A Novel Audio Watermarking Technique Based on Low Frequency Components

Hamad Alaryani
Department of Computer Science
The George Washington University
Washington, DC 20052
alaryani@gmail.com

Abdou Youssef
Department of Computer Science
The George Washington University
Washington, DC 20052
ayoussef@gwu.edu

Abstract

In this paper, we present a novel audio watermarking technique that utilizes the Low Frequency Components (LFCs) of an audio signal to identify the location of the embedded watermarks. The embedding takes place by modifying the amplitude of selected samples determined by the LFCs of the audio signal. The amount of modification to the amplitude is determined by the amount of distortion detected by the human ear. This technique is blind where the decoder does not need the original audio file to extract the watermarks. In this technique, we use a novel data recovery scheme to recover any ~~missing~~ watermarks that were lost because of an intentional or unintentional attempt of watermark removal (attack). Experimental results show that this technique is highly robust against single and double attacks with watermark ~~WM~~ recovery rates greater than 90%.

1. Introduction

The ~~wide spread~~large number of high~~-~~ speed internet users and the availability ~~and feasibility~~ of audio recording and editing software and devices, such as CD-writers and mp3 player, have put the music industry in a critical situation. The industry's intellectual properties are becoming in danger of being attacked by so called 'music pirates'. Internet applications such as Kasa and Morphous made it easier for music pirates to copy and distribute music and songs around the world illegally. Data protection techniques such as encryption are insufficient for protecting the music industry's intellectual properties, because once the music files are decrypted, they could be easily copied using the traditional analog audio recorders in the worst scenario. A feasible solution is to use digital watermarking by inserting the copyright identification digitally with the ability to survive attempts of removal or tempering.

There are many digital watermarking techniques that are intentionally designed for the sake of copyright protection for the music industry. Unfortunately, there is nothing perfect in this world. There is always a drawback for each technique. Some watermarking techniques are highly robust to many attacks and at the same time their robustness fails to a single attack.

The purpose of this paper is to present a robust audio watermarking algorithm that utilizes the Low Frequency Components (LFC) of an audio signal as its guide to locate and then insert the digital watermarks in the audio file. The insertion process is to modify the amplitudes of the audio signal at those locations. The number of locations reflects the amount of embedded watermarks in the signal and the amount of amplitude modification reflects the amount of distortion that occurs to the signal after embedding.

In this paper, we discuss in section 2 some background information and definitions that we use in our technique. In section 3 we review some of the related work. In section 4 we introduce our ~~highly robust~~ watermarking technique and its basic aspects. The experimental results obtained using our technique will be discussed in section 5. Finally, in section 6, we present final remarks and future work.

2. Background & Definitions

The Human Auditory System (HAS): Most of the current audio watermarking techniques depend mainly on the limitation and imperfection of the Human Auditory System [1]. HAS is insensitive to small amplitude changes, and some quiet sounds are fairly ignored in the presence of loud sounds. Additionally, there are situations where the distortion of a signal is ignored by the listener due to the common environmental distortions [2, 3]. Therefore, modifying the amplitude of an audio signal might not affect the perception of the quality of the sound by the human ear.

Formatted: Underline

Low Frequency Components (LFC): The audio signal in its original format is a mixture of high and low frequencies that form semi-sinusoidal waves with uneven amplitudes and wavelengths as shown in figure (1-a). When these waves are passed through a Low-Pass Filter (LPF), all high frequency components are reduced or eliminated, and only the low frequency components remain largely intact as shown in figure (1-b). The resulting signal is called the *Low Frequency Components* (LFCs).

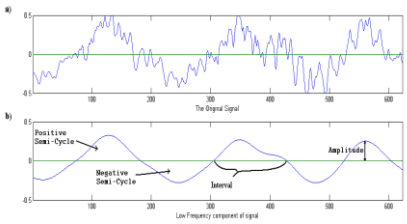

Figure 1. a) Audio Signal. b) Corresponding Low Frequency Components (LFCs).

Semi-Cycle: a *semi-cycle* is any all-positive or all-negative portion of the signal, as shown in figure (1-b). The length of a semi-cycle, measured as the number of samples the semi-cycle has, is called the *interval* of the semi-cycle. The average sample amplitude value in a semi-cycle is referred to as the *Average Amplitude* of the semi-cycle.

