International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:8, 2008

A Watermarking Scheme for MP3 Audio Files

Dimitrios Koukopoulos, and Yiannis Stamatiou

***Abstract*—**In this work, we present for the first time in our perception an efficient digital watermarking scheme for mpeg audio layer 3 files that operates directly in the compressed data domain, while manipulating the time and subband/channel domain. In addition, it does not need the original signal to detect the watermark. Our scheme was implemented taking special care for the efficient usage of the two limited resources of computer systems: time and space. It offers to the industrial user the capability of watermark embedding and detection in time immediately comparable to the real music time of the original audio file that depends on the mpeg compression, while the end user/audience does not face any artifacts or delays hearing the watermarked audio file. Furthermore, it overcomes the disadvantage of algorithms operating in the PCM-Data domain to be vulnerable to compression/recompression attacks, as it places the watermark in the scale factors domain and not in the digitized sound audio data. The strength of our scheme, that allows it to be used with success in both authentication and copyright protection, relies on the fact that it gives to the users the enhanced capability their ownership of the audio file not to be accomplished simply by detecting the bit pattern that comprises the watermark itself, but by showing that the legal owner knows a hard to compute property of the watermark.

***Keywords*—**Audio watermarking, mpeg audio layer 3, hard instance generation, NP-completeness.

I. Introduction

NOWADAYS there is a necessity for the development of reliable and robust schemes for protecting audio data, especially of those sent over the Internet because they are vulnerable to unauthorised copying and manipulation by malicious users. As it has been mentioned by previous works [1]-[6], [15] there is a number of properties an audio watermarking algorithm should have in order to be efficient:

- *Inaudibility*, the watermark embedding should not be accompanied by loss of audio quality.
- *Statistical invisibility*, the algorithm should prevent unauthorised watermark detection/removal or alteration.
- *Similar compression characteristics with the original signal*.
- *Robustness*, the algorithm should be robust against various attacks for malicious users.
- *Embedded directly in the data*.
- *Support multiple watermarks*.
- *Low redundancy*.
- *Self-clocking*.

The most proposed watermarking techniques in the bibliography embed watermarks to PCM-data (raw audio data) [1]-[4]. This ensures that these techniques will operate with all audio formats because it presupposes that the audio file will be uncompressed in order to embed the watermark and then recompress again. However, this approach creates two significant problems. Firstly, it is not sure that the watermark will survive the coding/decoding procedure and secondly it is time consuming, which is a luxury not available in real-time applications. In order to achieve real-time playability, audio data are commonly stored and transmitted in compressed format, such as mpeg audio. Thus, the watermarking schemes would be preferred to target the compressed data domain [5, 6, 14] when they are going to be used to real-time applications. There is the problem of watermark technique adaptability with this approach because these techniques are oriented to specific audio formats and they should be transformed in order to be able to operate with other audio formats.

An important issue in the context of audio watermarking is whether the original signal should be used for the detection of the watermark [2, 6] or not [3, 4, 5, 14]. It is desirable not to use the original signal for detection because it results in waste of big amount of storage space and it adds the danger of its usage by malicious users. Another issue that differentiates watermarking techniques is whether the watermark is embedded in the spatial or the frequency domain. Usually, when it is used the time domain [4, 5, 6, 14], time distortions are faced well, while it should be taken special care for spectral distortions. In the frequency domain [2, 3] the opposite happens. Some of the methods that use the frequency domain exploit the frequency characteristics of the audio signal in order to embed the watermark, minimising audible distortions even for high amplitude watermarks.

In this paper, a digital watermarking algorithm for mp3 audio files is presented that works in the compressed domain, makes its manipulations in the time and subband/channel domain and it does not need the original signal to make detection. This algorithm was implemented in a real-time environment taking special care for the efficient usage of the two limited resources of computer systems: time and space. The scheme was designed and implemented such that the delays to be decreased and the memory requirements to be as small as possible. Our algorithm overcomes the problem of the algorithms operating in the PCM-data domain that are vulnerable to compression/recompression attacks because it

Manuscript received July 8, 2005.

Dimitrios Koukopoulos is with Research and Academic Computer Technology Institute, 61 Riga Feraiou Str., 26110 Patras, Greece (corresponding author to provide phone: +30-26410-34473; fax: +30-26410-58075; e-mail: koukopou@ ceid.upatras.gr).

Yiannis Stamatiou, is with Research and Academic Computer Technology Institute, 61 Riga Feraiou Str., 26110 Patras, Greece (e-mail: stamatiu@cti.gr).

embeds the watermark in the scale factors domain and not in parts of the audio data that are masked or are not perceptible because of psycho-acoustic laws [1, 2, 4]. Also, it does not need to encode specially the audio data into an mp3 file as it is done by MP3Stego [7].

