Message Encryption in Audio Steganography

Daniel Hong, Grace Huang, Nitin Kumar

May 15, 2024

Contents

1 Background

Audio steganography, the practice of transmitting concealed information within an audio file (the cover audio), has various applications, including covert communication and digital audio watermarking. Besides their applications, the prevalence of audio files make them an interesting medium to transmit hidden information. Various challenges exist in effectively concealing information within cover audio, including the fact that the human auditory system (HAS) is more sensitive than other human perceptual senses, making it easier to hear differences between audio signals [2].

Many common approaches to audio steganography, such as Least Significant Bit encoding or Spread Spectrum, conceal and transmit plain text messages (that are converted into bit strings). Then, if an adversary gains information about what type of encoding was used to alter an audio file and are able to retrieve the hidden message, they can easily reconstruct it. As such, we are interested in exploring how data can be transmitted more securely through audio files using a combination of audio steganography and cryptographic techniques. In order to do so, we look at existing research on this front, identify properties that an effective cryptographic audio steganography scheme should satisfy, and propose a scheme that satisfies these properties.

1.1 Audio Steganography Approaches

There are a few common approaches to audio steganography. These approaches include Least Significant Bit (LSB) encoding, Echo hiding, Spread Spectrum (SS), and Phase coding [1]. To better contextualize audio steganography and how our project compares to existing audio steganography approaches, we will discuss how some of these steganography methods work.

1.1.1 Least Significant Bit encoding

Least Significant Bit (LSB) encoding is the simplest form of audio steganography. In LSB encoding, the hidden message is converted into a bitstring, and then the least significant bits of each audio sample are replaced with the bits of the bitstring. For 32-bit integer `.wav` files, one bit of the hidden message can be inserted per 32 bits of audio data.

LSB is prevalent because of its simplicity, but the main weaknesses of LSB are also due to its simplicity. Since language is not statistically random, statistical analysis on the least significant bits of stego audio is a big threat to the security of LSB encoding. Furthermore, small amounts of noise will easily destroy the hidden message, since the least significant bits will be modified.

There are many ways to enhance LSB to improve its security and efficiency, as it is easy to build off of. For example, combining LSB with a stream cipher (e.g. AES counter mode) by encrypting the plaintext before inserting it into the least significant bits allows the system to satisfy a higher standard of security.

1.1.2 Echo hiding

Echo hiding adds the hidden message to the audio in the form of an echo. This makes use of the idea that when audio is played in different environments, echos will be introduced to the audio signal anyways. As such, the human auditory system will often find it difficult to notice when echos are inserted into audio. The echo is defined by its amplitude, delay rate, and offset (which is different depending on whether the message bit is 0 or 1). The amplitude determines how strong the inserted echo is, and the delay determines how much later the echo is inserted from the original audio. When the parameters of this echo are tuned correctly, the altered stego audio will sound indistinguishable from the original audio [2].

1.1.3 Spread Spectrum

The Spread Spectrum method involves spreading a message throughout the frequency spectrum of the cover audio. This involves first converting the message to binary, then spreading it out over a larger bandwidth by artificially lengthening it. Noise is then added to this modified data and used to modulate the cover audio. This technique is more resistant to compression or manipulation than other methods.

The most simple and most commonly used approach to perform spread spectrum steganography is Direct Sequence Spread Spectrum (DSSS). In this mode, a bitstring b representing a secret message is converted into a longer string using a *Barker code* c. A Barker code is a code of ± 1s that has very low auto-correlation, or in other words does not follow any obvious patterns. For example, one such Barker code is the 11-bit Barker code

$$c_{11} = \{1, 1, 1, -1, -1, -1, 1, 1, -1, 1, -1\}.$$

Under the mapping $0 \to -c$, $1 \to c$, the sequence b is then converted into a sequence w by mapping each digit of b to its image in succession. Finally, the sequence w is used to modulate the audio, by adding δw to the first $|w|$ samples of the audio, where δ is small enough that the message cannot be heard. In order to retrieve the secret message, one can convolve c with the stego audio samples. Due to the low auto-correlation of c with itself, b can be retrieved by analyzing the peaks of this convolution.

