Audio Steganography Using Stereo Wav Channels

Douglas C. Farmer, Daryl Johnson
B. Thomas Golisan College of Computing & Information Sciences
Rochester Institute of Technology, Rochester, NY
{dcf2929,daryl.johnson}@rit.edu

ABSTRACT

This paper presents a new and novel method of audio steganography that allows for the encoding of a textual data within the audio channels of a wav file. This method differs from the more typical forms of audio steganography as it involves the modeling of existing audio channels in order to build a dictionary and then the addition of one or more channels containing encoded data in a form closely related to that of the carriers wav forms. This method presents a fairly robust, high-bandwith channel through which to communicate.

Keywords: audio steganography, channel, covert channel, covert communications, steganography, wav, wave

1. INTRODUCTION

Steganography is the art or practice of encoding some data, be it textual, visual, or of some other other form, inside another medium not specifically designed to carry such information. Audio Steganography then is the encoding of such information inside audio data. Such data hiding may take many forms. Common practices include encryption which attempts to conceal the secret itself using cryptography, generally in the form of some mathematical algorithm to encode the data to a not easily readable form. Steganography however is different in that the goal is to conceal that there ever was a communication taking place by hiding said communication within another expected form. The precipitous rise, and popularity of digital media in recent times provides many convenient new avenues in which to employ these practices.

2. RELATED WORK

There are several popular variants of Audio Steganography. These all address separate issues in dealing with the encoding of data in an auditory stream and have their own respective strengths and drawbacks. Some such encoding schemes include least significant bit encoding, phase encoding, and echo encoding.

2.1 Least Significant Bit Encoding

Least Significant Bit Encoding is by far the most common type of audio steganography in use. In this scheme the least significant bit of each audio frame is modified to encode binary information [2]. All data in Wav files is store in 8-bit bytes arranged in little endian format with the low-order (i.e. least significant) bytes first for multi-byte values. Changing this byte usually doesn't result in noticeable changes to the resulting waveform. This scheme is rather simple however, and unfortunately easily defeated. Techniques such as random bit shifting can interfere with the extraction process as can normal audio operations like compression and file conversion. Additionally due to the way binary information is represented its statistical shape is easily detectable when embeded in other media.

2.2 Phase Encoding

Phase encoding is much more difficult to detect. Phase Encoding involves breaking down audio into chunks, separated by phase groups and then shifting it based on the binary data to be encoded. This technique can still be succeptible to random bit shifts, but is much harder to detect. The main drawback of Phase encoding lies in the low bandwidth which is a direct result of encoding using the audio phase. SImply put there isn't a great deal of information that can be sent at one time.

2.3 Echo Encoding

Echo encoding is another common method for audio steganography. As the name implies this scheme involves the insertion of 'echoes' into an audio signal. The echo is varied along three parameters, initial amplitude, decay rate, and offset/delay. By using a short delay it is possible to hide data using this technique without noticeable effect on the resulting waveform. Likewise amplitude and decay rate can be set to values belove the audible range of the human ear. This makes this encoding scheme incredibly hard to detect. There is however a chance that some mix of echoes can combine to produce to a noticeable effect.

3. BACKGROUND

3.1 Wav Background

In order to understand this covert channel it is necessary to provide some background in more depth than previously covered. This channel deals primarily with the Waveform Audio File Format commonly referred to as wav due to its file extension. Wav is an audio file format standard for the

storing of audio bitstreams. This file format is the general raw, uncompressed format digital audio takes during recording and processing before being compressed into the various lossy and lossless streams developed for general distribution. Wav files are an application of the RIFF (Resource Interchange File Format) bitstream format for storing data chunks and thus closely follows the Riff file format. Wav files support any number of bit resolutions, sample rates and channels of audio[3] which we will take advantage of in the creation of our covert channel. They are a collection of different types of 'chunks' each containing important information. Because RIFF is a tagged format, interpreters of RIFF and Wav files are designed to interpret only tags they understand and ignore the rest. All Wav file interpreters though are expected to understand and read the 2 required chunks Format and Data. Figure 1 is an illustration of a single Wav file containing the two required chunks.

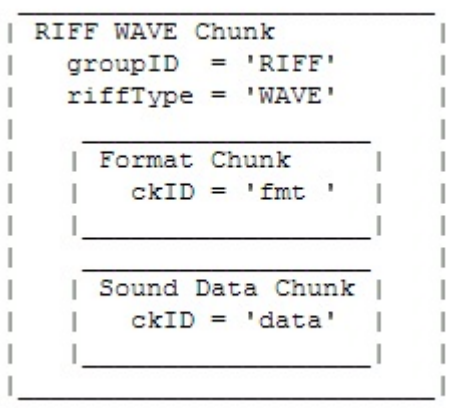

Figure 1: Wav File Require Chunks

The Format chunk describes the characteristics of the waveform data. These include fields such as sample rate, bit resolution, and number of channels. This chunk always has an id of 'fmt'. Analyzing this chunk is incredibly important for the robustness of the channel but will be remain largely unmodified.