Our algorithm exploits the ideas that have been appeared in [5, 6, 14] extending them in mp3 format with two ways: (i) applying a control mechanism that monitors the changes of the scale factors of the source file in the phase of watermark embedding, (ii) giving to the industrial user the enhanced capability of applying a crypto mechanism at the watermark creation part of the algorithm. This mechanism creates a watermarking key with unique semantic meaning for each user and strong robustness having the capability of surviving attacks that achieve to damage a significant percentage of it.

It should be mentioned that the presented algorithm is one of the first algorithms in our perception that can be used for authentication and copyright protection without the use of the original signal. This is because of the capability that is given by this algorithm to the industrial user to use a crypto-mechanism to create the watermarking key, which, now, is not a simple text as in other papers [5]. Here the key is the instance of a graph that can survive against inversion attacks (a malicious user subtracts his watermark from the original marked by the true owner) because it is spread all over the scale factors range and it can still be "operational" even if a large part of it has been damaged. Of course if the user does not want to use a key with semantic meaning he can use a simple text preserving all the benefits that has in the case of the key with semantic meaning, but then the key will be vulnerable to inversion attacks.

II. Properties of the Embedded Watermark–From Syntax to Semantics

Any bit-sequence may be seen under two different views:

1) *Syntactically*, i.e. how it looks like as a sequence of 0's and 1's. Then the sequence's characteristics and properties are determined simply by the pattern of 0's and 1's.
2) *Semantically*, i.e. whether in fact, it represents *by design* another entity/object converted into the bit-sequence under the action of a suitable encoding. This time the sequence, in addition to its syntactic characteristics, may also be seen as possessing characteristics and properties *inherited* from the entity/object from which it resulted.

A watermark, now, is a sequence of bits of a certain length and it usually provides evidence of ownership of a digital file by simply being inserted in suitable places in it and later, being detected and displayed to a referee during a copyright dispute. The fact that enables the resolution of the copyright dispute in favor of the legal owner of the file is that this bit sequence could not have been there by accident but purposefully.

One problem with many watermarking schemes that use this approach is that they usually cannot sustain attacks that destroy even small parts of the watermark. For example, in many audio watermarking algorithms, multiple copies of the watermark are embedded in the file to be protected and the detection is successful when a fraction of them are found in the file. However, if even a single bit of each of the embedded watermarks is destroyed after an attack (e.g. lossy compression/decompression), successful detection is impossible. Another problem is that if someone steals the watermark bit-sequence or manages to uncover it from a file after obtaining knowledge of the embedding mechanism; he/she may use it in order to prove ownership of a file personifying the legal originator of the file. The two aforementioned problems stem from the fact that the watermarks used for watermarking files are regarded as bit-sequences possessing a specific pattern (syntactic view). In order to solve these problems we suggest to view watermarks semantically and our proposal is to create and use watermarks that (i) represent a certain combinatorial object with a *specific* characteristic/property known only to his/her creator and (ii) the characteristic/property known to the creator of the watermark *still* holds for *any* part of the watermark (see [13] for an approach in image watermarking using Zero-Knowledge proofs). In essence, we propose to rely only on the *knowledge of the property* of the combinatorial object encoded by the watermark and not on the *bit pattern*. In addition, so as to guard against stealing of the watermark, we will use characteristics/properties that are *hard* to discover or compute if not known in advance so that whoever attempts to use the watermark, when asked to prove that he/she has created it by stating the property, of the object encoded by the watermark, he/she will be at difficulty to demonstrate such a knowledge.

A. Producing Objects/Watermarks Possessing Hard to Discover Characteristics/Properties

One of the most intriguing problems in complexity theory concerns the establishment and the determination of the *satisfiability threshold* for random Boolean formulas with 3 literals per clause (3-SAT problem). According to many experimental observations, there appears to exist a value r_0 for clause to variable ratio $r = m/n$ such that *almost all* randomly generated formulas with $r > r_0$ are *unsatisfiable* while allmost all randomly generated formulas with ratio $r < r_0$ are *satisfiable*. In addition, the formulas generated with variable to clause ratio close to r_0 seem to be among the most difficult formulas to test for satisfiability using the best of algorithms available. The belief that such a threshold point exists (theoretically) forms the *satisfiability threshold conjecture* and it is widely believed to hold true.

