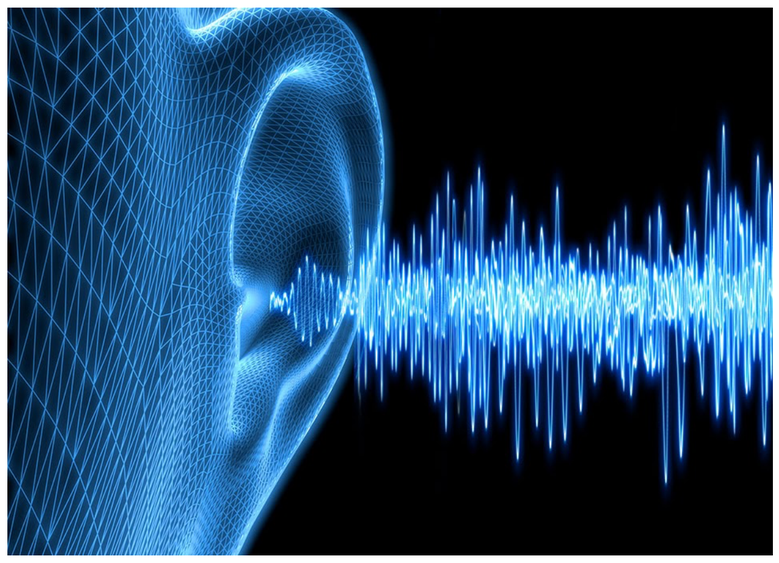
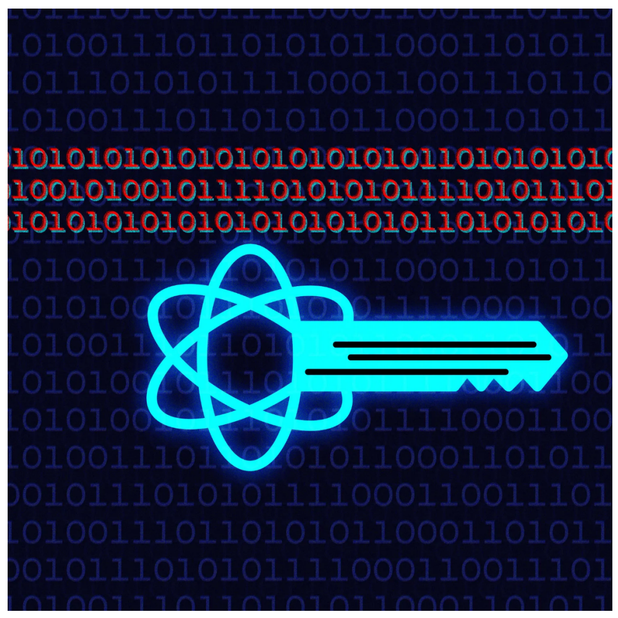
Acousto-Cryptography



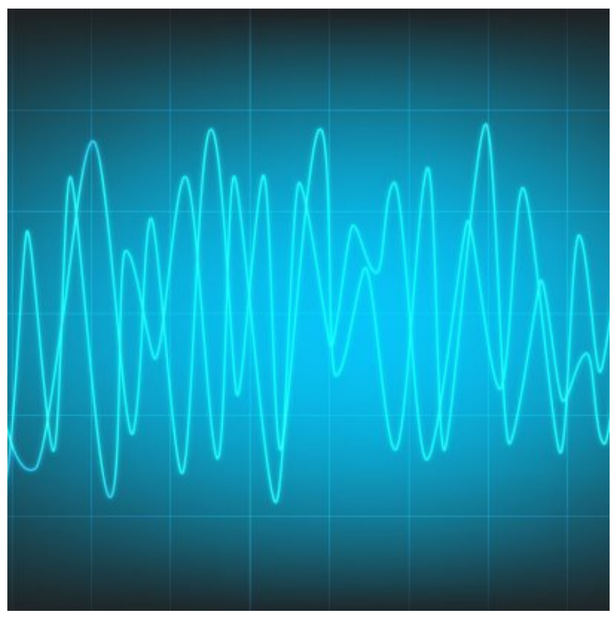
Audio Encryption

The goal of this project is to hide a secret message inside of an audio signal. The resulting encrypted signal should be indistinguishable from the original audio to a human listener. The final project is intended to be a MATLAB interface, which allows the user to enter a message, specify an audio file, and generate the encrypted file; or, enter an audio file containing an encryption and reveal the message contained.

[Data]



[Methods]



[DSP Tools]

[Progress Report]

[Final Report]

[Team/Contributions]

Image Credits

Main Image: https://tutanentertainment.com/science-of-sound/

Data Image: https://www.altlaw.co.uk/blog/the-ultimate-explainer-data-integrity-accuracy-quality

Cryptography Image: https://www.technologyreview.com/2019/07/12/134211/explainer-what-is-post-quantum-cryptography/

DSP Image: https://online.stanford.edu/courses/ee264-digital-signal-processing

Progress Report Image: https://stock.adobe.com/search?k=progress

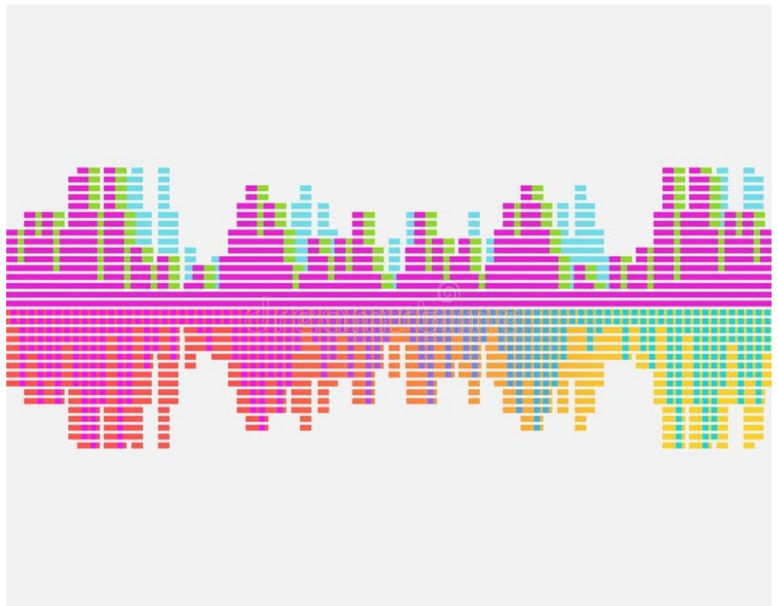
Final Report Image: https://www.perkins.org/resource/students-how-present-presentation/

Team/Contributions Image: https://www.entrepreneur.com/en-in/leadership/7-key-takeaways-in-building-a-global-team/285324

Data

Data

This project does not require much in the way of large datasets. Generally, we only need 2 signals: a secret message and a song to hide it in. Both of these are 1-dimensional signals (vectors) and the song is sampled at the music-industry standard of 44.1kHz. The secret message is either a string or character array. Neither of these signals is extremely long, which makes this project rather lightweight when compared with the memory usage of other DSP projects.



