Audio Watermarking for Copyrights Protection

Technical Report
School of Engineering Report No. 650

Yiqing¹ Lin and Waleed² H. Abdulla
Department of Electrical and Computer Engineering
The University of Auckland, Private Bag 92019,
Auckland, New Zealand

¹ylin124@ec.auckland.ac.nz  ²w.abdulla@auckalnd.ac.nz

2007
ABSTRACT

Digital watermarking is the process that imperceptibly watermarks the multimedia product with a specific watermark for the purpose of content authentication, data monitoring and tracking, and copyright protection. Amongst the applications, the most prominent usage of watermarking technique is helping in identifying the origins of different multimedia files and resolving ownership disputes. Specifically, the report is concerned with audio watermarking for copyright protection.

In the report, issues on audio watermarking for copyright protection are discussed in details. Firstly, introduction to audio watermarking is made in section 1, which is a glance of the state-of-the-arts in audio watermarking technique. Secondly, section 2 describes the principles of Psychoacoustics. Psychology of hearing, especially masking phenomenon, and psychoacoustic model are included. Thirdly, requirements of audio watermarking for copyright protection are stated in section 3, such as imperceptibility, robustness, security, capacity and speed. Actually, these prerequisites are what subjective and objective evaluations rely on. Section 4 is the extension of section 3. In the design of any audio watermarking scheme, robustness is a typical aspect worthy of considerations. So section 4 studies on common signal processing operations and deliberate manipulations, in order to get a better understanding of the mechanisms of various attacks. It is help for performing correct detection. Next, section 5 and 6 give detailed insight into different approaches on audio watermarking and also some supplemental techniques. Each basic system, i.e. LSB modification, phase coding, spread spectrum watermarking, cepstrum domain watermarking, wavelet domain watermarking and echo hiding, is implemented and evaluated, so as to compare their performances and find out their strengths and weaknesses. Based on the studies above, a robust and secured audio watermarking system using Gammatone auditory filterbank, coded image, multiple scrambling and adaptive synchronizing is developed in Section 7. Its embedding and detection methods are expounded and a complete assessment is given. Finally, Section 8 concludes the report and literature cited are listed in the last section.

Key words: Audio Watermarking, Information Hiding, Copyrights Protection, Multimedia Watermarking
# TABLE OF CONTENTS

1. Introduction to Audio Watermarking ................................................................. 1

2. Principles of Psychoacoustics ......................................................................... 3
   2.1 Psychology of hearing ............................................................................... 3
   2.2 Hearing Thresholds and Masking Phenomenon ....................................... 5
   2.3 Psychoacoustic Model ............................................................................. 10

3. Requirements of Audio Watermarking for Copyright Protection .................. 16
   3.1 Imperceptibility ..................................................................................... 16
   3.2 Robustness .......................................................................................... 17
   3.3 Security .................................................................................................. 19
   3.4 Capacity .................................................................................................. 19
   3.5 Speed ...................................................................................................... 19

4. Testing Items in Measuring Robustness ......................................................... 20
   4.1 Noise addition ...................................................................................... 21
   4.2 Resampling ............................................................................................ 21
   4.3 Lowpass Filtering .................................................................................. 22
   4.4 Echo Addition ....................................................................................... 23
   4.5 Data Compression .................................................................................. 24
   4.6 Random Samples Cropping and Zeros Inserting ..................................... 25
   4.7 Jittering .................................................................................................. 27
   4.8 Pitch-invariant Time Stretching ............................................................... 27
   4.9 Tempo-preserved Pitch Shifting ............................................................... 30

5. Audio Watermarking Techniques ................................................................. 33
   5.1 Least Significant Bit (LSB) Modification ............................................... 33
   5.2 Phase Coding .......................................................................................... 35
   5.3 Spread Spectrum Watermarking ............................................................. 37
5.4 Cepstrum Domain Watermarking ................................................................. 38
5.5 Wavelet Domain Watermarking ................................................................. 43
5.6 Echo Hiding .............................................................................................. 46

6. Supplemental Techniques ........................................................................ 52
6.1 Salient Points Extraction ......................................................................... 52
6.2 Power Points Extraction ......................................................................... 52
6.3 Envelope Peaks Extraction ..................................................................... 53
6.4 Silence Removal ....................................................................................... 53
6.5 Redundancy and Error Correction Coding .............................................. 55
6.6 Coded Image Enhancement ..................................................................... 55

7. A Robust and Confidential Audio Watermarking Scheme ...................... 58
7.1 Embedding Method .................................................................................. 58
7.2 Detection Method ..................................................................................... 61
7.3 Experimental Results and Discussions .................................................. 65

8. Conclusion and Future Work ..................................................................... 67

9. Reference .................................................................................................... 68

Appendix: Audio Watermarking Benchmarking .......................................... 72
A. STEP 2000 ................................................................................................. 72
B. StirMark for Audio .................................................................................... 73
C. SDMI Evaluation ...................................................................................... 75
1. INTRODUCTION TO AUDIO WATERMARKING

On-line distribution of digital multimedia including images, audio, video and documents has proliferated rapidly in recent years. In such an open environment, it is convenient to get the access to various information resources. Along with the ease by which the digital formatted data can be copied and edited, copyright infringement like illegal reproduction and distribution has arisen and greatly spoilt the originators’ passion for innovation. To prevent such iniquities, the enforcement of ownership management has claimed more and more attention. As a result, a novel watermarking technique is developed for copyright protection [1]. Digital watermarking is the process that embeds copyright information as ‘watermark’ into the multimedia object, so that the watermark can be extracted later to make an assertion about the ownership [2]. The general schematic diagram of watermarking is shown in Figure 1 [3].

From the point of view of information hiding, watermarking is a subdivision of steganography. Steganography refers to techniques of hiding messages, maybe secret signatures or public captions, in the host carrier for the purpose of identification, annotation and rights management [4]. For example, a common form of hidden writing is using ‘invisible’ inks. So the secret message can only be read by processed with some prescribed chemicals in a certain sort of way [5]. Roughly speaking, when the message content is related to the host carrier, then the hidden message is a watermark [5].

Digital watermarking techniques can be sorted into several kinds of categories on various bases [4].

1. Type of document to be watermarked
   - Audio watermarking
   - Image watermarking
   - Video watermarking
   - Text watermarking

2. Robustness of watermark
   - Robust: The watermark is hard to remove by modifications, mainly for copyright protection.
   - Fragile: The watermark is likely to be impaired by manipulations, mainly for data authentication.

3. Host signal required for extraction
   - Blind (or public): Host signal is not required for watermark extraction.
   - Non-blind (or private): Host signal is required for watermark extraction.

Specifically, our research is focused on embedding imperceptible, robust and secure watermarks for copyright protection. Different from cryptology for data security, watermarking techniques do not encrypt the host
information to restrict the access to approach, but embed one or more watermarks with separately specific meanings into the host carrier [6]. These watermarks are permanent signs, hard to clear up without degrading the quality of the host media. When proprietorial disputes happen, the watermark(s) could be extracted as reliable proofs for assuring the authorship [7].

Audio watermarking is the application on audio signals, a rather new field. Compared with images and video, inserting imperceptible, robust and secure watermark(s) into digital audio files presents special challenge. Since audio signals are represented by much less samples per time interval, the room for embedding is limited [8]. Moreover, the human auditory system (HAS) is much more sensitive than the human visual system (HVS), which means that inaudibility for audio is more difficult to achieve than invisibility for images [9]. Therefore, the advance of audio watermarking is slower than that of image or video watermarking.

During the previous decade, driven by prevalence of compressed audio data, audio watermarking technique has become an urgent need and made a great deal of progress. A variety of algorithms have been put forward for audio watermarking, such as LSB modification, phase coding, spread spectrum watermarking, cepstrum domain watermarking, wavelet domain watermarking and echo hiding [6,10-14]. Depending on the perceptual properties of HAS, they exploited the features of audio signals in different domains and embedded the watermark, a sequence of pseudo-random (PN) numbers or a coded image, by making unperceivable modifications to the host carrier. Some supplemental techniques, like salient point extraction, silence removal and error correction coding, are also utilized to provide some enhancement for robustness and security. In the detection, the watermark(s) could be extracted by performing corresponding lookup via secret keys. In the case of a coded image as the watermark, post-process could be applied for a better recognizability. Later, we will further discuss the basic ideas of audio watermarking schemes mentioned above and evaluate their performances. Based on these studies, a robust and secured audio watermarking scheme is implemented.

Obviously, audio watermarking is a promising technique for copyright protection and attracts the attentions of researchers from various universities and companies. However, this line of work still hasn’t been satisfactorily developed around the world currently. Whether any of the newer research work or the patent claims would survive to commercial exploitation is yet to be determined [5].
2. PRINCIPLES OF PSYCHOPHYSICS

Psychoacoustics is the science that studies the statistical relationships between acoustical stimuli and hearing sensations [15]. Because the process of embedding the watermark is required to be imperceptible, understanding the principles of psychoacoustics and making use of its perceptual properties are really helpful for watermarking implementation.

2.1 Psychology of Hearing [15]

Hearing is the sense by which sound is perceived and hearing of humans is performed primarily by the auditory system: sound as pressure waves is detected by the ear and transduced into nerve impulses that are perceived by the brain. The range of human hearing is 20Hz to 20KHz, a really broad period, where human speech mainly falls between 100Hz and 8KHz.

Without doubt, the ear plays an important role in human auditory system (HAS). Ear is subdivided into outer ear, middle ear and inner ear, as illustrated in Figure 2. The outer ear captures sounds and directs them to the ear canal ending at eardrum. Much of the middle ear’s function is to process sound waves into the vibrations of fluid within the cochlea of the inner ear. Finally, the cochlea transforms sound energy into nerve impulses entering the brain via auditory nerve fibers.

The cochlea is the main organ in the inner ear and actually the core of the ear. It is a snail shaped fluid-filled chamber and separated by two membranes, Reissner’s Membrane and Basilar Membrane, into three scalae, that is, scala vestibule, scala media and scala tympani. The basilar membrane is about 32mm long and the organ of Corti, the receptor organ for hearing, is rested on it. The organ of Corti contains specialized cells called “hair cells”, including the inner and outer hair cells, which translate fluid motion into electrical impulses for the auditory nerve. Experimental studies show that the basilar membrane is a resonant structure that has different
resonant properties at different points along its length, acting as a spectral analyzer. Its motion is like a traveling wave, being the greatest at the point where the frequency of the incoming sound matches that of the movement of the membrane, see samples in Figure 3. In the cochlea, low frequency signals will induce oscillations that reach maximum displacements at the apex of the basilar membrane near the helicotrema, while high frequency signals induce oscillations that reach maximum displacement at the base of the basilar membrane near the oval window, as summarized in Figure 4. In this sense, the cochlea performs a transformation from frequency to space, that is, mapping the frequencies of sound wave onto specific locations of basilar membrane. Consequently, the concept of critical bands could be introduced as representing equal distances along the basilar membrane. Furthermore, it leads to model the auditory system as an array of nonlinear band-pass filters with continuously overlapping pass-bands of bandwidths equal to critical bandwidths as given in Table 1, where the “critical band rate” \( z(f) \) (unit: Bark) can be approximated using 

\[
z(f) = 13 \times \arctan\left(0.76f/1\text{kHz}\right) + 3.5 \times \arctan\left( f/7.5\text{kHz} \right)^2
\]

Figure 3. Relative displacement envelopes of the basilar membrane for several different frequencies from [15] (The left side of the plot is in proximity of the oval window and the right is in proximity of the helicotrema.)

Figure 4. Frequency sensitivity along the basilar membrane from [15]

Such twenty-five critical bands cover frequencies up to 15.5kHz, near the upper limit of human hearing. For the basilar membrane is about 32mm long, each critical band corresponds roughly 1.3mm in basilar distance. Critical bands are closely related to masking phenomenon discussed later.
### Technical Report: Audio Watermarking for Copyright Protection

<table>
<thead>
<tr>
<th>Critical band rate $z$ (Bark) or band number</th>
<th>Lower edge $f_l$ (Hz)</th>
<th>Upper edge $f_u$ (Hz)</th>
<th>Centre frequency $f_c$ (Hz)</th>
<th>Critical bandwidth $\Delta f$ (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>100</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>200</td>
<td>150</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
<td>300</td>
<td>250</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>300</td>
<td>400</td>
<td>350</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>400</td>
<td>510</td>
<td>450</td>
<td>110</td>
</tr>
<tr>
<td>5</td>
<td>510</td>
<td>630</td>
<td>570</td>
<td>120</td>
</tr>
<tr>
<td>6</td>
<td>630</td>
<td>770</td>
<td>700</td>
<td>140</td>
</tr>
<tr>
<td>7</td>
<td>770</td>
<td>920</td>
<td>840</td>
<td>15</td>
</tr>
<tr>
<td>8</td>
<td>920</td>
<td>1080</td>
<td>1000</td>
<td>160</td>
</tr>
<tr>
<td>9</td>
<td>1080</td>
<td>1270</td>
<td>1170</td>
<td>190</td>
</tr>
<tr>
<td>10</td>
<td>1270</td>
<td>1480</td>
<td>1370</td>
<td>210</td>
</tr>
<tr>
<td>11</td>
<td>1480</td>
<td>1720</td>
<td>1600</td>
<td>240</td>
</tr>
<tr>
<td>12</td>
<td>1720</td>
<td>2000</td>
<td>1850</td>
<td>280</td>
</tr>
<tr>
<td>13</td>
<td>2000</td>
<td>2320</td>
<td>2150</td>
<td>320</td>
</tr>
<tr>
<td>14</td>
<td>2320</td>
<td>2700</td>
<td>2500</td>
<td>380</td>
</tr>
<tr>
<td>15</td>
<td>2700</td>
<td>3150</td>
<td>2900</td>
<td>450</td>
</tr>
<tr>
<td>16</td>
<td>3150</td>
<td>3700</td>
<td>3400</td>
<td>550</td>
</tr>
<tr>
<td>17</td>
<td>3700</td>
<td>4400</td>
<td>4000</td>
<td>700</td>
</tr>
<tr>
<td>18</td>
<td>4400</td>
<td>5300</td>
<td>4800</td>
<td>900</td>
</tr>
<tr>
<td>19</td>
<td>5300</td>
<td>6400</td>
<td>5800</td>
<td>1100</td>
</tr>
<tr>
<td>20</td>
<td>6400</td>
<td>7700</td>
<td>7000</td>
<td>1300</td>
</tr>
<tr>
<td>21</td>
<td>7700</td>
<td>9500</td>
<td>8500</td>
<td>1800</td>
</tr>
<tr>
<td>22</td>
<td>9500</td>
<td>12000</td>
<td>10500</td>
<td>2500</td>
</tr>
<tr>
<td>23</td>
<td>12000</td>
<td>15500</td>
<td>13500</td>
<td>3500</td>
</tr>
<tr>
<td>24</td>
<td>15500</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Critical bands of the auditory system from [15]

#### 2.2 Hearing Thresholds and Masking Phenomenon [16]

As discussed above, sound reaches the ear as pressure waves. Due to the wide coverage of human hearing, the range of relevant sound pressure is also broad, from $10^{-5}$ Pascal (Pa) to $10^2$ Pa. Two limits correspond to the absolute threshold of hearing and threshold of pain, respectively. To describe this large scale, we define the sound pressure level (SPL) in logarithm units as

$$\text{SPL} = 10 \log_{10} \left( \frac{p}{p_0} \right)^2 \text{ dB}$$

where the reference $p_0$ is equal to $20 \mu$ Pa. The hearing sensation relative to SPL is loudness, a means to convey perceived sound intensities. The loudness unit is the phon, where the phon describes a curve of equal loudness as a function of frequency [15]. Therefore, different hearing thresholds are obtained in hearing range, illustrated in Figure 5. Note that the threshold in quiet also named as the absolute hearing threshold represents the lowest
sound level that can be heard at a given frequency in noiseless environment. It means that even in extremely quiet conditions, the human ear cannot perceive sounds with SPLs below this threshold [15]. So the threshold in quiet is often used as the baseline when describing the hearing thresholds and it can be approximated by

$$TH_q(f) = 3.64(f/ \text{kHz})^{0.8} - 6.5 \exp[-0.6(f/ \text{kHz} -3.2)^2] + 10^{-3}(f/ \text{kHz})^4$$

The perception of a sound is related to not only its own loudness and spectrum, but also its neighbor components, which is the effect of masking phenomenon. Masking phenomenon means that a faint but audible sound (the maskee) becomes inaudible in the presence of another louder audible sound (the masker). Masking plays an important role in hearing sensation and is divided into two types: simultaneous masking and non-simultaneous masking.

![Figure 5. Hearing area from [15]](image)

**Simultaneous Masking**

Simultaneous masking, also called frequency masking, refers to masking between two sounds with close frequencies, where the low-level sound can be made inaudible by a simultaneously occurring stronger sound, as depicted in Figure 6. Signal $S_0$ is the masker, which produces a masking threshold named for the hearing threshold in the presence of masking. Other signals or frequency components below this curve will be masked by the presence of $S_0$. For example, the weaker signal $S_1$ and $S_2$ are completely inaudible and the signal $S_L$ is partially masked whose perceivable portion lies above the threshold. Hence, the masking threshold is a kind of the limit for just noticeable distortion. The reason for simultaneous masking is the fact that a masker creates an excitation in the cochlea’s basilar membrane that prevents the detection of a weaker sound exciting the basilar membrane in the same area [15].
The masking threshold relies on the characteristics of the maser and the maskee, which could be sinusoidal tone or narrow-band noise. Here, narrow-band means the bandwidth equal to or smaller than a critical band. So there are four cases, narrow-band noise masking tones, tones masking tones, narrow-band noise masking narrow-band noise and tones masking narrow-band noise. It is also worth mentioning that no matter what the type of the masker is, the effective masking ranges for maskers at different frequencies are defined by critical bandwidths. That is to say, if the maskee lies in the critical band of the masker, the maskee is more likely to be imperceptible. It is illustrated with two experimental results in Figure 7. In (a), there is a narrow-band noise at 2kHz centered between two sinusoidal maskers. The masking threshold is flat until the maskers are about 300Hz away from each other, at which point it drops off rapidly. A similar result is seen in (b), a sinusoid at 2kHz centered between two narrow-band noise maskers. Until two maskers are away from 300Hz, the masking threshold is rather flat. Thus, it is confirmed that the critical bandwidth $\Delta f$ only depends on the frequency of the masker $f_c$. The relationship between $\Delta f$ and $f_c$ is described by

$$\frac{\Delta f}{\text{Hz}} = 25 + 75\left[1 + 1.4\left(\frac{f_c}{\text{kHz}}\right)^2\right]^{0.69}$$

whose results correspond to the data in Table 1.

- Narrow-band noise masking tones

Most often, narrow-band noise masking tones happens. The masker is narrow-band noise and the maskees are...
tones within the same critical band.

Examples of the masking thresholds for tones masked by 60dB narrow-band noise maskers centered at 250Hz, 1kHz and 4kHz are given in Figure 8, whose frequency scale is in logarithm. The noise bandwidths are 100Hz, 160Hz and 700Hz respectively. The solid lines represent the masking thresholds, the levels for tones to be just audible, and the dashed line at the bottom is the threshold in quiet. From the graph, we see that the threshold from the lower frequency 250Hz is much broader than the ones from the higher frequencies, 1kHz and 4kHz, which is a general rule. Moreover, the 250Hz masker provides more masking, for the peak of 250Hz threshold is higher than that of the others and its signal to mask ratio (SMR) is the lest. SMR is defined as the difference in level between the signal and the masking threshold at a certain frequency. For the 250Hz, 1kHz and 4kHz thresholds, their SMR are 2dB, 3dB and 5dB respectively. Finally, in logarithmic frequency scale, the 1kHz and 4kHz thresholds are similar in shape and the 250Hz threshold appears to be different from the former two. As mentioned above, Bark scale is the unit for critical bands. So it is straightforward that we describe the masking thresholds in the Bark scale. Figure 9 shows the masking thresholds from narrow-band noise maskers at various frequencies in Bark scale. Note that the masking curves have been normalized to the same level (60dB) that of the masker by the addition of small offsets. As seen, all masking thresholds are similar in shape, regardless of the maskers’ frequencies.

