Encoding Secret Messages in Audio with Steganography

Least-Significant Bit Matching, Phase Coding, and Bipolar Backward-Forward Echo Hiding

Anastasia Grosch

CIS 542: Digital Forensics

Table of Contents

1 Goal

A person of interest may hide important information needed in an investigation within a media file. Audio files offer multiple frequency domain components (signal amplitude, phase, spectral bandwidth, etc.) to embed information in addition to binary-based techniques implemented in image steganography. The many different steganography options offered by audio files increases the desirability to embed messages in an audio file rather than an image. Challenges arise when implementing audio steganography, however, due to humans' higher sensitivity to audio changes. A digital forensic analyst must understand audio steganography techniques in order to successfully retrieve important, hidden information.

2 Scope

The project demonstrates how samples in audio files can be modified to contain secret message data without audibly changing the audio so that it can be secretly transmitted and decoded. This research explores three of the most commonly used audio algorithms: least-significant bit (LSB) matching, phase coding, and bipolar backward-forward echo hiding (BBFEH). A short cover audio file is embedded with a secret text message using each of the three steganography algorithms. The secret message is then extracted from the encrypted audio file.

3 Methodology

In each audio steganography algorithm, the secret text message is converted into a binary string based on each character's ASCII value. The binary string is then embedded into the cover audio. Each algorithm incorporates a procedure to extract the message without needing the original media file.

3.1 Least-Significant Bit (LSB) Matching

LSB algorithm focuses on minimizing the alterations made to the cover media to avoid detection. In standard LSB replacement, the least significant bits of a media file are replaced with the corresponding binary bits of the secret message. LSB replacement, however, lacks security due to its asymmetrical embedding method. Always adding 1 to even values ('0') and subtracting 1 from odd values ('1'), LSB replacement becomes vulnerable to structural attacks. LSB matching provides a more secure and efficient alternative by randomly adding or subtracting 1 from each LSB that differs from the corresponding secret message bit [3].

To further minimize the number of bits altered in the cover media, an end-of-text (EOT) descriptor defined by the ASCII table (0x03) is appended to the end of the secret message. The LSB of the audio file's first m bytes are modified to match the secret message, where m represents the length of the updated secret message in binary. Decryption simply requires reading the LSB of the first m bytes of the embedded audio file until reaching the EOT and removing the EOT.

3.2 Phase Coding

Phase coding addresses noise that LSB introduces into the output by hiding information in the phase component of the audio signal, which is not perceivable by the human auditory system [2]. This program implements an improved method of phase coding proposed by [5] which minimizes the computations by removing the phase difference propagation step.

The cover audio file divided into segments of length $L = 8192$ samples is converted to the frequency domain by applying the Fast Fourier Transform (FFT) to then retrieve its phase components. The length of the secret message in binary is embedded into the phase component of the first segment using the same procedure as the secret message. Equation 1 converts the binary secret message $msgBin$ into its corresponding phase components, to then embed into the mid frequency range of each segment's phase components with symmetry.

$$\phi_{data}[i] = \begin{cases} \dfrac{\pi}{2}, & \text{if } msgBin[i] = 0 \\ -\dfrac{\pi}{2}, & \text{if } msgBin[i] = 1 \end{cases} \tag{1}$$

The phase components ϕ_{data} are divided into the remaining segments and embedded into the mid frequency range of each segment ϕ_i with odd symmetry by

$$\phi_i\left[\frac{L}{2} - j + 1\right] = \phi_{data}[j], \quad j = 1, 2, 3, \ldots, \frac{m}{N} \tag{2}$$

$$\phi_i\left[\frac{L}{2} + j\right] = -\phi_{data}\left[\frac{m}{N} - j + 1\right], \quad j = 1, 2, 3, \ldots, \frac{m}{N} \tag{3}$$

where m represents the length of the secret message and N represents the number of segments. Applying the Inverse Fast Fourier Transform (IFFT) converts the data back to the time domain for outputting [5]. Decryption requires segmenting the embedded audio file and reading the phase shifts of the first segment to retrieve the length m of the embedded data. Extracting the secret message consists of reading $n = \frac{m}{N}$ phase shifts of each remaining segment where n represents the number of bits embedded in each segment.

3.3 Bipolar Backward-Forward Echo Hiding (BBFEH)

Echo hiding embeds information in a signal by adding unnoticeable echos corresponding to the bits of the message. Symmetrical echo impulses of BBFEH increase its robustness compared to other echo hiding methods. Delayed versions of the cover audio based on the 0-bit and 1-bit echo kernels shown in Figure 1 are added onto the cover audio with a mixer signal created from the hidden message [4].

$$h_0[n] = \delta[n] - \frac{\alpha}{4}\delta[n+d_1] + \frac{\alpha}{4}\delta[n+d_0] + \frac{\alpha}{4}\delta[n-d_0] - \frac{\alpha}{4}\delta[n-d_1] \tag{4}$$

$$h_1[n] = \delta[n] + \frac{\alpha}{4}\delta[n+d_1] - \frac{\alpha}{4}\delta[n+d_0] - \frac{\alpha}{4}\delta[n-d_0] + \frac{\alpha}{4}\delta[n-d_1] \tag{5}$$

