value 20% is a relatively small percentage which allows us to modify the audio track segments in a satisfactory level in order to detect the watermark and successfully extract it without affecting audio tracks' initial quality. Moreover, we choose to alter higher frequencies and thus the two segments are at the center of the magnitude vector. This is because high frequencies are less perceptible according to the human auditory system. What is more, at high frequencies audio tracks contain less information which means that information is less likely to be lost due to post alterations.

Fidelity. In order to evaluate the watermarked audio track quality obtained from our proposed watermarking method we used the Peak Signal to Noise Ratio (PSNR) metric. Our aim was to prove that the watermarked audio track is closely related to the original track proving the high fidelity attribute of our system. This is something vital as watermarking should not introduce audible distortions in the original audio track, as that would certainly reduce its commercial value.

Giving a short introduction to the PSNR metric, we should mention that it is defined as the ratio between the reference (or, original) signal and the distorted (or, watermarked) signal of an audio track and it is given in decibels (dB). It is well known that PSNR is most commonly used as a measure of quality of reconstruction of lossy compression codecs (e.g., for image or audio compression methods). The higher the PSNR value the closer the distorted signal is to

Filename	Gaussian Noise	Cropping	Resampling	Requantization	MP3 Compression
bach.wav	0	1	0	0	3.2
clarinet.wav	0	1	0	0	2.4
castanets.wav	0	1	0	0	5.2
elvis_riverside.wav	0	1	0	0	2.8
family_man.wav	0	1	0	0	0.2
high10sec.wav	0	1	0	0	3.0
low10sec.wav	0	1	0	0	2.0

Table 1: The hamming distance of the watermark w and the extracted watermark w^* after common signal attacks.

the original or the better the watermark conceals. We mentions that PSNR is a popular metric due to its simplicity.

For an initial audio signal S of size N and its watermarked equivalent signal S_w, PSNR is defined by the formula:

$$\text{PSNR}(S,S_w) = 10\log_{10}\frac{N_{max}^2}{MSE}, \qquad (2)$$

where N_{max} is the maximum signal value that exists in the original audio track and MSE is the Mean Square Error which is given by the following formula:

$$\text{MSE}(S,S_w) = \frac{1}{N}\sum_{i=0}^{N-1}(S(i)-S_w(i))^2. \qquad (3)$$

Comparing the original audio tracks with the watermarked ones, we immediately get to notice that they depict excellent fidelity according to the PSNR values that we have obtained. In every case PSNR is over 50 dB which proves that fact that there is a striking similarity between the original and the watermarked signal of an audio track.

Filename	PSNR
bach.wav	67.2
clarinet.wav	67.9
castanets.wav	68.2
elvis_riverside.wav	75.3
family_man.wav	73.8
high10sec.wav	58.8
low10sec.wav	64.5

Table 2: The PSNR values of the watermarked audio signals.

In Table 2 you can see the performance of our method as we demonstrate the PSNR values of some audio tracks that we used in this work. Each of them was sampled at 41.1 KHz and the duration was of about 10 sec. Additionally, each one has much different characteristics. More specifically, the audio tracks

- bach.wav, clarinet.wav and castanets.wav

where a concert, a clarinet and castanets solo respectively. The audio track

- elvis_riverside.wav

combines human voice with music, while the

- family_man.wav

contains only speech which means that it also has periods of silence. Lastly the audio tracks

- high10sec.wav and low10sec.wav

are some extreme cases of high and low frequency sounds.

Robustness. The watermarked signals were subjected to distortions or common signal attacks in order to evaluate the robustness of our audio watermarking algorithms. We tested the performance of each audio track under white noise addition, cropping, resampling, requantization and MP3 compression. Below we describe in more details each one of the five different attacks that we applied in our experiments.

(a) Gaussian Noise. A white gaussian noise of SNR 20 dB was added to the original audio signal.

(b) Cropping. A 10% of the beginning of the watermarked audio signal was cropped and subsequently replaced by zeros.

(c) Resampling. The watermarked signal, originally sampled at 44.1 KHz, is resampled at 22.05 KHz, and then restored back by sampling again at 44.1 KHz

(d) Requantization. The 24-bit watermarked audio signal is re-quantized down to 16 bits/sample and then back to 24 bits/sample.

(e) MP3 compression. The watermarked audio signal is compressed using a bit rate of 128 Kb/s and then decompressed back to the WAV format.

Since the watermark that we embed in our audio signal is a permutation, i.e. a vector over the set N_n ($n > 1$), we test after each attack the similarity of the

extracted watermark with the original one using the Hamming distance (Hamming, 1950).