![Figure 8. Masking thresholds for 60dB narrow-band noise maskers centered at 250Hz, 1kHz and 4kHz masking tones from [15]](image)

![Figure 9. Masking thresholds for 60dB narrow-band noise maskers at different frequencies masking tones (in Bark Scale) from [15]](image)

The effect of masking produced by narrow-band maskers also depends on the level of masker and thereby it is nonlinear. In Figure 10, the masking thresholds are from narrow-band noise maskers centered at 1kHz but with different SPLs, $L_{CB}$. The double peaks on the 100dB and 80dB thresholds are caused by non-linear effects in the
hearing, resulting in a second peak that approaches the second harmonic at high SPL. Also, it is obvious that the masking threshold is asymmetry around the masker’s center frequency 1kHz where the maximum locates. At frequencies lower than 1kHz, the curves show a steep rise from lower to higher frequencies. Then beyond 1kHz, the slopes decrease rapidly towards higher frequencies and become shallower with the higher masker levels. Figure 11 is the same to Figure 10, but in Bark scale.

Figure 10. Masking thresholds for a 1kHz narrow-band noise masker from at different levels masking tones from [15]

Figure 11. Masking thresholds for a 1kHz narrow-band noise masker at different levels masking tones (in Bark Scale) from [15]

- **Tones masking tones**

Masking experiments on measurements of pure tones masking pure tones is not easy because of the phenomenon of beating. That is, besides the masker and the maskee, additional tones near the masker’s frequency will be heard. Figure 12 shows the results for a 1kHz tone masker at different levels, obtained by Zwicker and Fastl. To avoid beating occurrence, the maskee is set 90 degrees out of phase with the masker when the maskee comes close to the masker’s frequency, 1kHz. It is seen that at lower SPLs, a greater spreading of the masking thresholds exists towards lower frequencies rather than higher frequencies. On the contrary, the slopes towards higher frequencies become shallower with increasing level of the masker.
Another point to notice is the difference in masking capability of narrow-band noise and pure tone, represented by SMR. In Figure 10, the narrow-band noise acts as the masker and the minimum SMR stays constant around 3dB for all levels. In Figure 12, a tone is used as the masker and the minimum SMRs are roughly 15dB. It means that the masking capability of narrow-band noise and tone is asymmetry and the noise is a better masker than pure tone. Figure 7 also indicates this asymmetry. There, both maskers are presented at a level of 50dB SPL, but noise masker provides 46dB masking threshold versus only 33dB given by tone masker. Therefore, noise or tone masker should be distinguished and their masking thresholds are calculated separately in psychoacoustic modeling.

- **Narrow-band noise or tones masking narrow-band noise**

The discussion on these two situations is very limited. In the case of noise masking noise, phase relationships between the masker and the maskee would largely affect the results. Just for reference, previous data show that the minimum SMRs of wide-band noise masking wide-band noise are about 26dB and those of tones masking narrow-band noise fluctuate between 20dB and 30dB. In general, the noise-like maskers will have a better performance on masking effect than tone-like maskers.

Typically, audio signals may contain several tone-like and noise-like components. It means that we hear their concurrent effects, a global masking threshold, rather than their individual masking thresholds. Therefore, when exploiting the psychoacoustic models, addition of masking curves is necessary.

- **Non-simultaneous Masking**

Non-simultaneous masking, also called temporal masking, may take place when two sounds appear within a small interval of time. There are two forms of temporal masking, pre-masking and post-masking. As their names indicate, pre-masking appears before the onset of the masker and post-masking occurs after the masker is removed [15], shown in Figure 13. We can see that a strong signal can mask a weaker signal that occurs after or before it. The physiological cause for temporal masking is that the auditory system requires a certain integration time to build the perception of sound, and louder sounds need longer integration intervals than softer ones.
Pre-masking
In general, the pre-masking lasts shorter than post-masking. In Figure 13, the duration of pre-masking is about 20ms and actually only effective in the few milliseconds preceding the onset of the 200ms masker. The duration of the masker might affect the duration of pre-masking, but no experimental results could specify the relation. Although just few milliseconds, pre-masking effect is significant in mitigating the pre-noise or pre-echo distortion [15].

Post-masking
Post-masking is the reflection of the gradual decrease of the masking level after the masker halts. As shown in Figure 13, post-masking is more apparent than pre-masking and continues for a longer period of time, about 150ms. Therefore, post-masking is more important in point of view of efficient coding. Similarly, post-masking heavily relies on the duration of the masker. And moreover, it also depends on the masker level, relative frequencies of the masker and the maskee [15].

2.3 Psychoacoustic Model

The perceptual property of HAS is that components below hearing threshold will be imperceptible. Furthermore, due to masking phenomenon, the areas around the maskers on the hearing threshold will be lifted. Therefore, certain noises can be hidden below the masking threshold [18]. In order to analyze the audio signal and compute the amount of masking effect, psychoacoustic model is developed accordingly. At present, psychoacoustic model is often used in audio coding for calculating the signal to masking ratio (SMR), which is the key for bit allocation. In audio watermarking, we must keep the noise produced inaudible to human ears, which is also controlled by psychoacoustic model. Typically, in spread spectrum or wavelet domain watermarking [19], watermark signal is added to the host signal like an additive noise. To prevent distortions being perceivable, amplitude shaping by the minimum masking threshold from psychoacoustic model is often adopted.

Modeling the effects of simultaneous masking

The essence of psychoacoustic model is exploiting the masking phenomenon, mainly simultaneous masking. First, excitation patterns arising from the maskers are derived from the experimental masking curves. Figure 9 and Figure 11 are two examples. Then, different spreading functions can be used to model these excitation patterns. For example, a spreading function is shown in Figure 14, which is adopted in ISO/IEC MPEG Psychoacoustic Model 1. The two-piece linear spreading function for upper and lower frequencies is to mimic
the masking characteristics of tone maskers [15]. Next, the masking thresholds for noise-like and tone-like maskers are given respectively. After that, as a result of adding up the individual masking thresholds, a global masking threshold is obtained. At last, by taking the minimum of the global threshold and the absolute hearing threshold in each subband, we get the minimum masking threshold (MMT).

The implementation of the psychoacoustic model is flexible, depending on the required accuracy and the intended application. Psychoacoustic model I and II in MPEG standard [20] are the ones commonly in use, where typically Model I is applied to MPEG Layers I and II and Model II to Layer III. Here, we just expound psychoacoustic model I that is adopted in our current audio watermarking scheme. The following steps describe the implementation of psychoacoustic model I [20, 21]

**Step1: Calculation of the power spectrum**
Each input frame $x(n)$ with 512 points ($N$) is weighted by hanning-window, $h(n)$.

$$h(n) = \frac{\sqrt{8/3}}{2} \left[ 1 - \cos \left( 2\pi \frac{n}{N} \right) \right]$$

So its power spectrum is computed as

$$X(k) = 10\log_{10} \left\{ \frac{1}{N} \sum_{n=0}^{N-1} x(n)h(n)\exp \left( -j2\pi \frac{nk}{N} \right)^2 \right\}$$

Then, $X(k)$ is normalized to a reference level of 96dB, so that its maximum is equal to 96dB. An example is shown in Figure 15 (a).
Step 2: Identification of tonal and non-tonal maskers

Since non-tonal (noise-like) components are better maskers than tonal (i.e. sinusoidal) components, we need separate tonal maskers from non-tonal ones.

Tonal components are special local maxima of the power spectrum. A local maximum refers to a spectral point satisfying $X(k) > X(k+1)$ and $X(k) \geq X(k-1)$. We take a local maximum as a tonal component if

$$|j| = -2, +2 \quad (2 < k < 63)$$

$$X(k) - X(k+j) \geq 7 \text{dB}, \quad |j| = -3, -2, +2, +3 \quad (63 \leq k < 127)$$

$$|j| = -6, ..., -2, +2, ..., +6 \quad (127 \leq k \leq 250)$$

Then, the SPL of every tonal component is calculated as

$$X_{tm}(k) = 10 \log_{10} \left[ 10^{X(k-1)/10} + 10^{X(k)/10} + 10^{X(k+1)/10} \right] \text{dB}$$

Finally, we put all tonal components into the tonal list, including its index number $k$ and SPL $X_{tm}(k)$.

The non-tonal components are calculated from the remaining components in each critical band, which means that the number of non-tonal components is that of the critical bands. The critical bands for different Layers and sampling frequencies are already tabulated (see Table D.2a, D.2b, D.2c, D.2d, D.2e, D.2f in [20]) and can be looked up directly. In Layer I, for example, 24 critical bands are used for sampling frequency of 44.1kHz. Then, within each critical band, the powers of the remaining components (except tonal component) are summed as the SPL of the non-tonal component in the corresponding critical band. Finally, we also put all non-tonal components into the non-tonal list, including its index number $k$ and SPL $X_{nm}(k)$.

Tonal and non-tonal maskers of the example audio signal are identified in Figure 15 (b).
Step 3: Decimation of the invalid tonal and non-tonal maskers
Firstly, both tonal and non-tonal components below the threshold in quiet are removed. Moreover, tonal components within a distance less than 0.5 Bark are disposed as well. Finally, $N_{tm}$ tonal maskers and $N_{nm}$ non-tonal maskers are preserved in their lists, which are regarded as effective maskers. In Figure 15(b), the components signed with a circle represent the removed ones.

Step 4: Calculation of individual masking thresholds
The individual masking threshold refers to the masking threshold for each masker, indexed by $j$, masking a group of the maskees. The spectral components considered as the maskees, indexed by $i$, are also predefined (see Table D.1a, D.1b, D.1c, D.1d, D.1e, D.1f in [20]). For instance, in Layer I at sampling frequency of 44.1 kHz, the number of the maskees is 106. Then, the individual masking threshold for tonal or non-tonal masker, $LT_{tm}$ or $LT_{nm}$, is computed as follows.

$$ LT_{tm}(z(j), z(i)) = X_{tm}[z(j)] + \text{av}_{tm}[z(j)] + v_f[z(j), z(i)] \text{ dB} $$
$$ LT_{nm}(z(j), z(i)) = X_{nm}[z(j)] + \text{av}_{nm}[z(j)] + v_f[z(j), z(i)] \text{ dB} $$

where $z(j)$ and $z(i)$ are critical band rates in Bark for the maskers and the maskees, respectively. For both tonal and non-tonal maskers, we can get $z(j)$ by mapping its index numbers $k$ into Bark scale. As for the maskees, their $z(i)$ are easy to obtain by lookup table. So $LT_{tm}(z(j), z(i))$ or $LT_{nm}(z(j), z(i))$ represents the value of masking threshold for the maskee at critical band rate $z(i)$ masked by the masker at critical band rate $z(j)$. After all the maskees are covered, each tonal and non-tonal masker has got a masking threshold. The term $X_{tm}[z(j)]$ or $X_{nm}[z(j)]$ is exactly the same to $X_{tm}(k)$ or $X_{nm}(k)$, where $z(j)$ in Bark corresponds to index number $k$. The term $\text{av}$ is called the masking index, the offset between the excitation pattern and the actual masking threshold. For tonal and non-tonal masker has different masking effect, $\text{av}_{tm}$ and $\text{av}_{nm}$ (dB) are defined separately.

$$ \text{av}_{tm} = -1.525 - 0.275 \ast z(j) - 4.5 $$
$$ \text{av}_{nm} = -1.525 - 0.175 \ast z(j) - 0.5 $$

The term $v_f$ (dB) is the masking function of a masker, characterized by different lower and upper slopes. It is related to the distance between the masker and the maskee, $d_z = z(j) - z(i)$.
\[ \begin{align*}
\eta_f &= 17 \cdot (dz + 1) - (0.4 \cdot X[z(j)] + 6) \quad \text{for } -3 \leq dz < -1 \text{ Bark} \\
\eta_f &= (0.4 \cdot X[z(j)] + 6) \cdot dz \quad \text{for } -1 \leq dz < 0 \text{ Bark} \\
\eta_f &= -17 \cdot dz \quad \text{for } 0 \leq dz < 1 \text{ Bark} \\
\eta_f &= -(dz - 1) \cdot (17 - 0.15 \cdot X[z(j)]) - 17 \quad \text{for } 1 \leq dz < 8 \text{ Bark}
\end{align*} \]

where \( X[z(j)] \) is \( X_{\text{tm}}[z(j)] \) or \( X_{\text{nt}}[z(j)] \) according to the kind of masker. When \( dz < -3 \text{ Bark} \) or \( dz \geq 8 \text{ Bark} \), the masking is no longer considered. So we set \( LT_{\text{tm}}[z(j), z(i)] \) and \( LT_{\text{nt}}[z(j), z(i)] \) to \(-\infty \text{ dB})\.

Figure 15(c) shows all individual masking thresholds for both tonal and non-tonal maskers.

**Step 5: Calculation of global masking threshold**

The ‘addition’ of all individual masking thresholds and the threshold in quiet \( LT_q \) is the global masking threshold. The global masking threshold at the \( i^{th} \) spectral component \( LT_g(i) \) is calculated as follows.

\[
LT_g(i) = 10 \cdot \log_{10} \left[ 10^{LT_q(i)/10} + \sum_{j=1}^{N_{\text{tm}}} 10^{LT_{\text{tm}}[z(j), z(i)]/10} \sum_{j=1}^{N_{\text{nt}}} 10^{LT_{\text{nt}}[z(j), z(i)]/10} \right]
\]

Actually for each \( i \), the range of \( j \) can be reduced to just encompass the masking components within -8 to +3 Bark from \( i \), and the others are needless to consider.

**Step 6: Determination of the minimum masking threshold**

In each subband that of equal bandwidth, the minimum of the global masking level is selected as the masking level of this subband \( LT_{\text{min}}(n) \). Furthermore, we spread each \( LT_{\text{min}}(n) \) over the corresponding subband and get the minimum masking threshold. As discussed above, any signal components below the minimum masking threshold are perceptually irrelevant.

In Figure 15(d), the power spectrum along with the global masking threshold and the minimum masking threshold is shown.
Figure 15 (d). Minimum masking threshold

◆ Modeling the effects of non-simultaneous masking

In addition to simultaneous or temporal masking, perceptual models exploit the effects of non-simultaneous masking as well. In [15], modeling temporal masking takes into consideration a time-sliding window, which has a larger weight to components near the center of the window as opposed to components near the edges. The shape of weighting function is designed to resemble the experimental data in Figure 13. Generally, it is assumed that the temporal smoothing occurs after the auditory filtering, i.e. applied to the signal spectrum, so that the output signal is smoothed. Differently, [22] approximates temporal masking effects using the modified envelope of the host audio, which increases with the signal and decays as exponential function $e^{-at}$. An example is shown in Figure 16.

Figure 16. Modeling the effects of temporal masking from [22]

3. REQUIREMENTS OF AUDIO WATERMARKING FOR COPYRIGHT PROTECTION

To function as an effective tool for copyright protection, the watermark embedded in the audio media must be
imperceptible, robust against attacks and self-secured. Moreover, capacity of embedding and speed of watermarking are also worthy of notice. In addition to these main requirements, the watermark also should be spread throughout the entire host audio, not just embedded in a header, to prevent deliberate removal. Finally, a powerful audio watermarking system should support multiple watermarking for multi authors and authorized distributors [23]. Note that trade-offs always exist between the requirements and the key of designing any audio watermarking scheme is to keep the balance.

3.1 Imperceptibility

It is obvious that watermarking must not degrade the perceptual quality of host audio signal. In other words, the watermark should be inaudible, imperceptible or transparent. Otherwise it is meaningless and the usage of watermarking is unacceptable at all. To evaluate whether a watermarked signal still keeps the original quality, subjective quality tests are carried out. The followings are two popular ones.

- **ABX Test** [24]

  A subjective assessment called ABX test is conducted in STEP 2000, which includes two steps.
  1. The listener listens to a sound recording with no watermark (A), a sound recording with watermarks embedded (B), and a sound recording which is one of the two (X).
  2. The listener listens to A and B alternately twice for 40 seconds each, and then listens to X for 40 seconds. The listener decides whether X is A or B.

Also, there are two requirements in ABX test to ensure its validity.

1. Elimination of (correct) contingency responses.
   a) In order to eliminate (correct) contingency responses, the above tests were conducted five times for each participating system.
   b) The listener is defined to have detected the embedded watermark if the same listener correctly determines whether the watermark is embedded or not on each of the five tests. Under this definition, significance of the responses are 95% or greater.

2. Four individuals each from the following occupation categories were selected as Golden Ears and Silver Ears from the recording industry.
   1) Recording engineers
   2) Mastering engineers
   3) Synthesizer manipulators
   4) Audio critics

- **Subjective Difference Grade (SDG)**

  A five-grade impairment scale is defined in ITU-R BS. 1116 (see Table 2), which rates the degradation of the test signal with respect to the reference. Difference Grade is equal to the subtraction between the subjective ratings given separately to the watermarked signal and the host signal. That is to say, a Diffgrade near 0.0 indicates a high quality of the watermarked signal, undistinguishable from the host signal. On the contrary, a Diffgrade near -4.0 means that the watermarked signal is far from the original version. By training a neural network, Perceptual Evaluation of Audio Quality (PEAQ) [25] could provide an ‘objective’ quality measure.
### Difference Grade | Description of Impairments
--- | ---
0 | Imperceptible
-1 | Perceptible but not annoying
-2 | Slightly annoying
-3 | Annoying
-4 | Very annoying

Table 6: Subjective Difference Grade (SDG) [25]

In addition to subjective quality tests, segmental signal to noise ratio (seg\_SNR) between the host signal and the watermarked signal is often used for estimating the quality of watermarked signal. To calculate segmental SNR, the watermarked signal is firstly divided into $M$ segments, each with $N$ samples. Then seg\_SNR is computed according to the following formula

$$\text{seg\_SNR} = 10 \cdot \log_{10} \left( \frac{1}{M} \sum_{m=1}^{M} \left( \frac{\sum_{n=1}^{N} [x_m(n)]^2}{\sum_{n=1}^{N} [x_m(n) - s_m(n)]^2} \right) \right)$$

where $x(n)$ and $s(n)$ are the host signal and the watermarked signal respectively. Moreover, $\Sigma [x_m^2(n)]$ is the power of host signal and $\Sigma [x_m(n) - s_m(n)]^2$ is the difference that is regarded as the noise power. Segmental SNR is an objective measure for quality test. However, it would not be valid under some specific situations, such as in phase coding watermarking scheme. In that case, the waveform of watermarked signal changes a lot due to phase alteration, so that segmental SNR would probably be lower than usual. But actually the perceptual difference caused by watermarking is still tolerable. That is to say, segmental SNR is invalid for phase coding watermarking scheme.

It is worth mentioning that the extent of the impairments depends on the intended audience and their surrounding conditions. For example, the timbre what a musician requires is different from what an ordinary person needs. Also there is a discrepancy in the same sound playing in a quiet studio or in the noisy environment. It seems that perceptibility, mostly, is a relatively subjective opinion. However, note that segmental SNR could never replace subjective quality tests, since human ear is decisive to verify the perceptibility. Moreover, sometimes although segmental SNR is high, the perceptual quality of the watermarked signal is not good. Therefore, in our experiment, both subjective quality test and segmental SNR are adopted to judge the quality of the watermarked signal. In subjective quality test, some general students and audio engineers are involved.

#### 3.2 Robustness

As a reliable tool for copyright protection, audio watermarking scheme should be able to ensure the detection accuracy when identifying products’ ownership. But in real applications, the watermarked data are often processed in some way during the transmission and communication [26], i.e. being processed by common signal operations or attacked by malicious attempts to remove the watermark. Then, the received file used for detection is different from the initial watermarked signal, which brings trouble to watermark extraction. Therefore, any practical watermarking scheme must be immune to modifications at a certain extent, which is a major criterion for watermarking schemes evaluation.
In our experiment, a robustness test is always carried out to assess the property of robustness for every audio watermarking technique. That is, various attacks are applied on the initial watermarking signal and then we perform watermark detection to check whether the watermark survives or not. It involves all the attacks that frequently used in robustness evaluation, for example, noise addition, resampling, low-pass filtering, echo addition, data compression, random samples cropping and zeros inserting, jittering, pitch-invariant time stretching, tempo-preserved pitch shifting, and multiple attack, combination of two or more single attacks.

