Digital Audio Watermarking

E4810 – Digital Signal Processing

December 12, 2002

Paul Chen

David Hoffman

For our project, we chose to investigate digital audio watermarking because we were both interested in learning more about how data can be embedded into audio samples without being perceptible. Since we wanted to do a practical project which involved more coding and hands-on work rather than pure research and concepts, we decided to focus on implementing and analyzing a specific simple algorithm for digital audio watermarking. We successfully implemented a simple encoding and decoding scheme that could be practically used to embed a watermark in any audio sample. By running extensive tests, we were able to thoroughly understand the advantages and the drawbacks to the algorithm.

Digital watermarking is a broad concept which encompasses hiding copyright information in a piece of digital multimedia such that it cannot be detected by the human senses nor can it corrupt the data in which it resides. The watermark resides permanently in the host data such as in picture, sound, or video files. In our case, we investigated digital audio watermarking. In audio data, the watermark must be inaudible and also robust against attacks and be statistically undetectable to ensure security and unwanted removal. In addition, it must also carry enough data to uniquely characterize each owner, therefore, the larger the data capacity, the better the watermark. In rendering the watermark inaudible, there is a tradeoff between data capacity and transparency, i.e. ease of detection. Lastly, the watermark should be easy to extract so that minimal time and processing power are spent authenticating the audio file.

Other watermark requirements for watermarks are application-dependent. For example, watermarks that carry arbitrary or secret data in unsuspected data fall into a class called steganography in which the watermark is not robust to attack. Sometimes watermark decoding must meet strict real-time requirements such as in video playback. In general, the watermark can be dual, requiring the original data file to detect its presence. The inherent advantage of this is that it can carry more data and be extremely resistant to attack; however, the original audio signal is not always available. Another problem that watermarks must overcome is deadlock, or in the event of multiple watermarks, it is still possible to

resolve the original copyright owner. The watermark should not be isolated to a specific location in the data nor to a separate bit stream since it can be readily removed.

Since watermarks identify a specific owner, robustness against attack is extremely important. Usually, attacks involve signal processing techniques such as filtering, adding noise, or re-sampling. Other pirates manipulate the host data by using compression, cropping, or D/A conversion. Their objective is to damage the host audio so that the watermark becomes undetectable or create a detection error by the decoding application. Given these possible attack mechanisms, an ideal watermark would be one such that the only way to damage it would be to damage the host audio itself.

There are many schemes used to encode watermarks but the optimal is one that models the human auditory system (HAS), also known as a perception-based model. The human ear can detect sounds within the 10 Hz to 20 kHz frequency range and consists of 26 band-pass filters (critical bands) in which the bandwidth increases with frequency. In this encoding scheme, the watermark is generated as a pseudo-random sequence by using two keys, one that is author-dependent and the other signal-dependent. The embedded watermark adapts to the host signal by varying its amplitude accordingly and also undergoes spectral and temporal shaping to resemble a noise-like sequence which is very difficult to remove. The model exploits the masking phenomena of the human auditory system in terms of its localized frequency and temporal detection.

Frequency masking takes advantage of the human ear's inability to detect a faint tone when competed with a louder tone which lies in the same critical band. A masking threshold is calculated which depends on the frequency, sound-pressure level (SPL), and tone-like or noise-like characteristics of the signal. Inaudible distortion to the original tone occurs when modification of the audio frequency component does not exceed the masking threshold. In temporal masking, the envelope of the host audio (decaying exponential) is used to determine the effects. The temporal mask guarantees that quiet regions are not disturbed by the watermark by spreading out the frequency-domain masking over time. Note that in both of these techniques, the masking effects are localized in its own domain.

Most of these algorithms were too complicated for us to use for this project. We decided to implement a relatively simple algorithm and explore its signal processing characteristics. In our sample algorithm, we encoded watermark data by shifting the DC level of portions of the input audio signal to a positive or negative level to indicate a 1 or 0 watermark bit respectively. A 64-bit pseudo-random watermark sequence is generated using a LFSR (linear feedback shift register) based on a 4-character user input key. The audio signal is divided into fixed-length frames and the frame mean and frame power are calculated. Each frame holds one bit of the watermark. A sequence of 64 frames contains one full 64-bit watermark sequence. The host audio signal is encoded with multiple complete watermark sequences.

To encode each bit of the watermark sequence, each frame is normalized to the DC level by taking the original signal and subtracting the frame's mean. Based on the watermark bit, the DC level of each frame is shifted by a value proportional to the frame power to make it inaudible. The exact shift value is determined by the following condition:

If the watermark bit is 0,

$$level_0 = - \text{ bias multiplier} * \text{frame power},$$

Else if the watermark bit is 1,

$$level_1 = + \text{ bias multiplier} * \text{frame power}.$$

Note that this encoding algorithm works well with audio signals since they are generally sinusoidal and have a DC bias that is already close to zero. However, if we tried this encoding algorithm on a signal with all positive samples, the encoded signal would be noticeably different from the original.

Decoding a sequence with an embedded watermark begins by calculating the frame mean for each frame. Note that this requires knowledge of the frame length which was used to do the encoding. If the frame mean is negative, then the encoded watermark bit is interpreted as a 0 and for a positive frame mean, the watermark bit is set to 1.