Testing Set

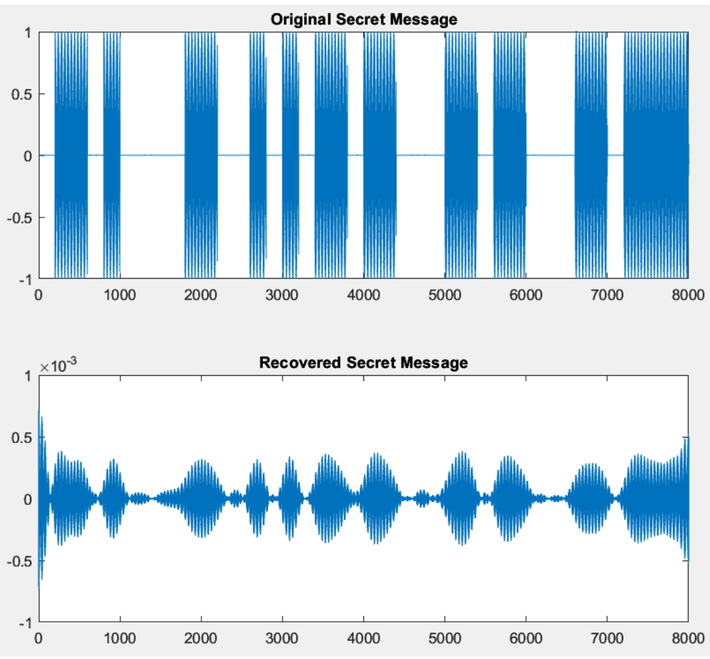
We developed a set of audio recordings to test out audio cryptography techniques. We wanted the dataset to be diverse and also have some more challenging recordings. So, we chose 10 music recordings and 10 speech recordings from the GTZAN Music/Speech dataset. We also took 10 miscellaneous recordings from the FSDKaggle2019 dataset that we thought may challenge our techniques.

Image Credits

Data Image: https://www.dreamstime.com/pixel-abstract-neon-frequency-wave-background-image269201003

Testing Set Image: https://www.videomaker.com/article/c16/7445-audio-glossary/

Amplitude Modulation



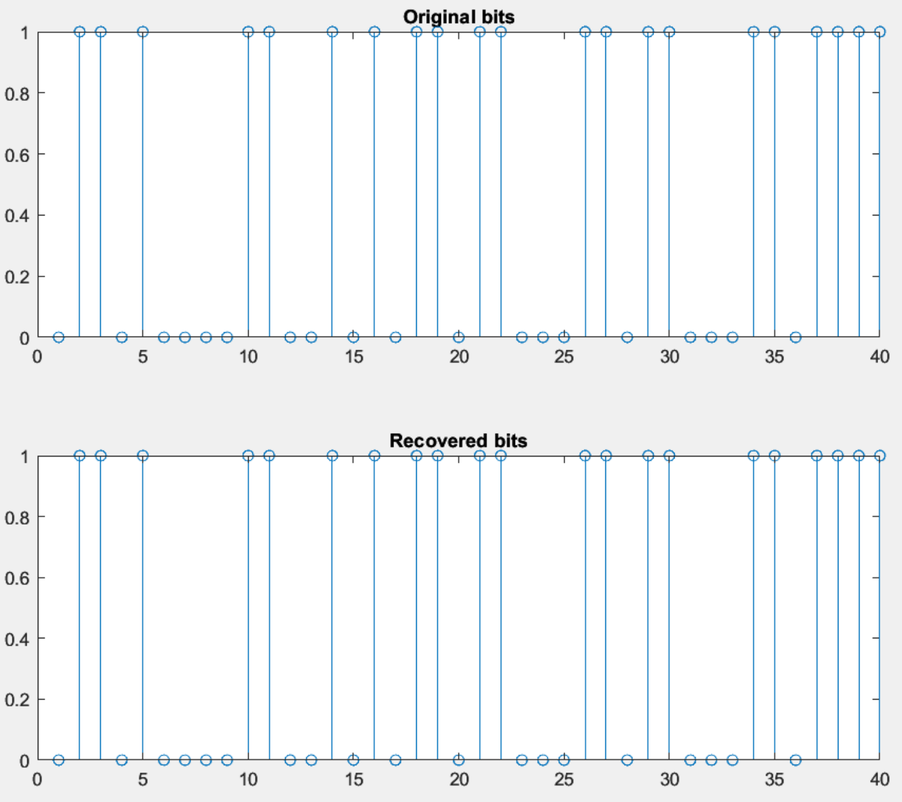
Comparison of the Amplitude-Modulation Signal generated from the binary message to the recovered sinusoid after filtering the combined audio.