For denoting the detection rate, the Bit Error Rate (BER) or the Ratio of Correct Bits Recovered (ROCBR) and the Ratio of Correct Letters Recovered (ROCLR) between the extracted watermark and the initial one are adopted. Note that ROCBR is complementary to BER, BER + ROCBR = 1. Here, terms ‘bit’ and ‘letter’ should be distinguished. A bit, ‘0’ or ‘1’, refers to the element of the embedding signal that is a vector, and a letter is one unit of the watermark. A letter could be one or two-dimensional array with bit ‘0’ or ‘1’. For example, the watermark to be embedded is a coded image ROBUST, which has six letters, R, O, B, U, S and T. Every letter is comprised of a 7 by 5 matrix with bit ‘0’ and ‘1’, like

\[
\begin{align*}
R &= [1 \ 1 \ 1 \ 1 \ 0; \\
&\quad 1 \ 0 \ 0 \ 1; \\
O &= [0 \ 1 \ 1 \ 1 \ 0; \\
&\quad 0 \ 0 \ 0 \ 1; \\
B &= [1 \ 1 \ 1 \ 1 \ 0; \\
&\quad 1 \ 0 \ 0 \ 0; \\
U &= [1 \ 0 \ 0 \ 0; \\
&\quad 0 \ 1 \ 1 \ 1; \\
S &= [0 \ 1 \ 1 \ 1; \\
&\quad 0 \ 0 \ 1 \ 1; \\
T &= [1 \ 1 \ 1 \ 1; \\
&\quad 0 \ 1 \ 0 \ 0; \\
&\quad 1 \ 0 \ 0 \ 0; \\
&\quad 0 \ 0 \ 1 \ 1; \\
&\quad 1 \ 0 \ 0 \ 0; \\
&\quad 0 \ 0 \ 1 \ 1; \\
&\quad 1 \ 0 \ 0 \ 0; \\
&\quad 0 \ 0 \ 1 \ 1; \\
&\quad 1 \ 0 \ 0 \ 0; \\
&\quad 0 \ 0 \ 1 \ 1];
\end{align*}
\]

In practice, letters are separated from each other by one spare column for easy identification. If the watermark is a sequence of pseudorandom numbers, like \(W(n)=[w_1, w_2, \ldots, w_n], w_n \in \text{to.}\) After triple repetitive-coding, each letter \(w_i\) will be represented by three bits with the same value. For instance, the watermark \(W(n)\) has two letters [0 1]. After tri-repetitive coding, the embedding message becomes [0 0 0 1 1 1] with six bits.

In addition, the similarity (\(sim\)) between the original watermark \(I_o\) and the extracted watermark \(I_e\) is also a meaningful factor for estimating detection rate.

\[
\text{sim}(I_o, I_e) = \frac{\langle I_o \cdot I_e \rangle}{\sqrt{\langle I_o \rangle \cdot \langle I_e \rangle}} = \frac{\sum_{i=1}^{p} I_o(i) \cdot I_e(i)}{\sqrt{\sum_{i=1}^{p} I_o^2(i) \cdot \sum_{i=1}^{p} I_e^2(i)}}
\]

It is more applicable to a watermark being a coded image, where \(P\) is the number of pixels (bits) of the coded image. The closer to 1 \(sim\) is, the more does extracted watermark resemble the original. For post-process like image enhancement may be applied on the extracted watermark, we use \(I'_e\) to represent the new version. Correspondingly, \(\text{sim}'\) is the similarity between \(I_o\) and \(I'_e\).
3.3 Security

The property of security is of great importance to watermarking schemes. It implies that the watermark should be impossible to be detected even though the algorithm is totally open to the public [27, 28]. Otherwise, the attackers could detect the watermark by reversing the embedding process or performing statistical detection. Then, they are able to modify the watermark without much impairment on the sound quality. For instance, in echo hiding watermarking schemes, decision on every bit depends on the peak position in cepstrum coefficients of each segment. Even $N_{\text{frame}}$ is unknown, the attackers could make a certain amount of trials with different values of $N_{\text{frame}}$. Then those peaks are found out through statistical data without prior knowledge. That is to say, the delays that decide the bits are unsecured under unauthorized detection. As a result, the original peaks may be eliminated and false peaks are added, which have catastrophic effect on the watermark detection. In the same way, cepstrum watermarking may suffer from the malicious intrusion as well.

Different solutions have been tried, like a scrambling all-pass filter in [12] and encryption/decryption on the coded-image in [13]. However, these operations are insufficient and cannot secure the watermarking systems. Suppose that an attacker has modified an encrypted watermark, it will definitely lead to the failure of decryption. So the coded-image cannot be recovered any more. In order to overcome such shortcomings, secret keys and/or scrambling should exist inside the embedding process to add randomness, so that the scheme is self-secured.

3.4 Capacity

The property of capacity describes the quantity of bits embedded per time interval. Usually, it is defined as the number of bits embedded in one-second fraction (bps). The theoretical capacity of an audio watermarking scheme relies on the proportion of silence to the host signal and the embedding algorithm. However, due to trade-off between capacity and robustness, it is always quite hard to compare the real capacities of two watermarking schemes. Therefore, the capacity is less important at present and our evaluations just give roughly estimation for each watermarking scheme.

3.5 Speed

Speed is important in practical, for real-time and low-delay processing is hot currently. The complexity of each watermarking scheme determines the speed. The most intuitive way to compare speed is to provide embedding and detection time separately in relation to the duration of the host audio file. As for a commercial product, we don’t care too much about the embedding time, as long as it is not weird. But the detecting stage is expected to spend time as less as possible. In addition, all the measurements should be performed on the platforms with the same computational capabilities, otherwise the results are not comparable.
4 TESTING ITEMS IN MEASURING ROBUSTNESS

As mentioned above, robustness test on the watermarked signal is important to claim that an audio watermarking scheme is practical. It is indispensable to watermarking schemes evaluation. A robustness test includes four steps. At first, apply attack on the watermarked signal and get the corrupted signal, i.e. the watermarked signal being attacked. For both common signal processing operations and hostile offences will introduce certain modifications, all of them are regarded as ‘attacks’. After that, perform detection process and extract the watermark from the corrupted signal. So BER or ROCBR and/or ROCLR can be calculated. If ROCBR is high enough, around 80%, the watermarking system is deemed to be robust against the tested attack. Next, repeat the former two steps by applying another attack on the watermarked signal until all the attacks are checked over. Finally, give a broad overview of the tested attacks and decide whether the watermarking system is robust or not.

A competent robustness test should cover a variety of attacks as many as possible. However, the extent of deterioration on the watermarked signal should keep with one fundamental premise. That is, the attack must be imperceptible to the audio file as well. If the corrupted signal sound far from the watermarked signal, such as with annoying noise, or discrepant timbres, then it is needless to execute watermark detection. The reason is that the corrupted signal is already ‘severely’ destroyed and not regarded as the counterpart of the host signal any more. So it is meaningless and not worthy of consideration. Thus, implementation of any attack should not exceed the bounds of perception, as the amplitude of noise and the rate of stretching are limited. For reference, three popular audio watermarking benchmarkings are listed in Appendix A, B, and C respectively, that is, STEP2000, SDMI and StirMark for Audio. All the robustness test items in each benchmarking can be found.

As we know, synchronization is a vital issue in communication systems. Similarly, loss of synchronization in time or frequency domain would result in failing to find the proper positions for detecting watermark bits in corrupted signal, which means that the whole watermark detection will be in a mess. Therefore, whether an attack will effect synchronization information or not is the principle to classify the possible attacks into two categories [13], summarized in Figure 17. On one hand, Type I attacks are called mis-synchronization attacks, such as noise addition, resampling, low-pass filtering, echo addition, and data compression. They are mainly
normal signal processing operations. On the other hand, Type II attacks are called synchronization attacks, for example, jittering, random samples cropping and zeros inserting, pitch-invariant time stretching and tempo-preserved pitch shifting. Those intentional tamperings are easy to get via some software tools. Without degrading the perceptual quality of audio files, they will heavily destroy the structure for synchronization.

If the mechanism of the attack is known, we are more capable of detecting watermark bits from the corrupted signal, probably just some alternations of detection parameters. In the following, the implementations of these attacks are stated one by one. Also, to be consistent in the expression, we define the attack representation for each tested attack.

4.1 Noise addition

Signal is often polluted by the noise during the transmission. To evaluate for robustness against noise addition, white Gaussian noise and normal distributed noise are applied separately to the watermarked signal. As determined by the criterion in the benchmarking, firstly, white Gaussian noise is added to the targeted signal and signal-to-noise ratio ($SNR$) will be specified as 40dB or 30dB. Then secondly, normal random numbers with zero mean and unit variance are added to the stamped signal and remember that these random number need to be scaled to get the proper amplitude. Due to waveform normalization in Matlab, $10^{-3}$ could be used as the scale factor, so that $SNR$ will reach 40dB. In reality, noisy signal of this level sizzles clearly. Figure 18 illustrates the corrupted signals by separately adding two kinds of noise to the watermarked signal.

![Figure 18. Effect of noise addition ($SNR=40$dB)](image)

When citing noise addition in the robustness test, the representation is like “Noise (snr)”, where the parameter ‘snr’ is value of SNR.

4.2 Resampling

Sampling rate, $F_s$, is an important parameter for digital signal. In our experiment, all the audio files are sampled at 44.1 kHz and quantized in 16 bits, with CD quality. So the sampling frequency of the watermarked signals is also equal to 44.1kHz. To examine the property of resisting resampling, the watermarked signal is first
downsampled to the sampling rate of 22.05 kHz or 11.025 kHz, then upsampled back to 44.1 kHz again. Here, the ‘down’ and ‘up’ operations are done by ‘Streambox Ripper’ tool. Note that owing to alteration of sampling rate, some samples in the end of the processed signal are invalid and the length will change slightly. Hence, when making comparison and calculating SNR, only effective and corresponding parts of the original and processed signal are taken into account. As showed in Figure 19, the resampling process will cause noticeable distortion in relation to the original signal.

![Figure 19. Effect of resampling (44.1kHz→22.05kHz→44.1kHz)](image)

When citing resampling in the robustness test, the representation is like “Resampling (fs1, fs2)”, where fs1 is the sampling frequency of the host signal and fs2 is the downsampling frequency. The default setting is “Resampling (44.1kHz, 22.05kHz)”.

### 4.3 Lowpass Filtering

With watermarks embedded in the frequency domain, low-pass filtering could effectively eliminate the embedded watermarks. Definitely, low-pass filtering attracts more attention than high-pass filtering, because the low frequency components of audio signals are much more perceptually significant. Although music has a wide dynamic range in frequency from 20Hz to about 20kHz, low frequency period contains most influential information. For example, human speech typically falls into the frequency range comprised between 100 Hz and 8 kHz. Thus, high frequency part probably could be filtered out, rather than the low frequency part. In our evaluation, low-pass filtering with a low cutoff frequency, such as 0.2 π , 0.3 π and 0.4 π (normalized frequency), will be applied to the watermarked signal. As usual, filtering on the sound is hoped to be inaudible. However, in the case of 0.2 π and 0.3 π , the audio often sounds dull due to much loss of high-frequency components. Therefore, 0.4 π is more reasonable. As for sampling rate of 44.1kHz, it equals to 0.4*44100/2 = 8820Hz.

Figure 20(a) gives an example for a 25th-order FIR lowpass equiripple filter, designed directly by Filter Design & Analysis Tool in Matlab. Its parameters include passband edge frequency ωp=0.4 π , stopband edge frequency ωs=0.5 π , passband ripple Rp=3dB and stopband attenuation Rs=40dB. Also, the outcome of a watermarked signal filtered by this lowpass filter is shown in Figure 20(b).
4.4 Echo Addition

‘Echo’ is, quite simply, the delayed version of original signal. In order to be inaudible when inserting an echo into the original signal, the echo should be designed with the proper amplitude and the time offset. If the offset is zero, two signals superpose each other completely and cannot be separated. With the offset increasing to a certain point \( d \), human ear can distinguish the original signal from the echo clearly. Between \((0,d)\), the echo is perceived as resonance to the host sound and does not produce uncomfortable noise [6]. However, this critical point is hard to determine exactly. It relies on the quality of the original audio file, the type of sound being echoed, and the listener. From the experience, this fusion occurs generally around 1/50 of a second (20ms) for most sounds and most listeners. So when sampling rate is 44.1kHz, 20ms covers 880 samples.

In the beginning, actually, echo addition is particularly introduced to attack echo hiding scheme. Now it is applied to all systems to be evaluated. Here, an echo with a delay of 200ms and an amplitude attenuation of 0.3 is added to the original audio signal. In this case, the echo can be clearly perceived by listeners and becomes undesired. Figure 21 exhibits the original signal and the echoed signal.
When citing echo hiding in the robustness test, the representation is like “Echo ($A_m$–$t_{delay}$)”, where $A_m$ and $t_{delay}$ is amplitude attenuation and delay time respectively. The default setting is “Echo (0.3, 100ms)”.

4.5 Data Compression

The capability against data compression of an audio watermarking scheme is tested using MP3 compression, implemented via ‘Streambox Ripper’. MP3 stands for ‘MPEG-1 Layer III’ and belongs to MPEG-1 compression standard among the MPEG (Moving Pictures Experts Group) Audio coding family. MP3 compression is a kind of lossy compression, removing perceptually irrelevant parts of the audio signal according to the perceptual properties of the human auditory system [29]. Therefore a real challenge is presented to the watermark detection, which was a major evaluation criterion for audio watermarking scheme in the past.

In our experiment, the watermarked audio file is firstly compressed by MP3 encoder at a bit rate $m$ kbps, where $m$ could be equal to 64, 96 and 128, etc. So the original watermarked signal $S(t)$, mostly in .wav format, is attacked and transformed to .mp3 file, $S_m(t)$. After that, MP3 decoder decompresses $S_m(t)$ to $C(t)$ (.wav file) again, in order to analyze watermark detection in Matlab. The whole procedure is illustrated clearly in Figure 22.

*Figure 22. Procedure of testing robustness against MP3 compression*

It won’t be taken for granted that these two conversions are exactly mutually inverse, like FFT and Inverse FFT, therefore $S_w(t)$ and $C(t)$ are likely to be the same. Figure 23 reveals the difference between $S_w(t)$ and $C(t)$, where a piece of watermarked signal is taken as $S_w(t)$ and bit rate $m$ equals to 96 kbps. And for being easily recognized, only a period with length of 1024 points is displayed. Also segmental SNR is calculated to show how much difference in them, seg. SNR = 25.3092dB.

*Figure 23. Effect of MP3 compression (m = 96kbps)*
However, one more thing must be mentioned that $C(t)$ in Figure 23 has already undertaken a preprocessing, not the output directly from Streambox Ripper. Because a certain amount of zeros will be added to the beginning of the processed audio file during compression/decompression, and at the same time, some points at the end of file will be discarded, like back-shifting slightly. It is similar to zeros inserting (introduced next). This happens for every MP3 encoder due to internal data organization in MP3 files, such as head frame, and points padded for converting bit rate from kbps to bps. For our case, there are exactly 1201 zeros inserted when the input audio file has a sampling frequency of 44.1kHz. in addition, $C(t)$ is shorter than $S_w(t)$. Here, in order to emphasize the effect of amplitude modification by MP3 compression and distinguish from mis-synchronization attack, we cut those zeros off and take the rest of sequence as $C(t)$. Hereunder it is referred to as MP3 Compression I. The steps for calculating the quantity of shifting points (1201 points) are as follows.

1. Pad $C(t)$ with trailing zeros to the length of $S_w(t)$, $N$.
2. Calculate the cross-correlation function of $S_w(t)$ and $C(t)$, $R_{ww}(t)$.
3. Find the location for the maximum value in $R_{ww}(t)$, $L_{max}$.
4. Subtract $L_{max}$ from $N$, so that get the amount of shifting points, $SP$.

Actually in practical robustness test, we neglect this preprocess and perform stamp detection on $C(t)$ directly. It is called MP3 Compression II to distinguish from the former. Since audio watermarking scheme should be able to resist multiple attacks. MP3 Compression II happens to be a dual-attack, combination of data compression and zeros inserting.

When citing data compression in the robustness test, the representation is like “Compression I (m)” or “Compression II (m)”, where m is compression bit rate. The default value for m is 96kps.

4.6 Random Samples Cropping and Zeros Inserting

Both random samples cropping and zeros inserting belong to geometric distortions, producing disastrous synchronization problem in a straightforward way.

Random samples cropping refers to deleting some samples at some randomly selected locations, at the beginning, somewhere in the middle, or in the end. Perception of cropping depends not only the amount of samples removed, but also the amplitude of that clipping. Roughly speaking, one cropping less than 10 samples would not give rise to obvious discontinuity under the normal volume, otherwise the click sound being sharp or flat can be heard. For our robustness test, a large number of samples, 500 and 1000, at a certain position will be cropping. Besides single cropping, double cropping which means cropping at two places simultaneously is also employed in test. Figure 24 illustrates single cropping, cutting 1000 samples off from the original signal.

On the contrary, samples inserting is not deleting the samples, but adding samples especially zeros to the stamped signal. So we call it zeros inserting, like placing some silence intervals in the audio file. Note that long muteness is not imperceptible and will result in a brief break during the smooth sound. As a rule, silence part longer than 20ms under the normal volume definitely has an effect on perception. When the sampling frequency is 44100Hz, the period of 20ms consists of 1000 zeros or so. Therefore, in the robustness test, inserting silence part with 2000 zeros is applied to attack the watermarked signal (see Figure 25). Similar to cropping, double inserting is adopted in our test as well.
Also, samples cropping and zeros inserting could be combined, that is, replacing some original samples with certain amount of zeros. Moreover, the length of samples substituted may be different from that of silence period. For example, 100 original samples are replaced with 120 zeros, depicted in Figure 26. In practical test, 1000 samples are cropped and then 1200 zeros are inserted at that position.

When citing random samples cropping and zeros inserting in the robustness test, the representations are like “Cropping (n₁)” and “Inserting (n₂)” where n₁ and n₂ are the number of samples cropped or inserting. As for their combination, the representation is simply defined as “C&I (n₁, n₂)”.

Figure 24. Random samples cropping (1000 samples)

Figure 25. Zeros inserting (200 zeros)
4.7 Jittering

Jittering is an evenly performed case of random samples cropping, which means randomly cropping $A$ samples out of every $B$ samples. As a continuous case of random samples cropping, jittering will lead to loss of synchronization sign too. In our survival test, 5 samples are cleared up out of every 20000 samples. Actually, series of clicks are distinctly audible under such situation.

When citing jittering in the robustness test, the representation is like “Jittering ($A/B$)”. The default setting is “Jittering (5/20000)”.  

4.8 Pitch-invariant Time Stretching

Pitch-invariant time stretching (PITS), also known as Time Compression/Expansion, is a kind of process of timescale modification, but preserving the pitch of audio file. In other words, the rate of speed at which a piece of music is played is changed, whereas its timbre remains. Obviously, if the timescale of the watermarked signal is altered, synchronization required in detection is destroyed. Since the intervals used for embedding watermarks, as well as watermarks themselves, will experience a certain distortion in time domain after Time Stretching. Moreover, the spectrum of the signal often needs to be modified so as to implement PITS. Hence, characteristics in both time and frequency domain vary simultaneously, which probably lead to the failure of watermark detection.

In audio watermarking benchmarks, usually time-stretching up to ±10% is adopted as the criterion. Figure 27 gives an example for a watermarked signal after 110% and 90% time stretching respectively, implemented via a powerful audio editing tool - ‘SoundForge 5.0’. For the watermark is approximately spread over the whole signal, the entire body is just the period of embedding watermark. In order to compare clearly, the lengths of three files are indicated in the graph, 1017598, 916967, and 1119264 points severally. Thus some deformations have occurred and mis-synchronization trouble is arisen.
In addition, there are two more points worth consideration. One is that although played in different duration, three pieces of music sound still in the same tone. The other is that the envelopes of signals after time stretching seem to be similar to that of original signal being compressed or expanded. However, they are not exactly the same. Therefore, Pitch-invariant time stretching cannot be performed using linear interpolation as the following Matlab codes, otherwise the pitch of signal will be changed at the same time when the timebase is recomposed. It is different from the way proposed in [30], where pitch shifting is performed using linear interpolation without anti-alias filtering.

% Matlab codes for implementation of time stretching by linear interpolation
% 110% and 90% time stretching
stretch_factor = 1.1 ;               % or 0.9
time_factor = 1/stretch_factor ;      % for new index
original_len = length(wmed_signal);  % length of the unattacked watermarked signal
original_vector = 1 : original_len;    % old coordinates
new_vector = time_factor : time_factor : original_len;  % new coordinates
% Linear interpolation
ts_wmed_signal = interp1(original_vector, wmed_signal, new_vector, 'linear') ;

The results obtained are showed in Figure 28. Compare Figure 28 to Figure 27, it can be seen that the waveforms of one pair resemble each other. A pair refers to twin signals with the same stretching factor. But in fact, (b) signal shown in Figure 28 sounds higher than original signal and (c) signal in Figure 28 sounds rather lower. So it is different from the proper way what we expect.

In practice, the Phase Vocoder is a popular method for Time Stretching. This approach is derived by Flanagan and Golden in 1966 and mainly utilizes Short Time Fourier Transform (STFT) to transform the signal between time and frequency domain at present. The idea is to reconstruct a spectrogram with the new timebase but the original phase relation. Accordingly, the temporal features of a sound are modified while its short-time spectra

Figure 27. Effect of pitch-invariant time stretching (±10%)
remain. If the analysis window in STFT is suitable, a bit longer than pitch cycle, it means the pitch of narrowband spectrum is preserved.