The next step is to compare the decoded sequence to an original watermark sequence. First we calculate the maximum number of complete watermark sequences that could be encoded in the audio signal, E_{max}. This is determined by dividing the length of the audio signal by the length of an epoch (the

frame length multiplied by 64). Then we cross correlate the two sequences and store the resulting sequence in an array. We sort the array and keep the greatest E_{max} values. Then we calculate the average of these values, R_{mean}, using it as our metric of similarity between the two sequences.

Since we are comparing two binary sequences, the maximum amplitude of the cross correlation can never be greater than the number of 1's in the shorter of the two sequences. For our application, this will always be the 64-bit watermark sequence. We count the number of 1's in the watermark sequence, multiply the count by an acceptance factor (we chose 0.9), and use this value, R_{thresh}, as the pass/fail threshold. If $R_{mean} > R_{thresh}$, then the watermark is considered to be present. Otherwise if $R_{mean} < R_{thresh}$, then the watermark is considered invalid.

To implement and test these algorithms, we wrote the following MATLAB functions:

```
wmenc( inFile, keyChar, outFile, frameDuration, biasMultiplier )
wmdec( inFile, frameDuration )
wmcor( keyChar )
wmfft( inFile1, inFile2 )
wmfil( inFile, outFile )
```

All of these files take a .WAV file format as input. To demonstrate our algorithm, we will be working with the four audio files:

`beatles.wav` [1]

`sinatra.wav` [2]

`regina.wav` [3]

`bonnie.wav` [4]

The bracketed numbers to the right correspond to the audio tracks included with this report. To begin, we encode an audio signal using the `wmenc` function.

```
wmenc('beatles', 'paul', 'beatles_enc', 25, 0.5)
```

This encodes the characters '*paul*' into `beatles.wav` with a 25ms frame length and a 0.5 bias multiplier. After the function has completed, the resulting watermarked signal, `beatles_enc` is played [5]. Note that there is no apparent change to the audio quality. The function also displays two graphs:

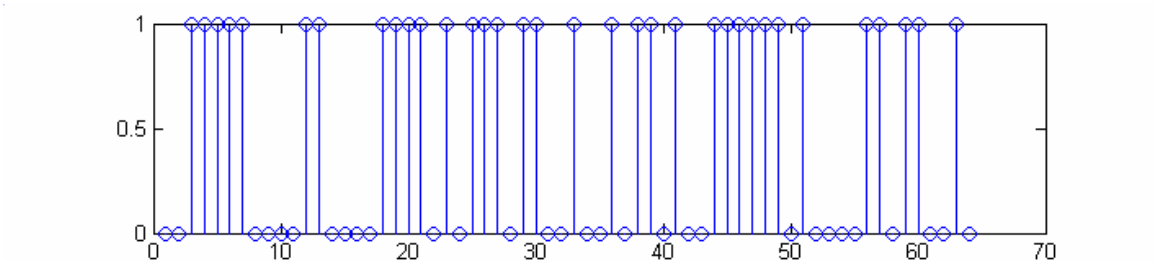

Figure 1: Watermark Sequence for 'paul'