Amplitude Modulation

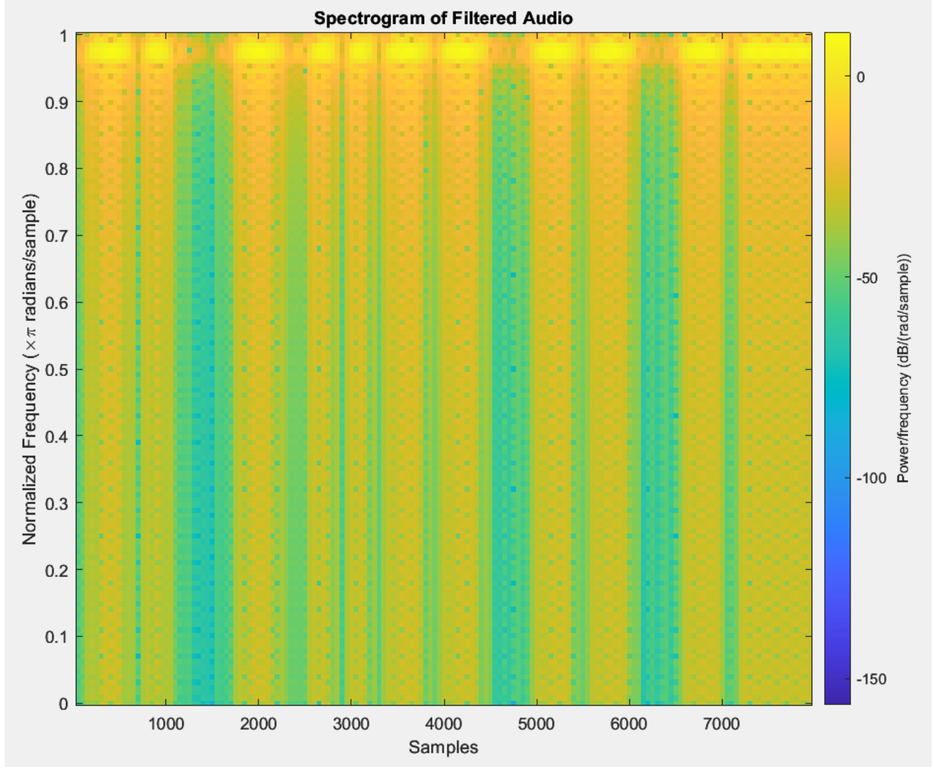
In Amplitude Modulation Encryption, the data is encoded in the amplitude of a sinusoid at a frequency higher than humans can hear. A binary 1 corresponds to high amplitude, while a 0 corresponds to low amplitude. The high-frequency signal can be added to an audio signal and because it is above the range of human hearing it remains undetected. The data is recovered by filtering away the rest of the audio and averaging the amplitude of the sinusoid

We have had moderate success in testing an Amplitude-Modulation-based encryption method. We were able to embed a signal inside a high-frequency carrier signal and insert it into an audio file. We were also able to reverse the process and extract the original signal. However, the maximum frequency we were able to use was 21.5kHz, which we were still able to hear when the encryption was played alone.

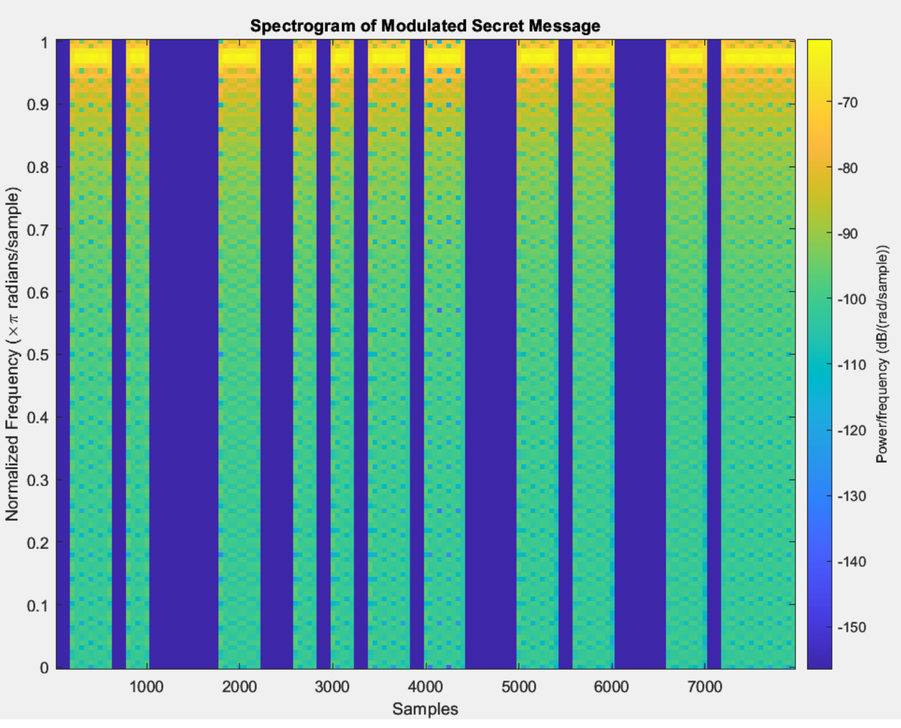
It's not all bad news though, as long as a cover audio is played overtop of the AM signal, it's very hard to make it out.



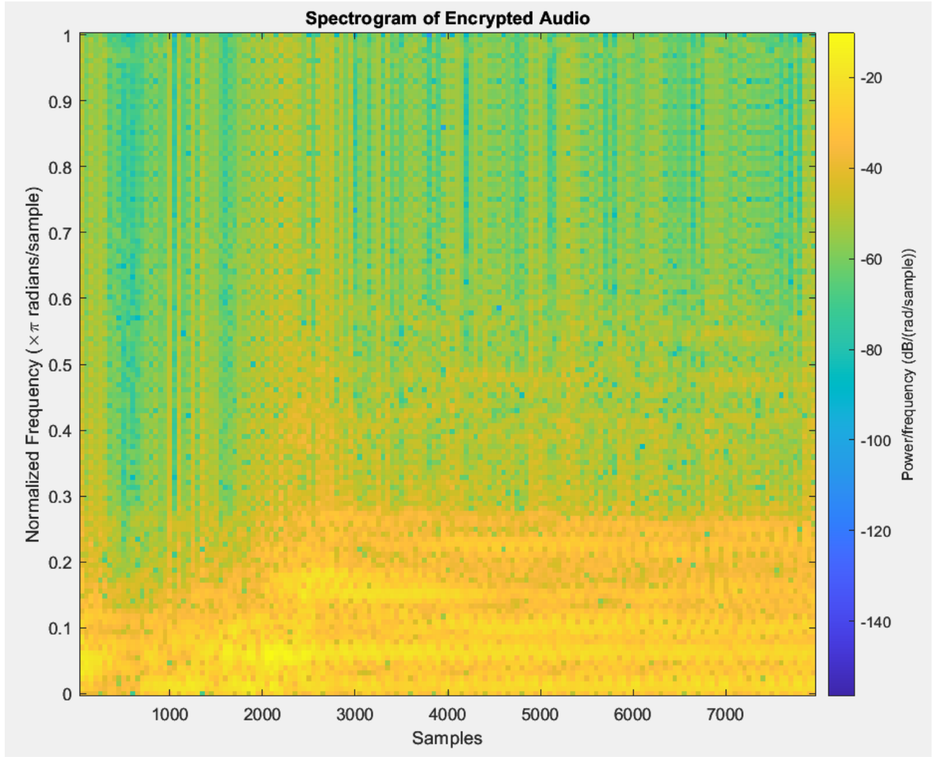
Comparison of bits from original secret message to the bits we were able to recover from the filtered audio. Notice that all bits were recovered sucessfully.