The following steps illustrate how a Phase Vocoder works [31].

1. Define parameters, such as size of frame $N$, stretch_factor and hop. stretch_factor is time-stretching factor, expressed as a rational number, and hop is the amount for overlap-and-add in STFT. When stretch_factor is less than 1, the duration becomes shorter, implying speeding up. On the contrary, if stretch_factor is larger than 1, it means slowing down.

2. Calculate the spectrogram of the original signal via STFT.

3. Establish a new timebase by sampling the original time array at a sequence of fractional value according to stretch_factor.

4. Modify the spectrogram based on the new timebase, including interpolating its magnitudes and fixing up the phases as before.

5. Transform back to time domain through inverse STFT.

The graphs in Figure 29 exemplify the outputs of a Phase Vocoder, acting as time stretcher.
Another time-stretching method is called time domain harmonic scaling (TDHS), working on different principle from Phase Vocoder. The two keys to this algorithm are estimating the correct fundamental frequencies of the signal and building the timebase by copying the input to the output in an overlap-and-add manner. In addition, more approaches, improved Phase Vocoder, adaptive basis transform algorithms, wavelet and multi-resolution techniques etc, are stated in details in [32].

Finally, for PITS changes the length of the watermarked signal greatly, an alternative detection is based on varying the frame size according to the time-stretching factor. That is, $N'_{frame} = \text{fix}(N_{frame} \cdot (1 \pm \text{stretch\_factor}))$. It is acceptable, since the length of host signal same to that of the watermarked signal can be transmitted as side information as well. Then the stretch\_factor is known.

When citing pitch-invariant time stretching in the robustness test, the representation is like “PITS (stretch\_factor)”, where 'stretch\_factor' is time-stretching factor in percentage. In our robustness test, the default setting is “PITS (±4%)”. Because the distortions brought by stretching exceeding ±4% have already audible. Moreover, the symbol $N_{frame}$ in PITS represents for using the original size of frame in the detection, and $N'_{frame}$ for the changeable case.

4.9 Tempo-preserved Pitch Shifting

Just as its name implies, tempo-preserved pitch shifting (TPPS) is opposite to pitch-invariant time stretching. Here the tempo is reserved, nevertheless the pitch will be upward or downward shifted.

There are five issues about tempo-preserved pitch shifting. Firstly, the pitch is related to the frequency of the fundamental sound wave. That is, the higher the frequency of a sound wave, the higher its pitch sounds. Accordingly, a change in pitch implies as change in frequency, which means frequency fluctuation occurs. It is fatal to synchronization, especially the watermarking system operating in the frequency domain. Secondly, pitch shifting is not linear for all frequency bins. In other words, lower frequency parts will not shift as far as higher frequency parts, illustrated in Figure 30. The underlying principal is that human pitch perception is logarithmic

![Figure 30. Nonlinearity in pitch shifting](image-url)
with respect to fundamental frequency, so that pitches are often represented in a linear scale based on logarithm of fundamental frequency. For instance, in MIDI standard, fundamental frequency $f$ is mapped to a real number $p$ as follows

$$ p = 69 + 12\log_2\left(\frac{f}{440}\right) $$

where 440 comes from the notion that the note A above middle C played on any instrument is perceived to be of the same pitch as a pure tone of 440Hz. Note that the perceived distance between the pitches “A220” and “A440” is the same as the perceived distance between the pitches “A440” and “A880”. Thirdly, in order to recognize the pitches of a sound and analyze pitch shifting, the maximum of autocorrelation of every short-time segment could be taken as one pitch of the signal. However, it is useful only in some simple cases, such as pure tone. As for complicated ones, a special toolbox ‘MAD’ is helpful. Fourthly, a slight change in frequency need not lead to a perceived change in pitch. Considering that the threshold at which a change in pitch is perceived, pitch shifting up to $\pm10\%$ is adopted in standards. Here, Cool Edit Pro 2.0, another audio editing tool commonly used, is taken for executing TPPS, showed in Figure 31. In fact, $\pm10\%$ shifting has already greatly exceeded the just noticeable difference of perception of pitch, so actually up to $\pm4\%$ is more appropriate in robustness test. Finally, although the pitch is just the focus of attention, the timebase of watermarked signal after pitch shifting will be adjusted slightly. So zeros will be padded at the rear of file so as to keep the original length. For example, it occurs when using SoundForge 5.0. Similarly, this could be taken as a dual-attack as well.

![Figure 31. Tempo-preserved pitch shifting](image)

Pitch shifting can be realized by some other approaches. Generally speaking, pitch shifting is the reciprocal process to time stretching, so pitch shifting could be achieved by altering the sample rate of the signal after time stretching. For instance, when Modified Phase Vocoder mentioned before is used for pitch shifting (referring to Figure 32), it belongs to frequency domain pitch correction [33]. Also, Pitch-Synchronous Overlap-Add (PSOLA) and Time Shifting complete pitch correction in time domain [34].

When citing tempo-preserved pitch shifting in the robustness test, the representation is like “TPPS (shifting_factor)”, where ‘shifting_factor’ is pitch-shifting factor in percentage. In our robustness test, the default setting is “TPPS ($\pm4\%$)”.  

32
5 AUDIO WATERMARKING TECHNIQUES

Currently, a variety of audio watermarking techniques have been proposed. Every approach has its own character, and exhibits diversely in robustness test. Avoiding being stuck in so many interlaced algorithms, we try to pick up the basic ideas lying in the existed schemes, for example, least significant bit modification, phase coding, spread spectrum modification, cepstrum domain modification, wavelet domain modification and echo hiding [35, 36]. By performing perceptibility and robustness evaluations on each basic scheme, we find out their merits and demerits of withstanding some typical attacks. It is useful for preparing to develop an extra robust audio watermarking system for copyright protection.

Note that in the evaluations, we use standard audio files from EBU’s Sound Quality Assessment Material (SQAM) as the host signals to be watermarked. The EBU SQAM disc tracks are WAVE files, sampled at 44.1kHz and 16 bit quantized [37] and specifically for the testing and evaluation of sound systems. Hereunder, track 40 is taken as the host signal to be watermarked and a binary PN sequence acts as the watermark. Sometimes repetitive coding is applied.

5.1 Least Significant Bit (LSB) Modification

Earlier audio watermarking methods employ the simplest way, ‘replacing’, to embed data into host signal. One popular ways is replacing the least significant bit (LSB) of each sample according to the watermark represented in a coded binary string. Probably not for the entire signal, some particular periods are selected for the sake of robustness or reliability. Moreover, it is more reasonable to modify LSB of frequency components, since the cochlea acts as the frequency filterbank as mentioned before. As is known, human beings are much more sensitive to low frequency part. Although high frequency components are perceptually insignificant, they are subjective to attacks as well, typically, lossy compression. Thus, we still work on the low frequency parts mostly. In addition, secret key may be conjointly used to increase the security.
The idea of watermarking scheme in [10] comes from LSB method and its schematic is shown in Figure 33.

![Figure 33. Schematic for audio watermarking scheme in [10]](image)

The embedding is based on the notion that rounding a certain decimal place of frequency spectrum coefficients to an even number stands for watermark ‘1’, or to an odd number for watermark ‘0’. The decimal place depends on the perceptibility and security, probably the fourth or fifth decimal place of Matlab data. Besides, there are two additional features in the scheme. One is employing four times repetitive encoding to ensure reliability. The other is using PN sequence to randomize the quantization scale to enhance further the security, which is regarded as secret key. Here, the fourth decimal place is for ‘1’ and the fifth for ‘0’. As for the watermark detection, the procedure is exactly the first part of embedding. The watermark bit is determined as ‘0’ or ‘1’ on the basis of whether the specific decimal place is even or odd.

Figure 34 exemplifies track 40 and its watermarked signal. Here, an eight-level complementary filter bank is used. Since the length of the period used for embedding can only be in a power of 2, only the beginning 524288 (=2¹⁹) points are watermarked in the implementation, in spite of the length of host signal. The evaluation results are listed in Table 2.

<table>
<thead>
<tr>
<th>Audio File</th>
<th>Number of Bits Embedded</th>
<th>Seg_SNR (dB)</th>
<th>ROCBR (%)</th>
<th>ROCLR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>No attack</td>
<td>Noise (50dB)</td>
</tr>
<tr>
<td>Track 40</td>
<td>512</td>
<td>77.20</td>
<td>98.75</td>
<td>53.99</td>
</tr>
</tbody>
</table>

Table 2. Simulation results for LSB watermarking scheme
Such an audio watermarking scheme has high segmental SNR, for we only modify the least significant bits. As a result, the watermarking is likely to be perceptually transparent and a very few distortion is caused. At the same time, however, it is easy to understand that this method is quite susceptible to any attacks. Even just quite slight modifications will make the system crash. For example, we add imperceptible white Gaussian noise (SNR=50dB) to the host signal, and then ROCBR and ROCLR drop dramatically. Consequently, LSB method cannot be utilized in most robust audio watermarking system, unless there are some very strong low frequency components in the host audio. Those components could help the host signal mask the perceived noise produced. Accordingly, the capacity will lower down quite a lot.

5.2 Phase Coding

The basis of phase coding watermarking is that human ears could hardly perceive the absolute phase coefficients of the sound, but sensitive to relative value. Therefore, it is feasible to substitute the phase of an initial audio segment with a reference phase representing the watermark, and then adjust the phase of subsequent segments to preserve the relative phase between segments [6]. As is known, the Fourier Transform of the real series $x(n)$ ($1\leq n\leq N$) is conjugated symmetry around the Nyquist frequency, like

$$X_k = X^*_{N-k}$$

where * denotes complex conjugation and the subscripts are interpreted modulo $N$. It means that we can just modify the first half of complex DFT spectrum and the second half must conform to this symmetry. Therefore, only half of spectrum is involved in the process of watermarking. The whole procedure is depicted as follows.

1. Partition the original audio signal $s(n)$ ($n=1,\ldots,N$) into some short segments, each with $Q$ points. $Q$ depends on the tradeoff between perceptibility of phase distortion and capacity. So there are $K$ segments totally, $s_k(n)$ ($k=1,\ldots,K$), where $K=\text{floor}(N/Q)$.

2. Apply $P$-point FFT (probably $P=2^m$) on each segment $s_k(n)$ and get its frequency spectrum $F_k(\omega)$, so as to set up magnitude and phase matrix $A$ and $\Phi$, respectively. Here the part with superscript $^l$ or $^h$ represents the former or the later half of the spectrum.
\[ F(\omega) = A \cdot \left[ \sin(\Phi) + j \cdot \cos(\Phi) \right] = [F^l(\omega), F^h(\omega)] \]

\[
A = \begin{bmatrix} A^l & A^h \end{bmatrix} = \begin{bmatrix}
(a_{11},...,a_{i(P/2)}) & (a_{(i+1)(P/2)},...,a_P) \\
(a_{21},...,a_{2(P/2)}) & (a_{(i+2)(P/2)},...,a_{(i+1)p}) \\
\vdots & \vdots \\
(a_{K1},...,a_{K(P/2)}) & (a_{(K+1)(P/2)},...,a_{Kp})
\end{bmatrix}
\]

\[
\Phi = \begin{bmatrix} \Phi^l & \Phi^h \end{bmatrix} = \begin{bmatrix}
(\phi_{11},...,\phi_{i(P/2)}) & (\phi_{(i+1)(P/2)},...,\phi_{P}) \\
(\phi_{21},...,\phi_{2(P/2)}) & (\phi_{(i+2)(P/2)},...,\phi_{(i+1)p}) \\
\vdots & \vdots \\
(\phi_{K1},...,\phi_{K(P/2)}) & (\phi_{(K+1)(P/2)},...,\phi_{Kp})
\end{bmatrix}
\]

(3) Calculate phase differences between the adjacent segments and get the phase difference matrix,

\[
\Delta \Phi = [\Delta \Phi_k] = \Phi^l_k - \Phi^l_{k-1} = [\Delta \phi_{li}] = [\phi_{li} - \phi_{(i-1)l}]. \quad (k=2,...,K; \quad i=1,...,P/2).
\]

\[
\Delta \Phi = \begin{bmatrix}
\Delta \phi_{21} & \Delta \phi_{22} & \ldots & \Delta \phi_{2(P/2)} \\
\Delta \phi_{31} & \Delta \phi_{32} & \ldots & \Delta \phi_{3(P/2)} \\
\vdots & \vdots & \ldots & \vdots \\
\Delta \phi_{K1} & \Delta \phi_{K2} & \ldots & \Delta \phi_{K(P/2)}
\end{bmatrix} = \begin{bmatrix}
\phi_{21} - \phi_{11} & \phi_{22} - \phi_{12} & \ldots & \phi_{2P} - \phi_{1P} \\
\phi_{31} - \phi_{21} & \phi_{32} - \phi_{22} & \ldots & \phi_{3P} - \phi_{2P} \\
\vdots & \vdots & \ldots & \vdots \\
\phi_{K1} - \phi_{(K-1)1} & \phi_{K2} - \phi_{(K-1)2} & \ldots & \phi_{KP} - \phi_{(K-1)P}
\end{bmatrix}
\]

(4) Define an initial phase array \( \Phi^l = [\phi^l_{11}, \phi^l_{21}, ..., \phi^l_{iP/2}] \) according to the watermark \( W(i) \) (\( i=1,...,P/2 \)). For example, if \( W(i) = 1 \), then \( \phi^l_{ji} = \pi / 2 \). Otherwise, if \( W(i) = 0 \), then \( \phi^l_{ji} = -\pi / 2 \).

(5) Recreate the new phase matrix \( \Phi^l \) by using \( \Phi^l \) and \( \Delta \Phi \). That is,

\[
\Phi^l = \begin{bmatrix} \Phi^l_1 & \Phi^l_2 \\
\ldots & \ldots \\
\Phi^l_{(K-1)} & \Phi^l_{(K-1)} \end{bmatrix} + \begin{bmatrix} \Delta \Phi_2 \\
\Delta \Phi_3 \\
\vdots \\
\Delta \Phi_{K}
\end{bmatrix} = \begin{bmatrix}
\phi^l_{11} + \Delta \phi_{21} & \phi^l_{12} + \Delta \phi_{22} & \ldots & \phi^l_{1(P/2)} + \Delta \phi_{2(P/2)} \\
\phi^l_{21} + \Delta \phi_{31} & \phi^l_{22} + \Delta \phi_{32} & \ldots & \phi^l_{2(P/2)} + \Delta \phi_{3(P/2)} \\
\vdots & \vdots & \ldots & \vdots \\
\phi^l_{(K-1)1} + \Delta \phi_{K1} & \phi^l_{(K-1)2} + \Delta \phi_{K2} & \ldots & \phi^l_{(K-1)(P/2)} + \Delta \phi_{K(P/2)}
\end{bmatrix}
\]
(6) Combine the original magnitude matrix, $A_l$, with new phase matrix, $\Phi'$, to build the first half of new frequency spectrum $\hat{F}(\omega)$. Then the second half is the inverse complex conjugation of the first half. Hence, new spectrum is created, $\hat{F}(\omega) = \left[ \hat{F}_l(\omega), \hat{F}_h(\omega) \right]$. The pseudo codes are given as follows.

\[
N_2 = \frac{N_{\text{frame}}}{2} ; \\
\text{amplitude} = \text{abs}(\text{fft(frame)}) ; \quad \text{% magnitude of one frame} \\
\text{phase} = \text{angle}(\text{fft(frame)}) ; \quad \text{% phase of the frame} \\
\%
\text{Reconstruct the signal by using Euler's formula} \\
\text{Fr}_l = \text{amplitude}(1:N_2) \cdot \cos(\text{phase}(1:N_2)) ; \quad \text{% the first half of real part} \\
\text{Fi}_l = \text{amplitude}(1:N_2) \cdot \sin(\text{phase}(1:N_2)) ; \quad \text{% the first half of imaginary part} \\
\text{Fr}_h = [0, \text{Fr}_l(N_2-1:2)] ; \quad \text{% the second half of real part} \\
\text{Fi}_h = [-0, \text{Fi}_l(N_2-1:2)] ; \quad \text{% the second half of imaginary part} \\
\text{Fr} = [\text{Fr}_l, \text{Fr}_h] ; \quad \text{% the real part} \\
\text{Fi} = [\text{Fi}_l, \text{Fi}_h] ; \quad \text{% the imaginary part} \\
\text{freq\_frame} = \text{complex}(\text{Fr}, \text{Fi}) ; \quad \text{% reconstructed frame} \\
\text{re\_frame} = \text{real}(\text{ifft(freq\_frame)}) ; \quad \text{% IFFT back to time domain}
\]

(7) Turn each $\hat{F}_k(\omega)$ back to $s'_k(n)$ in time domain by applying IFFT, and concatenate them into the overall watermarked signal $s_W(n) = [s'_1(n), s'_2(n), ..., s'_k(n)]$.

The detection is very simple, setting a threshold based on two values of initial phases and then checking the former half phase spectrum of the first segment. Here, for initial phases are $\pm \pi/2$, the threshold $T = 0$. If the phase value is larger than zero, the watermark bit is ‘1’. Otherwise, the watermark bit is ‘0’.

Figure 35 shows the host signal and the watermarked signal by phase coding. Here, one segment has 4096 points and actually 2047 bits ($= 4096/2-1$) are embedded. No repetitive coding or BCH coding is used. Note that the actual number of bits embedded equals to half size of the segment minus 1, because the first frequency bin is DC component and invalid for phase coding. Also, we can see that the watermarked signal differs from the host signal, since the absolute phases of consecutive segments have been altered. So segmental SNR is meaningless, as mentioned in section 3.1. However, the relative phase differences in the same frequency bins of adjacent segments are preserved. As a result, the quality of watermarked signal is just a bit away from satisfaction, although there are still some perceptible clicks due to phase dispersion. It is caused by a break in the relationship.
of the phases between each segment, for instance, substituting the initial phase with the binary code. In order to relieve the spiny situation, the magnitude of the phase modifier needs to be close to the original value. Moreover, by smoothing transitions between phase changes, audible distortions could be reduced. Finally, the results of robustness test are listed in Table 3.

<table>
<thead>
<tr>
<th>Audio File</th>
<th>Number of Bits Embedded</th>
<th>ROCBR (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No attack</td>
</tr>
<tr>
<td>Track 40</td>
<td>2047</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3. Simulation results for phase coding watermarking

It is seen that phase coding is better than LSB method. Because phase coding watermarking is more robust against noise addition compared to LSB method. But it is still quite vulnerable to cropping, even just five samples. It is quite straightforward. One watermark bit corresponds to one phase coefficient in turn, so synchronization is especially important. Moreover, due to the watermark just accreted to the first segment of host signal, this portion is crucial. Any loss on this part will lead to failure of watermark detection. Also, this segment must not be an absolute silence part, for zero magnitude spectrum is impossible to retain phase information. Generally speaking, phase coding method is not robust enough and rather hazardous.

5.3 Spread Spectrum Watermarking

In a broad sense, any watermarking scheme for which the watermark signal is spread over a wide range of frequencies may be referred to a spread spectrum (SS) watermarking scheme [38]. Rather than embedding watermark in narrow bands of the host signal, it is more difficult to intercept and remove the watermark carried in SS watermarking scheme. However, perceivable distortion is often introduced into the host signal. As a result, amplitude shaping by masking threshold based on psychoacoustic model is always crucial to keep the noise inaudible to human auditory system [39]. Thus, in a SS watermarking scheme, the watermark message is first shaped according to psychoacoustic model and then modulated by a pseudorandom sequence called the key. Finally, it is added to the host signal to form the watermarked signal. During the decoding stage, the same key is
used to demodulate the watermarked signal so as to recover the watermark embedded.

Some of the most influential watermarking schemes are founded on the basis of SS principle, such as [28], [40] and [41]. Our basic watermarking scheme from [11] also belongs to a Direct Sequence Spread Spectrum (DSSS) watermarking system, which will be expounded in the next section.