Modifiable Semi-Cycle: A semi-cycle is said to be *modifiable* if its average amplitude can be modified (increased or decreased) without any perceptible distortion to the original signal. Therefore, there are limits for the semi-cycle average amplitude and intervals, beyond which distortions become perceptible. The average amplitudes and intervals must then be within certain ranges.

Quartets: A *quartet* is a group of four consecutive modifiable semi-cycles; the first and third must be positive semi-cycles, while the second and fourth are negative

3. Related Work

~~Many researchers have done work~~Considerable research has been done on audio watermarking that deals directly with audio amplitude. The *Amplitude Modification* [4] technique is one of them. It takes a number of blocks of a predetermined length (in samples), say N. Then, it calculates the energy of that block. Consider two consecutive blocks, A and B, with energies E_A and E_B, respectively. The energies of blocks A and B are modified so that $E_A > E_B$, to denote a **0** embedding. Similarly, they are modified so that $E_A < E_B$, to denote a **1** embedding. This algorithm has a serious problem. Assume we want to embed for example a zero and E_B is much larger than~~t~~ E_A. This technique will attempt to increase the energy of block A, or decrease the energy of block B. This will require a very large change in the amplitude, making the watermark signal highly perceptible ~~[4]~~.

Another technique that deals with amplitude is called *echo hiding* [5]. This technique is very successful. It uses a type of distortion to the signal, which is undetected by the human auditory system. An echo in audio signals is nothing but a copy of the portion of the signal. The echo is decreased in energy and delayed by a small period of time. Therefore, three parameters ~~can be~~are introduced in echo hiding ~~[5]~~: the initial amplitude, the delay rate, and the offset. By varying these three parameters, one could hide data in the audio signal. This technique ~~of data hiding~~ makes the resulting sound ~~becomes~~ *rich*, compared to the original sound. However, the richness of the sound makes it obvious for the attacker to know that there is data embedding in the sound. Another disadvantage of the echo hiding technique is its high complexity in the watermark-identification process [4]. Furthermore, there are some algorithms that could retain an accurate estimation of the delay when an echo is added to the original signal [6].

4. Our Watermarking Technique

Our technique was based on an assumption that the Low Frequency Components (LFCs) of an audio signal do not change much when subjected to common audio signal manipulation and processing such as filtering or compression. When we say '*do not change much*' we mean that ~~the LFCs shape and low frequency value are~~ largely preserved since it is the base of the sound that, when changed or destroyed, causes ~~the sound~~ audio to sound different~~gets changed or destroyed~~. To test our assumption, we performed exhaustive tests to audio signals by applying a variety of audio processing techniques to them and compared the original LFCs to the resulting once. Over 90% of the resulting LFCs survived

Formatted: Underline
Formatted: Font: Italic, Complex Script Font: Italic
Formatted: Font: Not Italic, Complex Script Font: Not Italic
Formatted: Underline
Formatted: Font: Italic, Complex Script Font: Italic
Formatted: Font: Italic, Complex Script Font: Italic
Formatted: Font: Italic, Complex Script Font: Italic
Formatted: Underline
Formatted: Font: Italic, Complex Script Font: Italic
Formatted: Underline
Formatted: Font: Italic, Complex Script Font: Italic

the audio manipulation. The LFCs that didn't survive had a very poor sound quality which means that if a manipulation cannot preserve the sound quality then it cannot be considered as a successful attack. In our techniques, as we mentioned earlier, the locations of the embedded watermark (WM) are based on the LFCs of the audio signal. As long as the sound quality is preserved, we can almost make sure that we are able to get to the locations of our embedded WM; therefore we could possibly retain the embedded WM. The audio manipulations that we used include MP3 compression with 6 different bit rates, filtering, re-sampling, re-quantization and rescaling.

4.1 The Watermark-Embedding Process

As shown in figure (2), the original signal is temporarily passed through a Low-Pass Filter (LPF) to extract the LFCs. Then we look for the modifiable semi-cycles from which we determine the quartets. This step is needed because the encoder does not know *a priori* the locations of the quartets since they are dependent upon the audio file. Each quartet will be used to embed a single bit of the WM. Once we know the locations of all the quartets **in the LFCs**, the corresponding locations of the quartets **in the original signal** are marked, and the quartets in the original signal are modified in a way that encodes the 0s and 1s of the intended watermark. The details are covered next.