Spectrogram of the audio after filtered using a bandpass filter centered on the AM Carrier Frequency. Notice how it is very similar to the original spectrogram of the AM signal.



Spectrogram of the Amplitude-Modulated signal that gets added to the audio. This signal contains our secret message. Notice the scale, even the brighter, intense regions are in the -60 dB range. This keeps them from being easily heard when a louder song is played over top of them.



Spectrogram of the audio signal with the Amplitude-Modulated signal added to it. Notice how the frequency band containing our secret message seems rather empty. When listened to, this sounds like normal music, despite our manipulations.

Lessons Learned...

While investigating amplitude modulation, we've learned about discrete modulation. While we were exposed to continuous-time modulation in EECS216, discrete-time modulation has remained a mystery to us up till now. During our investigation, we discovered that discrete-time modulation simply multiplies each element of the signal by a shifted delta function and the discrete sinusoid at the (digital) modulation frequency. Amplitude Modulation is easily accomplished in Matlab with the ammod() and amdemod() commands.

We've also learned a little about human hearing and speaker systems. A quick Google search tells us that humans can hear sounds in the range of 20Hz-20kHz. But when we play a tone at 21.5kHz, we're still able to hear it. How can this be? We were able to come up with two reasons that might explain why we can hear the 21.5kHz signal. First, we can see on the spectrogram that even though the majority of the AM signal's power is held in the 21.5kHz band, there are lower-frequency components that we might be hearing instead. Second, the speakers on the laptop that we used to play the sounds are apparently not designed to play sounds at the edge of human hearing. In order to avoid having to design high-frequency hardware, some speaker designers simply pitch-shift the higher frequencies down so they can be played.

Audio Watermarking

Introduction

Audio watermarking refers to the process of embedding information into a signal in a way that is difficult to remove. This can be achieved in time domain (or spatial domain), frequency domain, or both (mixed domain).

The time domain methods we are investigating now include the Least Significant Bit Encoding technique, the Echo Hiding technique, and the frequency domain method spread spectrum, which requires generating some noise signals to hide the information.

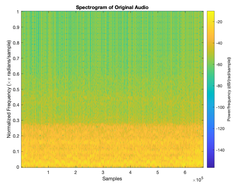
Least Significant Bit Encoding

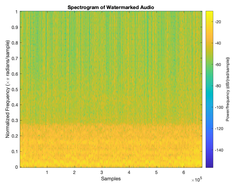
Least Significant Bit Encoding is a technique used in digital steganography, including audio watermarking, to hide information within a signal. In the context of audio, it involves modifying the least significant bits of the audio sample data to embed a watermark—often a digital pattern or message without perceptibly altering the audio to the average listener.

This is a rather straightforward way of hiding information, and the extraction process is also simply taking the least significant bits to retrieve the hidden data.

Spread Spectrum

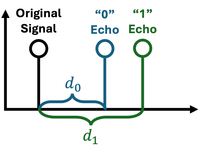
The core idea behind spread spectrum communication is to spread the signal of interest over a wide bandwidth. This spreading makes the signal less susceptible to interference and jamming. In watermarking, this principle embeds data into a host media file robustly against attempts to remove the watermark (e.g., compression, cropping, filtering).





Echo Hiding

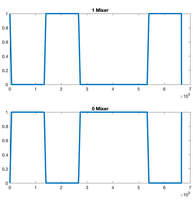
The echo hiding method is a time-domain method that embed information by creating "echoes" of different delays. When the delay times are small enough, human ears cannot resolve the created echo and the original audio as different sources, and thus the hiding is achieved.



Echo System Functions

The time delay can be easily achieved by passing the signal through a system function constructed with two (scaled) delta functions:

h[n] = δ[n] + aδ[n-d], in which d is the delay corresponding to the data bit and a is the scaler that specifies the amplitude of the echo.



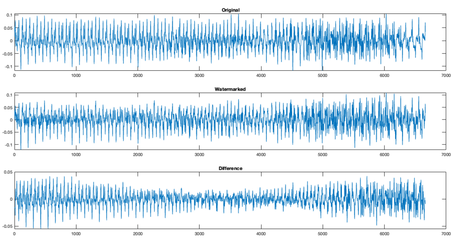
Hiding Process

To hide n-bit binary data into the original cover audio, the cover audio will be segmented into n pieces and 1 bit of data will be embedded into each of these segments with the above-mentioned system function.

To guarantee smooth transitions between each segment and the imperceptibility of the watermarked signal, two mixer signals are introduced. They filter out the segments we want for each binary bit from the data and introduce linear transition ranges to ensure smooth connection between segments. An example is shown on the left for binary data "01001".

The cover audio will first go through each of the system functions to be delayed by either delay d0 or delay d1. Then, the delayed signals will be masked by the according mixer signal before they can be added to produce the final watermarked signal.

The whole process has been coded using MatLab, and the same piece of the signal's original amplitude, watermarked amplitude, and their difference is shown below.

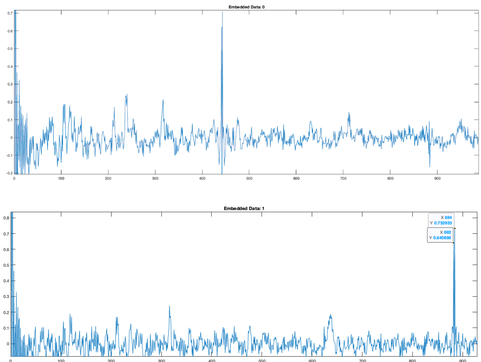


Extraction Process

To extract the hidden signal, we will calculate the cepstrum autocorrelation of the segments of the watermarked signal and identify the peaks of the autocorrelation to identify whether it's a one-echo or a zero-echo.

Below are two graphs demonstrating the different autocorrelation curves for different binary data embedded. The d0 delay we used for 0 is 441 and the d1 delay used for 1 is 882. From the graphs, it can be shown that these values are indeed peaks of the autocorrelation graph, and with these peaks, we can successfully extract the binary data embedded.

We tried different scaling factors for the system functions, and it turns out that scaling factors of over 0.5 can guarantee successful extraction of the embedded data, while for those below 0.5, only around 85% of the extracted bits are correct. This is probably because the audio we're using has some repetitive melodies which will also be captured by the autocorrelation metric.



Lessons Learned...