The Data chunk contains the actual sample frames. Sample frames contain all the wavform data for all channels of the audio. This format chunk while generally the largest chunk in terms of actual bytes contains only three fields generally. An ID that is always 'data', a field denoting the number of bytes contained in the waveform data, and the waveform data itself. This data is arranged by sample frame. Every channels first frame comes first followed by every channels second frame and so on and so forth. For a stereo track containing 5 frames, a typical waveform data array might appear as L1R1 L2R2 L3R3 L4R4 L5R5 where L and R denote the Left and Right audio channels and the number the frame. As mentioned before all data in Wav file is stored as 8-bit bytes in little endian format as pictured in Figure 2. It is also important to note that a Wav file should only ever contain one data chunk.

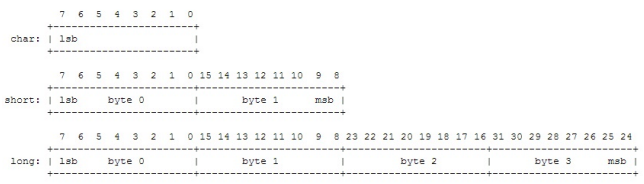

Figure 2: Wav File Byte Organization

3.2 Audio Channels

Most of the previously outlined encoding techniques for audio steganography involve modifying the pre-existing waveform to add significance to data that wouldn't otherwise carry meaning. This method is different in that it doesn't actually involve the changing of the pre-existing data but in fact adding to it. Audio files may contain many different audio channels. These channels act as a storage device for multi-track recording and playback. Monaural sound or mono refers to a single channel where as stereophonic sound or stereo refers to more than one channel. Most commonly stereo is 2 channels but there is no real limit on the number of channels that can comprise a stereo sound. 5.1 and 7.1 Surround Sound both refer to stereo audion with 5 and 7 full range channels respectively and .1 to reflec the limited range of the Low Frequency Channel (e.g. bass) . This is useful in the creation of a covert channel using audio steganography because of the way these channels are interpreted by audio playback devices.

Mono systems can have multiple speakers but because there is only one audio signal it is simply replicated over each playback device, read speakers. Stereo on the other hand may contain level and time/phase information to simulate direction cues or in other cases contain explicity different audio for each channel. In stereo systems each channel is mapped to a different speaker. The increasing number of channels defines more and more precise sound targeting. One of two things generally happen when a stereo sound with more channels than a system has capabilities to decode is played. Either the excess channels are ignored completely or they are sent to the closes matching speakers. For example on a 5.1 stereo system 7.1 audio would have it's extra two channels which correspond to surround back left and back right would be played by the speakers corresponding to surround left and surround right.

4. DESIGN OF CHANNEL

4.1 Encoding Method

The proposed covert channel aims to take advantage of stereo audio processing by adding channels to an original audio file and then encoding data in it. This process has several steps. Before begining to write data the file must first be read and parsed for relevant information from the Format chunk. Specifically, the framerate, number of frames, frame width, and number of channels is required. Because the data chunk is read as a byte string it must be parsed in order to be useful. Thus the frame width and number of channels is needed. The frame width tells how many bytes constitute a discrete sample while the number of frames combined with the number of channels denote how many samples each frame has. This information is important when writing the file later.

Using the python module numpy the byte string can be converted to an array of integers of the proper size as denoted by our sample width. Even further it can be reshaped to an array of arrays in which each inner array represents a frame in the original audio file and each index in an array is a sample from a channel of audio. Remember that wav data is represented by channel by frame in the data chunk.

In the simplest form an ascii message can be obtained and converted to an array of it's decimal representation and written directly into a new audio channel. It is important to ensure that the new channel is the same number of frames as the other channels and so the new channel must be padded. The pad value makes no difference, however using zero's results in no white noise. When writing the file, one denotes the framerate, new number of channels, and the sample width. The number of frames will be updated after writing is complete.

4.2 Decoding Method

Decoding follows the same steps as reading and parsing an audio file for writing. The format chunk is read and parsed. Framerate is unimportant for decoding but number of channels, sample width, and number of frames are again key. Numpy is used to covert the binary string to a useful format. Now each sample in the last channel can be iterated over and used in conjunction with the python function 'chr' to convert and append the binary representation to its ascii equivalent.

5. RESULTS

This proposed program was implemented in two Python3.3 programs title encode_wav.py and decode_wav.py. As previously mention numpy was used extensively as a more convenient way to work with formated byte data. Additionally the built in python Wave and Struct modules were used to read and pack wav files for writing respectively. The last module used was the Matplotlib module Pylab which was used to generate graphical representations of the sample distribution. Only 16-bit audio was tested however, methods for 8, 24, and 32 bit audio would be trivial.