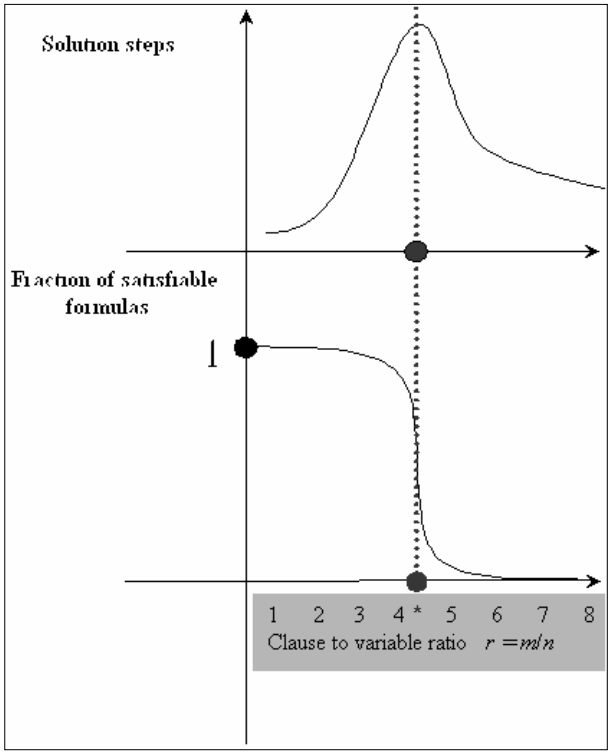

Fig. 1 Threshold phenomena in 3-SAT

In Fig. 1, we show two diagrams that depict schematically the conjecture (see, also, [12]). In the lower diagram, we show the probability that a randomly generated formula is true as a function of the clause to variable ratio r. Observe the sudden change of this probability from 1 to 0 as the ratio passes over the value 4.2. At the same time, in the upper diagram, we see that the number of solution steps taken by the algorithm (recursive calls of the Davies-Putnam procedure) suddenly increases at the value 4.2 and then falls again. Similar threshold phenomena can also be observed in connection with properties of other kinds of combinatorial problems besides the satisfiability problem. In 1991, Cheeseman, Kanefsky, and Taylor in [10] considered the problem of *graph colouring*. Before considering the threshold behaviour of this problem we will give some preliminaries. Given a graph G, the *3-colouring problem* asks us to discover a colouring of its vertices using at most 3 colours, so that no two adjacent colours receive the same colour. Such a colouring is called 3-colouring of G. The problem of *discovering* a 3-colouring of a *given* graph is computationally intractable or NP-complete like the 3-SAT problem we considered above (see [8, 9]). This fact means that there is strong evidence that no fast (polynomial) time algorithm exists that solves the problem. On the other hand, it is very easy to *create* a graph with n vertices that possesses a 3-colouring: given three non-zero real numbers p_1, p_2, and p_3 that sum up to 1 (for a further constraint, see below), each of the n vertices is assigned independently of the others to one of three colour classes C_1, C_2, and C_3 with probability p_1, p_2, and p_3 respectively. Then each pair of vertices that belong to different classes becomes adjacent independently of the other pairs with probability p. We now have an instance of the computationally intractable 3-colouring problem with a *known* to us 3-colouring and colour classes of (expected) sizes $|C_1| = p_1n$, $|C_2| = p_2n$, and $|C_3| = p_3n$. One algorithm that accomplishes this is the following:

Step 1: Let p_1, p_2, and p_3 be real numbers such that $p_1 + p_2 + p_3 = 1$ and p_1, p_2, and $p_3 > 0$.
Step 2: Generate a random permutation i_1, i_2, …, i_n of the numbers 1, 2, …, n (vertices of the graph).
Step 3: For each $j = 1$, …, n, vertex v_j is assigned to color class C_k with probability p_k, $k = 1, 2, 3$.
Step 4: For each pair u, v of vertices that do not belong to the same color class, introduce the undirected edge (u,v) with probability p.

Furthermore, due to the intractability of the problem, it would be difficult for someone else to discover quickly a 3-colouring of our graph and we, thus, can use the graph along with our knowledge of the colouring to prove ownership of a file into which we have inserted the adjacency matrix representation of the graph (an $n \times n$ matrix containing 1 at position (i,j) if vertices i and j are adjacent or 0 otherwise).

However, the fact that 3-colouring is NP-complete does not guarantee difficulty to solve of a specific instance. The determination of the i*nstance complexity* of computationally intractable problems is a major open issue in complexity theory and, to the best of our knowledge, no useful practical characterization of such instances is known as of today. It is a fact that if a problem is computationally intractable then there exists an infinite set of instances that can not be solved in polynomial time by any algorithm. The set of these instances form the *complexity core* of the problem and it would be desirable for us to pick instances that belong to this set. A possible way to sample this set while constructing our instance, is to use the theory of *threshold phenomena* of intractable combinatorial problems.

Let G be a randomly generated graph with m edges and n vertices and let $r = m/n$. Returning to the work of Cheeseman, Kanefsky, and Taylor in 1991, they reported a remarkable experimental observation concerning the 3-colorability property of such graphs. They discovered that for graphs generated with ratio r around the point 2.3, either almost all of them were 3-colourable ($r < 2.3$) or almost all of them were not 3-colourable ($r > 2.3$). Thus the value $r_0 = 2.3$ seems to mark a threshold through which *almost certain* colourability switches into *almost certain* non-colourability. However, the most important observation for our purposes, was that these graphs were the most difficult to handle using the most efficient search algorithms available. So an idea may be to modify the graph creation procedure we presented before by setting $p = (p_1p_2+p_1p_3+p_2p_3)r_0/n$ in order to achieve an *expected* ratio for the generated graphs equal to the threshold point $r_0 = 2.3$. And since partitions of vertices of about equal

