Digital Watermarking for MPEG Audio Layer 2

Jana Dittmann - Martin Steinebach - Ralf Steinmetz
GMD Forschungszentrum Informationstechnik GmbH, Institut IPSO, Darmstadt
Email: { dittmann, steinebach, ralf.steinmetz}@darmstadt.gmd.de

Abstract

MPEG-Audio has become a standard in the area of audio compression. It is used for a wide range of applications like online music distribution or in the audio parts of MPEG-Videos. In this paper we show how to secure the audio stream by watermarking without conversion to PCM-Wave-Data and without encoding the MPEG-Data in a special way. The original MPEG-file is not necessary to read the embedded information.

The key to our watermarking algorithm is to change the scale factor information of MPEG-frames. Small patterns are created producing a stream of information bits hidden in the data stream. Multiple streams can be included by using different patterns without critical reduction of perceived quality or robustness of the single watermarks.

Keywords

digital watermarking for MPEG audio, copyright protection, scale factor manipulation

1. Motivation

Today audio watermarking is used mainly for copyright protection. We want to provide watermarking technologies for authentication and the prove of integrity and research robust and fragile audio watermarking schemes. This new approaches can be used to improve security of multi media streams.
A possible attack against the integrity of an audio recording would be the removal of words, so that a sentence like “I am not guilty” could be changed into “I am guilty”. A watermark could be used to verify if the original has been changed by embedding a time stream.

2. Digital Watermarking

Digital watermarking is a way to embed data within another data stream or signal using aspects of the carrier signal like quantisation noise in A/D-conversion.

The technique of watermarking digital audio data has been the object of previous researches, e.g. [BTH1996], [SZAT1998], [SM1998] and [CKLL1997]. They provide a number of attributes which are important for watermarking:

- **perceptual transparent** : The watermark should not produce audible artifacts or reduce the quality of the audio data
- **robust**: The watermark should not be removable without seriously damaging the carrier audio data
- **statistical invisible**: Even when the algorithm for insertion of the watermark is known to the public, it should not be possible to destroy, fake or remove it without knowing a special key
- **self-clocking**
- **embedded directly in the data** (as headers can be removed or replaced easily)
- **multiple watermarks** should be possible
- the **expense** necessary to embed the watermark should stand in relation to its estimated effect
- **compression characteristics** of watermark and original data should be the same or at least similar
- **unambiguous**: the watermark should reliably identify the owner

Most of the previous works are based on marking PCM-Data. Many claim to be robust against MPEG-encoding, but tests have shown that the coding and decoding deletes most watermarks.

Watermarking algorithms are not robust against MPEG compression if they are based on the same principles: Parts of the audio data are masked or are not perceptible because of psycho-acoustic laws ([LAKa1996], [LAKb1996]) .

So either we have to embed the watermark in a perceptible area of the data, which would mean loss of quality, or we have to work directly on the MPEG Data.

With MP3Stego ([Pet1998]) there is an algorithm that inserts a watermark in layer 3 files. But to do so the audio data has to be encoded to MPEG by the algorithm. The resulting watermark is not robust against de- and re-compression and may not be robust against MPEG-attacks.

3. MPEG Audio Layer 2

The audio signal we used to test our algorithm is a MPEG-Audio Layer 2 stereo signal, 44.1 kHz and 160 bps. The basic idea of the algorithm makes it compatible with every Layer 1 or Layer 2 MPEG audio stream.

Only scale factor information of the MPEG stream is used in our algorithm. The following graph shows how the needed bits are extracted:

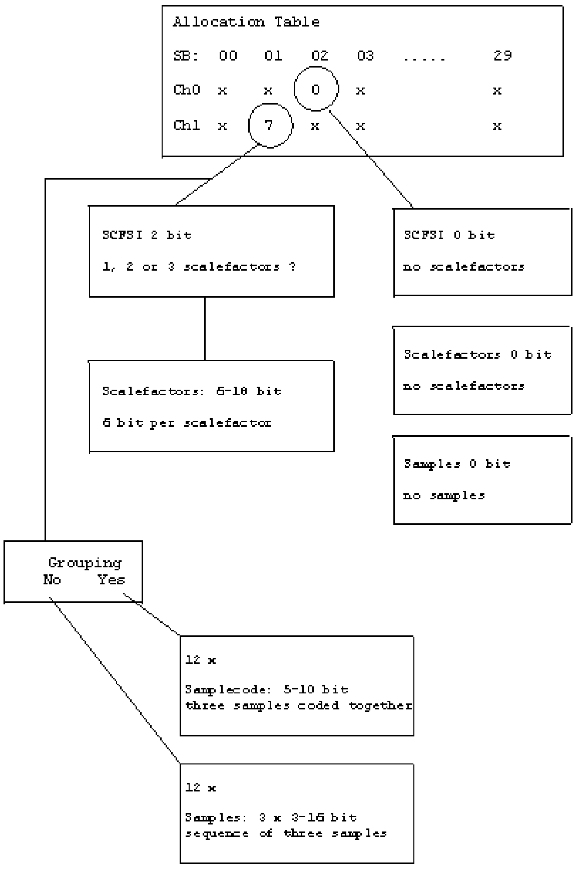

Figure 1: Scale factor location in MPEG-Stream

In Layer 3 scale factors are encoded in a different way. So to embed data in a Layer 3 audio stream we will have to change the extraction algorithm.

The data is divided in channels, subbands and one to three scale factors per frame. The allocation table tells which channels and subbands are encoded in the frame. Then we have to look at the scale factor selection information (SCFSI) to see how many scale factors are used for the samples. There are four different possibilities that use up to three scale factors. When we know how many scale factors are used we can extract them for further use. Knowledge of the used size of the samples is only necessary to stay synchronous with the data stream.

4. Watermarking principle

Figure 2 shows an overview of our algorithm. Given a MPEG-file, a text to embed and a group of three patterns we encode the text into a sequence of patterns and extract the scale factors from the frames of the MPEG-file.
Difference patterns based on this scale factors are calculated and the central algorithm changes these patterns until a sufficient number matches our desired sequence of patterns.
The whole watermark is inserted in this way, if there are more frames than needed the watermark is inserted multiple times.
Then the new scale factors are inserted in the source file, overwriting the old ones and so creating a watermarked MPEG-file.

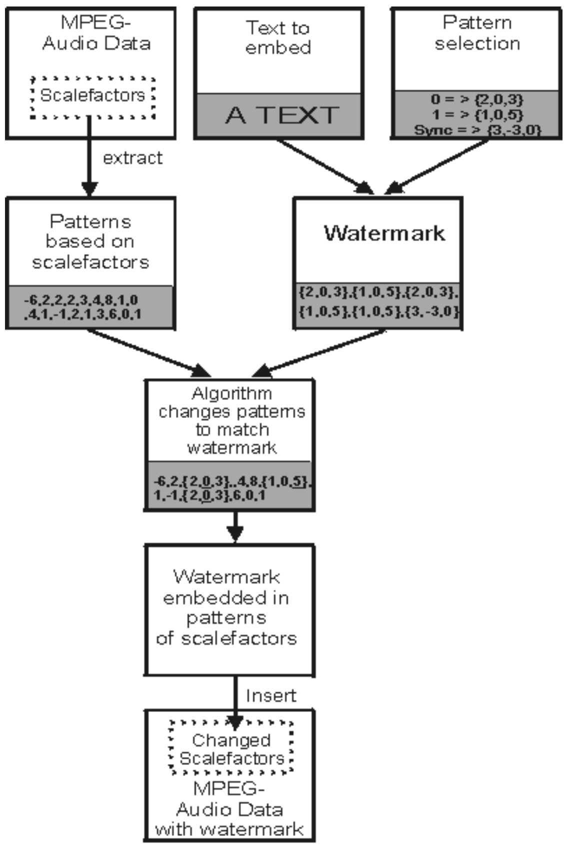

Figure 2: Watermarking principle

Embedding watermarks in the scale factor information has already been proposed in [NQ1998]. While our algorithm has first been developed uninfluenced by this work, we used some knowledge and experiences gained in [NQ1998] to improve audio quality. We also choose another way to embed and receive data and offer more security against attacks that would destroy the watermarks proposed in [NQ1998].
A main difference between the two algorithms is that the one in [NQ1998] needs the original signal do read the watermark. This also makes it resistant against inversion attacks where a third party could prove ownership by subtracting their watermark from the original marked by the true owner. As we build our algorithm for applications where ownership can be proven by other ways, it did not have to be resistant against this attack and therefor

does not need the original to read the watermark. Only a very small amount of data has to be transferred if it is used online - a huge benefice when a fast solution is necessary.

The key to our watermark is always a group of three patterns, one to code „0“, one for „1“ and another for „sync“. The last one is used for self clocking and robustness against cropping. These patterns consist of a few numbers that must match the differences between a starting point and the following scale factors in the data stream. An example: Given the list of scale factors {10,8,12,14} the first one would be used as a starting point and the pattern would be {-2,2,4}.