Autocorrelation Analysis

Autocorrelation is a mathematical operation that measures the similarity of a signal with a delayed version of itself as a function of the delay. It gives an idea of the repeating patterns or periodicity within the signal, or whether certain parts of the signal are predictable from other parts.

To calculate the autocorrelation of a signal in the time domain, we convolve the signal with the "time-flipped" version of itself.

$$Rxx = x[n] * x[-n]$$

This process is a polynomial-order operation, and the efficiency of which can be further improved (nlogn) if done while taking the cepstrum.

Cepstrum Analysis

Cepstrum analysis is a mathematical procedure used in signal processing where the spectrum of a signal is transformed to the quefrency domain by taking the inverse Fourier transform (IFT) of the logarithm of the signal's spectrum. This process effectively separates different sources of signal alterations, such as the effects of a room's acoustics on a sound signal or the vocal tract's shape on a speech signal.

Cepstrum of a signal can be obtained following the steps below:

1. Apply Fourier Transform
2. Get the complex logarithm
3. Apply Inverse Fourier Transform

Fun Fact:

cepstrum = Inverse(spec)trum

quefrency = Inverse(frequ)ency

Reference

1. Echo Hiding

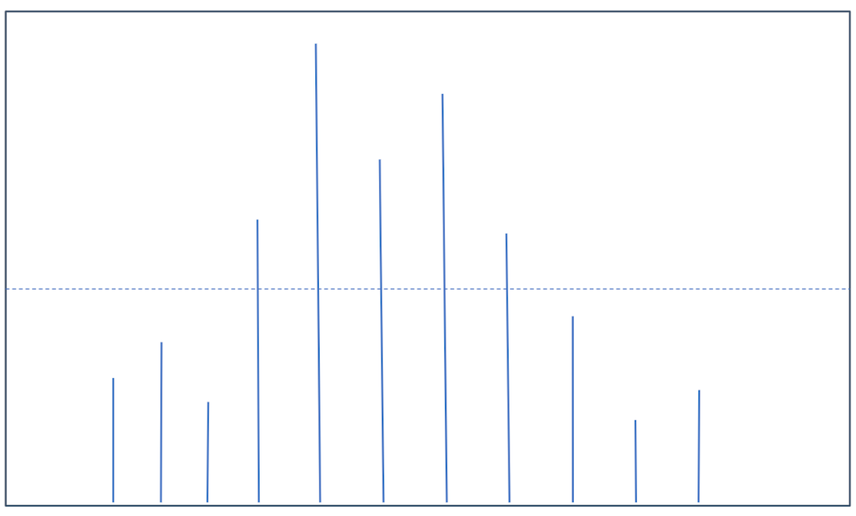
   Daniel Gruhl, Anthony Lu, and Walter Bender, Massachusetts Institute of Technology Media Laboratory