size have been experimentally observed to make the colouring problem more difficult, we may set $p_i = 1/3$ and, thus, fix our p to be equal to $3r_0/n$. This choice of p samples (according to experimental observations in threshold phenomena) from the complexity core of the 3-colouring problem and, thus, to enable us to construct hard to solve 3-colouring instances to use as watermarks. Then exhibiting the 3-colouring to a referee proves ownership of both watermark and the file that contains it. The referee may, then, easily check that this is indeed a legal 3-colouring (using a Zero Knowledge Interactive Proof protocol). Someone who illegally claims ownership will be in a difficulty to demonstrate a 3-colouring of our graph. Moreover, if he/she tries to demonstrate another property not belonging to a class of hard to compute properties or if he/she simply claims that he has put a string that "happens" to resemble a graph, then his/her credibility will be lower than ours since we have demonstrated a 3-colouring of this string when seen as a graph which we could not possibly do unless we purposefully had constructed this string to be a graph possessing the colouring we demonstrated.

III. Main Algorithm

Our algorithm extends the one that was proposed for mpeg audio layer 2 files in [5]. The main differences are: (i) the application of a mechanism that produces unique crypto-keys that are robust in malicious attacks because they can be detected correctly even if a big percentage of them has been damaged, (ii) the use of a control mechanism that ensures that the embedding of the key in the mp3 audio file scale factors is done suitably (no audible distortions and preservation of scale factors bits number), and (iii) the specification of similar patterns with the one that is wanted to be embedded extremely fast as an one step procedure, improving extremely the time complexity of the algorithm.

As input the algorithm takes an mp3 audio file, a unique key that is produced with the use of the watermarking key creation algorithm or simple text and a set of three patterns that represent the binary digits "0" and "1" and a synchronization bit, which is used between the bits or between pre-specified group of bits and at the end of the embedding key for self-clocking and robustness against cropping.

The next step is the transformation of the binary string that represents the watermarking key into a sequence of the given patterns and the extraction of the scale factors from the frames of the mp3 audio file. The scale factors in mp3 files have variant number of bits that depends from 0 to 4 bits, while in mpeg audio layer 1 and 2 formats they consist of a fixed number of bits equal to 6. Because of that along with the scale factors another stream of information is extracted by an mp3 audio file, which specifies the number of bits used in the bit representation of each scale factor. Based on the scale factors, difference patterns are calculated. These patterns correspond to the difference between the first scale factor that can be considered as a starting point and the following scale factors in the scale factor stream. Assuming as an example the list of scale factors {6, 8, 5, 3}, the first scale factor is considered as a starting point and the pattern is {2, -1, -3}.

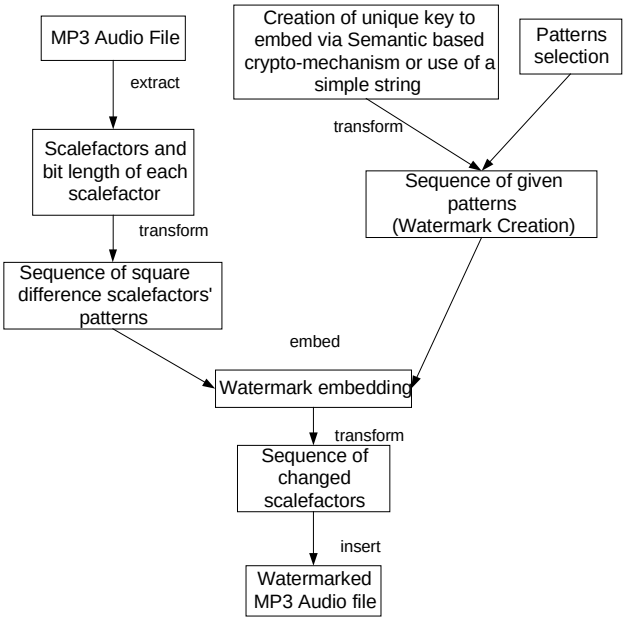

Fig. 2 Main algorithm

In the following step the sequence of patterns produced by the watermarking key is embedded into the sequence of difference patterns produced by the scale factors of the mp3 audio file. The embedding is achieved by a mechanism that changes the scale factors difference patterns until a sufficient number of them matches the desired sequence of patterns. The mechanism guarantees that it is not introduced an audible distortion in the watermarked audio file and that each modified pattern does not violate the number of bits the underlying scale factors consist of. The second goal of the applied mechanism is achieved by applying a control sub-mechanism that takes as input the stream that carries the information about the number of bits each scale factor consists of. Finally, the produced sequence of patterns from the previous step is transformed to a sequence of scale factors that are embedded in the source audio file, creating a watermarked mp3 audio file.

A. Embedding Mechanism

The embedding mechanism takes as input the pattern we want to embed, the unwanted patterns and the scale factors of frames where we have decided to embed our pattern. The embedding mechanism, also, uses as parameters the minimal number of patterns in the specified area of frames that must be equal to the pattern of the information bit we want to embed in order to consider that the wanted pattern has been embedded, the maximal tolerance that determines the changes that the patterns can suffer (as concerns the audibility) to match the wanted pattern and the stream of bit lengths of the scale factors, which patterns examined.