An information bit (0,1,sync) is present in a part of the stream when its number of occurrences is higher then the ones of the other two patterns. Therefor the flow of scale factors is searched for matching or nearly matching patterns while the other two patterns are destroyed. Then the nearly matching patterns are changed by adding or subtracting small numbers so that they finally match the requested pattern. Imagine we were trying to insert the pattern {-2,2,2} in the example above. We would have to change the last number (14) by -2 to match the pattern. So the work can be done easily by subtracting the existing pattern from the wanted pattern (a) and applying the result to the scale factors (b):

(a) {-2,2,2} - {-2,2,4}= {0,0,-2}

(b) {8,12,14}+ {0,0,-2} = {8,12,12}

This would be the final list of scale factors. By doing so we create an area in the MPEG-stream where one of the patterns is found quite often while the other two are found rarely or not at all.

5. The central algorithm

Based on the idea explained above, we created an algorithm that changes patterns in the sequence of scale factors so that two of three patterns are subdued and one is inserted a certain number of times.

The parameters of the algorithm are:

- the three patterns for „0“, „1“ and „sync“
- a maximum tolerance that states how strong the changes in the scale factors may be to create matching patterns
- a number of frames in which the changes take place
- the minimal number of patterns that must be equal to the pattern of the information bit we want to encode in the area of frames

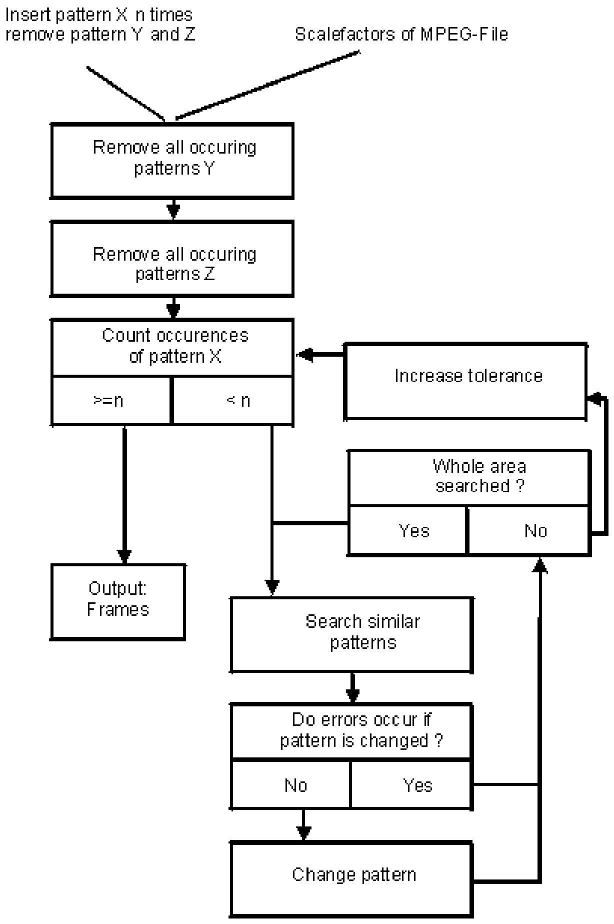
Insert pattern X n times
remove pattern Y and Z
Scalefactors of MPEG-File
Remove all occuring patterns Y
Remove all occuring patterns Z
Count occurences of pattern X
>=n
< n
Increase tolerance
Whole area searched ?
Yes
No
Output: Frames
Search similar patterns
Do errors occur if pattern is changed ?
No
Yes
Change pattern

Figure 3: Pattern insertion

As one can see in figure 3 the central algorithm consists of one loop. Given the wanted and unwanted patterns, minimum of matching patterns, tolerance and the sequence of scale factors, the algorithm first distorts any appearing pattern of the two unwanted and then starts to count the patterns corresponding to the information bit to be embedded. Usually there won’t be enough matching patterns to satisfy the given minimum. Now the algorithm will start to look for patterns similar to the wanted one.
Based on the sum of the squares of the difference between found and wanted patterns it will decide which patterns could be changed to the desired pattern without serious loss of quality. In the first round of the loop only patterns which differ by one will be changed, in the next the used tolerance is increased until either the given maximal tolerance or the minimal number of patterns is reached.
This means that at low levels of maximal tolerance the algorithm will not insert as many patterns as desired. This will protect audio quality by the loss of security.
The reason why the sum of the squares and not only the sum is used is that large changes are more audible than multiple changes, so it is better to change three scale factors by 1 (which means a difference of 3 ($1^2+1^2+1^2$)) than one by 3 (which would mean a difference of 9 ($3^2+0^2+0^2$)).