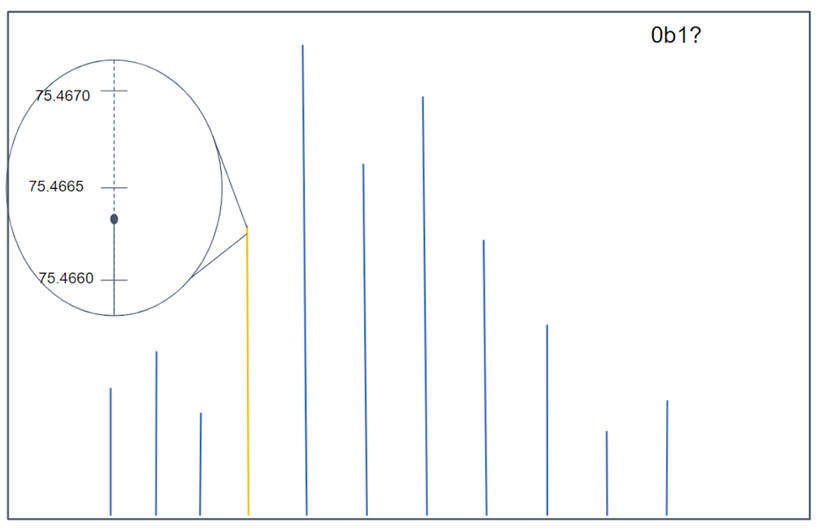
Quantization Index Modulation

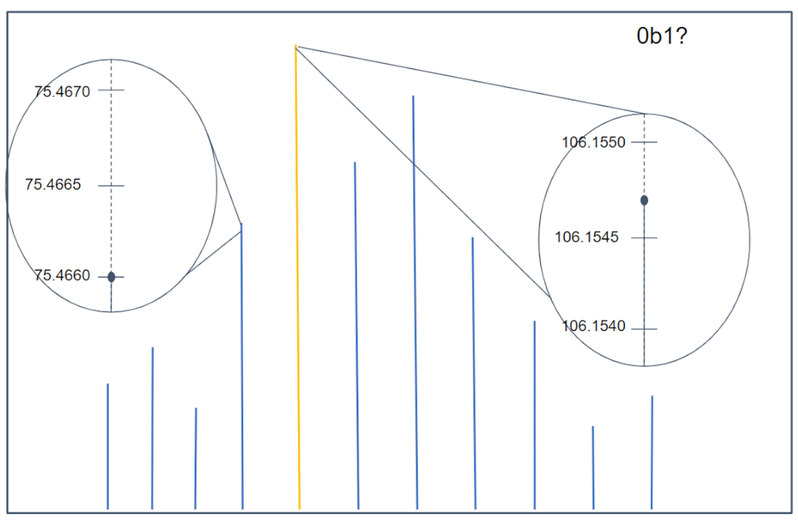
QIM Encryption

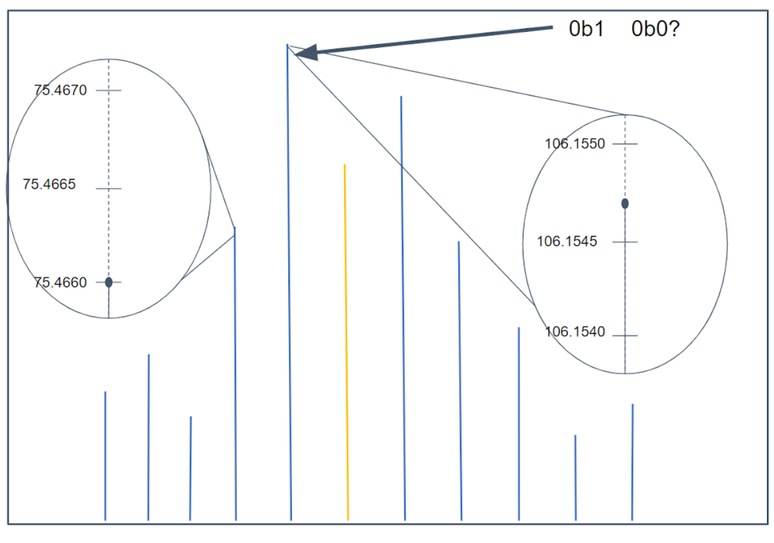
In Quantization Index Modulation (QIM), a signal is quantized by rounding to the third decimal place. Information can then be stored in the residual value. An audio signal is sampled at a rate based on the length of the message to be encoded. Values above a given threshold are analyzed and manipulated based on the encrypting message. The determining cutoff is 0.0005. Values above this are considered to be 1 and values below are considered 0. To encode the signal, the sample is iterated and quantized until a value is found that will equate to the bit you are trying to hide. It then moves on to the next value and next bit until all 8 bits of the character are hidden in the sample. To decode, the quantized value is subtracted from the actual value leaving only the residual to analyze.