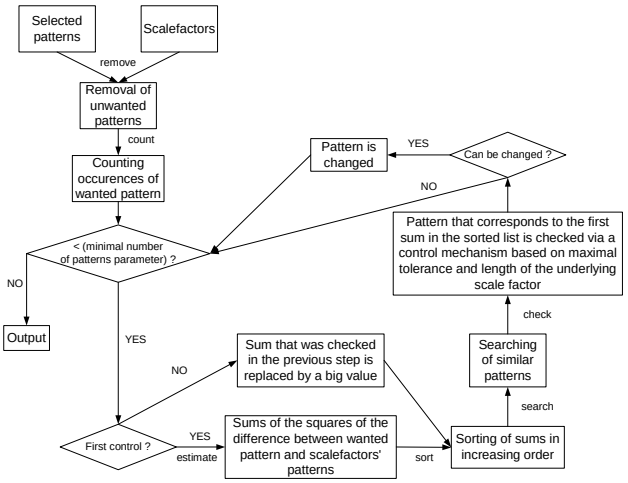

Fig. 3 Watermark embedding

The embedding mechanism does not consist of one loop based on the parameter of tolerance as in [5]. The first step is the investigation of the scale factors pattern stream for unwanted patterns and their removal by changing them slightly. The next step is the looking for patterns similar to the wanted one and their counting. If the wanted pattern appears more times than the parameter that specifies the minimal number of its appearances then the mechanism is satisfied and it extracts the pattern as a result. Otherwise, the mechanism tries to find similar patterns to the wanted one. The specification of the similar patterns is achieved using a control mechanism that uses as parameters the maximal tolerance concerning audibility and the number of bits each scale factor consists of. Firstly, the sums of squares of the difference of the patterns of all the given scale factors with the wanted pattern are estimated and sorted in increasing order. Then, the pattern, which sum is at the beginning of the sorted list, is selected as candidate for changing. This pattern is checked by the control mechanism to be specified if the tolerance parameter is satisfied and the bit size of the underlying scale factor is not violated. If the check is successful, it is replaced by the wanted pattern. Then, it is checked if the minimal number of patterns parameter is satisfied. If not, the corresponding sum in the sorted list is replaced by a big value, the list of sums is sorted again and the same procedure is repeated. This procedure can be considered as a one-step procedure comparing to the procedure in [5] where the searching for similar patterns consists of a loop that is based on the increase one by one of the tolerance parameter until the maximal tolerance is reached. Thus, most of the changes will affect patterns that differ as little as possible from the wanted pattern minimizing the distortion in audio quality, whereas the number of bits of the underlying scale factors is not violated.

We should always take in care the variant feature of the number of bits that are used for the scale factors because otherwise the watermarked audio file can be damaged due to overwriting different data from scale factors or overlapping between scale factors. This may affect the audibility of the audio file because it can lead to pattern changes that are not at the lowest levels of tolerance. However, in practice with suitable selection of the set of three patterns, there is only a slight deterioration of the quality of the watermarked audio file.

B. Detection Mechanism

The detection mechanism depends basically on the method we choose to embed the watermarking key and the crypto mechanism with the help of which the semantically unique crypto-key is created. Thus, if a fixed number of frames has been used for the embedding of each information bit, the given fixed region of each embedding pattern, which is given as input to the detection mechanism is searched and the presence of the three patterns is counted. The pattern with the most hits is selected. This method is sufficient when there is no trimming. If trimming exists then the synchronization bits should be used for the re-synchronization of the algorithm at the beginning of each watermark. Our algorithm handles this problem efficiently because the watermarking key is long enough to spread all over the scale factors.

If the number of frames/bit used for watermark embedding is not given, but we know the minimal number of equivalent consecutive patterns and the tolerance, then the detection procedure is more complicated because it demands the algorithm to search for the dominant of the three patterns in the frames with respect to the parameter of the minimal number of patterns. When the dominant pattern changes, then the previous dominant pattern is considered as a found one. In this method, the synchronization bit-pattern is used for the separation of information bit-patterns along with weighting of the frames and filtering of very short dominance phases in order our algorithm to be robust against noise and false detection. Also, the crypto key plays a big role towards the minimization of noise and false detection artifacts because it is embedded all over the scale factors and it is robust against significant damages of it. If the file containing the graph (adjacency matrix) that is used for the creation of the watermarking key is attacked, then we expect that at least half of the embedded (in the scale factors) adjacency matrix will be left intact (assuming that each of its bits is flipped with probability ½). But even then, we can still show a 3-colouring (subset of the 3-colouring of the original graph) surviving, in this way, the attack.

IV. Algorithmic Engineering

The use of algorithms in a real-time system requires several modifications in order their needs to adjust to the limited resources of a computer system. Two crucial factors that should be taken into account are time and space. The delays, from which the used algorithms suffer, should be decreased and the memory requirements should be as small as possible. In this section, we will present the engineering solutions we gave to these problems in order to make our algorithm efficient for real-time application. Also, we present several issues that improve the audio quality.