Figure 4 shows a small excerpt of a file marked with two watermarks at a bit rate of 1 bps. It was computed by comparing the scale factors of the original MPEG file and the marked one. Wherever gray bars are visible, scale factors exist. The white bars are the ones that were changed. The height of the bars is depended on the size of the scale factor.

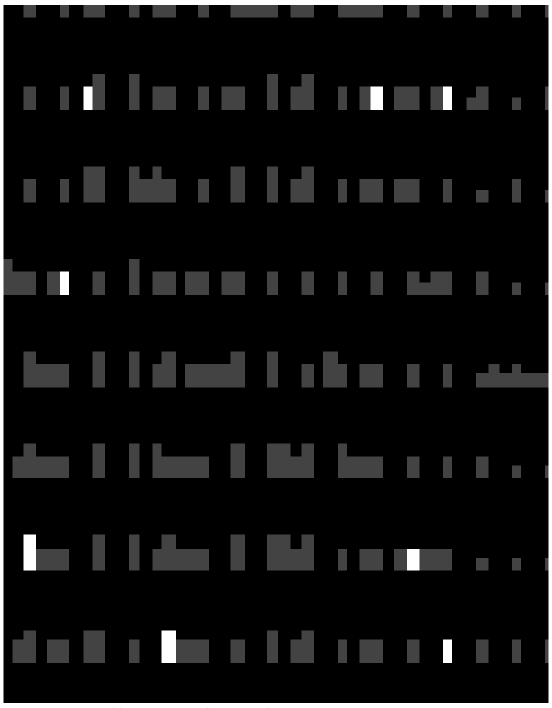
Figure 4: Differences of scale factors

The distribution of the scale factors is subband/channel in the Y-axis and SCFSI-number and frame in the X-axis. The same as used while searching for patterns.

5.1 Pattern selection

It is vital for the resulting audio quality and the security of the watermark to carefully choose the right patterns. Some patterns occur quite often and should not be used for data embedding as they would have to be changed each time they are found where another pattern should be inserted.

First steps in research have lead to the following „Top-20“ of pattern occurrences:

Rank	Pattern			Abs	Hits
1	0	-1	-1	2	393
2	0	0	0	0	375
3	-1	-1	-1	3	367
4	1	0	0	1	351
5	0	0	-1	1	331
6	0	-1	0	1	322
7	-1	-1	-2	4	316
8	0	0	1	1	316
9	0	1	1	2	310
10	1	1	1	3	310
11	0	1	0	1	309
12	1	1	0	2	306
13	0	-1	-2	3	303
14	0	-2	-1	3	302
15	1	0	1	2	301
16	-1	0	0	1	299
17	-1	-2	-2	5	297
18	-1	-1	0	2	294
19	0	-2	-2	4	294
20	1	0	-1	2	292

Abs: Sum of pattern absolutes
Hits: How many times the pattern has been found in so far four MPEG-Files with 63831 patterns.

Table 1: Occurrences of patterns

In our tests patterns with at least one number as large as three (e.g.{0,0,3}) or with steps larger then two (e.g. {0,-2,2}) worked fine. Of course the used patterns should differ as much as possible when using multiple watermarks, as two similar patterns would be changed to match each other and then the watermark embedded first would be destroyed or at least heavily distorted.

5.2 Security

The knowledge about the used patterns is the key to the security of our algorithm. To further improve security the patterns can be modulated. Thereby detection of key patterns becomes harder.

Example(pattern shifting):

starting pattern	{2,0,2}
modulation step 1	{0,2,2}
modulation step 2	{2,2,0}

The modulation process can be synchronized with the sync-bit, for example at the beginning of each encoded letter.

Embedding with a very high redundancy r1 enforces a lot of matching patterns which can be located more easily. A sensible threshold has to be found here.

The text information can be encrypted before embedding it, so even finding the right pattern combination will not necessarily lead to identifying it as no readable output is produced.

If an attacker knows the combination of the three patterns, he can easily destroy, change or replace the watermark.

5.3 Robustness

An example how to attack our watermarks inside of MPEG-files would be to change the scale factors randomly. But as our patterns are distributed over the whole range of subbands and are embedded in two directions (time and subband/channel) a large amount of scale factors would have to be changed. This would mean an audible loss of quality.