How it works

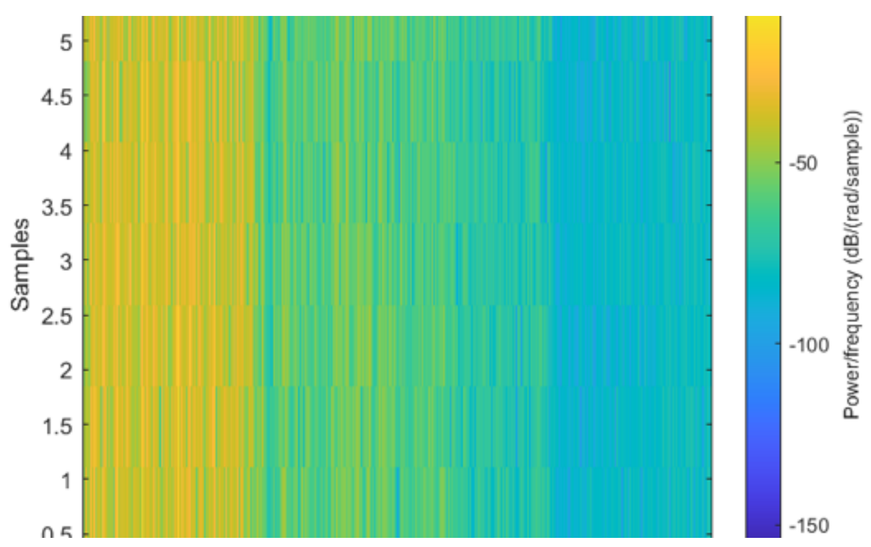




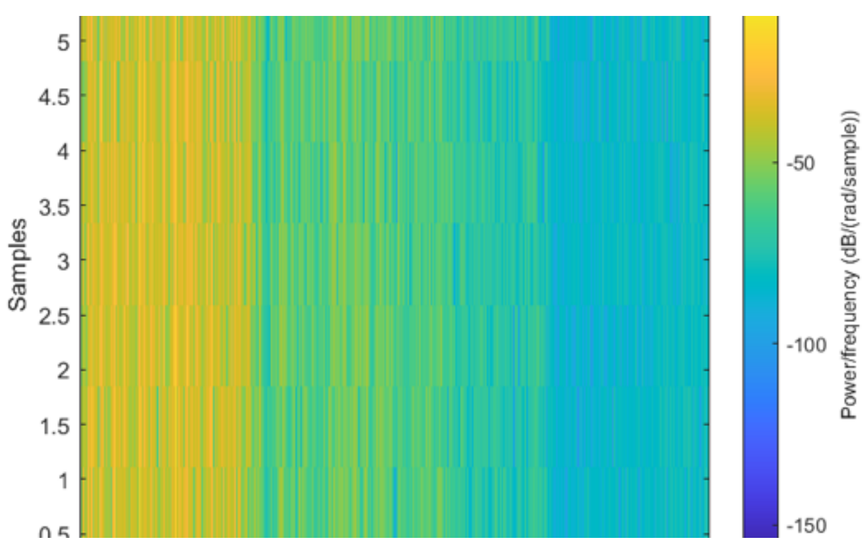




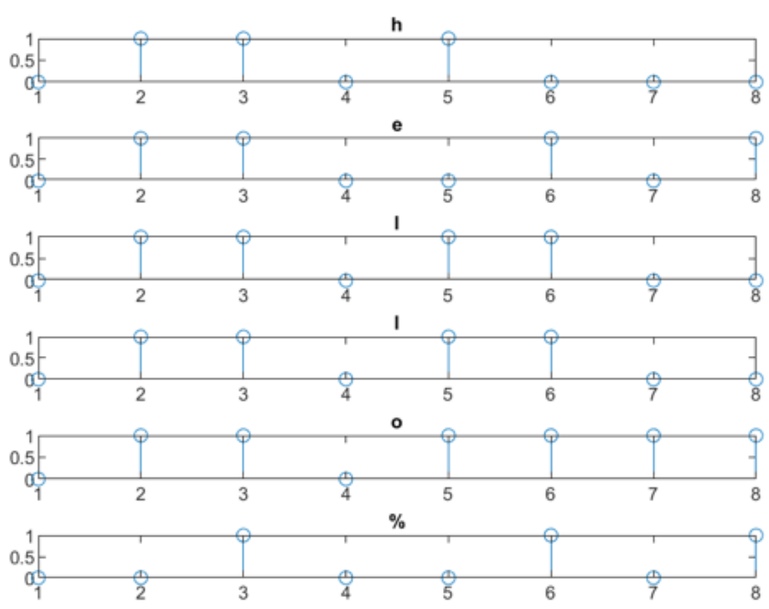
Results



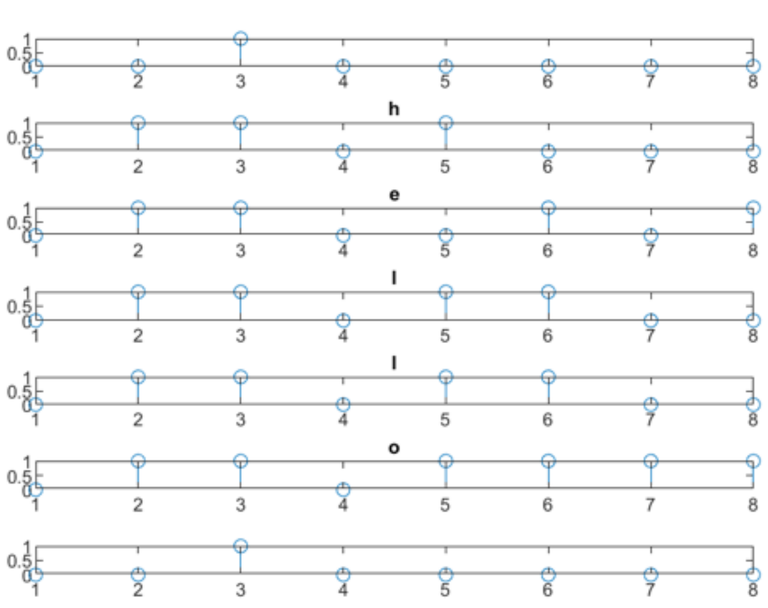
Original Signal



Encoded Signal



Message Sent



Decoded Message

Audio Comparison Method

Feature Extraction Method

One of the issues we faced was that we did not have a way to quantify how similar two audio files would sound to the human ear. Simply using the difference between audio files is not a good measurement of this similarity, so, we developed a method that used Audio Feature Extraction to compare audio files. The technique can be described in a few steps:

1. Load two audio files
2. Use the Matlab "audioFeatureExtractor" to get a set of feature vectors for each recording. Specifically, we extract the Spectral Centroid, Spectral Rolloff Point, and Spectral Flux
3. Normalize the feature vectors
4. Compute the sum of the Euclidian Distances between feature vectors corresponding to each file

This gives us a numeric way to determine how much our encryption methods were altering the sound of the audio inputs.

DSP Tools

In-Class DSP Tools

Filtering

Filtering is the process of running a signal through a system which has a gain that depends on frequency. A high gain corresponds to passing the frequency, while a low gain corresponds to filtering out the frequency. Most systems can be described using filters.

Filtering was used in the Amplitude Modulation method, where a bandpass filter centered at the carrier frequency was needed to extract the encoded data from the audio signal. The filter was designed using MATLAB's bandpass() command to have an attenuation of -60db in the stop band and linear phase in the pass band. This allows the lower frequency components that make up the covering audio to be removed, while the high-frequency secret message passes through with only a slight delay from the phase shift.

Change of Basis/Fourier Analysis

The Discrete Fourier Transform (DFT) is a change of basis into the Fourier (complex exponential) domain. The DFT changes a time-domain signal into a set of coefficients describing the different frequencies it contains. It is accomplished using FFT methods to decompose a large and computationally-intensive calculation into something more manageable.

Many of the original methods required calculating the DFT. Amplitude Modulation uses it to identify frequency bands beyond human hearing, Quantization-Index Modulation directly modifies the DFT coefficients to hide its data, and Spread Spectrum Watermarking adds a procedurally-generated noise signal into the DFT coefficients, Echo Hiding uses multiplication in the Fourier domain to delay signals.

While it was used heavily at the start of the project, it has since been dropped from all encryption methods except Echo Hiding, where the time-delaying systems are described by their DFT coefficients. Though not using the DFT directly, the spectrograms (discussed below) generated by our project have many internal calls to FFT functions.

Spectrograms

Spectrograms are a way of visualizing an audio vector. They divide the original vector into several sections using a windowing function, then calculate the DFT of each section and display the results as a heatmap. This allows us to see how the frequency make-up of the audio changes with time and is incredibly useful as a way to combine data from both the time and frequency domains.

We generated spectrograms many times while testing our methods as a way to "see" what was going on. A scroll through the methods section will show plenty of our early spectrographic work. Spectrograms also feature in our final application, where the majority of the user interface is taken up by a large spectrogram of the audio file currently being worked on. This allows you to see the changes you're making to the sound.

Additional DSP Tools

Quantization

Quantization is the process of representing an analog signal as a digital one. The real world is described by numbers with infinite decimal places, so in order to process it with computers, we break it down into chunks that are close-enough to the original values to be considered accurate representations. These chunks are saved as binary numbers. The amount of data lost when quantizing is sometimes referred to as the "quantization" of the original data.

Quantization-Index Modulation uses the quantization of the data to hide information. More data being lost, in other words, a higher quantization, corresponds to a binary 1, while less data being lost, or a lower quantization, corresponds to a binary 0. Because the digital values used by Matlab have such high precision, we can change the quantization without causing a change that a human listener would hear.

Auto-Correlation

Autocorrelation is a mathematical operation that measures the similarity of a signal with a delayed version of itself as a function of the delay. It gives an idea of the repeating patterns or periodicity within the signal, or whether certain parts of the signal are predictable from other parts.

To effectively calculate the autocorrelation of a signal in the time domain, we convolve the signal with the "time-flipped" version of itself.

$$Rxx = x[n] * x[-n]$$

This process is a polynomial-order operation, and the efficiency of which can be further improved (nlogn) if done while taking the cepstrum.

Cepstrum Analysis

Cepstrum analysis is a mathematical procedure used in signal processing where the spectrum of a signal is transformed to the quefrency domain by taking the inverse Fourier transform (IFT) of the logarithm of the signal's spectrum. This process effectively separates different sources of signal alterations, such as the effects of a room's acoustics on a sound signal or the vocal tract's shape on a speech signal.

Cepstrum of a signal can be obtained following the steps below:

1. Apply Fourier Transform
2. Get the complex logarithm
3. Apply Inverse Fourier Transform

Fun Fact:

ceps trum = Inverse(spec)trum

quef rency = Inverse(frequ)ency

Modulation

Modulation in discrete time is simply element-wise multiplication of a signal with a sinusoid at a given frequency. This results in the same data as the original signal, being centered at the carrier frequency of the modulating sine wave. Demodulation is accomplished by dividing by the known amplitude of the discrete sinusoid at each sample point.

Modulation was used heavily in the Amplitude Modulation method, which (as the name implies) is based on using modulation to encode data. This method focuses on the fact that after modulation, a signal's DFT coefficients become recentered around a new carrier frequency. This makes modulation an easy way to "move" the frequency of the original signal into a frequency band of our choosing, and if we choose a frequency band beyond the human-hearing range, we can hide secret messages.

Progress Report

Click to Check Out Project Topic Finalization Report

In our Project Finalization Document, we identified four methods of encryption that we believed had potential. That has been narrowed down to three methods which were able to make substantial progress toward successful encryption.

We now intend to move forward with further developing these methods and integrating them into a single interface to allow for a flexible, user-friendly encryption program.