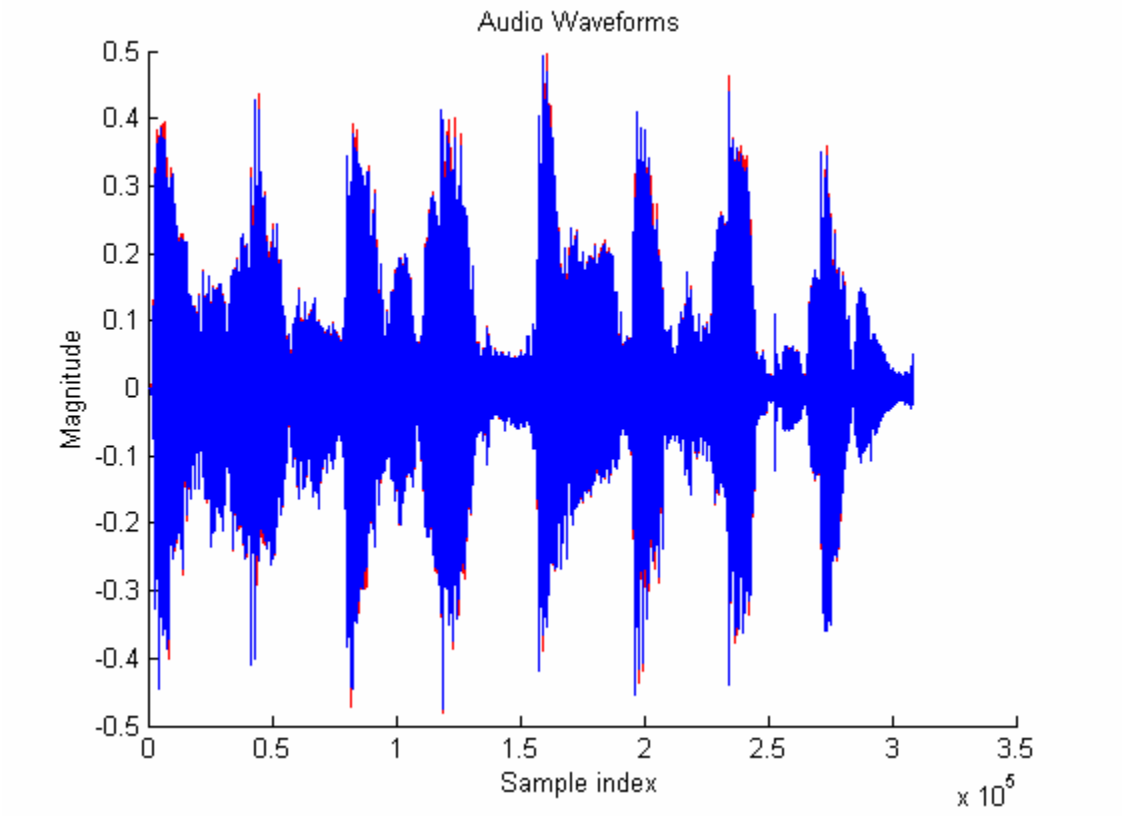

Figure 2: Original and Watermarked Audio Waveforms

The first graph shows the actual 64-bit watermark sequence representing the characters '*paul'*. The second graph is of the watermarked audio signal (`beatles_enc.wav`) in red with the original audio signal (`beatles.wav`) in blue superimposed over it. We can see a slight difference between the two signals at the envelope of the waveform.

To extract the watermark from this waveform, we use the following command:

```
wmdec('beatles_enc', 25)
```

This brings up a graph of the frame means in blue and their binary interpretation by the decoding algorithm in red. In this graph, a -0.1 corresponds to a 0 and +0.1 corresponds to a 1. If we zoom in to the beginning of the decoded sequence, we see that it matches the original watermark sequence in Figure 1.

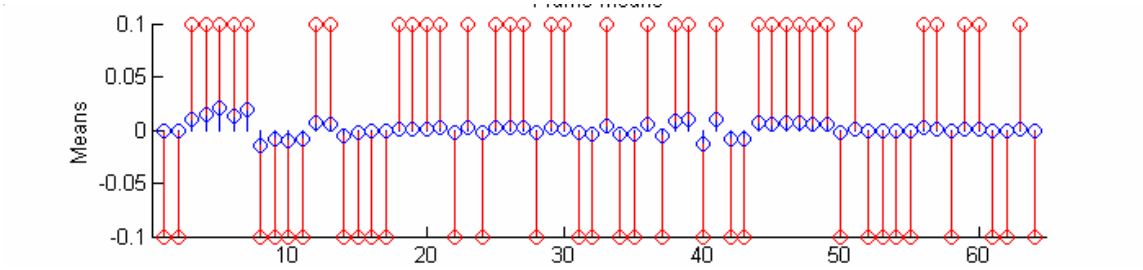

Figure 3: Decoded Watermark Sequence 'paul'

To ensure that the watermark is actually present throughout the entire signal we use:

```
wmcor('paul')
```

This brings up a graph of the cross correlation between the watermarked audio signal and the watermark sequence. The red line shows the R_{thresh} for this sequence. The four distinct maxima tell us that the watermark sequence was repeated fours times in `beatles_enc`. The function also displays a message to the MATLAB terminal informing us that it was, in fact, able to locate the watermark '*paul*' in this signal.

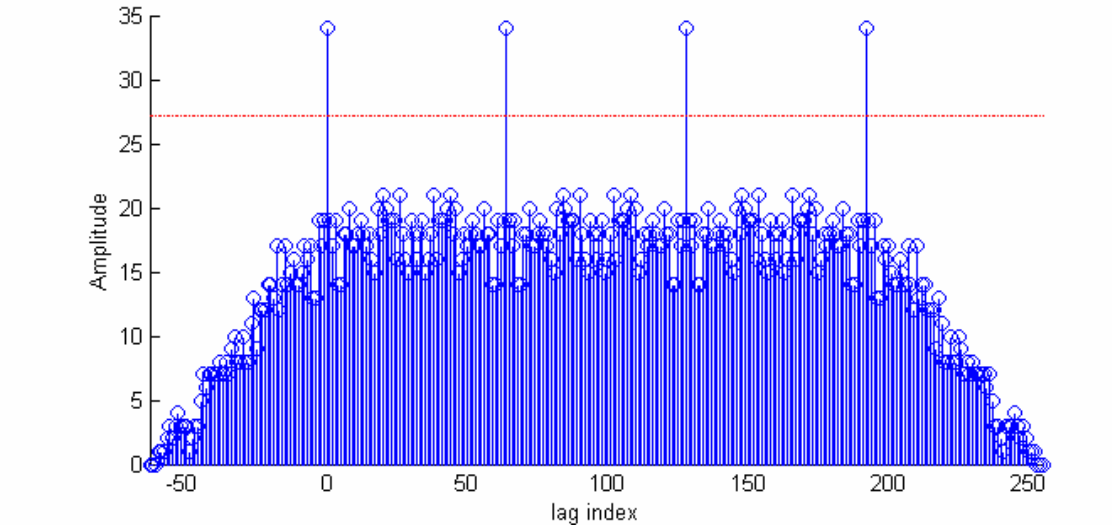

Figure 4: Cross Correlation of Encoded Audio Signal with its Watermark Sequence

If we wanted to perform an attack on this watermarking algorithm, we need to find a way to remove the watermark without compromising the audio quality of the host signal. Having a way to distinguish between the host information and the embedded watermark could be a first step. We noticed earlier that there was only a slight difference between the two signals visually in the time domain. So we turn to the frequency domain to see if this yields any helpful information. We use the following function to display the original audio signal and the watermarked audio signal in the frequency domain:

```
wmfft('beatles','beatles_enc')
```

This function uses the built-in MATLAB Fast Fourier Transform (FFT) function. Again, the watermarked audio signal is in red with the original audio signal in blue. At first glance, there only appear to be a small discrepancy between the two signals. However, if we zoom into the lower frequencies we see a very different picture.