5.4 Cepstrum Domain Watermarking

Cepstrum refers to the spectrum in cepstral domain, like frequency spectrum in frequency domain. The complex cepstrum \( \tilde{x}(n) \) for a sequence \( x(n) \), is calculated by finding the complex natural logarithm of Fourier Transform (FT[.]) of \( x(n) \), and then taking inverse Fourier transform (FT\(^{-1}\)[.]) of the resulting sequence, as described in Figure 36 (a).

\[
\tilde{x}(n) = \text{FT}^{-1}\left[ \ln \left( X(e^{j\omega}) \right) \right]
\]

where

\[
\text{Fourier Transform: } X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x(n)e^{-j\omega n} = \left| X(e^{j\omega}) \right| e^{j\arg[X(e^{j\omega})]}
\]

\[
\text{Complex Natural Logarithm: } \ln \left( X(e^{j\omega}) \right) = \ln \left| X(e^{j\omega}) \right| + j \arg[X(e^{j\omega})]
\]

\[
\text{Inverse Fourier Transform: } FT^{-1}\left[ \ln \left( X(e^{j\omega}) \right) \right] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln \left( X(e^{j\omega}) \right) e^{j\omega n} d\omega
\]

So, the complex cepstrum \( \tilde{x}(n) \) is given by

\[
\tilde{x}(n) = FT^{-1}\left[ \ln \left( FT[x(n)] \right) \right] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln \left( X(e^{j\omega}) \right) e^{j\omega n} d\omega
\]

Similar to Fourier Transform analysis and synthesis, the original signal could be recovered from complex cepstrum, see Figure 36(b).

\[
x(n) \xrightarrow{\text{FT}[.]} X(e^{j\omega}) \xrightarrow{\text{exp}(.)} x(n)
\]

59
In fact, the real part of complex cepstrum is often taken as ‘cepstrum’, hereunder what we use in most applications. It should be distinguished from “real cepstrum”, \( \hat{x}_{\text{real}}(n) \), which is defined as the inverse Fourier transform of the natural logarithm of magnitude of Fourier transform of \( x(n) \), like

\[
\hat{x}_{\text{real}} (n) = FT^{-1}\left[\ln\left|FT\{x(n)\}\right|\right] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln|X(e^{j\omega})| \cdot e^{j\omega} d\omega
\]

Real cepstrum is based only on the magnitude of the Fourier transform for the sequence and phase information is excluded, so there is no inverse real cepstrum transformation to complete reconstruction.

Currently, cepstrum analysis has been extensively used in speech/speaker recognition and homomorphic filtering, whereas it is a comparatively new area for audio watermarking. Inspired by the research result that cepstrum coefficients experience much less disturbance after most common signal processing attacks than original samples in time domain, [12] and [42] propose to embed watermark by manipulating the statistical mean (SMM) of a subset of cepstrum coefficients. It is a kind of patchwork watermarking. Figure 37 shows a portion of an audio signal and its cepstrum. As is seen, a typical cepstrum has large coefficients around the center and small coefficients spreading over both sides. The large coefficients contain important envelope information and are mostly perceptually significant, whereas the small coefficients are floating around zero. Moreover, from Figure 38, it is seen that most attacks could change individual cepstrum coefficient dramatically, but their statistical mean only varies very few. So the statistical mean is an attack-invariant feature.

A basic watermarking scheme is implemented on the basis of SMM. In the embedding stage, the statistical mean of cepstrum coefficients of each frame is enforce coercively to two different watermarking depths, \( \alpha_1 \) or \( \alpha_2 \), according to the watermark bit being ‘1’ or ‘0’. Both \( \alpha_1 \) and \( \alpha_2 \) are different real numbers. Then, the detection could be accomplished by comparing a threshold \( TH(\alpha_1, \alpha_2) \) to the statistical mean of every watermarked frame. Studies show that negative-mean enforcement is detrimental to the watermarked signal, that is, experiencing much larger variation after attacks. Hence, we set a positive mean to carry bit ‘1’ and zero to carry bit ‘0’.
The following pseudo codes describe the basic scheme.

% Embedding
chop the host signal \( host(n) \) into a certain number of frames
for every frame \( x(n) \)
transform to cepstrum domain, \([c(n), nd] = cceps[x(n)]\)
calculate the mean of cepstrum coefficients, \( m = \text{mean}[c(n)] \)
remove bias-mean from the original coefficients, \( mc(n) = c(n) - m \)
if watermark bit being ‘1’
enforce the mean to a positive real number, \( nc(n) = mc(n) + \alpha \)
end
transform back to time domain, \( wm_x(n) = icceps[mc(n), nd] \)
end
join every frame end to end so as to build the watermarked signal, \( wmed(n) \)

% Detection
chop the watermarked signal \( wmed(n) \) into a certain number of frames
for every frame
transform \( wm_x(n) \) to cepstrum domain, \([wm_c(n), nd] = cceps[wm_x(n)]\)
calculate the mean of cepstrum coefficients, \( wm_m = \text{mean}[wm_c(n)] \)

Figure 38. Comparison between several cepstrums after different attacks
determine threshold, \( TH = N_{frame} \alpha \times 75\% \) (experiential value)

\[
\text{if } wm_m > TH \\
\quad \text{the bit detected being '1'} \\
\text{otherwise} \\
\quad \text{the bit detected being '0'}
\]

end

determine threshold, \( TH = N_{frame} \alpha \times 75\% \) (experiential value)

\[
\text{if } wm_m > TH \\
\quad \text{the bit detected being '1'} \\
\text{otherwise} \\
\quad \text{the bit detected being '0'}
\]

After simulation (see Table 4), two problems are found existing in the basic cepstrum domain system are unsatisfied. First, non-overlapping between consecutive segments lead to obvious ‘break’ sound, although segmental SNR is high, 54.90dB. It is seen that segmental SNR is not always applicable. We must use windowing and overlapping to smooth the edges of adjacent frames. Since the detection is guaranteed by the mean of the whole segment, it’d better preserve most of original cepstrum. In our experiment, ladder window with 1/4 overlapping is employed for providing a soft transition, shown in Figure 39. Second, silence parts in host signal are sensitive to cepstrum coefficients modification. A little alteration will produce distinct noise, described in Figure 40 (b). Moreover, detection errors often occur in such areas. That is why ROCBR is not 100% even no attack applied on the watermarked signal. Therefore, embedding should skip over the silence part. So we check every \( N_{frame}/8 \) samples followed by the proceeding pointer. If its power is less than 0.01, this portion will be regarded as silence part and the pointer will hop to next portion.

![Ladder window](image)

Figure 39. Ladder window
Two sets of simulations are carried out to prove the efficiency of the improved cepstrum domain watermarking. In the first experiment, there is no repetitive coding used and we just calculate ROCBR. In the second one, tri-repetitive coding is applied on PN watermark, so ROCBR and ROCLR are computed to check the detection rate. For both tests, the size of each frame $N_{frame}$ is 2048 points, so the theoretical capacity is 28.7 bps ($= 44100/2048/0.75$). Because the watermarking discards the muteness in the audio signal, the actual capacity is a bit lower than theoretical value. Here, 480 bits are embedded into track 40, which has 1042752 samples and lasts 23.6 seconds. The real capacity is 20.4 bps ($= 480/23.6$). In addition, watermarking depth $\alpha$ is set as 0.002 compared to the maximum of audio signal, 0.3941. Then detection threshold $TH$ is set to 3.4 (experiential value, $TH=N_{frame}*\alpha*85\%$). All simulation results are summarized in Table 4.

In the subjective listening test, the new watermarked signals from improved scheme are perceptually undistinguished from the host signal. It is much better than the old scheme. Their segmental SNRs are around 30dB, still fairly high. Moreover, it is confirmed from Table 4 that robustness of the improved scheme is comparative to that of the old scheme, even stronger via repetitive-coding. Towards PITS, if $N_{frame}$ is altered accordingly, the detection rate will increase a lot. However, it is necessary to point out that echo addition is still a potential trouble to the new scheme. When the delay exceeds 50ms, detection rate here will decrease significantly, because adding such an echo has changed the statistical property of the watermarked signal. Solutions should be found to help the scheme resist echo addition with a delay of 150ms or so.
### Table 4. Simulation results for cepstrum domain watermarking

<table>
<thead>
<tr>
<th></th>
<th>Old Scheme $(TH=3.1)$</th>
<th>Improved Scheme $(TH=3.4)$</th>
<th>No Coding</th>
<th>Tri-repetitive Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROCBR (%)</td>
<td>ROCBR (%)</td>
<td>ROCBR (%)</td>
<td>ROCLR (%)</td>
</tr>
<tr>
<td>No attack</td>
<td>99.21</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Noise (30dB)</td>
<td>99.21</td>
<td>100</td>
<td>89.38</td>
<td>100</td>
</tr>
<tr>
<td>Resampling</td>
<td>98.82</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>LP filtering</td>
<td>99.21</td>
<td>82.08</td>
<td>84.17</td>
<td>84.38</td>
</tr>
<tr>
<td>Echo (0.3, 40ms)</td>
<td>98.63</td>
<td>96.25</td>
<td>88.75</td>
<td>98.75</td>
</tr>
<tr>
<td>Compression I</td>
<td>98.82</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Compression II</td>
<td>74.07</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Cropping (1000)</td>
<td>84.48</td>
<td>76.88</td>
<td>98.54</td>
<td>100</td>
</tr>
<tr>
<td>Inserting (1000)</td>
<td>84.48</td>
<td>100</td>
<td>98.54</td>
<td>100</td>
</tr>
<tr>
<td>Jittering</td>
<td>99.21</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>PITS +4%</td>
<td>$N_{frame}$ 52.26</td>
<td>56.04</td>
<td>54.17</td>
<td>53.13</td>
</tr>
<tr>
<td></td>
<td>$N'_{frame}$ 79.76</td>
<td>63.96</td>
<td>82.71</td>
<td>81.88</td>
</tr>
<tr>
<td>PITS -4%</td>
<td>$N_{frame}$ 54.81</td>
<td>48.75</td>
<td>53.54</td>
<td>53.75</td>
</tr>
<tr>
<td></td>
<td>$N'_{frame}$ 76.62</td>
<td>62.03</td>
<td>85.42</td>
<td>95</td>
</tr>
<tr>
<td>TPPS +4%</td>
<td>92.34</td>
<td>90.21</td>
<td>94.58</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>97.25</td>
<td>92.29</td>
<td>88.13</td>
<td>96.88</td>
</tr>
<tr>
<td>TPPS -4%</td>
<td>92.34</td>
<td>90.21</td>
<td>94.58</td>
<td>100</td>
</tr>
</tbody>
</table>

For optimizing the performance, prior researchers tried to utilize some supplemental techniques in watermarking systems based on cepstral domain feature modification. In [43], psychoacoustic model is employed to shape the watermark signal to control the audibility of introduced distortion. In addition, cepstrum domain method attempted to be combined with Salient Points Extraction to obtain synchronization. It means modifying the cepstral mean values only in the frames following salient points [44]. However, in practice, the result of implementation is not ideal. The biggest problem that arises is the restricted capacity due to the selection of steady salient points. For the salient points are starting points of frames to be watermarked, we expect that salient points stay put or just shift a little bit after attacked. Therefore, they must be quite distinctive points out of normal samples. At the same time, the size of one frame should be large enough to insure the stability of statistical mean as well. Consequently, the number of salient points extracted is very limited. For example, in track 40, there are only 32 effective salient points. It means just 32 bits could be embedded in the host signal. The capacity is too low and does not come up to our expectation.

Another such kind of watermarking system is presented by Lee and Ho [45]. According to the distribution of cepstral coefficients and also the frequency masking characteristics of human auditory system, a pseudorandom sequence is weighted in cepstrum domain and then taken as the watermark signal, so as to minimize its audibility. However, in the detection, the cross-correlation between the recovered and the original watermarks is measured to verify the presence of watermark, which differs from SMM. This watermarking scheme is reported...
to be robust to multiple watermarks, mp3 compression and additive noise. But it is not a blind watermarking system, since the original watermark is required in the detection.

In conclusion, cepstrum domain watermarking method is rather robust against mis-synchronization, especially with the help of repetitive coding. This is the biggest advantage. However, security is its weakness, since the attacker could deliberately adjust the cepstral mean of the watermarked signal slightly. As a result, we are unable to accurately extract the watermark embedded.

### 5.5 Wavelet Domain Watermarking

Wavelet watermarking algorithm is employing wavelet decomposition/reconstruction to realize embedding watermark in the host signal and then extracting it from the watermarked signal probably being attacked.

Wavelet Transform is powerful for time-frequency representation of non-stationary signal, such as speech and audio signals. For non-stationary signals have frequency content and properties changing with time or the length of signal, Fourier transform is less useful under such situation. More advanced than Short-time Fourier Transform (STFT), Wavelet Transform (WT) is able to decompose non-stationary signal into multilevel components. That is, by passing a pair of complementary filter, the primitive signal will be decomposed to high and low frequency elements, whose terms are *approximation* and *detail*. Then *approximation* part would be further decomposed over and over. In theory, decomposition process is of infinite levels. However, suitable amount of decomposition levels is chosen according to the practical requirement. A DWT tree for a three-level decomposition is shown in Figure 41.

![DWT tree for a three-level decomposition](image)

As a result, DWT provides greater time resolution and less frequency resolution for high frequencies, while providing greater frequency resolution and less time resolution for low frequencies. In this respect, DWT exhibits similar time-frequency resolution characteristics to human ear. Therefore, a good performance from DWT watermarking scheme is especially expected. Furthermore, inspired from the idea embodied in cepstrum domain watermarking algorithm, statistical features of wavelet coefficients are taken into considerations. The mean of wavelet coefficients at the coarsest approximation part is adopted in [13] and [46], for their experiments show that these means experience much less variance after attacks than original sample in the time domain.
Thus, based on this attack-invariant feature, a wavelet-based watermarking scheme is implemented as follows. It is also a kind of patchwork. First, employ a three-level decomposition on each overlapped segment of the host signal with ‘db4’ or ‘haar’ wavelet. Second, slightly modify the mean of ‘A3’ subband corresponding with the watermark bits. Finally, reconstruct signal from the whole wavelet decomposition structure ([A3, D3, D2, D1]) again. The detection is not complicated, that is, decomposing every segment in the same way as used above and then checking the mean of its ‘A3’ subband.

% Embedding
Segment the host signal into overlapping frames
for each frame
    Apply hamming-window
    Perform three-level ‘haar’ wavelet decomposition
    Restore all approximation and detail coefficients, A3, D3, D2 and D1
    Remove the biased mean of A3 subband and get mA3
    if watermark bit is ‘1’
        Add a small number $\alpha$ to mA3
    else
        Subract $\alpha$ from mA3
    end
    Reconstruct the watermarked frame via mA3, D3, D2 and D1
end
Join all frames together and obtain the watermarked signal

% Detection
Chop the overlapped frames from the host signal
for each frame
    Perform three-level ‘haar’ wavelet decomposition
    Calculate the mean of A3’ subband
    if the mean is positive
        The watermark bit embedded in this frame is ‘1’.
    else
        The watermark bit embedded in this frame is ‘0’.
    end
end

Similar to the improved cepstrum watermarking scheme, repetitive coding (five-time) and silence removal are utilized to improve the overall detection performance. The experimental parameters are set as follows. Each frame has 2048 samples and two adjacent frames are 50% overlapped. To balance the tradeoff between perception and robustness, the embedding depth $\alpha$ is set to be 0.004. In this manner, the segmental SNR is around 21 dB and the watermarked signal sounds identical to the host signal. Results of robustness test are summarized in Table 5.
Technical Report: Audio Watermarking for Copyright Protection

<table>
<thead>
<tr>
<th></th>
<th>No Coding</th>
<th>Tri-repetitive Coding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROCBR (%)</td>
<td>ROCBR (%)</td>
</tr>
<tr>
<td>No attack</td>
<td>80.43</td>
<td>96.74</td>
</tr>
<tr>
<td>Noise (30dB)</td>
<td>80.65</td>
<td>96.96</td>
</tr>
<tr>
<td>Resampling</td>
<td>80.43</td>
<td>96.52</td>
</tr>
<tr>
<td>LP filtering</td>
<td>80.65</td>
<td>96.74</td>
</tr>
<tr>
<td>Echo (0.3, 200ms)</td>
<td>78.48</td>
<td>95.87</td>
</tr>
<tr>
<td>Compression I</td>
<td>80.43</td>
<td>96.74</td>
</tr>
<tr>
<td>Compression II</td>
<td>55.22</td>
<td>87.83</td>
</tr>
<tr>
<td>Cropping (1000)</td>
<td>77.61</td>
<td>95.43</td>
</tr>
<tr>
<td>Inserting (1000)</td>
<td>78.26</td>
<td>96.30</td>
</tr>
<tr>
<td>Jittering</td>
<td>80</td>
<td>96.74</td>
</tr>
</tbody>
</table>

Table 5. Simulation results for wavelet watermarking scheme

We can see that with the help of repetition coding, wavelet watermarking is robust to almost all attacks except time stretching. Because five-times repetition ensures that the samples over a large range have the same statistical characteristic, which is the foundation of correct detection. Although that of adjacent frames may be confused, statistical means of A3 subband in middle frames are preserved. By virtue of decision based on majority, the right watermark bit could be detected.

To further combat synchronization attacks like random cropping and time-scale modification, the previous author proposed to embed watermark in steady high-energy local regions that represent music edges [47]. These regions are of great importance to the understanding of music and will not changed much for maintaining high auditory quality, so that the embedded watermark is potential to survive all kinds of distortion. Three methods are presented to select such embedding regions. The first one is like Peak Points Extraction and Salient Points Extraction. According to a criterion defining high-energy points, reference points are selected from original audio waveform directly. The second takes the peaks on audio envelop as reference points. The last approach is based on music content analysis. Reference points are selected as peaks in ‘D3’ subband from a five-level wavelet decomposition.

As seen, statistical wavelet watermarking is similar to the previous cepstrum domain watermarking, but modify the means in different domains. So security is an ineluctable trouble. To increase security of the scheme, random chaotic sequences produced via the hybrid chaotic dynamical system is proposed to encrypt the watermark message in [13].
5.6 Echo Hiding

Echo hiding embeds watermark bits into the host signal by introducing two echo kernels with different offsets [6]. As discussed before, if both offsets are below the audible threshold of human ear, the watermarking would not generate noticeable noise. Correspondingly, watermark extraction depends on echo detection, where cepstrum analysis plays an important role. Such minute space is hard to discern in time domain.

The essence of embedding is passing every portion of host signal through one of two possible delay units according to its desired bit, ‘1’ or ‘0’. Two echo kernels are illustrated in Figure 42, where $d_1$, $d_0$ are delay offsets and $\alpha_1$, $\alpha_0$ are echo amplitudes.

![Figure 42. Echo kernels for ‘1’ and ‘0’](image)

The echo kernel is formulated as

$$h(n) = \delta(n) + \alpha \cdot \delta(n - d)$$

So the echoed signal is expressed as

$$s(n) = x(n) \otimes h(n) = x(n) + \alpha \cdot x(n - d)$$

where $\otimes$ denotes convolution, and $x(n)$ and $s(n)$ represent host signal and its echoed version respectively.

To detect the echo embedded, we compare the peak values at two possible locations in cepstrum domain. This is based on the fact that complex cepstrum will show a spike at $d_1$ or $d_0$, which indicates the echo offset. It is proved as follows [48]. The complex cepstrum of echoed signal is given by

$$5(n) = FT^{-1}[\ln(FT[s(n)])]$$

$$= FT^{-1}[\ln(FT[x(n)])] + FT^{-1}[\ln(FT[h(n)])]$$

$$= FT^{-1}[\ln(X(e^{j\omega}))] + FT^{-1}[\ln(H(e^{j\omega}))]$$

$$= c_x(n) + c_h(n)$$

where $c_x(n) = FT^{-1}[\ln(X(e^{j\omega}))]$, $c_h(n) = FT^{-1}[\ln(H(e^{j\omega}))]$ and $H(e^{j\omega}) = FT[h(n)] = 1 + \alpha \cdot e^{j\text{rad}}$
Since
\[ \ln(1+\alpha) = \alpha - \alpha^2/2 + \alpha^3/3 - \ldots \text{, when } |\alpha| < 1, \]
then
\[ \ln(H(e^{j\omega})) = \alpha \cdot e^{-j\omega d} - \alpha^2/2 \cdot e^{-2j\omega d} + \alpha^3/3 \cdot e^{-3j\omega d} - \ldots \]
So
\[ c_{\delta}(n) = \text{FT}^{-1}\left[ \alpha \cdot e^{-j\omega d} - \alpha^2/2 \cdot e^{-2j\omega d} + \alpha^3/3 \cdot e^{-3j\omega d} - \ldots \right] \]
As result,
\[ \tilde{s}(n) = c_{\delta}(n) + \alpha \cdot \delta(n-d) - \alpha^2/2 \cdot \delta(n-2d) + \alpha^3/3 \cdot \delta(n-3d) - \ldots \]
It shows that a series of pulses with exponentially decaying amplitudes repeatedly appear. But the first spike at \( n=d \) is dominant and its amplitude is \( \alpha \). Moreover, it is seen that the delay offset should be chosen carefully to ensure not only inaudibility of echo but also high detection possibility.