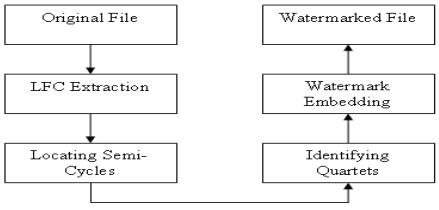

Figure 2. Embedding Process

4.2 Watermark Embedding

A watermark is a (short) sequence of bytes. To enhance the robustness of the watermark, each bit in a byte of the WM will be encoded with 4 "physical" bits: the first physical bit will be identical to the ~~(~~Watermark Bit~~)~~ ~~WB bit~~ being encoded, and the other 3 physical bits encode the location of WB bit within its byte.

We mentioned earlier that when embedding a WM, we simply modify the average amplitude of modifiable semi-cycles. Recall that each quartet represents a single bit of WM embedding, and each consists of a group of 4 modifiable semi-cycles. Therefore, each modifiable semi-cycle has a meaning in the embedding. We use the first semi-cycle to represent the first physical bit, that is, the value of the embedded WM bit, and the remaining three semi-cycles to represent the other 3 physical bits that encode the within-byte location. For example, suppose we were to embed the fifth bit of an 8-bit WM word that has a value of '1' as shown in figure (3).

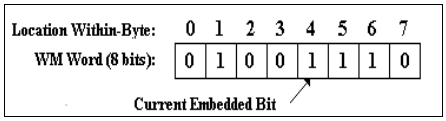

Figure 3. An 8-Bit WM Word.

The 4-modifiable semi-cycles will be modified to represent the values 1, 1, 0 and 1. (1 for the bit value, and 101 for the location within-byte). This process is repeated across the entire file. The first embedded WM bit will have the location within-byte value to be 0 (000). The last 8th embedded WM bit will have the location within-byte value to be 7 (111). ~~The next location within-byte value will reset to 0 (000).~~ Since the number of quartets in a typical audio piece is larger that the watermark size by several factors, the WM will be embedded multiple times in the signal. ~~This means the WM code is embedded more than once.~~ This will increase the robustness of our technique against file chopping. Now the only remaining question ~~comes~~is, how do we modify a semi-cycle to represent a 1 or 0? The answer is covered in the next section.

Formatted: Font: Bold, Complex Script Font: Bold

Formatted: Font: Bold, Complex Script Font: Bold

4.3 Semi-Cycle Modification

A modifiable semi-cycle needs to be in a certain range in terms of semi-cycle length in samples (interval) and average amplitude, ~~in a~~so ~~way~~ that no perceptible distortion to the sound occurs if modified. Therefore, we introduce the following parameters:

A_{max}: The maximum allowed average amplitude for a modifiable semi-cycle to be considered.

A_{min}: The minimum allowed average amplitude.

A_{mid}: The middle value between A_{min} and Amax.

I_{max}: The maximum allowed interval for a modifiable semi-cycle to be considered.

I_{min}: The minimum allowed interval.

If a semi-cycle is to be modified to represent 1, its average amplitude is modified to $A_1=[A_{mid}+A_{max}]/2$, and if to be modified to represent 0, then its average amplitude is modified to $A_0=[A_{min}+A_{mid}]/2$. Of course, the modifications are done directly to the original signal and not to the LFCs. LFCs are only used to determine the modification locations.

4.4 Embedding Example

Let's suppose we are embedding a WM bit of value '1' and location within-byte equal to 5 (101). For simplicity, let us assume that we experimentally found out that reasonable values for A_{min}, A_{max}, I_{min} and I_{max} are 0.3V, 0.5V, 50 Samples and 100 Samples, respectively. Therefore, A_{mid}, A_0 and A_1 equal 0.4V, 0.35V and 0.45V, respectively as shown in figure (4-a).

Figures (4-b & 4-c) show the original quartet and the quartet after modification. By looking at the left semi-cycle of the original signal, the average amplitude was 0.4465V. After embedding '1', the new average amplitude became 0.45V (A_1). Note that the original LFC was intentionally redrawn with the watermarked signal to make comparison easier.

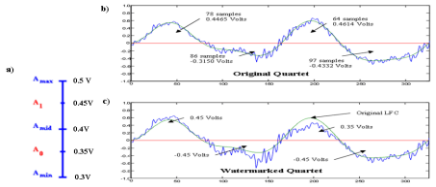

Figure 4. a) Average Amplitude Parameters. b) Original Quartet. c) Watermarked Quartet.