Methods to improve DSSS have been explored: one such method involves converting the audio to the frequency domain first by using the modulated complex lapped transform (MCLT), a close variation of the Discrete Fourier Transform. This method resulted in stego-audio that was harder to detect encryption [3]. An example of work that has combined encryption with audio steganography involves combining an encryption scheme and a with DSSS [4]. The message is first encrypted using a Vigènere cipher. Then, the encrypted message is then converted into a bitstring, and the DSSS method is used to embed the message into the audio file. A pseudo-random number sequence, seeded off of a secret key, is used to determine how the message is modulated, inserting additional randomness into the system while still allowing the receiver to decode the message [4].

While this approach does improve the security of the message transmission, this combined scheme requires carefully choosing a threshold value δ for modulation in order to make sure that the modulation contains enough noise to mask the original message while not adding too much noise to the audio. More importantly, the use of the Vigènere cipher presents an opportunity for improvement, as it is not a secure encryption scheme by itself.

2 Design Goals

Audio steganography techniques are most commonly evaluated on transparency, capacity, and robustness. Transparency refers to how hidden the steganography is to listeners; a human ear should not be able to differentiate between the original audio and the modified audio. Capacity refers to the average number of bytes of ciphertext that the steganography method can transport per byte of audio data. Finally, robustness refers to how well the hidden information survives common audio processing techniques (i.e. compression, noise addition, or filtering).

We extend these properties to identify a set of properties that an effective cryptographic audio steganography should satisfy:

1. **Transparency**: The stego audio should be aurally indistinguishable from the cover audio. By aurally indistinguishable, we mean that humans should not be able to hear the difference between the stego audio and the cover audio.
2. **Security**: The altered bits of the stego audio should look computationally indistinguishable from a random alteration of the same bits in the cover audio. That is, to an adversary, the alteration of bits in the stego audio should look random. In the case of audio steganography, since most bits might be left untouched and we want the stego audio to sound like the original audio, we only look at the altered bits themselves (for example, the least significant bits for LSB) instead of all of the bits in the audio.

3. **Recovery Complexity**: An attacker should not easily be able to recover the encrypted ciphertext from the audio. This principle is included to prevent 'harvest now, decrypt later' attacks, where adversaries could potentially store the altered stego audio file indefinitely to try to decrypt the audio.
4. **Robustness**: If the stego audio is tampered with, the receiver should be able to recover very little, if any, of the original message. This principle is included to prevent MITM or impersonation attacks.

Although the approaches described in Section 1.1 achieve transparency and some, such as DSSS, achieve robustness, they fall short because the only knowledge an adversary needs to decrypt the secret message is the steganographic procedure. Furthermore, these methods encrypt the message in the audio *unencrypted* and *sequentially*, meaning that given just a small portion of the stego-audio allows the corresponding portion of the message to be decrypted, and because language is not uniform, much less cryptographically secure, so the security constraint is also not satisfied.

In order to better meet these goals and scope our implementation, we assume and require that the cover audio is not too synthesized, such that specific bits in the audio are not predictable. When audio is more professionally edited or more fully synthesized, noise is often removed, such that periods of silence are fully silent. This makes it more difficult to satisfy our properties. Specifically, transparency poses a larger concern, since it is easy to tell the difference between complete silence and a small noise. Audio from voice recordings tend to have sufficient white noise (40-60 dB) for our use cases.

3 System Design

As a proof of concept of how a cryptographic audio steganography scheme could work, we designed and implemented a scheme that satisfies many of the intended goals specified above. For this implementation, we use AES and assume that the sender and receiver both know the encryption/decryption key.