However, in real experiment, the result of the cepstrum is still unsatisfied. The cepstrum peak is small relative to the host signal and is subject to be buried by surrounding peaks. Therefore, autocorrelation of cepstrum is taken.

\[ \text{FT}^{-1}\left[ \ln(\text{FT}[s(n)])^2 \right] \]
The first peak is amplified significantly (see Figure 43) and the remaining pulse trains are suppressed.

![Figure 43. Autocorrelation of cepstrum for echo detection](image)

At last, the bit embedded is decided based on comparison between \( \tilde{s}(d_1) \) and \( \tilde{s}(d_0) \). That is, if \( \tilde{s}(d_1) > \tilde{s}(d_0) \), the bit is more likely to be ‘1’. On the contrary, the bit is supposed to be ‘0’ when \( \tilde{s}(d_1) < \tilde{s}(d_0) \).

Similar to before, overlappings between the adjacent segments to prevent abrupt changes is also necessary in echo hiding watermarking. Here, we use the previous ladder window with 1/4 overlapping. Moreover, the embedding should keep away from the silence parts. Finally, in order to enhance the security, a secret key sequence is involved. That is, the key along with watermark bit will decide to use one possible offset out of four,
The first subscript stands for key bit and the second is for watermark bit. For example, if both the key and the watermark bit are ‘1’, \( d_{11} \) is selected. In the detection, one set of offsets at a time is chosen according to the given key. If the key for one frame is ‘1’, peak values at \( d_{11} \) and \( d_{10} \) in autocorrelation of cepstrum will be compared to find out the bit embedded. Otherwise, if the key is ‘0’, \( d_{01} \) and \( d_{00} \) will be under consideration.

The following codes are written for the amended scheme.

```matlab
% Embedding
set parameters \( N_{frame} \), \( a \), \( d_{11} \), \( d_{10} \), \( d_{01} \) and \( d_{00} \), etc;
define a key sequence \( k(n) \) with the same length of watermark message \( w(n) \);
c_1 = 1;  \% initializing the first counter for counting \( w(n) \)
c_2 = 1;  \% initializing the second counter, worked as the proceeding pointer
while ( \( c_1 \) ≤ length\( [w(n)] \) )
    temp = s( \( c_2 \) : \( c_2 + N_{frame}/8 - 1 \) );  \% temporary array
    if power(temp) > power_threshold \% regarded as non-silence part
        temp = s( \( c_2 \) : \( c_2 + N_{frame} - 1 \) );  \% process one frame
        if \( k(c_1) = = 1 \)
            if \( w(c_1) = = 1 \)
                d = \( d_{11} \);
            else
                d = \( d_{10} \);
            end
        else
            if \( w(c_1) = = 1 \)
                d = \( d_{01} \);
            else
                d = \( d_{00} \);
            end
        end
    end
    \( \hat{z} = temp + echo \);  \% add echo to host frame
    \% windowing and overlapping to patch echoed frames up
    c_1 = c_1 + 1;  \% increment 1
    c_2 = c_2 + \text{step};  \% \text{step} is set based on overlapping percent
else
    keep temp unprocessed and patch up
    c_2 = c_2 + length(temp);
end
end

% Detection
\( N_{frame} \), \( k(n) \), \( d_{11} \), \( d_{10} \), \( d_{01} \) and \( d_{00} \) are given;
c_1 = 1;  \% for counting \( w(n) \)
c_2 = 1;  \% worked as the proceeding pointer
```
while ( \( c_1 \leq \text{length}[w(n)] \) )

\[
\begin{align*}
t & = \tilde{s}(c_2; c_2 + N_{\text{frame}}/8 - 1); \\
\text{if power}(t) & > \text{power\_threshold} \\
& = \tilde{s}(c_2; c_2 + N_{\text{frame}} - 1); \\
autocep & = \text{real} \left( \text{ifft} \left( \log(\text{fft}(s(n))^2) \right) \right); \\
& \quad \% \text{calculate autocorrelation of cepstrum} \\
& \quad \text{if } k(c_1) = 1 \\
& \quad \text{if autocep}(d_{01} + 1) > \text{autocep}(d_{00} + 1) \\
& \quad \quad \text{recovery\_wm}(c_1) = 1; \\
& \quad \quad \text{else} \\
& \quad \quad \quad \text{recovery\_wm}(c_1) = 0; \\
& \quad \text{end} \\
& \quad \text{else} \\
& \quad \text{if autocep}(d_{01} + 1) > \text{autocep}(d_{00} + 1) \\
& \quad \quad \text{recovery\_wm}(c_1) = 1; \\
& \quad \quad \text{else} \\
& \quad \quad \quad \text{recovery\_wm}(c_1) = 0; \\
& \quad \text{end} \\
& \quad \text{end} \\
c_1 & = c_1 + 1; \\
c_2 & = c_2 + \text{step}; \\
& \quad \text{else} \\
c_2 & = c_2 + \text{length}(t); \\
& \end{align*}
\]

During the experiment (\( N_{\text{frame}} = 8192, d_{1i} = 400, d_{i0} = 410, d_{0i} = 420, d_{00} = 430 \)), it is found that instead of Fourier Transform (FT) in cepstrum analysis, discrete cosine transform (DCT) would give a better performance in detection, especially when echo amplitude \( \alpha \) is small. Moreover, other than \( \text{autocep}(d+1) \), the average \( [\text{autocep}(d) + \text{autocep}(d+1)]/2 \) is adopted for comparison in DCT cepstrum detection. In Table 6, case (a) and (b) stand for FT and DCT cepstrum respectively.

\[
\begin{align*}
\text{autocep\_a} & = \text{real} \left( \text{ifft} \left( \log(\text{fft}(s(n))^2) \right) \right) \\
\text{autocep\_b} & = \text{idct} \left( \log(\text{dct}(s(n))^2) \right)
\end{align*}
\]

From Table 6, we can get two conclusions. On one hand, echo hiding scheme is susceptible to PITS and TPPS. For PITS, if the stretching factor is known, we can change the size of frame in detection, thereby improving ROCBR drastically. But at present, there is no way to resist TPPS. On the other hand, the behavior of autocorrelation of new DCT cepstrum is always superior to that of traditional FT cepstrum. Particularly, when echo amplitude is comparatively small (\( \alpha = 0.1 \)) and there is no obvious projecting point in cepstrum at all. Usually the new detection process obtains about 15 percent higher ROCBR than the old one. Capability of detecting concealed peak is extremely important. Not to mention the perceptible artifacts produced, it is beneficial to the security. As said above, suppose an echo hiding watermarking scheme has strong echo. Although possibility of detection will be high, the attackers can randomly chop one portion from the
watermarked signal and easily recognize the peaks, then find the approximate value of delay. After that, they deliberately minify the peak or add an interfering echo. As a result, extracting watermark will definitely fail in the end. So the smaller $\alpha$ is, higher security the system has. Afterwards, more theoretical analyses are necessary to prove the efficiency of the new cepstrum.

<table>
<thead>
<tr>
<th></th>
<th>$\alpha = 0.4$</th>
<th>$\alpha = 0.2$</th>
<th>$\alpha = 0.1$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Segmental SNR (dB)</strong></td>
<td>8.6423</td>
<td>14.6523</td>
<td>20.6692</td>
</tr>
<tr>
<td><strong>ROCBR (%)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a)</td>
<td>(b)</td>
<td>(a)</td>
<td>(b)</td>
</tr>
<tr>
<td>No attack</td>
<td>94.74</td>
<td>100</td>
<td>93.42</td>
</tr>
<tr>
<td>Noise (40dB)</td>
<td>93.42</td>
<td>98.68</td>
<td>85.53</td>
</tr>
<tr>
<td>Resampling</td>
<td>97.37</td>
<td>100</td>
<td>88.16</td>
</tr>
<tr>
<td>LP filtering</td>
<td>93.42</td>
<td>100</td>
<td>84.21</td>
</tr>
<tr>
<td>Echo (0.3, 200ms)</td>
<td>92.11</td>
<td>98.68</td>
<td>93.42</td>
</tr>
<tr>
<td>Compression II</td>
<td>90.79</td>
<td>100</td>
<td>78.95</td>
</tr>
<tr>
<td>Cropping (1000)</td>
<td>94.74</td>
<td>98.68</td>
<td>90.79</td>
</tr>
<tr>
<td>Inserting (1000)</td>
<td>93.42</td>
<td>98.68</td>
<td>86.84</td>
</tr>
<tr>
<td>Jittering</td>
<td>96.05</td>
<td>100</td>
<td>86.84</td>
</tr>
<tr>
<td><strong>PITS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+4%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_{frame}$</td>
<td>51.32</td>
<td>59.21</td>
<td>50</td>
</tr>
<tr>
<td>$N'_{frame}$</td>
<td>89.47</td>
<td>100</td>
<td>86.84</td>
</tr>
<tr>
<td>-4%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$N_{frame}$</td>
<td>53.95</td>
<td>56.58</td>
<td>50</td>
</tr>
<tr>
<td>$N'_{frame}$</td>
<td>94.73</td>
<td>100</td>
<td>89.47</td>
</tr>
<tr>
<td><strong>TPPS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+4%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>56.58</td>
<td>39.47</td>
<td>47.37</td>
</tr>
<tr>
<td>-4%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>51.32</td>
<td>53.95</td>
<td>46.05</td>
</tr>
</tbody>
</table>

Table 6. Simulation results for echo hiding watermarking

Since Bender [6] introduced the concept of echo hiding scheme for watermarking in 1996, several novel methods are utilized to pursue a higher performance. In [49], the echo kernel is designed as multiple echoes comprised by both positive and negative pulses with different offsets, so that noise perceptibility is reduced considerably. Moreover, the implementation of Hyoung [48] confirmed that a combination of backward and forward kernels can enhance the watermark detection rate greatly.

In conclusion, echo hiding is a simple method for embedding watermark in audio signals. It is very straightforward and easy to implement. Moreover, by fine-tuning the parameters of echo kernel, it is indeed possible to make watermarking transparent in perception. In spite of that, echo hiding scheme has inherent weakness and suffer from unavoidable mistakes. For example, natural sound may contain inborn echoes, which cause false-positive error [48]. In addition, type of audio signals is also a decisive factor. For instance, if the audio file is polyphonic and has large power, echo is easy to hide. On the contrary, if the music is composed of intermittent beats, its spiny cepstrum will interfere with echo peaks and leads to failure of detection. Figure 44 gives two examples for ideal and non-ideal cepstrums. Therefore, maybe sound content analysis needs to be done so as to select some component for watermarking.
6 SUPPLEMENTAL TECHNIQUES

In order to enhance the robustness and increase the detection rate, some supplemental techniques are merged into different watermarking schemes. For example, we can look for attack-sensitive regions where the watermark is embedded by sound content analysis. Such attack-sensitive regions start with distinct points, which contain important features of host signal and are sensitive to human ears. That is to say, if those special points are destroyed, noticeable distortions will occur inevitably. Also, these tags are useful for self-locking without introducing any distortion, other than inserting a specific synchronization signal. Salient points extraction, power points extraction, and envelope peaks extraction are working on this aspect. Moreover, silence removal and repetitive coding have already been put into use, and exhibited their virtue. Finally, post-process on coded image watermark will conduce to better recognizability and ROCLR will gain an extra improvement.

6.1 Salient Points Extraction

Salient points are defined as the positions where the audio signal energy is fast climbing to a peak, so that the regions following each salient point would contain high energy [26]. The procedure of searching for salient points is stated as follows [44].

1. For each sample of host signal, \( x(n) \) \( (n=1,2,\ldots,N_{frame}) \), calculate energy of regions before and after this point.

\[
E_{before}(n) = \sum_{i=r}^{-1} x^2(n + i)
\]

\[
E_{after}(n) = \sum_{i=d}^{r-1} x^2(n + i)
\]
2. Calculate the ratio of $E_{\text{before}}(n)$ and $E_{\text{after}}(n)$, and compare with an energy threshold. If $\text{ratio}(n) = E_{\text{after}}(n)/E_{\text{before}}(n) > \text{threshold}$, then $x(n)$ is labeled as a raw salient point.

3. Classify all the raw salient points into separate groups, for they usually cluster.

4. Center points of each group are taken as salient points.

A good salient point extraction method should extract approximately the same set of salient points from signals at different stages, such as the original signal, the watermarked signal and the corrupted signal. However, in practice, almost every salient point is more or less shifted by a few points and some salient points may disappear or be created after processing. It means that if a watermarking scheme embeds and detects watermark based on the exact positions of salient points, it is infeasible. Therefore, salient points extraction is more suitable for patchwork watermarking schemes, that work on statistical characteristics.

### 6.2 Power Points Extraction

Peak points extraction (PPE) is also an energy-feature-based synchronization scheme [50]. Watermark is embedded between two valid peak points. The steps for PPE are stated as follows.

1. Perform special shaping on the host signal to exaggerate the energy difference. It is done by raising the sample value to a high power, for example, $x'(n) = x^4(n)$.

2. Compare the power of every sample in $x'(n)$ with a threshold and pick out the points with higher power than the threshold.

3. Assemble peak points and regard the last point of every group as a raw peak point.

4. Discard some redundant peak points if the distance between two consecutive peak points is not enough for embedding one bit and get a set of effective peak points.

Power points extraction is similar to salient points extraction, but with different energy rules. Therefore, those power points have the same properties as salient points and some displacements will often occur. Moreover, power points in audio signals are more ambiguous than the ones in speech signals, so that less synchronization points are extracted in audio watermarking schemes. Therefore, in our experiments, we prefer salient points extraction rather than PPE.

### 6.3 Envelope Peaks Extraction

Audio envelope roughly reflects the tendency of the amplitude variation, which is usually relatively invariant to change in the time and frequency domain compared with the original audio samples [47]. Therefore, the peaks on the audio envelope are picked out and taken as the starting points of embedding frames. It is reported that by embedding the stamp in those frames, the scheme is safer under certain degree of time stretching attacks.

The audio envelope can be extracted by sliding window. First, let a sliding window whose increment is one move along the full-wave rectified signal. Then, the means of every portion are joined up and consist of the raw audio envelope. Finally, get the smoother envelope by denoising. An audio signal and its envelope are shown together in Figure 45.
Silence parts in audio signals are always intractable in almost all watermarking schemes, because embedding bits into mute intervals while keeping noises inaudible is extremely impracticable. Moreover, the bits embedded in silence areas are hard to decide and detection errors often occur. Thus, the silence parts in the host signal should be ‘removed’ during the process of watermarking. Not deletion, but skip. It means that we pre-process the host signal and choose the suitable regions for watermarking. Actually, it also helps in resisting zeros inserting attack, for zeros inserted may be ignored automatically. For example, some zeros will be added to the beginning or the end of the processed signal, like MP3 compression by ‘Streambox Ripper’. Then, silence removal can ‘get rid of’ these zeros incidentally. Note that except for absolute muteness, some quasi-silent parts and trifle intervals should be omitted as well. Therefore, same starting points of each period should be extracted from signals at different stages to avoid introducing unnecessary mis-synchronization.

Silence removal includes two steps. First, we divided the host signal into small fractions (6ms) and compare their power with an appropriate threshold. So silence and non-silence fractions are identified and corresponding blocks are formed. Second, on the assumption that a silence part with more than 25ms is a valid silence and a non-silence part with more than 300ms is worthy to be regarded as a valid embedding period, the useless non-silence portions are discarded and false silence portions are incorporated into vicinal non-silence parts. Consequently, some stable embedding periods are picked out, as depicted in Figure 46.
Redundancy coding such as repetitive coding and cyclic redundancy check (CRC) coding and error-correction coding like Bose-Chaudhuri-Hocquenghem (BCH) coding have different effect on various watermarking schemes. Generally, application of redundancy or error correction coding would result in a trade-off between robustness and capacity.

Repetitive coding is the simplest redundancy coding, which repeats every watermark letter for several times. For example, the watermark [1 0] after three-time repetitive coding becomes [1 1 1 0 0 0]. In decoding, decisions are based on majority principle, so that the repetitive times should be odd. Usually, three or five times are reasonable and resultful. On the whole, repetitive coding is more effective in patchwork watermarking, such as cepstrum domain and wavelet watermarking schemes. Because repetition makes the samples in a broad area possess the same statistical characteristic, the detection rate is guaranteed.

BCH code belongs to convolutional code and could correct multiple random errors in the received codes. Binary BCH code is often in form $(n,k,t)$, where $n$ is codeword length, $k$ is message length and $t$ is the error-correction
capability. The larger \( n \) is, the larger \( t \) is. Therefore, in order to correct more errors, we should define a bigger \( n \). But it is obvious that the capacity will fall down at the same time. Roughly speaking, if the BER is less than 10\%, BCH coding could be utilized to correct a certain amount of errors. Otherwise, it is needless to correct quite a lot of errors on the expense of capacity.

6.6 Coded Image Enhancement

As mentioned above, coded image as watermark has several distinct advantages over chaos or \( PN \) sequence. Usually, the coded image consists of one or more words with special meaning and works as the unique logo, for example, \textbf{ROBUST}. It conveys message straightforwardly and is more suitable for the purpose of copyright protection. The extracted watermark can be clearly recognized even under noisy background. Better than depending on the similarity between the original and the extracted pseudo-random sequence, visual identification of ownership logo is more reliable. Moreover, scrambling on the coded image can contribute to more security. Even the cracker detects all the embedded bits, he can not know the meaning of coded image without secret keys. The last but not least, separate post-process could be done to enhance the coded image. So it has better recognizability and \textit{ROCLR} will gain an extra improvement.

However, eliminating unwanted noise points from a binary image with only ‘0’ and ‘1’ is not easy, especially in our watermarking schemes where the watermark is not elaborate enough. The normal filtering methods, such as linear filtering, median filtering and adaptive filtering, are useless here. Morphology is a technique of image processing based on shapes. In Image Processing toolbox, morphological operations include contrast enhancement, noise removal, thinning, skeletonization, filling, and segmentation. For example, small isolated objects could be removed by \texttt{bwareaopen(.)} command, which is a way of size thresholding. That is to say, this technique will remove each segment containing fewer than a given threshold number of pixels from the image [51]. An example is showed in Figure 47. It is seen that noise B and C have been got rid of. However, it is extremely hard to erase noise point A, which is connected with image body. Skeletonization is another potential method for removing noise effectively. By checking the similarity between skeletons of watermarked signal and standard letters in database, we can get the watermark exactly right. But it is imaginable that such a procedure is quite time-consuming.

![Figure 47. A coded image as watermark](image)

A good choice is by virtue of neural network (NN). Neural network can function as an associative memory and has been used widely in character recognition problem. Its general architecture is shown in Figure 48 [52].
Therefore, we can design and train a network to identify the noisy characters on the watermark. However, before neural network correction, we need to standardize the alphanumeric characters in the coded image, like 26 capital letters of the alphabet. Each object is represented as a 5 by 7 matrix. For example, letter A and a sample logo are shown in Figure 49.

Here, we employ back-propagation neural network. The principle of standard back-propagation is minimizing sum of squared error measured at the output by gradient descent algorithm. In our case, the network will receive a 35-element input vector and finally identify the letter by responding with a 26-element output vector. Each 26-element output vector represents a letter, who has the only ‘1’ in the corresponding position and all other values are ‘0’. For instance, the output vector representing letter A is \([1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]’.

Since the extracted watermark probably has been attacked and becomes noisy, the network should be trained on both ideal and noisy vectors with noise of mean 0.1 and 0.2 added. Note that there is no optimal method to decide the number of processing units of the hidden layer. By guesswork and experience, we set 10 neurons here. If a higher accuracy is needed, the network can be trained for a longer time or retrained with more neurons in its hidden layer, or be trained on input vectors with greater amounts of noise if greater reliability are needed for higher levels of noise. Also, the resolution of the input vectors can be increased to a 10-by-14 grid. Figure 50 displays a noisy coded image ‘WATERMARK’, probably an extracted watermark, and the result from neural network. As is seen that the original coded image has been recovered successfully.
7 A ROBUST AND CONFIDENTIAL AUDIO WATERMARKING SCHEME

Besides good imperceptibility, strong robustness and high-level security are two crucial properties to an effective and practical audio watermarking scheme for copyright protection. In this section, a robust and confidential audio watermarking scheme based on Gammatone auditory filterbank (GTF), multiple scrambling and adaptive searching for the best synchronization position is presented in details.