Another kind of attack would be the separation of one channel, thereby creating a mono channel out of one of the two a stereo channels. Almost all patterns distributed over the subband axis would be destroyed, but the ones in the time axis survive this attack.
In the current implementation the watermarks are not robust against decoding to PCM-Wave and back to MPEG. In our tests we found almost no similarity between the patterns of the original and the re-coded MPEG-file. Massive changes occurred which made the survival of patterns impossible. But we have to remember that decoding and re-coding to MPEG always means a serious loss of quality, which will stop pirates form choosing this way.

7. Detection of a watermark

Thanks to the open design of the algorithm we tested different ways to embed and extract data.

The first method was to use a fixed number of frames for every information bit. This makes it simple to extract the data if no trimming occurred: In the given region the three patterns are searched for and counted. The one with the most hits is the embedded bit.
For robustness against trimming the sync-bits can be used. With a fixed number of frames and a fixed number of bits to encode a letter the sync bit can be used as a header to resynchronize the algorithm at the beginning of each watermark.

The second method uses no fixed number of frames. Only the minimum of embedded bits and a tolerance are given. The algorithm steps through the frames and changes patterns if the tolerance allows it. When the minimum number of patterns to embed is reached, the algorithm continues with the next information bit. Thereby a given level of quality is always ensured, but the watermark is also embedded. The used number of patterns can vary as needed.

The detection process is more complicated as in method 1: We search for the dominant of the three patterns in the frames, and every time the dominance changes, the previously dominating bit is treated as a found information bit. To make this process robust against noise and false detection, weighting of the frames and filtering of very short dominance phases is used. The sync-bit is used to divide the “0” and “1” bits.

This method can be used to embed bit patterns in the MPEG stream. The detection rate of these patterns is high.

7. Test results

Most of the attributes of a watermark we mentioned in part 2 were realized in our algorithm. While points like security and robustness have already been mentioned, the following test results will try to complete the picture.

7.1 Transparency

We are still testing in this area. Right now you can say that a certain loss of quality is not deniable, but it is not strong enough to be found annoying. The following results are based on a test with ten students. The audio material has been burned on CD and was played on a usual stereo set in a natural listening environment.
The examples were rated on a scale from one to five where one was „no audible difference“ and five was „bad FM-receiver or scratches on a record“.

The test contained the following audio data:

- original (CD-Quality)
- only MPEG-encoding (44,1 kHz, 160 bps)
- MPEG with one watermark
- MPEG with two watermarks.

All examples were about 30 seconds long.
The watermarks were embedded with a bitrate of 1bps but with the requirement of twenty matching patterns.

The examples were:

- **Form**: Electronic dance music
- **Sheila**: Female ethnic singing with synthesizer background
- **Waaberi**: Male ethnic singing with native instruments
- **Crowd**: Talking group of people
- **Serenade**: Spoken poem

The averaged results in table 2 show us that in no case the perceived quality with the watermarks was a full step worse than the one with only the quality loss produced by MPEG-compression. Most results are in the range of two, which meant a difference like between two stereo-sets.

The last example, „Serenade“ was the one where the changes were heard most often. The algorithm produces some slightly audible noise in the reverb between the words .

Example	MPEG	1 wm	2 wms
Form	1,60	1,90	2,10
Sheila	1,60	1,90	2,10
Waaberi	1,60	1,90	2,10
Crowd	1,90	2,40	2,40
Serenade	1,90	2,40	2,70

Table 2: Averaged results of test

A second test was done to show if a listener could hear differences between the unmarked and the marked MPEG file. While the listeners heard the original at the beginning of each sequence, now only the unmarked MPEG was given to compare with.

We created sequences of ten 4 second long audio pieces of the examples „Form“, „Sheila“ and „Serenade“. Five types of material existed:

- unmarked
- 20 frames / 2 bps
- 10 frames / 4 bps
- 5 frames / 8 bps
- 3 frames / 14 bps

The given number of frames tells how many frames where used to encode at least 20 patterns according to the desired information bit. As a frame in this case is 23 ms long, there are about 43 frames per second. The bits per seconds (bps) are calculated by dividing these 42 frames by the used number of frames.

Ten sequences were created. The six listeners had to determine whether the actual piece of the sequence was a marked one or not.

The results of the test are shown in table 3. The last column tells how may percent of the times an audio piece of the according type was found to be changed. We can see that the unmarked pieces were chosen more times (10,4 %) than the ones with 2 and 14 bps (7,2% and 8,9 %). The ones with 4 and 8 bps were most often selected as marked, but not even in 20% of the times they occurred.