The encryption scheme consists of two main layers, given a secret message m. For ease of explanation, we describe the layers in the opposite order they are applied:

- **Layer 2:** We separate m into data chunks of 16 bytes each. We will eventually hide the chunks in random locations in the original audio. To allow the receiver to find the data chunks in the correct order, we form a linked list by associating each data chunk with a standard 8-byte pointer to form a node. The first chunk is slightly different: it contains an 8-byte nonce for the AES cipher (more on this later) and 8 bytes for the length of the message in place of the 16-byte data chunk. This way, all the nodes are 24 bytes long. We choose random locations for the nodes in the audio file, except for the first node, which is placed at the beginning of the audio file, so that the receiver has an entry point to the linked lists. Then, we place the nodes in the least significant bits of the original audio file. Optionally, we can randomly change the other least significant bits that would not have otherwise have been affected, in order to remove any detectable patterns in the least significant bits, which makes it computationally impossible to detect which bits store data.

- **Layer 1:** We use AES in counter mode to encrypt our data. This way, the length of the data does not change. The cipher uses a public nonce, which we leave unencrypted. Everything else that we place in layer 2, including the length of the message, all of the data chunks, and all of the pointers, are encrypted. A big advantage to counter mode for AES is that we can decrypt byte-by-byte, instead of needing the full message at once. This property is necessary, because we must decrypt the pointers one at a time to follow the linked list.

The receiver can decrypt the message by starting from the first chunk and decrypting the length and first pointer. They can then follow the pointer to the location of the next message chunk, decrypting the associated message data and attached pointer again. This process is repeated until the length of the decrypted message matches the expected length from the first chunk. Then, the receiver can concatenate the data chunks together in order and recover the original message.

3.1 Implementation

Our code can be found here: `https://github.com/dantaxyz/audio-stego`.

3.2 Strengths

Our implementation has several strengths. First, it has high capacity. Most modern audio files have 44.1kHZ sampling rate, which means that 44100 audio samples are present for every second of audio. This then means that a three minute audio file includes 7.94 million least significant bits. Each of the nodes are 24 bytes long and encode 16 bytes of data, which means that in total, we can transmit over 661 kilobytes of data.

Additionally, our implementation improves upon the effectiveness of existing audio steganography approaches as framed by the properties outlined in section 2. That is, because we use LSB encoding in the second layer, we satisfy the transparency property. This is because only the least significant bit is changed in every sample, and when every sample is 32 bits, this last bit will be insignificant, especially when taking into account our assumption that there will be some inherent noise in the audio file. The security property is also satisfied, because AES is used to encrypt the original message.

On the recovery complexity front, even though an adversary can recover the least significant bits from the audio, the encrypted message itself is stored in chunks across the audio file, which makes it harder for an adversary to piece the original message back together. In order to store enough information to encrypt the entire message, an adversary must store all the least significant bits of the audio. If the least significant bits are tampered with in the audio, the receiver will either be able to recover the entire message or it will fail to recover the message at all, because if any pointer is tampered with, the remainder of the decrypted message will be incorrect.

Lastly, it is easy to extend this implementation to other audio steganography approaches. That is, instead of using LSB in order to insert the bits into the audio file, we could use the altered bits with other techniques like Echo Hiding or DSSS in order to further improve on the security of the algorithm. Some extensions will be further explored in section 4.3.

3.3 Limitations

Given that our implementation is largely a proof of concept, there are a few limitations that need to be addressed in order for it to be more complete. First, our implementation only works on uncompressed audio files, such as `.wav` files. This limitation has a few consequences. Particularly, it is harder to apply to more general audio files, as many audio files are compressed, in `.mp3` format or otherwise. Additionally, uncompressed audio files are generally larger than compressed audio files, which makes processing time slower.

The current implementation of the system also does not work when audio is being streamed live, as the encrypted message chunks can be inserted out of order in the actual audio file itself. This limitation prevents users from sending covert messages through their audio signals *while* they are in communication with each other.

Furthermore, if the cover audio has locations where the least significant bit is predictable, it can be hard to satisfy the transparency property, as changes to the audio can easily be distinguished by the human ear. That is, if there is a period of complete silence, even altering the least significant bit can make it easy to tell that the cover audio has been altered.