Our Plan

There are many ways of attempting this, so we decided to pursue a set of 3 possible techniques to determine whether or not they were viable methods for this project. The results of this investigation are detailed in the **methods** section.

Going forward we have set these tasks to perform:

1. Gather a semi-large dataset of audio recordings to thoroughly test our methods
    1. Songs
    2. Talking
    3. Ambient noises
    4. etc.
2. Unifying the three methods within a graphical user interface so that we may speed up the testing process
    1. Allow adjustment of certain parameters like amplitude thresholds and sampling rates
    2. Allow custom messages to be written and encrypted
    3. Allow a section of source audio and encryption method
3. Determine measurements of error to analyze differences in the original and encrypted signals
    1. Root means squared difference
    2. Compress through MP3 and analyze
    3. Overall determine if differences are noticeable to the human ear

Our Progress

There are several possible ways to encrypt information into audio, the methods we tried to investigate include:

Amplitude Modulation

Audio Watermarking

Quantization Index Watermarking

What We Learned So Far

- Discrete Modulation & Human Hearing and Speaker System
- Cepstrum & Autocorrelation

Image Credit

Progress Report Image: https://www.goanywhere.com/blog/what-is-zero-knowledge-encryption

Who We Are...

Callen Fields

Researched and Implemented Quantization Index Modulation.

Yiming Liu

Researched and Implemented Frequency-Domain Watermarking and Echo Hiding.

Jake Pattok

Researched and Implemented Amplitude Modulation. Integrated tools into Application.

Gabriel Romero-Brownell

Designed and Updated the MATLAB Application.

Danny Samuel

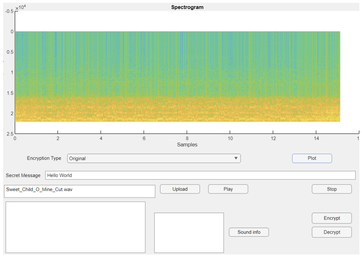
Researched and Implemented Feature Extraction. Compiled Library of Sounds for Testing.

Final Report

What We Did

Over the past few months, we've researched steganographic techniques used to embed data into audio, implemented several methods in MATLAB, and finally integrated them into a single application interface. We then gathered a library of sounds and tested our application to see if we could encode and recover the messages.

The Final Product



Our project's final deliverable is our MATLAB application which contains multiple cipher methods and can operate on any sound file that is saved in a format that MATLAB can open.

The application allows the user to upload an audio file to work on. Once uploaded, the encryption type can be selected from a drop-down menu and the secret message specified in a textbox. The tool can generate or extract secret messages from encrypted audio files.

For a review of its effectiveness, see the "Data Analysis" section below.

The Work

See our methods page to see how we developed the cryptography algorithms.

See the DSP Tools page to see how we used DSP tools from the class, and outside the class

See the data page to see what data we use to test our project.

The project itself can be downloaded from the zip file below:

AudioCrypt_EECS351Project_FinalSubmission.zip 1 item

Name	Last modified	File size
AudioCrypt_EECS351Project_FinalSubmission	-	19 MB

Examples/Demos

The application is included in the .zip file of all our source code. Here's a quick procedure to encrypt/decrypt your .WAV files:

Encryption

1. Press "Upload" and search for your audio file (it must be in the MATLAB path).
2. Under "Encryption Type," select "Original" and click "Plot." The graph at the top of the application will now display a Spectrogram of the audio file that was uploaded. Clicking the "Play" button will play the sound from the computer speakers, this can be stopped by clicking "Stop."
3. Select an encryption method from the "Encryption Type" dropdown menu. See the methods page for a description of what each method entails.
4. In the "Secret Message" textbox, enter the string you would like to encrypt into the audio file.
5. After selecting a method and entering a message, click "Plot" and the graph will update with a Spectrogram of the encrypted audio. Clicking the "Play" button will play the sound of the encryption from the computer speakers.
6. Click "Encrypt" to write the encrypted audio to a .wav file in the same folder as the application. The file will be named "SuspiciousAudio.wav"

Decryption

1. After generating an encrypted file, press "Upload" and select the "SuspiciousAudio.wav" file (it should generate in the MATLAB path)
2. Under "Encryption Type," select the cipher method used to generate the file
3. If using "Quantization-Index Modulation" or "Echo Hiding," you will need to enter the length of the original secret message in the "Message Length" textbox.
4. Now press "Decrypt." After a moment or two, the message should appear in the textbox on the lower left corner.

Data Analysis

	AM	QIM	Echo Hiding
Initial test			
Sweet Child O Mine	0.63	0.00	37.67
FSDKaggle			
2f8dd247	22.94	0.01	19.41
8fb2b69f	82.77	0.00	32.14
9ce0562b	5.99	0.00	27.58
GTZAN Music			
Vlobos	0.21	0.00	27.50
Glass	0.16	0.02	31.46
Pop	0.15	0.00	22.49
GTZAN Speech			
Serbian	1.59	0.01	15.86
Vegetables	0.54	0.00	24.66
Acomic	0.49	0.01	16.34

No errors
1 error
2 errors
3+ errors

Shown above is a chart of data collected by encrypting the message "Hello World" in several test audio recordings. The recorded numeric data points represent the Euclidean distance between feature vectors of the original and encrypted audio recording, while the highlight color shows the number of errors when attempting to decrypt the original secret message.

Quantization index modulation seems to consistently provide an encrypted audio file that is similar to the original. This gives evidence to show that QIM is secure from being noticed. However, this method is not the most reliable as it is susceptible to error in the received messages. QIM often misidentifies 1 character in the decrypted message, and struggles the most with the "everyday sounds" from the Kaggle dataset.

Amplitude modulation seems to be the most reliable method as it consistently provides the correct message, but did struggle somewhat when operating on human speech. This method also seems fairly secure as, besides a small number of outliers, it provides low feature vector distances; it had the most difficulty with remaining hidden in the "everyday sounds" category.

Echo hiding seems to be not as reliable as the other two. However, it should be noted that the character error data may be skewed toward the negative. In many of the single character errors, the last character was simply not present in the received message. This may give rise to errors in processing the result of echo hiding encryption. Additionally, echo hiding may not be successful amongst the fairly short audio clips provided in this test suite. This method relies on autocorrelation. The decoded peaks of the autocorrelation method may become harder to identify as the signal provides less data points. The results of this test do not correlate with the results in initial testing which show that 80 bits can be successfully recorded in a 15-second audio recording with 100% accuracy.