4.5 Extraction Process

The watermark extraction process is similar to the embedding process. The watermarked audio file is passed through a Low–Pass Filter to extract its LFCs. Note that no matter what happens to the watermarked file, as long as its sound quality is acceptable, it will preserve its LFCs. Therefore, using these LFCs, the decoder tries to locate all modifiable semi-cycles ~~that their conditions (parameters) are pre-set at embedding time~~, from which the quartets are determined. Once it has all quartets, the watermark extraction takes place. All retrieved WMs go through an error correction and recovery phase as shown in figure (5).

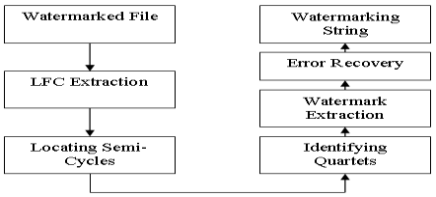

Figure 5. Extraction Process

4.6 Single Quartet Detection

As shown in figure (6), the decoder found 4 consecutive semi-cycles that satisfy the preset parameters A_{min}, A_{max}, I_{min} and I_{max}, mentioned earlier. Therefore, it considers them as a valid quartet. The decoder then inserts this quartet values into two pre-initialized arrays. The first array represents the bit values and the second represents the location-within-byte of those bit values.

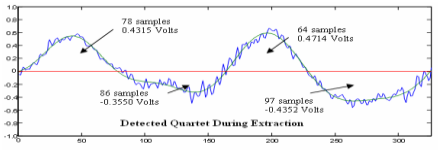

Figure 6. Single Quartet Detection

The values are determined according to the following two conditions:

- *If the semi-cycle average amplitude is greater than A_{mid} (0.4V), the extracted value is '1'.*
- *If the semi-cycle average amplitude is less than A_{mid} (0.4V) the extracted value is '0'.*

According to the figure above, the first semi-cycle average amplitude is 0.4315V. Therefore, the extracted value will be '1', which is inserted to the first array (bit values). The second, third and forth semi-cycles' average amplitudes were 0.3550V, 0.4714V and 0.4352V, respectively. Therefore, the decoder will insert 3 (011) into the second array (location-within-byte).

4.7 Full WM Extraction & Error Recovery

Remember that each extracted bit value in the first array has a corresponding location-within-byte in the second array. The first extracted bit location-within-byte should be 0. However, this might not be true in all cases. If, for example, the file had been chopped so that the first extracted bit location-within-byte is 4(100), the decoder will insert the missing location-within-byte to the second array and will insert their corresponding bit values in the first array to x (*don't care values*). If any of the bits following the current extracted bit is missing, the decoder will insert its location-within-byte in the second array and will insert its corresponding value into the first array with a bit value to be x (~~*don't care*~~ *still-unknown value*). Figure (7) illustrates this situation. Each consecutive 8 bits now compose a full WM code. As shown in the figure, there are 3 full WM codes that were extracted. Of course, some of the extracted bits could be erroneous due to signal manipulations by the attacker or due to non-malicious processing such as compression.

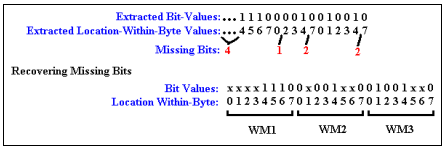

Figure 7. Missing Quartets Recovery

After obtaining a number of WM codes, the decoder tries to recover the bits that have '~~*don't care*~~*still-unknown*' values and~~, or those~~ the bits that have errors, using a voting scheme by taking the average values of the extracted bit for each location-within-byte. Let's suppose that the original WM code was (01001110) and the decoder extracted 5 WM codes as show in Table 1. The average is calculated by finding the sum of the extracted bits divided by the number of bits (excluding the *still-unknown* ~~*don't care*~~ bits). The average is then is rounded to the nearest integer to obtain an error free WM bit. Even though there were some bits that had errors and were used in calculating the average, the decoder was still able to get the correct bit value. To make it easier for the reader, the incorrect bits were displayed in the table as underlined text as show in location 0 of the extracted WM4.

Table 1. The Values of 6 Occurrences of a 1-byte WM, and the Recovered WM.

Location	WM1	WM2	WM3	WM4	WM5	Average	Rounded
0	0	0	X	*1*	0	1/4 = 0.25	0
1	1	X	1	1	*0*	3/4 =0.75	1
2	0	*1*	0	X	0	1/4 = 0.25	0
3	X	0	0	0	0	0/4 = 0	0
4	1	1	1	1	1	5/5 = 1	1
5	1	X	1	X	1	3/3 = 1	1
6	1	X	1	*0*	1	3/4 =0.75	1
7	0	0	0	X	*1*	1/4 =0.25	0

5. Experimental Results

Our experimental results are divided into two parts. The first part is to calculate (on average) how many bits of watermark per second (of time) of original signal we can obtain using our

technique (Bandwidth). The second part is to measure the robustness of our technique.