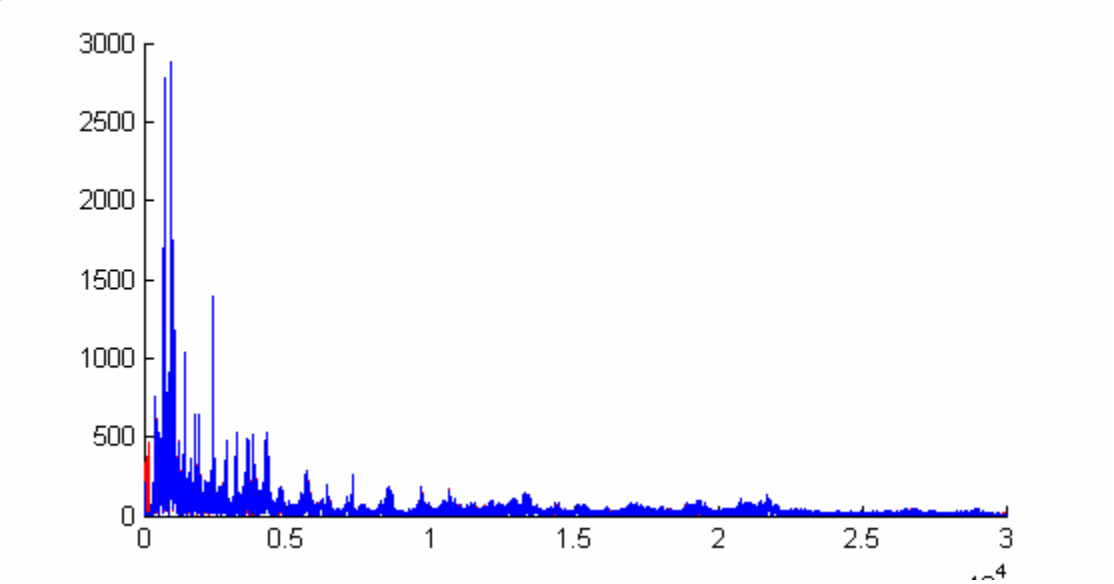

Figure 5: Original and Encoded Audio Signals in Frequency Domain

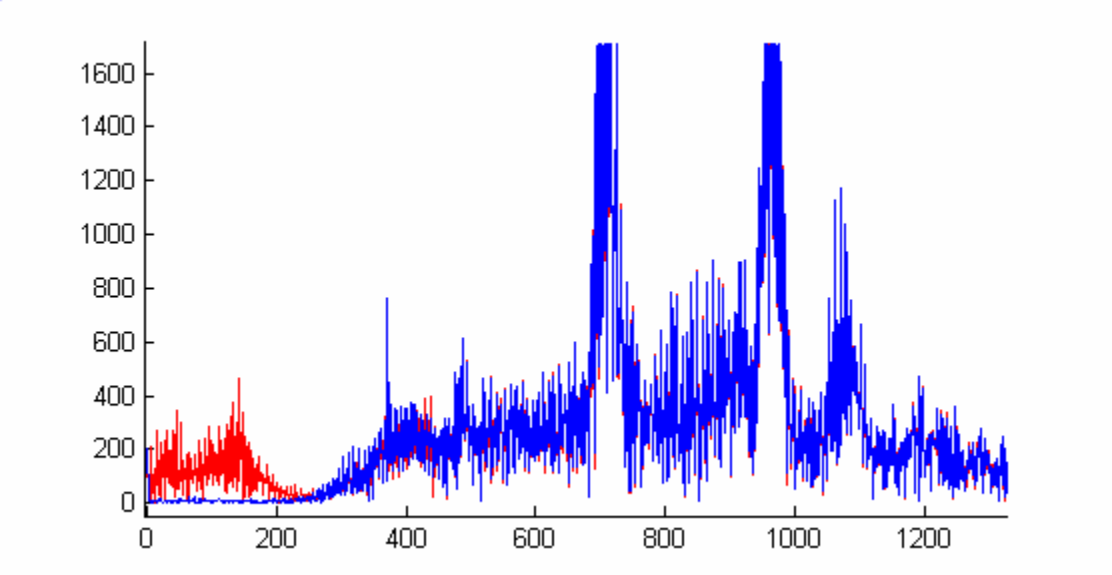

Figure 6: Close-up of the Original and Encoded Audio Signals in Frequency Domain

We see that the watermarked audio signal contains some additional low-frequency components that the original signal does not contain. This, in fact, is our watermark. If we want to remove the watermark, we simply need to design a high-pass filter and apply it to the watermarked signal. The following function performs this filtering operation:

```
wmfil('beatles_enc','beatles_enc_fil')
```

This function implements a high-pass Chebychev Type I filter. The magnitude and frequency response of the filter are shown below. After the filter has been applied, the resulting audio signal is played [6]. Again, there is no apparent change in audio quality.