In order to achieve the first goal, delays reduction, we should search for the components of the algorithm that are time consuming and then to see if they can be improved or not. It can be noticed that the software module, which is responsible for scale factors extraction from mpeg audio files, contributes a significant portion to the total delay, which depends on the file size. However, this delay cannot be decreased, as it is proportional to the real music time of an audio file. The module, which is responsible for the transformation of the watermarking key to a sequence of given patterns, is proportional to the key length, but it is very fast. Therefore, it contributes a minor portion to the total delay that is less than a second.

Another important module is the one that embeds the watermark to the scale factors of the audio file. The initial algorithm for watermark embedding in mpeg layers 1 and 2 audio files that was proposed by Dittman *et al.* [5] consists of a loop that is used to find out similar patterns with the pattern that is wanted to be embedded in the scale factors patterns stream of the original audio file. In each round of this loop the patterns of the scale factors are processed one by one for the estimation of the sums of squares difference with the wanted pattern. In the first round of the loop only patterns, which sum of squares difference to the wanted pattern is one, will be changed, in the next round the used tolerance is increased until the given maximal tolerance or the minimal number of patterns is reached. This loop was found that delays the watermarking procedure unexpectedly because it can't find the appropriate patterns fast enough. In order to decrease the delay we tried to find a way to break the loop. The chosen solution was proved that improves the time duration of the whole procedure dramatically. The solution we gave is based on the idea that the patterns with the smallest sum of the squares difference with the wanted pattern can be found if we estimate the sum of squares differences of the patterns of all the given scale factors with the wanted pattern and sort them in increasing order. Then, the wanted patterns are the patterns at the beginning of the sorted list that can be picked if the tolerance parameter is not violated. This solution can be applied after the counting of wanted pattern occurrences in scale factors' patterns sequence for the location of similar patterns.

In order the watermark embedding module not to consume much time, we only change the scale factors of the frames that are enough to embed the watermark once. So, we save time not only at this module, but as considers the software module that embeds the changed scale factors in the original audio file, too. The changed scale factors embedding procedure in the source audio file is limited only to the embedding of the changed scale factors frames. As considers the watermark detection part of our system, it doesn't need the original audio file to extract the embedding watermark and it is very fast as the only consuming part of it is the scale factors extraction mechanism applied.

Another factor that affects the delay of the system is the selection of patterns for the representation of binary digits "0" and "1" and the synchronization bit. The selected patterns should appear rare in the audio stream. Especially care should be taken when mp3 audio files are watermarked where scale factors have variant sizes. An improper selection of patterns in that case could result in big delay for finding the appropriate scale factors for modification in order the watermark to be embedded. For example if the chosen pattern has in its triplet a number bigger than four and the scale factors have sizes one or two bits in many consecutive frames it will have as a result big delay. In order to handle this problem in the mp3 case the patterns that have been selected are triplets that consist of numbers close to zero.

Moreover, the predefined number of frames for the embedding of a bit's pattern plays big role in system delay. If we embed each bit in a small number of frames the system will be fast but the audio quality will be low because the changed factors will be concentrated in a small region, if we choose a big number of frames to embed our bit then we will face biggest delays because we will have many more scale factors to process in each step. So, an appropriate trade-off should be found. After exhaustive testing, we decided that a good trade-off for fast system execution and acceptable audio quality in the resulted watermarked audio file is a selection of 5 to 15 frames for the embedding of each bit. Also, we found out that in the mp3 case we can use less number of frames to embed each bit than in the other cases.

As concerns the space requirements, we introduce a mechanism in the scale factors extraction procedure and in the changed scale factors embedding procedure that permits the process of audio data in small portions of 9800 bytes in each step. Thus, we don't need to preserve big arrays for storing audio data and other data that are required for the needed manipulations. With this way we make our system independent of the memory resources of the system, in which it is executed, as it needs to use small amounts of memory.

Another important issue is the processing of audio data in mp3 audio files for watermark embedding and detection. The problem starts from the fact that audio data of a frame are not located in the frame as happens in mpeg layers 1 and 2. They can be in the audio data field of previous frames. The solution we selected is the loading of audio data in a special buffer and their processing as a whole. Also, we use a special mechanism for the specification of the starting and ending byte of each scale factors region of each frame because the scale factors of a frame can be spread in many frames and the audio data field that contains scale factors has a special organization in granules and channels where Huffman bits intervene between scale factors areas.

Finally, it should be mentioned the fact that the watermarking algorithm for mp3 audio files not only implemented, but, also, designed by our team. It's the first compressed-domain watermarking algorithm proposed for mp3 audio files. In its design and implementation we faced a lot of problem mainly because of the different encoding it uses for audio data and scale factors respectively. Furthermore, another basic problem is owed to the variant nature of scale

factors size. Scale factors have 0 to 4 bits size. This fact differentiates mainly the scale factors extraction algorithm and watermark-embedding algorithm. In the scale factors extraction phase not only the scale factors should be extracted but their sizes should be extracted, too. Also, in the watermark embedding phase when we search for similar patterns in scale factors pattern stream we should check not only the sum of squares difference to be as small as possible for audio quality preservation but, also, we should control the size of scale factors that are proposed to be changed not to be violated. This is done using a special control mechanism.