Lastly, given our current implementation, an adversary that knows how the message was encrypted can just retrieve the least significant bits from the audio to decrypt later or to delete from the actual audio file, which can be harmful in use cases like communication, as the receiver might not get any message at all. Furthermore, even though compression and tampering will make it impossible to recover the original message, the receiver may not be able to detect tampering surely (although nonsense pointers that loop or point outside the audio file can sometimes be sufficient to detect tampering). This problem arises because of our use of LSB, which inserts each bit of the encrypted message as is into the audio file.

4 Extensions

To address some of the limitations in our current implementation, we propose a few extensions of our algorithm as potential future work.

4.1 Using DSSS instead of LSB

As mentioned in the limitations section, while LSB is easy to implement and use, an adversary can easily retrieve the altered bits of the stego audio by just taking the least significant bit of every sample. Furthermore, the message can easily be written over; an adversary can simply write over all least significant bits with 0s to delete the secret message completely.

In order to address this limitation, we can change how we insert the encrypted message chunks and pointers back into the cover audio in layer 2. Instead of using LSB, we could potentially use Direct Sequence Spread Spectrum encryption. Compared to LSB, DSSS would be much more resistant to compression and would make it harder for a man-in-the-middle adversary to destroy the message. In particular, we can similarly encrypt and decrypt blocks using DSSS, along with pointers to the next block.

Even further, by using the method described in [3], we can encrypt the secret message into the frequency transform of the audio instead of the audio itself, which would make the encryption less noticeable and harder to destroy.

4.2 Supporting Live Streaming

In order to support live streamed audio, we can change the implementation to ensure that the chunks are placed in the order that they should be decoded in within the audio. So, instead of choosing random locations for each of the nodes in the cover audio, we can place the randomly generated locations in increasing order, then insert the nodes according to this ordered list. This ensures that the receiver does not have to rewind the streamed audio at any point to retrieve a message chunk.

4.3 Authentication

The receiver in our implementation simply follows pointers until they get to the message length. However, tampered stego-audio is still difficult to detect if an adversary fills the LSBs with random values. If the sender and receiver share an authentication key, the sender can insert this authentication key in some manner after the final block, and this will act as a *signature* that verifies that the stego-audio was not tampered with.

4.4 Public Key Encryption

In our implementation, we used AES, which assumes that both the sender and the receiver have the shared private encryption key. Because our encryption scheme is separated from the rest of the implementation, a public key encryption scheme like RSA can be used in place of AES for greater security, at the cost of requiring more compute to calculate. For situations in which the stego audio is being live streamed, using a quicker encryption and decryption scheme like AES can be more appropriate still, as the receiver needs to be able to decrypt the pointer to the next encrypted data chunk before it is streamed in the audio itself.

5 Acknowledgements

The system design in our paper was brainstormed together as a group. The implementation of the system design was mostly done by author 2, while authors 1 and 3 examined strengths and limitations of the design compared to known methods, and considered extensions of the current design to address these limitations.

We would like to thank the 6.5610 course staff for their guidance on the project. Specifically, we would like to thank our TA Leo for meeting with us and providing us helpful comments on what goals might be important and what could be improved about our design.

References

[1] F. Djebbar, B. Ayad, K. A. Meraim, and H. Hamam. Comparative study of digital audio steganography techniques. *EURASIP Journal on Audio, Speech, and Music Processing*, (25), 2012.

[2] D. Gruhl, A. Lu, and W. Bender. Echo hiding. In *Information Hiding: First International Workshop Cambridge, UK, May 30–June 1, 1996 Proceedings 1*, pages 295–315. Springer, 1996.

[3] D. Kirovski and H. S. Malvar. Spread-spectrum watermarking of audio signals. *IEEE transactions on signal processing*, 51(4):1020–1033, 2003.

[4] A. A. Krishnan, C. S. Chandran, S. Kamal, and M. Supriya. Spread spectrum based encrypted audio steganographic system with improved security. In *2017 International Conference on Circuits, Controls, and Communications (CCUBE)*, pages 109–114, 2017.