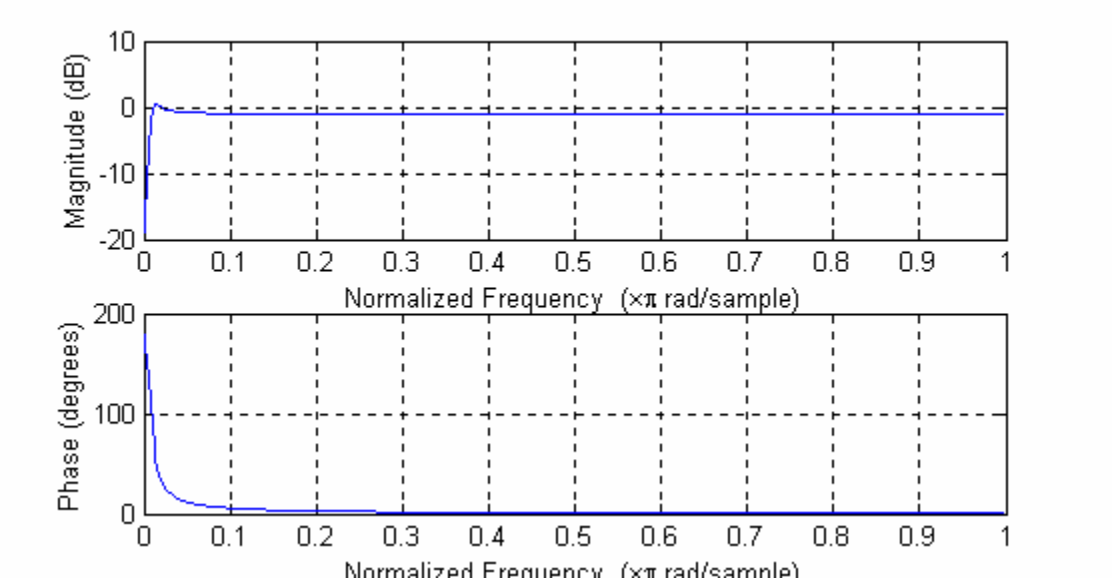

Figure 7: Magnitude and Phase Response of High-Pass Filter

Now we try to decode the watermark again with the following two commands:

```
wmdec('beatles_enc_fil', 25)

wmcor('paul')
```

The correlation function informs us that the watermark '*paul*' is no longer present in the audio file. We can see this is obviously true from the graph, i.e. all of the cross correlation samples are well below R_{thresh}. Thus, we have successfully removed the watermark without damaging the audio signal.

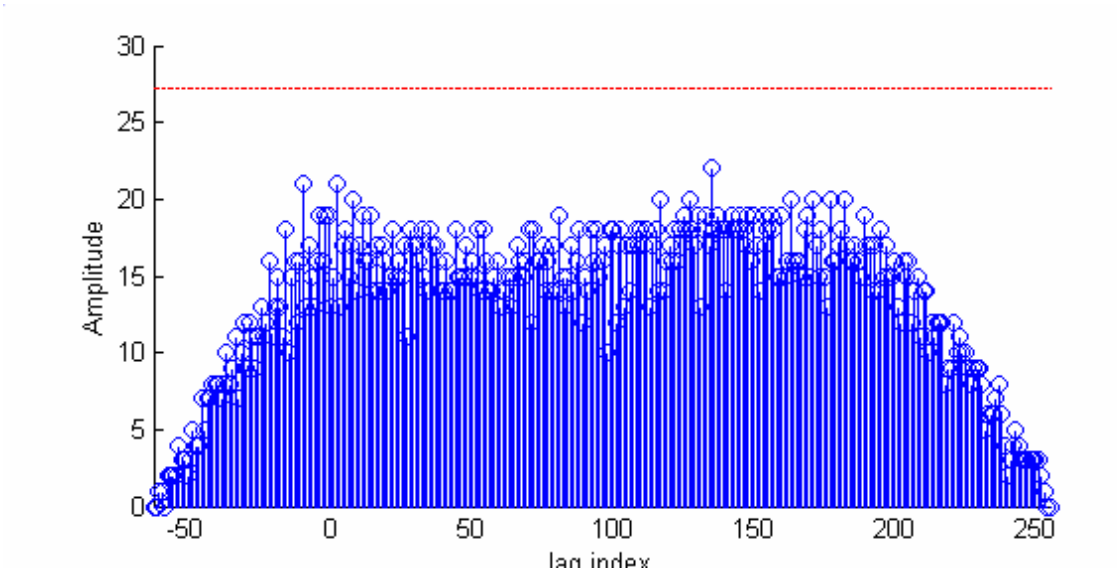

Figure 8: Cross correlation of Filtered Audio Signal with its Watermark Sequence

The frequencies at which the watermark resides in the encoded signal are a function of the watermark sequence and the frame length. Since we are shifting the DC bias of the original signal in almost every frame (assuming an essentially random binary watermark sequence), the result is additional low frequency components centered around 1/(frame duration) in the encoded signal.