Superior to [11] and [30], Gammatone filterbank instead of uniform filterbank is utilized to obtain non-linear subbands in the scheme, which contributes to achieving transparent perception in the process of embedding the watermark. Moreover, the system is self-secured by integrating multiple scrambling operations into the process of embedding. Not only encryption on the watermark which is a coded-image, but also randomly selecting certain subbands for embedding and further randomizing their orders of encoding, do ensure that unauthorized detection without correct secret keys is extremely difficult. In addition, adaptive searching is used to increase the accuracy of detection in the case of mis-synchronization caused by random samples cropping/inserting, pitch-invariant time stretching, and tempo-preserved pitch shifting. Experimental results of listening evaluation and robustness tests demonstrate that the scheme preserves transparent perception and strong resistance to typical attacks for audio watermarking, with minor degradation on its performance.

7.1 Embedding Method

The basic idea of the embedding method is amplitude modification on time-frequency plane, similar to spectrogram. In particulars, for every segment to be watermarked, the corresponding inaudible watermark signal is constructed in frequency domain and added to the host after transforming back to time domain. Therefore, an increase or decrease in amplitude will depend on the sign of watermark signal, which is determined by the multiplication of the coded-image bit and the predefined pseudo-random array (PRA). In the course of embedding, Gammatone filterbank is employed to partition the spectrum into nonlinear subbands and masking threshold from psychoacoustic model I is utilized to obtain transparent perception. Furthermore, multiple scrambling is emerged to secure the scheme.

Before explaining the algorithm, we first clarify some terms in use and describe the embedding structure. To keep synchronization, the watermarking scheme is aimed to be self-locked. Thereby, we divide the spectrogram of host signal into \( N_{\text{block}} \) pattern blocks, avoiding interaction of parts. At the same time, the coded-image watermark \( B_{n(m)} \) \((m=1,2,\ldots,N_{\text{wm}})\) is also separated into sub-watermarks \( B_{\text{sub}(i)} \) \((i=1,2,\ldots,N_{\text{block}})\), each with \( N_b \) bits \((N_b=N_{\text{wm}}/N_{\text{block}})\). That is to say, every sub-watermark is embedded into one block in turn. Note that the binary coded-image consists of ‘1’ and ‘0’, where ‘0’ is not appropriate for standing for sign, hence bit ‘0’ in \( B_{n} \) need to be mapped to ‘-1’. Moreover, synchronization bits \((B_s)\) that are always represented by bit ‘1’ will be embedded as well, which are crucial to the design. Furthermore, one block has \( N_u \) units and one unit fixedly consists of four FFT frames. Hence, one block has \( 4N_{\text{u}} \) frames totally. Frame refers to one segment’s frequency representation, that is, frequency coefficients. Then, we call each coefficient as a FFT bin. By Gammatone filterbank, the frame is separated into \( N_G \) non-linear subbands along the frequency axis. Note that each subband crosses the entire block. The intersection of subband and unit is slot, each of which will be assigned with a pseudorandom number (PN) and used to embed one bit. Accordingly, a pseudorandom array (PRA) is formed.
The finer element of the slot is \textit{tile}, the primitive entity for amplitude modification. Symbol \( T_{t,b} \) denotes the tile in the \( b^{th} \) subband of the \( t^{th} \) frame. These components are shown in Figure 51.

As mentioned above, here Gammatone filterbank is responsible for settling subbands, different from [11]. Gammatone filterbank (GTF) is a bank of overlapping bandpass filters, which mimics the frequency response of the human cochlea [53], wider bandwidths at higher frequencies and narrower bandwidths at lower frequencies. Figure 52 is the frequency response of a 32-channel GTF, covering 100-22050Hz band.

By bounding the carrying band by \( f_H \) Hz and \( f_L \) Hz and also defining the number of channels \( N_G \), we can get centre frequency \( f_c \) Hz and the bandwidth \( ERB \) of every subband as follows. First, overlapping spacing \( v \) is calculated by
Then the centre frequency and bandwidth of each subband are obtained by

\[
u = \frac{9.26 \ln \left( \frac{f_H + 228.7}{f_L + 228.7} \right)}{N}
\]

Then the centre frequency and bandwidth of each subband are obtained by

\[
f_c = -228.7 + (f_H + 228.7)e^{\frac{\nu}{9.26}}
\]

\[
ERB = 24.7 \times (1 + 4.37f_c)
\]

where \(1 \leq n \leq N_G\). In our experiment, \(N_G = 32\).

However, channels from GTF are usually overlapped and discontinuous, which will cause confusion in embedding bits and insufficient usage of spectrum. Here, we take the lower limits of channels as the partitions for nonlinear subbands, so that the whole spectrum is covered. It also could improve the robustness.

Another key in the watermarking is two-dimensional pseudorandom array (PRA), which is for deciding the sign of watermark signal. In the detection, we correlate PRA with watermarked signal so as to extract the watermark. As said before, one PN (‘1’ or ‘-1’) is assigned to each slot. Consequently, a PRA is formed for the pattern block. Then in every slot, its PN will be expanded to its four tiles by modulus operator \(C_r = (+1,+1,-1,-1)\). That is, if the sign of slot is positive, the signs of the first two tiles in that slot are positive and that of the last two tiles are negative. On the contrary, the signs reverse. By this way, PRA spreads through the whole plane. Here is an example. Sub-watermark \(B_{sub(i)} = [B_1, B_2]\) is intended to embed in every block and the PRA, a 6 by 3 matrix, is built as Figure 53.

![Figure 53. An example of PRA](image)

Here, five specified slots signed with ‘I’ or ‘II’ are allocated to \(B_1\) and \(B_2\) respectively and the rest blank slots signed with ‘S’ are for \(B_s\). Symbol \(\oplus\) and \(\ominus\) represents PN ‘1’ or ‘-1’ designated to that slot, and symbol \(\oplus\) means \(PN \times C_r\), i.e. PN is expanded by \(C_r\).

Note that every pattern block has the identical PRA. Since in the detection, we resort to all tiles for embedding \(B_s\) to find out the most probable beginning of each block. At the same time, different tiles for each \(B_w\) will be picked up to decide what the bit is. Therefore, the positions for embedding different bits and the values of PNs should be the same in each block. So the PRA is often created in advance. Although PRA is unalterable, the locations
for each bit and the values of PN’s are randomly set, which is a kind of secret key beneficial to the security.
As usual, the embedding cannot proceed in silence areas. A pre-selection process for silence removal is applied
on the host signal, in order to pick up regions for embedding the watermark. Then for every frame with \(N_{frame}\)
points, its watermark signal is constructed in frequency domain correspondingly. The magnitude is estimated by
psychoacoustic model I, so that the noise is inaudible to human ears. To avoid unnecessary phase distortions, the
watermark signal has the same phases with that of the host segment. As for the sign, we combine watermark \(B_w\)
with \(PRA\) to determine the sign of each FFT bin. It means the products of \(B_w\) and \(B_s\) with \(PRA\) finalize the signs.
Next, the real and imaginary parts of frequency spectrum of the watermark signal are calculated by
\[
\text{Real part} = \text{sign} \times \text{magnitude} \times \cos(\text{phase})
\]
\[
\text{Imaginary part} = \text{sign} \times \text{magnitude} \times \sin(\text{phase})
\]
It is worth mentioning that we only construct half of spectrum here, and another half is formed conforming to
principle of conjugated symmetry. By inverse FFT, the spectrum is transformed into time domain. Then, the
watermarked signal is the summation of the host signal and the computed watermark signal. Finally, segmental
watermarked signals are smoothly concatenated to form the overall watermarked audio.

The process of embedding is concluded in Figure 54.

![Diagram of embedding process](image)

To increasing security of the scheme, multiple scrambling is adopted throughout the embedding process. Firstly,
the coded-image is encrypted into incomprehensible cipher. Secondly, we randomly select \(N_s\) subbands out of \(N_G\)
subbands available for watermarking. Secondly, as said above, the amount and the value of PN’s allocated to
each slot plus the locations for each bit are randomly set as well. Finally, the order of embedding is randomized.

### 7.2 Detection Method

The detection algorithm correlates the magnitudes of all tiles with \(PRA\), so that the watermark could be
extracted from the received signal, mostly the corrupted signal. Thus, successful detection heavily depends on
pattern block synchronization, i.e. detecting the beginning of each block precisely. As we know, mis-synchronization attacks such as random samples cropping, zeros inserting, pitch-invariant time stretching,
and tempo-preserved pitch shifting will destroy synchronization information badly. To ameliorate the situation, adaptive searching for the best synchronization position is performed. Note that it is different from [30] which aims at improving the robustness against pitch shifting performed by linear interpolation and random stretching implemented by omitting or inserting a random number of samples. Actually, random stretching in [30] is what we call random samples cropping/inserting. It is less complicated than pitch-invariant time stretching which alters the timescale nonlinearly, as seen in Figure 30. Moreover, linear pitch shifting in [30] will change the time as well as its frequency. It is not tempo-preserved pitch shifting which is commonly used in attacking audio watermarking systems.

The following steps describe the whole process of extraction.

1. Apply the same pre-selection process on the received signal in order to trace the valid embedding periods.
2. Divide each period into the overlapped segments with \( N \) samples and employ hanning-window on them.
3. Get frequency spectrum of every segment via FFT and normalize the magnitude spectrum by its average.

\[
\hat{F}_i = F_i \left( \frac{1}{N/2} \sum_{n=1}^{N/2} F_{i,n} \right)
\]

where \( F_i \) is the frequency spectrum of the \( i^{th} \) segment and \( F_{i,n} \) is the \( n^{th} \) bin of \( F_i \).
4. Calculate the logarithmic magnitude spectrums as

\[
\log F_i = 20 \log_{10} \left( \hat{F}_i \right)
\]
5. Amplify the effect of modulus operator by

\[
\log \hat{F}_i = \log F_i - \log F_{i+2}
\]
6. Calculate magnitudes of all tiles in each period.

\[
Q_{t,b} = \left( \sum_{n=FL_b}^{FH_b} \log \hat{F}_{i,n} \right) \left( FH_b - FL_b + 1 \right)
\]

where \( Q_{t,b} \) is magnitude of the tile in the \( b^{th} \) subband of the \( t^{th} \) frame. \( FH_b \) and \( FL_b \) are the lowest and highest bins of the \( b^{th} \) subband, respectively.
7. Determine the \( i^{th} \) sub watermark \( B_{sub(i)} \) \((i=1,2,...,N_{block})\) using the following procedure.
   (a) Let PRA slide on the spectrogram and assume every frame in the \( i^{th} \) block is the starting frame. Then calculate the synchronization strength \( S_d \) for the \( d^{th} \) frame by cross-correlating with the \( Q_{i,b} \)s assigned to embedding \( B_r \).

\[
S_d = \sum_{k=1}^{D_k} \omega^S_k (Q_{(i,k),b} - \overline{Q}) / \sqrt{\sum_{k=1}^{D_k} (\omega^S_k (Q_{(i,k),b} - \overline{Q}))^2} \quad (d=1,2,...,4N_u)
\]

where \( D_k \) is the number of tiles assigned to \( B_r \), \( \omega^S_k \) is the \( k^{th} \) PN for \( B_r \) and \( \overline{Q} = 1 / D_s \sum_{k=1}^{D_k} Q_{(i,k),b} \).
Note that $Q_{t(k,d),b}$ used here represents the tiles chosen for embedding $B_s$, where the subscript $t(k,d)$ means the value of $t$ depends on $k$ and $d$. Take the settings in Figure 53 as an example. As seen in Figure 55, eight slots ($D_s = 8$) are for embedding $B_s$ in each block. When assuming the $d^{th}$ frame ($d=1, 2, \ldots, 12$) as the starting frame of the 1\textsuperscript{st} block, $Q_{t,b}$ for calculating $S_d$ are listed as follows.

- $d=1$, $Q_{t,b} = \{ Q_{1,1}, Q_{1,3}, Q_{3,2}, Q_{3,5}, Q_{5,6}, Q_{9,1}, Q_{9,3}, Q_{9,4} \}$
- $d=2$, $Q_{t,b} = \{ Q_{2,2}, Q_{2,3}, Q_{5,2}, Q_{6,5}, Q_{6,6}, Q_{10,1}, Q_{10,3}, Q_{10,4} \}$
- $d=3$, $Q_{t,b} = \{ Q_{3,2}, Q_{3,3}, Q_{7,2}, Q_{7,5}, Q_{7,6}, Q_{11,1}, Q_{11,3}, Q_{11,4} \}$
- $d=4$, $Q_{t,b} = \{ Q_{4,2}, Q_{4,3}, Q_{8,2}, Q_{8,5}, Q_{8,6}, Q_{12,1}, Q_{12,3}, Q_{12,4} \}$
- $d=5$, $Q_{t,b} = \{ Q_{5,2}, Q_{5,3}, Q_{9,2}, Q_{9,5}, Q_{9,6}, Q_{13,1}, Q_{13,3}, Q_{13,4} \}$
- $d=6$, $Q_{t,b} = \{ Q_{6,2}, Q_{6,3}, Q_{10,2}, Q_{10,5}, Q_{10,6}, Q_{14,1}, Q_{14,3}, Q_{14,4} \}$
- $d=7$, $Q_{t,b} = \{ Q_{7,2}, Q_{7,3}, Q_{11,2}, Q_{11,5}, Q_{11,6}, Q_{15,1}, Q_{15,3}, Q_{15,4} \}$
- $d=8$, $Q_{t,b} = \{ Q_{8,2}, Q_{8,3}, Q_{12,2}, Q_{12,5}, Q_{12,6}, Q_{16,1}, Q_{16,3}, Q_{16,4} \}$
- $d=9$, $Q_{t,b} = \{ Q_{9,2}, Q_{9,3}, Q_{13,2}, Q_{13,5}, Q_{13,6}, Q_{17,1}, Q_{17,3}, Q_{17,4} \}$
- $d=10$, $Q_{t,b} = \{ Q_{10,2}, Q_{10,3}, Q_{14,2}, Q_{14,5}, Q_{14,6}, Q_{18,1}, Q_{18,3}, Q_{18,4} \}$
- $d=11$, $Q_{t,b} = \{ Q_{11,2}, Q_{11,3}, Q_{15,2}, Q_{15,5}, Q_{15,6}, Q_{19,1}, Q_{19,3}, Q_{19,4} \}$
- $d=12$, $Q_{t,b} = \{ Q_{12,2}, Q_{12,3}, Q_{16,2}, Q_{16,5}, Q_{16,6}, Q_{20,1}, Q_{20,3}, Q_{20,4} \}$
- $d=13$, $Q_{t,b} = \{ Q_{13,2}, Q_{13,3}, Q_{17,2}, Q_{17,5}, Q_{17,6}, Q_{21,1}, Q_{21,3}, Q_{21,4} \}$

(b) Select the frame providing the maximum $S_d$ ($S_d^{\text{max}}$) is selected as the beginning frame, $t_{\text{sync}}$. It is the position that PRA should match best with the tiles allocated to this block. This procedure is called pattern block synchronization.

\[
    t_{\text{sync}} = \arg \left( \max_{d=1}^{4N_s} S_d \right)
\]

where $N_s$ is the number of frames per block.

However, it is not always like what is expected. As is known, without attacks, $t_{\text{sync}}$ must be one or two. Due to attacks, $t_{\text{sync}}$ of some blocks will shift and misalign to the usual position. The situation becomes worse.
when serious mis-synchronization occurs. From the experience, $t_{\text{sync}}$ should be larger than a certain threshold, otherwise it is useless. Take cropping $n$ frames from the watermarked signal as an example. After cropping, $PRA$ embedded in the watermarked signal is actually shifted backward along time axis. Keep in mind that every block has the same $PRA$. Therefore, when we correlate the current tiles with the original $PRA$, $t_{\text{sync}}$ would be $(4N_u-n+1)$, which is bad for bit detection. Moreover, if not be corrected immediately, it will lead to failure of synchronization of the subsequent blocks. So during all blocks’ synchronization, $t_{\text{sync}}$ need to be automatically updated forward or backward by resetting it as $(4N_u+t_{\text{sync}}+1)$ or $(4N_u-t_{\text{sync}}+1)$ respectively, and eventually reach a value lower than the threshold. At last, the threshold with the maximum accumulated synchronization strength $A_S$ is the desired one.

$$A_S = \frac{1}{\sqrt{N_h}} \sum_{n=1}^{N_{\text{max}}} S_{d(\text{max})}^{(n)}$$

where $S_{d(\text{max})}^{(n)}$ refers to $S_{d(\text{max})}$ of the $i^{th}$ block. Here we borrow the name “accumulated synchronization strength” from [30]. However, the meaning of $A_S$ is different. For [30] sets several stretched detectors on some parts of the whole signal and calculates $A_S$ for each stretched detector. But $A_S$ we use is calculated by taking all blocks into consideration. In fact, the thresholds in a range often provide similar result. Therefore, the thresholds can be chosen at intervals to reduce the computation.

Except for pitch-invariant time stretching and tempo-preserved pitch shifting, the threshold can be directly set as half of the block size, $2N_u$, under most attacks. Still take cropping to illustrate how this works. We define the threshold as $2N_u$. So if $t_{\text{sync}}$ is over the threshold, it is reset as $(4N_u-t_{\text{sync}}+1)$, like shifting backward. Then $t_{\text{sync}}$ must be under $2N_u$. Moreover, the amount of this displacement must be kept for all the succeeding blocks. By this means, even there are several cropings in the corrupted signal, the right synchronization positions will not be lost.

For the timescale stretching and pitch fluctuation, the threshold is various for different signals. Hence, trial thresholds from $2N_u$ to $4N_u$ need to be tested and find the best threshold according to $A_S$’s equation. Also, to increase the detection’s accuracy, we assume the length of original signal is known. It is acceptable. Thus, time stretching and pitch shifting can be distinguished based on the length of corrupted signal. Then, positive or negative time stretching is further identified. Therefore shifting forward and backward can be specified to positive and negative shifting respectively. But for pitch shifting, we cannot differentiate them, so only backward or forward shifting could be used for both positive and negative ones. In our experiment, we adopt backward shifting.

(c) Calculate the bit strength $G_j$ for the $j^{th}$ bit ($B_j$) at $t_{\text{sync}}$.

$$G_j = \frac{D_B}{\sum_{k=1}^{D_B}} \omega_k^B \frac{(Q_{t_{\text{sync}},b} - \bar{Q})^2}{\sqrt{\sum_{k=1}^{D_B} \omega_k^B (Q_{t_{\text{sync}},b} - \bar{Q})}}$$

where $D_B$ is the number of tiles assigned to each $B$, $\omega_k^B$ is the $k^{th}$ $PN$ for $B$, and $\bar{Q} = \frac{1}{D_B} \sum_{k=1}^{D_B} Q_{t_{\text{sync}},b}$. 


Technical Report: Audio Watermarking for Copyright Protection

Note that \( Q_{i,h} \) used here represents the tiles chosen for each \( B_{\text{info}} \).

(d) Decide the value of \( B_j \) according to \( G_j \).

\[
B_j = \begin{cases} 
1 & (G_j \geq 0) \\
0 & (G_j < 0) 
\end{cases}
\]

8. Combine all sub watermarks to recover the watermark.

7.3 Experimental Results and Discussions

To assess the performance of audio watermarking scheme above, perceptual evaluation and robustness test are conducted based on the principles discussed in [5, 54, 55].

Three pieces of EBU SQAM disc tracks, track 40, 47 and 48, are utilized in our experiment. In order to embed a coded-image with 7 × 55 pixels into every host audio signal, we repeat the basic audio chips and separately recompose three test audio files to about 40s, that is, Harpsichord.wav, Bass.wav and Quartet.wav. Then, each frame has 512 samples and half overlaps the adjacent frames. A sub message with 4 bits is embedded into every block, which has 11 units and 32 nonlinear subbands. Here, 28 subbands are randomly selected for embedding, so that the probability is

\[
P^{28}(32) = \frac{32!}{28!} \approx 2.5 \times 10^7
\]

There are 11 * 28 = 308 slots totally, where 30 slots are assigned to each bit and the rests, 308 - 4 * 30 = 188 slots, are for embedding synchronization bit.

(a) Perceptual Evaluation

Informal subjective listening tests are performed to assess the quality of watermarked audio signals and ten listeners are involved. Except that one or two feel slight difference in the bass line of Harpsichord.wav and its watermarked signal, most of them are hard to distinguish the original and the watermarked audio in perception. In addition, the segmental signal-to noise ratio (SNR) between the host and watermarked signal is calculated as an objective measure, that is, 23.5dB, 27.4dB and 24.8dB respectively.