The Hamming distance $d(x,y)$ between two vectors x and y is the number of coefficients in which they differ (Hamming, 1950). The Hamming distance equals to zero, i.e., $d(x,y) = 0$, if x and y agree in all coordinates; it happens if and only if $x = y$. In our case the Hamming distance is computed between the watermark $w = \pi^*$ that we embedded into the audio track and the watermark π^*_{ext} that we extract from the audio. If $d(\pi^*, \pi^*_{ext}) = 0$ the watermark $w = \pi^*$ successfully extracts from the attacked audio signal. Additionally, it is worth noting that if $d(\pi^*, \pi^*_{ext})$ is relatively small, then the watermark π^* can be reconstructed with high probability by exploiting the self-inverting properties of the permutation π^*.

In Table 1 we demonstrate similarity results between the watermark that we embedded into the audio track and the watermark that we extracted after various signal processing attacks. As the experimental results show, our audio watermarking algorithm is robust against additive gaussian noise of SNR 20 dB, cropping, resampling and requantization. Evaluating our method's robustness over lossy compression we tested it using the MP3 encoding format with a bit rate of 128 Kb/s. In order to optimize the results as high frequency information is mostly lost using MP3 we made the appropriate adjustments concerning the width of the segments to be marked as well as the additive value $c = c_{opt}$ (see, Algorithm Embed_SiP-to-Audio). For most cases the results were positive as despite not being able in every case to successfully extract all the elements of the watermark, using the properties of self-inverting permutations recovery of the initial watermark can be successfully operated.

Closing the robustness evaluation of our method, we should point out that a drawback of our method is actually when we want to watermark an audio track with extreme high frequencies; it is something that could be encountered on future work.

Capacity. The capacity of our audio watermark method has been computed by measuring the percentage of the watermarked parts of an audio track over the length of the entire audio track. Our method partitions the audio track into $n^* \times n^*$ frames, where n^* is the length of the permutation π^*, and marks only one frame of a set of n^* frames; recall that our embedding method groups the frames into n^* sets each containing n^* frames (see, Algorithm Embed_SiP-to-Audio). That means, a total n^* over $n^* \times n^*$ frames are marked, so the ratio of the watermarked part over the entire length of the audio track is $\frac{n^*}{(n^* \times n^*)}$. Thus, our audio watermarking method has $\frac{1}{n^*}$ capacity.

6 CONCLUDING REMARKS

In this paper we presented an audio watermarking technique which efficiently and invisibly embeds information, i.e., watermarks, into an audio digital signal. Our technique is based on the same main idea of a recently proposed image watermarking technique expanding thus the digital objects that can be efficiently watermarked through the use of self-inverting permutations.

We experimentally tested our embedding and extracting algorithms on WAV audio signals. Our testing procedure includes the phases of embedding a numerical watermark $w = \pi^*$ into several audio signals S, storing the watermarked audio S_w in WAV format, and extracting the watermark $w = \pi^*$ from the audio S_w. We obtained positive results as the watermarks were invisible, they didn't affect the audio's quality and they were extractable.

The performance evaluation of our audio watermarking technique on several other attacks remains a problem for further investigation.

REFERENCES

Alsalami, M. A. and Al-Akaidi, M. M. (2003). Digital audio watermarking: survey. *De Montfort University*, pages 1–14.

Bender, W., Gruhl, D., and Morimoto, N. (1996). Techniques for data hiding. In *Proc. IBM systems journal*, volume 35(3.4), pages 313–336.

Collberg, C. and Nagra, J. (2010). *Surreptitious Software*. Addison-Wesley.

Cox, I. J., Miller, M. L., Bloom, J. A., Fridrich, J., and Kalker, T. (2008). *Digital Watermarking and Steganography*. Morgan Kaufmann, 2nd edition.

Golumbic, M. (1980). *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, Inc., New York.

Gonzalez, R. C. and Woods, R. E. (2007). *Digital Image Processing*. Prentice-Hall, 3rd edition.

Grover, D. (1997). *The Protection of Computer Software - Its Technology and Applications*. Cambridge University Press, New York.

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160.

Hartung, F. and Kutter, M. (1999). Multimedia watermarking techniques. In *Proceedings of the IEEE*, volume 87(70), pages 1079–1107.

Ingle, V. K. and Proakis, J. G. (2010). *Digital Signal Processing using Matlab*. Cengage Learning, 3rd edition.

Sedgewick, R. and Flajolet, P. (1996). *An Introduction to the Analysis of Algorithms*. Addison-Wesley.

Sharma, S., Rajpurohit, J., and Dhankar, S. (2012). Survey on different level of audio watermarking techniques. *Int'l Journal of Comput. Applications*, 49(10):41 – 48.