Type	n.o. app.	marked	%
none	40	25	10,4
2 bps	23	10	7,2
4 bps	13	12	15,4
8 bps	9	9	16,7
14 bps	15	8	8,9

n.o.app.: number of appearance in a total of 100 pieces

marked: how many times did the six listeners hear a loss of quality one of the pieces of the according type

Table 3: Results of watermark perception test

This shows that it is very hard to detect the changes made by our algorithm, even at higher bit rates like 14 bps. Only in 39 of 360 cases the watermark was heard, which is a percentile chance of 10,8 %. This is almost the same chance as the one of false detection (10,4 %).

7.2 Bitrate / Capacity

In our tests we ensured 20 matching patterns in a area of 3 frames, which was audible, but not annoying and good enough for most internet movies or previews. With about forty frames per second the given bit rate would be fourteen.

For our first audio test we used a bit rate of one bit per second, but with two parallel watermarks. For the second test we used bitrates up to 14 bps. In both cases detection success of the embedded data was 100 %.

Further research will be necessary to find out if there is correspondence between MPEG-bitrate and possible embedding-bitrate and which bitrates can be used with MPEG Audio Layer 1 and 3.

7.3 Complexity

Our algorithm has shown to be quite fast as the used calculations are mainly additions and subtractions on integers or bytes.
It also uses only a few kilobyte of memory. This is because we only have to look at a small part of the scale factor information at one time and can leave the rest of the file unchanged.
It can be assumed that a real-time application for detecting watermarks can be implemented without serious changes in the algorithm.

7.4 Self-clocking

Self-clocking was achieved with the „sync“ information bits: Depended on the way we insert our data, we can determine the number of frames used for one information bit.
In our current tests, we use one „sync“ bit to separate letters and two „sync“ bits to mark the end of the message.
But we could also reduce the amount of inserted data by only embedding a sequence of „1“s and „0“s and using a „sync“ at the end of the message. Then we would have to look for the number of frames used for this „sync“ and use this information

to decode the other bits if we did not know the number of used frames.

8. Conclusion

We introduced a way to mark MPEG-Audio Layer 2 files with one or multiple watermarks. Tests have shown that it doesn’t reduce audio quality significantly even with multiple watermarks.
The strength of our algorithm is that it works on existing MPEG-Files without the need of decoding to PCM-data and that it doesn’t need the original MPEG-file to read the watermark. Only small transfer rates would be necessary if used online, and due to low complexity of the used calculations the algorithm works very fast.

It is build to be secure against attacks imaginable against MPEG-files by the way the data is distributed in the scale factor information.

Removal is possible by decoding the audio file to a PCM-file and back again, but this would result in a high quality loss. A loss not acceptable in most situations a watermark is necessary.

9. References

[BTH1996] Digital watermarks for audio signals L. Boney, A.H. Tewfik, K. N. Hamdy

[SZAT1998] Audio watermarking and data embedding - Current state of the art, challenges and future directions Mitchell D. Swanson, Bin Zhu, and Ahmed H. Tewfik

[NQ1998] Non-Invertible Watermarking Methods For MPEG Encoded Audio, Lintian Qiao and Klara Nahrstedt, June, 1998

[SM1998] So this is Convergence? Technical, Economic, Legal, Cryptographic, and PhilosophicalConsiderations for Secure Implementations of Digital Watermarking, Scott Moskowitz, Blue Spike inc., 1998

[CKLL1997] Secure Spread Spectrum Watermarking for Multimedia, Ingemar J. Coxy, Joe Kiliany, Tom Leightonz and Talal Shamoony Lambda, Published in IEEE Trans. on Image Processing, 6, 12, 1673-1687, 1997

[LAKa1996] Laurence Boney, Ahmed H . Tewfik , and Khaled N. Hamdy , "Digital Watermarks for Audio Signals," 1996 IEEE Int. Conf. on Multimedia Computing and Systems June 17-23, Hiroshima, Japan, p. 473-480.

[LAKb1996] Laurence Boney, Ahmed H . Tewfik , and Khaled N. Hamdy , "Digital Watermarks for Audio Signals," EUSIPCO-96, VIII European Signal Proc. Conf., Trieste, Italy, September, 1996.

[Pet1998] MP3Stego, Fabien A.P. Petitcolas, Computer Laboratory, Cambridge, August 1998