V. Evaluation

In the context of this algorithm a lot of issues arise. The first of them is how somebody should choose the appropriate group of patterns to represent the information bits and the synchronization bit in order to enhance the security of the watermark and the inaudibility of watermark embedding. Basically, there are two rules for this selection that became evident after experimental evaluation. The first rule states that patterns, which occur often in the context of scale factors, should not be selected and the second one that patterns with larger steps than two among the components of their triple should be preferred. Also, when multiple keys are inserted inside the scale factor stream then the used patterns in different keys should differ as much as possible. Our algorithm is more secure than others with similar characteristics because it uses a key with semantic meaning. Thus, even if somebody guesses correctly the used patterns, he will not be able to *deduce* the semantics of the key.

Another important issue is that of robustness. Our algorithm operates at the compressed domain and not at the time domain. That is, we embed the watermark bits in the scale factors regions of the frames of an mp3 audio file that are special fields inside the audio data stream and not in the original raw audio data. Therefore, our algorithm is robust against attacks in audio data. Also, it is robust against local attacks or even random attacks to the whole region of scale factors because the watermarking key patterns are distributed over the whole range of subbands and the key allows its detection even if a percentage of it has been damaged. Such kind of attacks is not really a danger because they lead to audible distortions and damage the audio quality. We should mention that, as far as it concerns time-domain attacks that are applied to raw audio data, they require the watermarked mp3 audio file to be first decompressed. But, our algorithm cannot handle the decoding of the mp3 audio file and the recoding of it. However, the malicious users do not prefer this kind of attack, because it leads to a serious loss of quality. This occurs because in a watermarked mp3 audio file, we have modified some of the scale factors, which are used for the production of the real audio data at the decompression phase. This modification precludes decompression without affecting the sound quality. So in this sense, we can claim that our algorithm is robust against such attacks. Another kind of attack is the creation of a mono channel from the two stereo channels. This attack is handled efficiently because the patterns in the time axis survive this attack even if the patterns over the subband axis are damaged. Finally, in the context of robustness we should mention that our algorithm could be used both for authentication and copyright-protection. This is because of the nature of our watermarking key, which is a graph with a known 3-colouring, that can survive against inversion attacks (a malicious user subtracts his watermark from the original marked by the true owner) because it is spread all over the scale factors range and the colouring of even a small part of it can be used by the legal owner for proving ownership of the file that contains it.

Our algorithm has been tested in laboratory conditions where has been proved that the watermark embedding inserts a slight distortion but not a strong one. It should be mentioned that the distortion is more obvious in the case of spoken poems when there is no background sound. Our algorithm can be used for online distribution of audio files and in CD-ROMS and DVD from music industry for authentication and copy-protection purposes because it needs only small transfer rates for online use and the used calculations have low complexity because they are just additions or subtractions on integers or bytes. Also, the key's nature makes it essential for copyright protection.

The audio files that have been used in our experiments are:

- *Youthinasia*: Disco music (44.1 kHz, 128 kbps),
- *Dream*: Male ethnic singing with native instruments (44.1 kHz, 128 kbps),
- *Ipomoni*: Greek folk music (44.1 kHz, 128 kbps),
- *Blue*: Greek pop music (44.1 kHz, 96 kbps)

The following experiments have been conducted in a Pentium 4 at 3.4 GHz. Repeating these time tests in PCs with different processor speeds from 733 MHz to 3.1 GHz it was observed that the estimated times (embedding-detection time) are proportional to the raw processor speed. All the times that have been presented in the following experiments are in the same range as the real music time on the described platform. Furthermore, we should mention that these times refer to the users of our algorithm that want to make watermark embedding and detection and not to the audience of watermarked audio files who do not face any artifacts or delay hearing the watermarked audio files.

TABLE I
Averaged Results of Test

Example	MPEG	1 wm
Youthinasia	1.5	1.6
Dream	1.5	1.6
Ipomoni	1.5	1.6
Blue	1.5	1.6

Table I shows that the perceived quality of watermarked files is approximately the same as the quality loss produced by MPEG-compression. Most results are in the range of two, which means a difference like between two stereo-sets.

TABLE II
WATERMARK EMBEDDING TIME TESTS

File Name-Size	Embedding time (sec)	Real music time (sec)
Youthinasia-103Kb	6	6
Dream-1.3Mb	55	78
Ipomoni-2.9Mb	128	178
Blue-2.2Mb	92	180

Table II gives some test results of the time needed in our system to embed a watermark in mp3 audio files. Again the scale factors extraction component of the system is the most time consuming. We observe that less bit rate leads to more frames, which results in the encoding of more audio data. That's way the Blue audio file has more real music time than Ipomoni while it has less size.