If we use a smaller frame size with the following commands the audio quality become increasing tinny and flat over the bass notes:

```
wmenc('sinatra', 'abcd', 'sinatra_enc20', 20, 0.5)     [7]

wmenc('sinatra', 'abcd', 'sinatra_enc10', 10, 0.5)     [8]

wmenc('sinatra', 'abcd', 'sinatra_enc5', 5, 0.5)       [9]
```

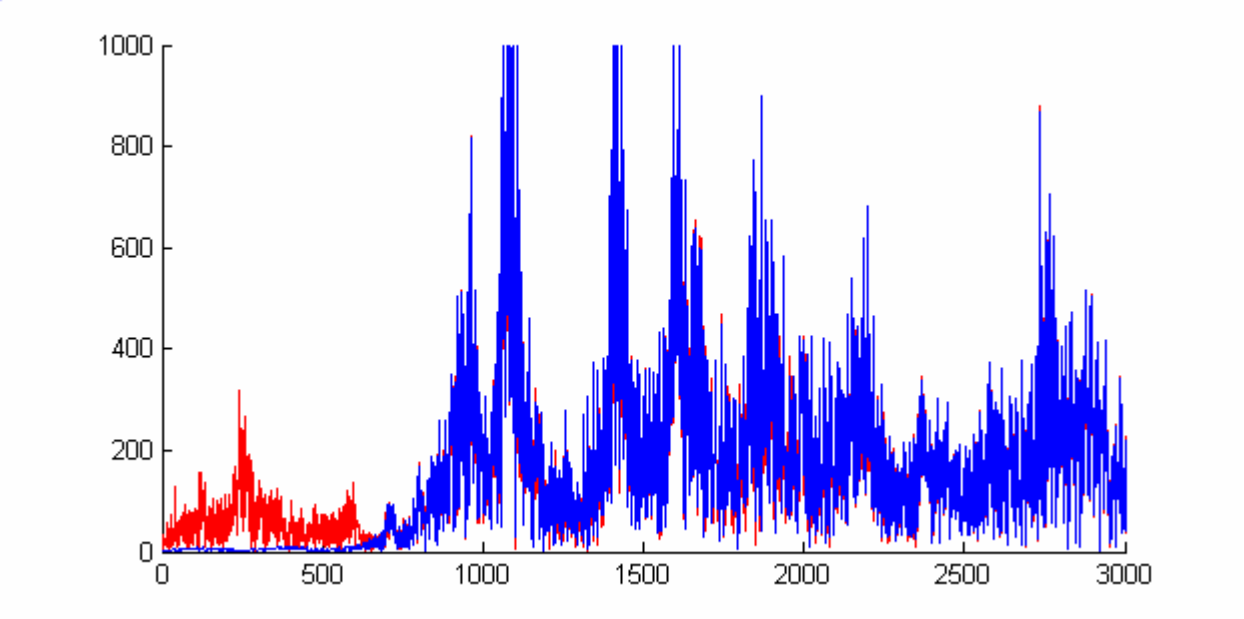

Figure 9: Frequency Domain Plot of Encoded Signal with 20ms Frame

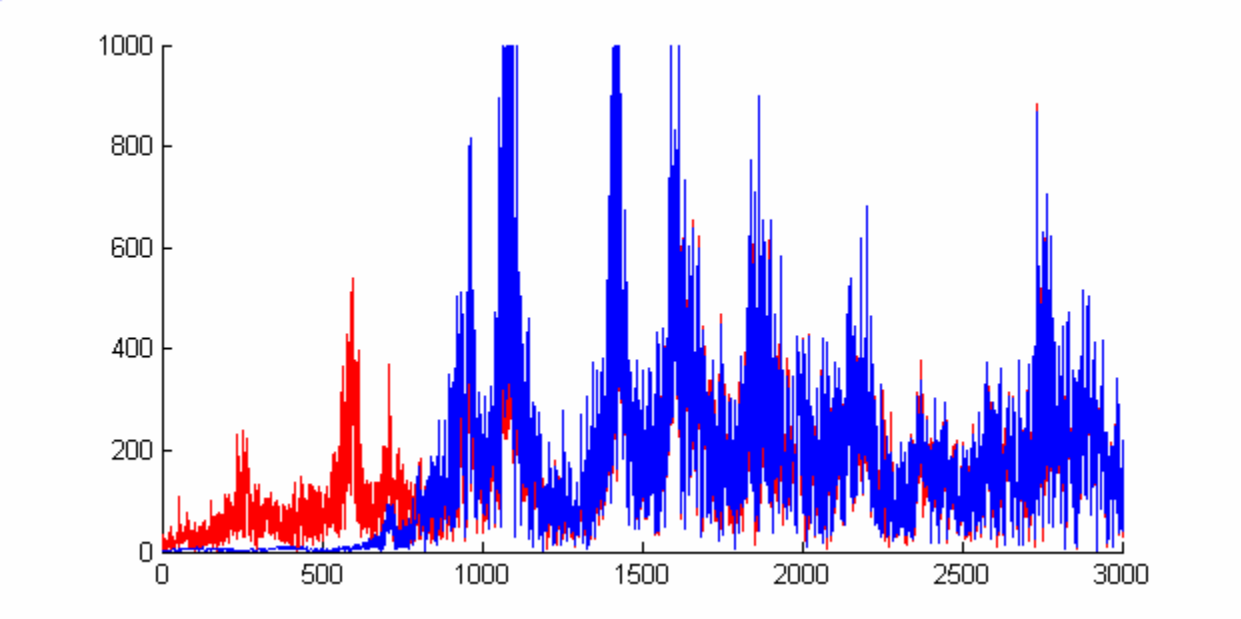

Figure 10: Frequency Domain Plot of Encoded Signals with 10ms Frame

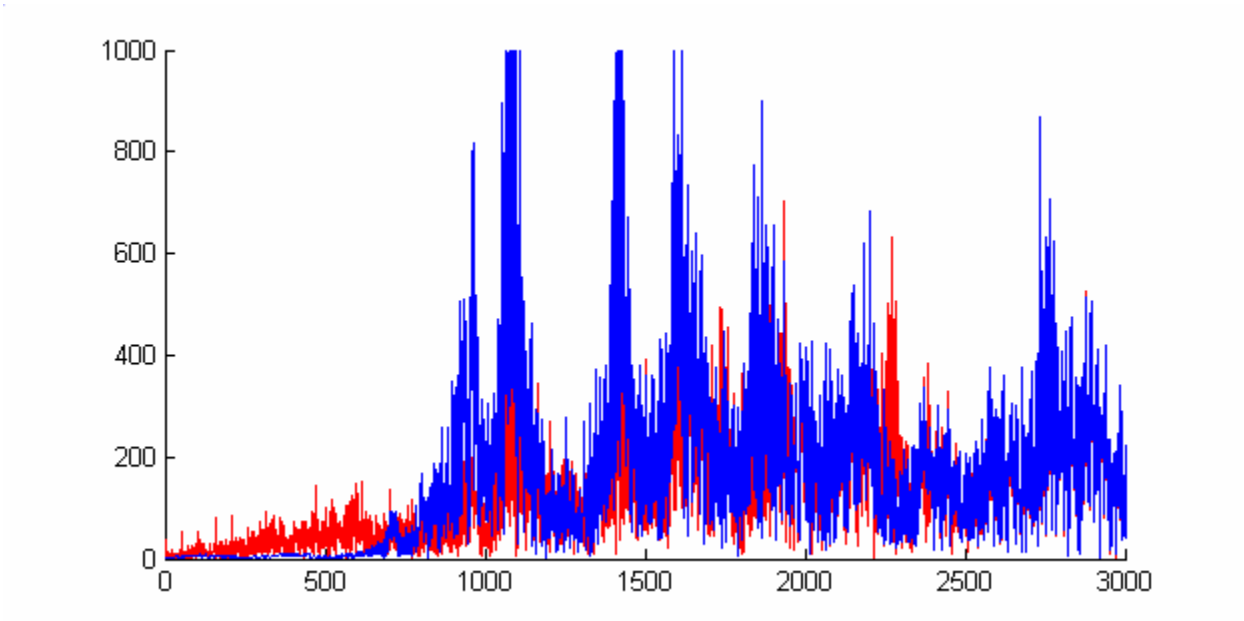

Figure 11: Frequency Domain Plot of Encoded Signals with 5ms Frame

As the frame size is decreased, the watermark data moves up the frequency axis and collides with more of the original audio signal.

Another form of watermark attack is to resample or re-encode the signal in a different format. For a watermark to survive such an attack, the watermark must have sufficient signal strength. For our algorithm, we will use a higher bias multiplier to add more strength to the signal:

```
wmenc('regina', 'wxyz', 'regina_enc2', 25, 2)          [10]
wmenc('regina', 'wxyz', 'regina_enc4', 25, 4)          [11]
wmenc('regina', 'wxyz', 'regina_enc8', 25, 8)          [12]
```

The increased multiplier causes clipping which results in a crackling static sound in the encoded audio signal. However, the signals with bias multipliers of 4 and 8 survived a conversion from .WAV to .MP3 and back to .WAV again. A similar test was performed by changing the .WAV sampling rate from 44.1 kHz down to 16 kHz. All three encoded signals survived this attack.

The last aspect of watermarking that we explored was the problem of deadlock. A deadlock occurs when one watermark is encoded on top of another one. In most cases, applying a second watermark to a signal that already has a watermark overwrites the first watermark. The following example is an interesting exception to this rule:

```
wmenc('bonnie', 'abcd', 'bonnie_enc', 25, 0.5)
wmenc('bonnie_enc', 'wxyz', 'bonnie_enc2', 25, 0.5)
wmdec('bonnie_enc2', 25)
wmcor('wxyz')
wmcor('abcd')
```