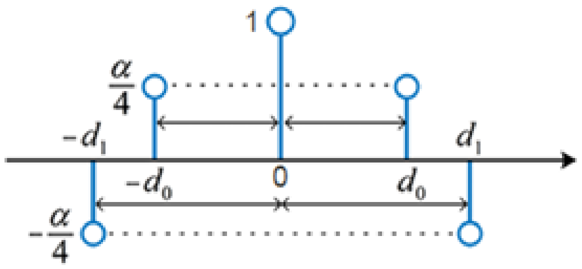

Figure 1: Bipolar backward-forward echo kernel (reprinted from [4])

BBFEH includes a mixer created from the binary secret message in which each bit takes length $L = (12)1024$ and undergoes Hann smoothing. If the secret message falls short of the maximum permitted characters, an EOT descriptor is appended to the message and padded with zeros as needed. Decryption consists of segmenting the audio file and comparing the delay points d_0 and d_1 of each segment n's real cepstrum c_n to determine the message bit value using Equation 6.

$$msgBin[i] = \begin{cases} 0, & \text{if } c_n[d_0] > c[d_1] \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

4 Results

Each audio steganography algorithm successfully embeds and extracts a secret text message in a cover audio file. The amount of data that each algorithm can encrypt caused challenges to arise, but adding restrictions on the input files as further explained in Section 5 resolved the issues. The program supports `.mp3` and `.wav` cover audio files and `.txt` message files for encryption and/or decryption with LSB matching, phase coding, or BBFEH.

5 Challenges

Sampling the cover audio with MATLAB's `audioread` function and creating the embedded audio output file with MATLAB's `audiowrite` function added single bit errors due to flaws in the functions themselves. This created a challenge in the decryption process due to the high sensitivity to bit errors. To resolve this, phase coding and BBFEH added Hamming (8,12) error correction to each message character. While resolving the bit error issue, increasing the number of bits each character needs from 7-bits to 12-bits for Hamming error correcting drastically decreases the permitted number of characters.

Each algorithm presented its own limitations, which created a challenge for selecting cover audios and messages that work with every algorithm. The acceptable message length varies greatly between algorithms due to the different amount of data needed to embed each bit. Considering this variation, the program checks the acceptable message length before encoding the message to prevent length-based errors. Equation 7 defines the maximum length of a message max_{length}

accepted by each algorithm for encryption based on the cover audio size $size_{cover}$:

$$\begin{aligned}
\text{LSB Matching} : max_{length} &= \frac{size_{cover}}{7} - 1 \\
\text{Phase Coding} : max_{length} &= \begin{cases} \left\lfloor \dfrac{\frac{L}{4}\left(\frac{size_{cover}}{L} - 1\right)}{12} \right\rfloor, & \text{if } max_{length} \leq 2^{24} - 1 \\ \dfrac{2^{24} - 1}{12}, & \text{otherwise} \end{cases} \\
\text{BBFEH} : max_{length} &= \frac{size_{cover}}{L/12}
\end{aligned} \tag{7}$$

Additionally, LSB matching accepts `.mp3` and `.wav` files as the cover audio. Unfortunately, phase coding and BBFEH are not compatible with `.mp3` files due to MATLAB's limitations in sampling `.mp3` signals. Therefore, the programs restricts the cover audio files types for each algorithm.

6 Conclusions

There exist many techniques for unassumingly hiding information within an audio file, which people of interest may use to hide information important to an investigation. Therefore, digital forensic analysts must understand ways to retrieve information from audio files when necessary. Additionally, recent research presents a new use for audio steganography: authenticating a media file by embedding unique metadata. Ensuring an audio file is real and unaltered may pose useful in an investigation as well. A commonly known steganography method, LSB matching, used among all media forms includes very little variation in formatting and thus can easily be decrypted by trial and error. Phase coding and BBFEH, on the other hand, contain higher formatting variations which can increase decryption difficultly if not knowing the formatting beforehand. Fortunately, established standards in each algorithm simplifies the process of trial and error, such as the standard segment lengths of 1024 and 8192 samples. Understanding the encryption process of common steganography algorithms in addition to the parameter standards provides an analyst better chances of uncovering important hidden information.

7 Link to Tool

The full code can be found here:

`https://github.com/anagrosch/audio_steganography`

References

[1] G. for Geeks. Hamming code in computer network. `https://www.geeksforgeeks.org/hamming-code-in-computer-network/`, July 2024. [Online; accessed 10-November-2024].

[2] A. Katta. Audio steganography using phase encoding. `https://medium.com/@achyuta.katta/audio-steganography-using-phase-encoding-d13f100380f2`, September 2021. [Online; accessed 10-November-2024].

[3] D. Lerch. Lsb steganography in images and audio. `https://daniellerch.me/stego/intro/lsb-en/`. [Online; accessed 10-November-2024].

[4] K. Tekeli and R. Asliyan. A comparison of echo hiding methods. *The Eurasia Proceedings of Science Technology Engineering and Mathematics*, pages 397–403, 2017.

[5] G. Yang. An improved phase coding audio steganography algorithm. *ArXiv*, abs/2408.13277, 2024.