TABLE III
WATERMARK DETECTION TIME TESTS

File Name-Size	Detection time (sec)	Detection (%)
Youthinasia-103Kb	4.5	100
Dream-1.3Mb	54	100
Ipomoni-2.9Mb	108	100
Blue-2.2Mb	90	100

Another important part of our algorithm is the watermark detection mechanism that suffers from the significant time needed to extract the scale factors from the original file. Table III gives some time results of the detection procedure along with the percentage of successful watermark detection.

Except from the above basic time tests, we used *Dream* audio file to study the impact of the use of different watermarking keys in the same audio file on watermark embedding and detection time (Table IV).

TABLE IV
DIFFERENT WATERMARKING KEYS TIME TESTS

Dream	Embedding time (sec)	Detection time (sec)
1-*char watermark (w)*	57	54
1-*char watermark (p)*	56	54
2-char watermark (re)	53	54

In Table IV we can observe that using different one-character keys (9-bit watermarks-8 bits the character + 1 the synchronization bit) there is a slight difference in watermark detection time, while using a two-characters key (17-bit watermarks) the detection time becomes bigger than the embedding time. These results are logical if we have in mind the embedding algorithm that searches for similar patterns to use them for watermark embedding. Therefore, it is logical different keys to have different probabilities of finding similar patterns in the audio stream. The second result in which the use of a two-characters key results in more detection time than embedding time comes to strengthen the previous conjecture. This implies that the selection of the watermarking key is more crucial for watermark embedding/detection time than the size of the key.

VI. CONCLUSION

The contribution of the presented watermarking scheme is two-fold: (i) It adopts and integrates in the audio watermarking concept ideas, which applicability and efficiency is well-known in cryptography giving to the users the enhanced capability their ownership of the audio file to be accomplished by showing that the legal owner knows a hard to compute property of the watermarking string. (ii) It exploits efficiently the speed and the memory of the PC system where it runs offering to the industrial user the capability of watermark embedding and detection in time immediately comparable to the real music time of the original audio file, while the end user/audience does not face any artifacts or delays hearing the watermarked audio file.

REFERENCES

[1] W. Bender, D. Gruhl, N. Morimoto and A. Lu, "Techniques for data hiding," *IBM Systems Journal,* Vol. 35, No. 3&4, pp. 313-336, 1996.
[2] L. Boney, A. Tewfic and K. Hamdy, "Digital watermarks for audio signals," *IEEE International Conference on Multimedia Computing and Systems*, pp. 473-480, 1996.
[3] M. Arnold and S. Kanka, "MP3 robust Audio Watermarking*," DFG VIIDII Watermarking Workshop 1999*, Erlangen, Germany, 1999.
[4] V. Basia, I. Pitas and N. Nikolaidis, "Robust Audio Watermarking in the time-domain," *IEEE Transactions on Multimedia*, Vol. 3, No. 2, pp. 232-241, June 2001.
[5] J. Dittmann, M. Steinebach and R. Steinmetz, "Digital Watermarking for MPEG Audio Layer 2," *Multimedia and Security Workshop at ACM Multimedia*, October 1999.
[6] L. Qiao and K. Nahrstedt, "Non-Invertible Watermarking Methods for MPEG Video and Audio," *Multimedia and Security Workshop at ACM Multimedia*, pp. 93-98, September 1998.
[7] F. Petitcolas, "MP3Stego," Computer Laboratory, Cambridge, 1998.
[8] C.H. Papadimitriou, *Computational Complexity.* Addison-Wesley, 1994.
[9] M. Garey, and D. Johnson*, Computers and Intractability, a guide to the theory of NP-completeness*. W.H. Freeman and Company, 1979.
[10] P. Cheeseman, B. Kanefsky, and W. Taylor, "Where the really hard problems are," *International Joint Conference on Artificial Intelligence*, Vol. 1, pp. 331-337, 1991.
[11] B. Hayes, "Computing Science: Can't Get No Satisfaction," American Scientist, March-April 1997.
[12] S. Kirkpatrick and B. Selman, "Critical behavior in the satisfiability of random Boolean expressions," *Science* 264, pp 1297-1301, 1994.
[13] S. Armeni, D. Christodoulakis I. Kostopoulos, Y.C. Stamatiou and M. Xenos, "Proving copyright ownership using hard instances of computationally intractable problems," *8th Panhellenic Conference on Informatics*, Nicosia, Cyprus, November 2001.
[14] D. K. Koukopoulos, Y. C. Stamatiou, "A Compressed-Domain Watermarking Algorithm for Mpeg Layer 3," *Multimedia and Security Workshop at ACM Multimedia*, pp. 7-10, October 1999.
[15] J. Seok, J. Hong and J. Kim, "A Novel Audio Watermarking Algorithm for Copyright Protection of Digital Audio," *ETRI Journal,* Vol. 24, No. 3, pp. 181-189, June 2002.