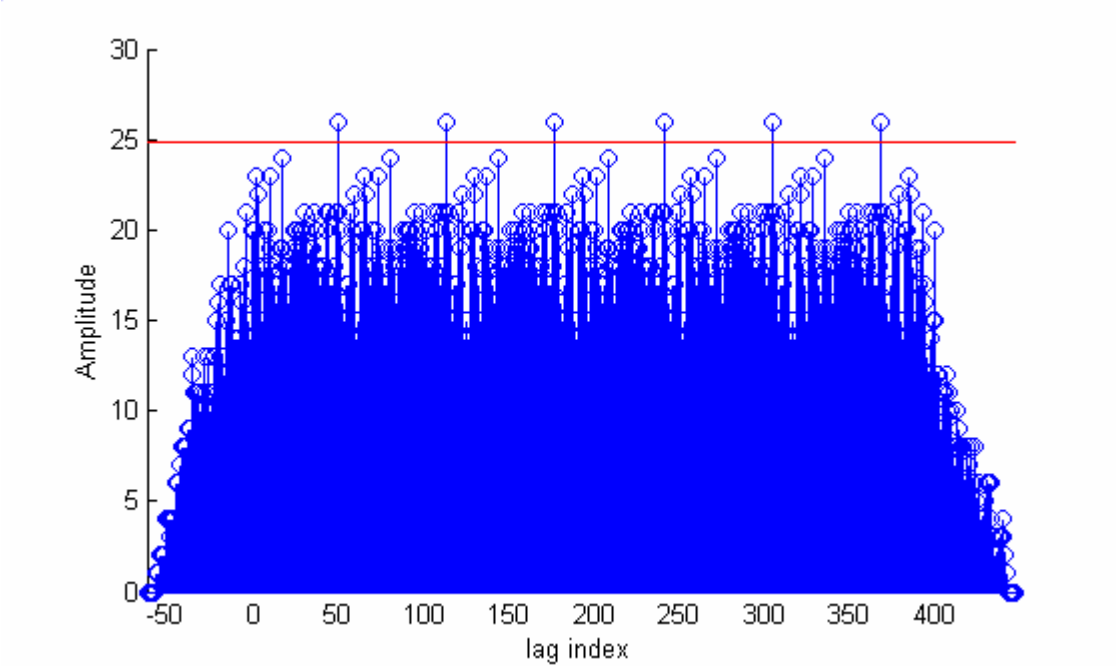

Figure 12: Cross Correlation with 'wxyz' of Sequence with Two Watermarks

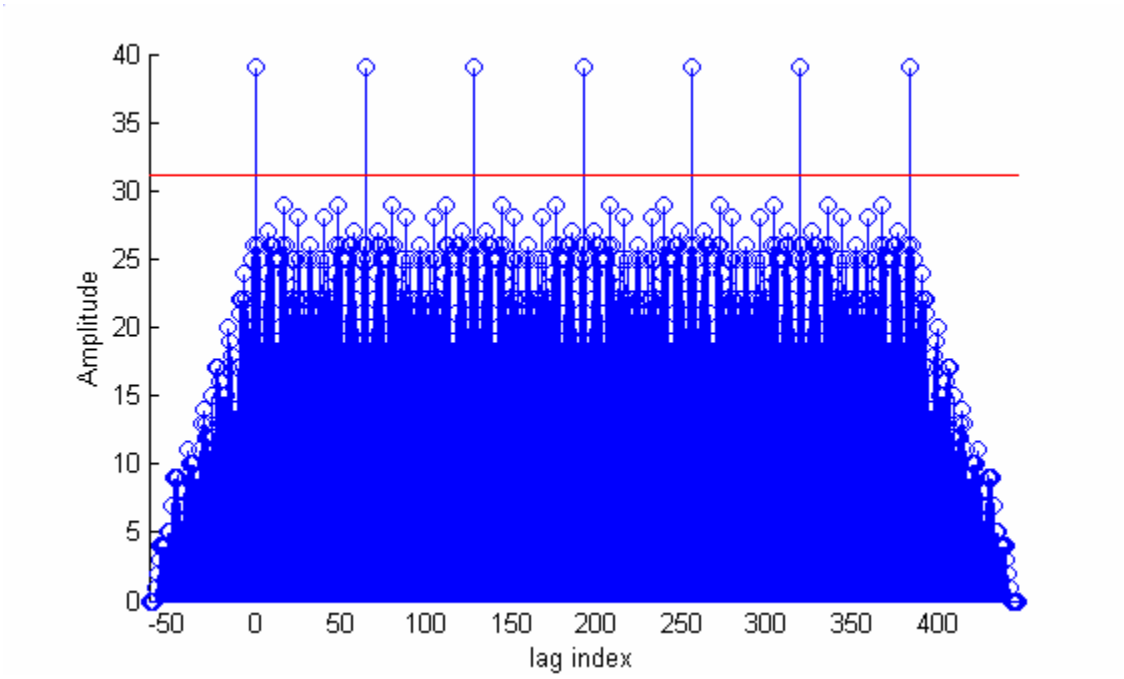

Figure 13: Cross Correlation with 'abcd' of Sequence with Two Watermarks

From figures 12 and 13, both '*abcd*' and '*wxyz*' have been encoded in the same audio signal! This situation results in deadlock since it is impossible to determine which watermark was applied first.

This project has given us the opportunity to explore some of the topics covered in class during the semester. These include time and frequency domain representations of signals, filter design, and cross correlation. Although this algorithm was relatively easy to implement, our experiments have shown that this algorithm lacks many of the features that the more robust watermarking algorithms offer. However, the algorithm did prove to be an exceptionally good introduction to the characteristics of watermarking algorithms and showed us the considerations and difficulties in designing and implementing a robust watermarking scheme that can be widely used to serve its purpose.