5.1 Bandwidth

A database of over 100 music and song files was used. The average file length ~~of~~ was 3.5 minutes. The average number of bits of watermarks that can be inserted was found to be 733 bits/file. Therefore, the average number of bits of watermark per second is 733/(3.5x60) = 3.49 bps. This means that it is possible to encode a watermark word of length 32 bits in a single file that is as short as 10 seconds. This also means that in a typical song, a 32-bit WM can be embedded 23 times. This redundancy is very valuable for error tolerance, WM extraction, and watermark robustness against chopping.

5.2 Robustness

For the sake of testing the robustness of our technique, we randomly selected 15 (out of 100) watermarked files and applied single and double attacks to them. The attacks we used were MP3 compression (bit rate = 32 kbps), filtering (cut-off frequency = 3 kHz), re-sampling (sampling rate = 22050Hz), re-quantization (quantization = 4 bits per sample), and chopping (15 seconds). Table (2) shows the WM recovery rates obtained.

Table 2. WM Recovery Rates from Single/Double Attacks

	MP3	Filtering	Re-Sample	Re-Quantize	Chopping
MP3	98.7%	97.1%	97.8%	97.1%	91.4%
Filtering	97.1%	98.1%	96.4%	96.7%	93.8%
Re-Sample	97.8%	97.4%	99.6%	99.1%	94.3%
Re-Quantize	97.1%	96.7%	99.1%	99.3%	95.2%
Chopping	91.4%	93.8%	94.3%	95.2%	95.4%

The diagonal shows the results of single attacks. Any other cell in the table shows the result of the double attacks of the crossing table headers. The ~~worst~~ smallest recovery rate obtained was 91.4% when applying mp3 compression to the watermarked file and then chopping it. Even in the low 90s, these recovery rates are very high. What is more encouraging is that in the case where the recovery rates are in the low 90s, the sound quality of the attacked watermarked files was quite low, which means an attacker will not benefit from the resulting file after attack because of the resulting poor sound quality.

6. Conclusion and Future Research

In this paper we introduced and evaluated a novel, highly robust watermarking techniques for audio files. A particular advantage of this technique is that the decoder does not need a reference copy of the original un-watermarked audio file. It only needs to know the pre-set parameters A_{min}, A_{max}, I_{min} and I_{max}. The decoder, in this case, is called "*informed detector*" [7]. Another advantage is that it is computationally possible to decode the watermark while playing the audio file, since it is possible to get the LFCs of the audio file using a Low Pass Filter. This feature makes the technique practical in identifying the copyright tags of songs that are broadcasted by radio stations, if the tags were saved as watermarks.

Further research is needed to determine standardized parameters for all of types of audio files to make the technique completely blind instead of being an "informed detector".

7. References

[1] S. Katzenbeisser and F. A. P. Petitcolas, *Hiding Techniques for Steganography and Digital Watermarking*, S. Katzenbeisser and F. A. P. Petitcolas, Eds. Boston, MA: Artech House, 2000.

[2] Walter Bender, Daniel Gruhl, Norishige Morimoto, Anthony Lu. "*Techniques for Data Hiding*", IBM Systems Journal, Vol. 35, No. 3&4. 1996. pp. 313-336

[3] Nedeljko Cvejic (2004) "*Algorithms for audio watermarking and steganography.*" Oulu: University of Oulu. ISBN 951-42-7383-4

[4] Kim, H. J., "Audio Watermarking Techniques," Pacific Rim Workshop on Digital Steganography, Kyushu Institute of Technology, Kitakyushu, Japan, July 3-4, 2003.

[5] Whitiak, Deborah A. "The Art of Steganography." 2003. The Sans Institute. 14 Oct 2004 <http://www.giac.org/practical/GSEC/Deborah_Whitiak_GSEC.pdf>.

[6] K. Kaabneh, A. Youssef "Muteness-Based Audio Watermarking Technique". Dsc. Dissertation, the George Washington University, 2000.

[7] Ingemar Cox, Je rey Bloom, and Matthew Miller. Digital watermarking. In Digital Watermarking. Morgan Kaufmann, 2001.