(b) Robustness Test

In robustness test, typical signal manipulations for audio watermarking are adopted to attack the watermarked signal, such as adding Gaussian noise, resampling, lowpass filtering, adding echo, MP3 compression, jtering, random sample cropping, zeros inserting, pitch-invariant time stretching and tempo-preserved pitch shifting. All experimental results are tabulated in Table 7, where BER represents bit error rate for every detection, not BER just based on correct detection in [11] and [30].

As seen from Table 7, although the performance slightly varies with different audio files, the coded-image embedded in three pieces of audio files can be extracted and clearly identified under almost every tested condition. Within expectation, the scheme is able to withstand common attacks, for example, adding noise, resampling, lowpass filtering and MP3 compression. It is worth mentioning that the scheme also possesses
excellent resistance against some kinds of malicious tampering, such as jittering, random samples cropping and zeros inserting, which belong to mis-synchronization attacks. Their bit error rates are quite attractive, approaching zero. Compared with that of the others, bit error rates of the extracted watermark attacked by time stretching and pitch shifting are a bit higher, which will be focused on in future study. In addition, it is expectative that bit error rates would be minimized further if repetitive or error correction coding is employed.

In conclusion, the proposed audio watermarking scheme based on Gammatone auditory filterbank, multiple scrambling and adaptive searching has a prominent property for copyright protection. The scheme is strictly protected and anyone without all sets of secret keys is impossible to ascertain or destroy the watermark embedded. Moreover, coded-image acting as the visual watermark makes the extraction more straightforward, better than the message represented by a sequence with random numbers. At last, the robustness against serious mis-synchronization attacks such as random sample cropping/inserting, pitch-invariant time stretching and tempo-preserved pitch shifting is improved by adaptive searching for the best synchronization position in the detection, which makes the scheme more useful in practice.

<table>
<thead>
<tr>
<th></th>
<th>Harpsichord.wav</th>
<th>Bass.wav</th>
<th>Quartet.wav</th>
</tr>
</thead>
<tbody>
<tr>
<td>BER</td>
<td>Extracted Image</td>
<td>BER</td>
<td>Extracted Image</td>
</tr>
<tr>
<td>No Attack</td>
<td>0% SECURITY</td>
<td>0% SECURITY</td>
<td>0% SECURITY</td>
</tr>
<tr>
<td>Noise (40dB)</td>
<td>0% SECURITY</td>
<td>3.13% SECURITY</td>
<td>1.82% SECURITY</td>
</tr>
<tr>
<td>Resampling</td>
<td>0.78% SECURITY</td>
<td>0.26% SECURITY</td>
<td>0% SECURITY</td>
</tr>
<tr>
<td>LP filtering (0.3 π )</td>
<td>2.34% SECURITY</td>
<td>2.08% SECURITY</td>
<td>1.56% SECURITY</td>
</tr>
<tr>
<td>Echo (0.3, 200ms)</td>
<td>0.78% SECURITY</td>
<td>1.56% SECURITY</td>
<td>1.04% SECURITY</td>
</tr>
<tr>
<td>Compression II</td>
<td>5.21% SECURITY</td>
<td>5.73% SECURITY</td>
<td>1.82% SECURITY</td>
</tr>
<tr>
<td>Jittering</td>
<td>1.30% SECURITY</td>
<td>0.26% SECURITY</td>
<td>0.78% SECURITY</td>
</tr>
<tr>
<td>Cropping (2*1000)</td>
<td>0% SECURITY</td>
<td>0.52% SECURITY</td>
<td>0% SECURITY</td>
</tr>
<tr>
<td>Inserting (2*1000)</td>
<td>0.26% SECURITY</td>
<td>0% SECURITY</td>
<td>0.26% SECURITY</td>
</tr>
<tr>
<td>PITS</td>
<td>+4% 5.47% SECURITY</td>
<td>8.85% SECURITY</td>
<td>7.55% SECURITY</td>
</tr>
<tr>
<td></td>
<td>-4% 4.95% SECURITY</td>
<td>4.17% SECURITY</td>
<td>7.03% SECURITY</td>
</tr>
<tr>
<td>TPPS</td>
<td>+4% 10.49% SECURITY</td>
<td>8.07% SECURITY</td>
<td>10.42% SECURITY</td>
</tr>
<tr>
<td></td>
<td>-4% 8.33% SECURITY</td>
<td>8.33% SECURITY</td>
<td>7.03% SECURITY</td>
</tr>
</tbody>
</table>

Table 7. Results of robustness test
8 CONCLUSION AND FUTURE WORK

This report is concentrated on the audio watermarking techniques for copyright protection. We started with psychoacoustic model for transparent perception, and then studied on general evaluation rules including subjective perceptual test and robustness test. After that, several prevalent audio watermarking schemes with some supplemental techniques are implemented and analyzed. Finally, a robust and confidential audio watermarking scheme based on Gammatone filterbank, multiple scrambling and adaptive synchronizing is presented and discussed in details.

According to the conclusions obtained from the studies above, we find that LSB watermarking and phase coding are somewhat vulnerable, because the watermark detection depends on the properties of some single points. It means that any displacement or modification will lead to unsuccessful detection. By comparison, patchwork watermarking via repetitive coding, especially cepstrum domain watermarking, has a good resistance to mis-synchronization attacks, which are more intractable than ordinary signal processing. As for wavelet domain watermarking, it can withstand pitch shifting, but fails at time stretching. On the contrary, echo hiding is relatively insensitive to time stretching, but can’t survive pitch shifting. Moreover, they all suffer from the problem of security in common. Deliberate alterations probably destroy the watermark as a consequence. So it seems that spread spectrum watermarking is superior to others on the whole, represented by the proposed scheme. It is rather robust and self-secured. However, adaptive searching is a bit time-consuming and also the behavior of the scheme varies sometimes. Therefore, more efforts will be put on this aspect afterwards. In addition, other ideas like using sinusoidal patterns [59-61] and zero-crossing rate (ZCR) [18] will be absorbed into the design of our audio watermarking scheme.

On the other hand, evaluation for audio watermarking will be amended further. For example, false alarm, the probability of declaring an unwatermarked audio as watermarked by the detector [47], need to be taken into consideration as a complementary criterion for detection rate. Moreover, new attacks often come forth, such as reverberations and reflections [61]. We should include them in the robustness test, as well as multi-attack. In addition, subject perception assessment, embedding capacity and speed are required to be more objective.

Researches on audio watermarking technique are progressing very fast and we will continue focusing on developing workable audio watermarking schemes for copyright protection.
9 REFERENCE


www.petitcolas.net/fabien/software/mpeg


http://www.jasrac.or.jp/watermark/ehoukoku.htm

http://www.TSP.ECE.McGill.CA/MMSP/Documents


http://www.ee.columbia.edu/~dpwe/resources/matlab/pvoc/
http://www.dspdimension.com

http://cnx.org/content/m11715/latest/

http://cnx.org/content/m11711/latest/


http://sound.media.mit.edu/mpeg4/audio/sqam/


   http://amsl-smb.cs.uni-magdeburg.de/stirmark/doc/index.html


Appendix: Audio Watermarking Benchmarking

A. **STEP 2000** [24]

SETP2000 is a joint international evaluation project for audio digital watermarking technology, developed by the Japanese Society for Rights of Authors, Composers and Publishers (JASRAC), Nomura Research Institute, Ltd. (NRI), International Associations of Copyright Management Societies (CISAC) and BIEM. It is the first of its kind initiated by copyright management bodies. The objective of STEP200 is “to certify the aptitude of digital watermark technologies, with a view towards promoting its utilization” met with enthusiastic responses from many technology enterprises.

The evaluation of submitted digital watermark technologies is conducted with two focuses. One is audibility, that is, whether Golden Ears and Silver Ears can detect if data watermarks have been embedded in music that is played back in a recording studio environment. The other is robustness, referring to whether the watermark can be extracted after the following various processes on host music.

<table>
<thead>
<tr>
<th>Testing Item</th>
<th>Overview of Processing Involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>D/A, A/D transition</td>
<td>Digital → Analog → Digital</td>
</tr>
<tr>
<td>Altered number of channels</td>
<td>Stereo (2ch) → mono</td>
</tr>
<tr>
<td>Down sampling</td>
<td>44.1kHz/16bit/2ch → 16kHz/16bit/2ch</td>
</tr>
<tr>
<td>Amplitude compression</td>
<td>44.1kHz/16bit/2ch → 44.1kHz/8bit/2ch</td>
</tr>
<tr>
<td>Time and pitch compression and decompression</td>
<td>Time compression / decompression: ±10%</td>
</tr>
<tr>
<td>Linear data compression</td>
<td>Pitch shift compression / decompression: ±10%</td>
</tr>
<tr>
<td>MPEG 1 Audio Layer 3 (MP3): 128kbps</td>
<td></td>
</tr>
<tr>
<td>MPEG 2 AAC: 128kbps</td>
<td></td>
</tr>
<tr>
<td>ATRAC: Version 4.5</td>
<td></td>
</tr>
<tr>
<td>ATRAC 3: 105kbps</td>
<td></td>
</tr>
<tr>
<td>RealAudio: ISDN</td>
<td></td>
</tr>
<tr>
<td>Windows Media Audio: ISDN</td>
<td></td>
</tr>
<tr>
<td>Non-linear data compression</td>
<td>FM (FM multiple broadcast, terrestrial hertzian TV broadcast)</td>
</tr>
<tr>
<td>AM (AM broadcast)</td>
<td></td>
</tr>
<tr>
<td>PCM (Satellite TV broadcast: communications satellite, broadcasting satellite)</td>
<td></td>
</tr>
<tr>
<td>Characteristic transformation of frequency response</td>
<td>FM (FM multiple broadcast, terrestrial hertzian TV broadcast)</td>
</tr>
<tr>
<td>AM (AM broadcast)</td>
<td></td>
</tr>
<tr>
<td>PCM (Satellite TV broadcast: communications satellite, broadcasting satellite)</td>
<td></td>
</tr>
<tr>
<td>Noise</td>
<td>White noise: S/N: - 40dB</td>
</tr>
</tbody>
</table>
B. StirMark for Audio [56]

StirMark for Audio is a benchmarking tool for audio watermarking schemes. It is derived from StirMark3.1, a generic tool for robustness testing of image watermarking algorithms.

The goal of StirMark Benchmark is to provide a fair environment for evaluating robust digital watermarking systems by using some standardized attacks and relative parameters, as outlines in the table.

<table>
<thead>
<tr>
<th>Attack Name</th>
<th>Description</th>
<th>used parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddBrumm</td>
<td>Adds buzz or sinus tone to the sound. The unit of three values are samples and for the frequency hertz (Hz).</td>
<td>AddBrummfrom AddBrummt0 AddBrummstep AddBrummFreq</td>
</tr>
<tr>
<td>AddDynNoise</td>
<td>This attack adds a dynamic white noise part to the samples. The given parameter sets the maximum noise value.</td>
<td>Dynnoise</td>
</tr>
<tr>
<td>AddFFTNoise</td>
<td>Adds white noise to the samples in the FFT room. The value &quot;FFTNoise&quot; sets the power of this attack to add the noise.</td>
<td>FFTSIZE FFTNoise</td>
</tr>
<tr>
<td>AddNoise</td>
<td>Adds in different steps white noise to the samples. The unity is in sample values. The value &quot;0&quot; adds nothing and &quot;32768&quot; the absolute distorted maximum.</td>
<td>Noisefrom Noiseto Noisestep</td>
</tr>
<tr>
<td>AddSinus</td>
<td>Adds a sinus signal to the sound file. With this attack you can insert a disturb signal in the frequency band where the watermark is located. The unity of the frequency parameter is hertz (Hz) and samples.</td>
<td>AddSinusFreq AddSinusAmp</td>
</tr>
<tr>
<td>Amplify</td>
<td>Changes the loudness of the audio file. For example the value &quot;100&quot; does not change the amplify and a value &quot;50&quot; means a half loudness.</td>
<td>Amplify</td>
</tr>
<tr>
<td>Compressor</td>
<td>This attack works like a compressor. You can increase or decrease the loudness of quitely passages. The unit of the threshold is decibel (dB). The &quot;CompressValue&quot; describes how the sample can be changed. &quot;2&quot; means that the loudness of all samples in the threshold will be the half. If the value is less than &quot;1&quot;, the compressor is an expander and will increase the loudness.</td>
<td>ThresholdDB CompressValue</td>
</tr>
<tr>
<td>CopySample</td>
<td>Is like the FlippSample attack but this attack copies the samples between the samples with a distance of FlippDist=.</td>
<td>Period FlippDist FlippCount</td>
</tr>
<tr>
<td>CutSamples</td>
<td>Removes samples from the audio file. If the value of &quot;Remove&quot; is &quot;10000&quot; then this attack removes every &quot;10000&quot; samples &quot;RemoveNumber&quot; samples periodic.</td>
<td>Remove RemoveNumber</td>
</tr>
<tr>
<td>Echo</td>
<td>Adds an echo to the sound file. The given value means the distance the echo.</td>
<td>Period</td>
</tr>
<tr>
<td>Exchange</td>
<td>Swaps two sequent samples for all samples</td>
<td></td>
</tr>
</tbody>
</table>
### Technical Report: Audio Watermarking for Copyright Protection

<table>
<thead>
<tr>
<th>Attack</th>
<th>Description</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExtraStereo</td>
<td>Increases the stereo part of the file. If the file does not have a stereo part (e.g., mono), then this attack does not have an effect.</td>
<td>ExtraStereo from ExtraStereoStep</td>
</tr>
<tr>
<td>FFT_HLPassQuick</td>
<td>Is like the RC-High- and RC-LowPass attack and uses the same frequencies from the parameter file. FFT window size can be set with the &quot;FFTSIZE&quot; parameter. This attack does not fade between the FFT windows so it is possible to hear knocks.</td>
<td>FFTSIZEx HighPassFreqLowPassFreq</td>
</tr>
<tr>
<td>FFT_Invert</td>
<td>Inverts all samples (real and imaginary part) in the FFT room.</td>
<td>FFTSIZES</td>
</tr>
<tr>
<td>FFT_RealReverse</td>
<td>Reverses only the real part from the FFT.</td>
<td>FFTSIZES</td>
</tr>
<tr>
<td>FFT_S tat1</td>
<td>Attack .....</td>
<td></td>
</tr>
<tr>
<td>FFT_Test</td>
<td>I will do some tests on this attack. At present I swap some samples inside from FFT.</td>
<td>FFTSIZES</td>
</tr>
<tr>
<td>FlippSample</td>
<td>Swaps samples inside the sound file periodically. It swaps every &quot;Period&quot; &quot;FlippCount&quot; samples with samples which have a distance of &quot;FlippDist&quot;. Important: Period &gt; FlippDist &gt; FlippCount! (PICTURE)</td>
<td>Periodx FlippCountx FlippDist</td>
</tr>
<tr>
<td>Invert</td>
<td>Inverts all samples in the audio file.</td>
<td></td>
</tr>
<tr>
<td>LSBZero</td>
<td>This attack sets all LSB to &quot;0&quot; (zero).</td>
<td></td>
</tr>
<tr>
<td>Normalize</td>
<td>Normalize the amplify to the maximum value.</td>
<td></td>
</tr>
<tr>
<td>Nothing</td>
<td>This attack does nothing with the audio file. The watermark should be retrieved. If not, the watermarking algorithm can be a snake oil!</td>
<td></td>
</tr>
<tr>
<td>RC-HighPass</td>
<td>Simulates a high pass filter using a resistance (R) and a capacitor (C).</td>
<td>HighPassFreq</td>
</tr>
<tr>
<td>RC-LowPass</td>
<td>Simulates a low pass filter like RC-HighPass.</td>
<td>LowPassFreq</td>
</tr>
<tr>
<td>Resampling</td>
<td>Changes the sample rate of the sound file.</td>
<td>SampleRate</td>
</tr>
<tr>
<td>Smooth</td>
<td>This attack smooths the samples. The setting sample value depends on the samples before and after the modify point.</td>
<td></td>
</tr>
<tr>
<td>Smooth2</td>
<td>Is like Smooth, but the neighbor samples are voted somewhat different.</td>
<td></td>
</tr>
<tr>
<td>Stat1</td>
<td>Attack .....</td>
<td></td>
</tr>
<tr>
<td>Stat2</td>
<td>Attack .....</td>
<td></td>
</tr>
<tr>
<td>VoiceRemove</td>
<td>Is the opposite to ExtraStereo. This attack removes the mono part of the file (mostly where the voice is). If the file does not have a stereo part (e.g., only mono) then everything will be removed.</td>
<td></td>
</tr>
<tr>
<td>ZeroCross</td>
<td>This attack is like a limiter. If the sample value is less than the given value (threshold - blue line), all samples are set to zero (green line).</td>
<td>ZeroCross</td>
</tr>
<tr>
<td>ZeroLength</td>
<td>If a sample value is exactly &quot;0&quot; (zero) then this attack inserts more samples with the value &quot;0&quot; (zero). (PICTURE)</td>
<td>ZeroLength</td>
</tr>
<tr>
<td>ZeroRemove</td>
<td>This attack removes all samples where the value is &quot;0&quot; (zero).</td>
<td></td>
</tr>
</tbody>
</table>
C. SDMI Evaluation

Secure Digital Music Initiative (SDMI) is an international consortium aiming at developing open technology specifications that protect the playing, storing, and distributing of digital music.

In “SDMI portable device specification Part 1 (Version 1.0)”[57] and “Call for Proposals for Phase II Screening Technology (Version 1.0)”[58], digital watermarking technology that is able to identify unauthorized content is highly proposed and key technical factors for evaluation in accordance with the claimed performance are included. Inaudibility of the watermark embedded is evaluated during Listening Test 1 and 2. As for robustness requirements, the watermarking scheme is expected to withstand each of the following signal processes when applied to the audio files at 44.1 or 48kHz, 16 bit.

<table>
<thead>
<tr>
<th>Signal Process</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D/A, A/D</td>
<td>D/A, A/D, converting twice</td>
</tr>
<tr>
<td>Equalization</td>
<td>Typical case: 10-band graphic equalizer with the following characteristics</td>
</tr>
<tr>
<td></td>
<td>Freq.[Hz] 31 62 125 250 500 1k 2k 4k 8k 16k</td>
</tr>
<tr>
<td></td>
<td>Gain [dB] -6 +6 -6 +6 -6 +6 -6 +6 +6</td>
</tr>
<tr>
<td>Band-pass filtering</td>
<td>100Hz – 6kHz, 12dB/oct.</td>
</tr>
<tr>
<td>Linear speed change</td>
<td>+/- 10%</td>
</tr>
<tr>
<td>Codecs (at typically used data rates)</td>
<td>ISO/IEC 13818-7:1997 (“AAC”)</td>
</tr>
<tr>
<td></td>
<td>ISO/IEC 14496-3:1999 (MPEG-4 AAC with Perceptual noise substitution)</td>
</tr>
<tr>
<td></td>
<td>ISO/IEC 11172-3:1993 Layer III (MPEG-1 Audio Layer 3 “MP3”)</td>
</tr>
<tr>
<td></td>
<td>Q-Design</td>
</tr>
<tr>
<td></td>
<td>Windows Media Audio</td>
</tr>
<tr>
<td></td>
<td>Twin-VQ</td>
</tr>
<tr>
<td></td>
<td>ATRAC-3</td>
</tr>
<tr>
<td></td>
<td>Dolby Digital AC-3 ATSC A_52</td>
</tr>
<tr>
<td></td>
<td>ePAC</td>
</tr>
<tr>
<td>Noise addition</td>
<td>Adding white noise with constant level of 36dB lower than total averaged music power (S/N: 36dB)</td>
</tr>
<tr>
<td>Time scale modification</td>
<td>Pitch-invariant time scaling: +/- 4%</td>
</tr>
<tr>
<td>Wow and flutter</td>
<td>0.5% rms, from DC to 250Hz</td>
</tr>
<tr>
<td>Addition echo</td>
<td>Maximum delay: 100ms</td>
</tr>
<tr>
<td></td>
<td>Feedback coefficient: up to 0.5</td>
</tr>
<tr>
<td>Down Mixing and Surround Sound Processing</td>
<td>6 channel to stereo</td>
</tr>
<tr>
<td></td>
<td>SRS</td>
</tr>
<tr>
<td></td>
<td>Spatializer</td>
</tr>
<tr>
<td></td>
<td>Dolby Surround</td>
</tr>
<tr>
<td></td>
<td>Dolby Headphone</td>
</tr>
<tr>
<td>Sample Rate Conversion</td>
<td>48 kHz to 44.1 kHz</td>
</tr>
<tr>
<td></td>
<td>96 kHz to 48/44.1 kHz</td>
</tr>
</tbody>
</table>
| Dynamic Range Reduction | Threshold: 50 dB  
16dB max compression  
Rate: 10ms attack, 3s recovery |