Figure 3 and Figure 4 are graphical representations of two test wav file, the former mono, and the latter stereo. These plots were normalized from -1 to 1. Note the full spectrum spread. Both the mono channel and the various stereo channels in Figures 3 and 4 are well distributed.

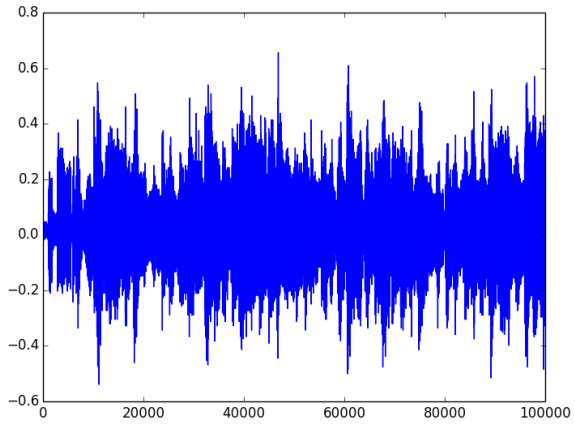

Figure 3: Mono Channel Non-Encoded Woman.wav Plot

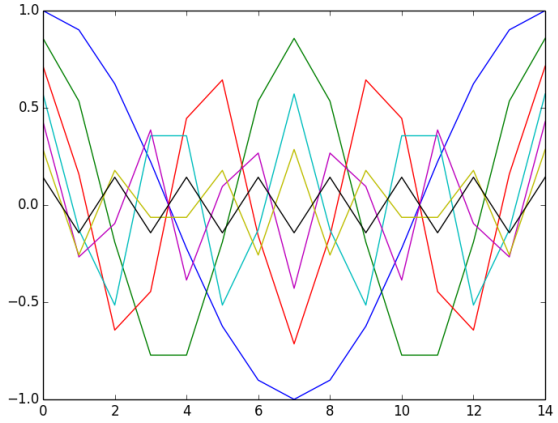

Figure 4: 7 Channel Non-Encoded Plot

Several individuals were asked to listen test audio files both before and after the encoding of data. When adding channels to mono (1 channel) audio there was a discernable change in the audio but no real degredation. Adding another channel cause the original channel to be played as a front left audio channel while next to nothing was played on the new channel containing the encoded data. Stereo tracks faired much better as there was no discernable difference in the audio.

Figure 5 and Figure 6 are both stereo channels that have had a message encoded in them. The wavforms are unchanged but if one notes the almost horizontal line around 0 one can see the where a message has been encoded. Because the data has been encoded using 0 to 255 in a field that ranges to 32,767 there is barely any appreciable slope. Normalizing these values based on the existing wav data would greatly increase the covertness of this channel.

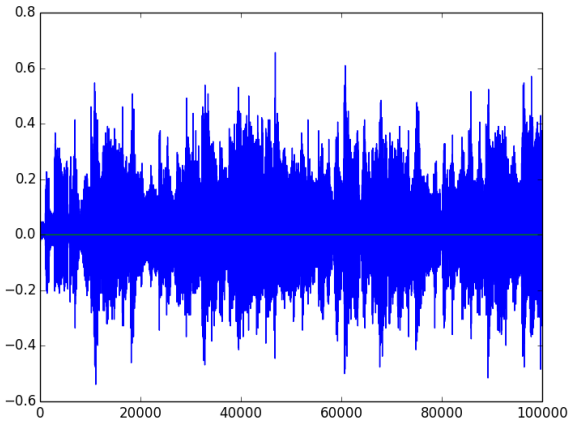

Figure 5: 2 Channel Woman.wav Plot

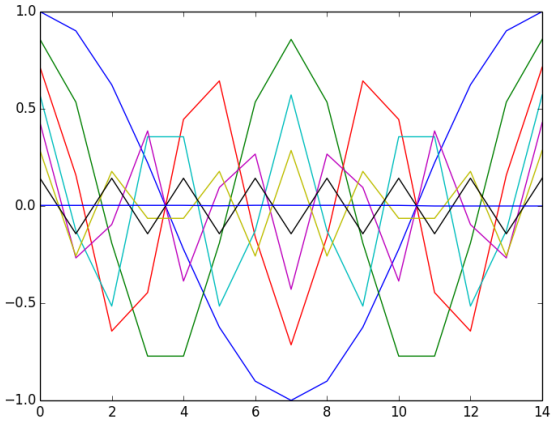

Figure 6: 8 Channel Encoded Plot

5.1 Throughput

In a 9 second clip of audio, a 16-bit mono wav file contained 100,000 frames. Upon encoding each frame using the 16-bit representation contains a single signed short with value ranging from -32,768 to 32,767. Even using only the extended ascii range from 0 to 255 and a single frame per character the channel was able to encode 100,000 characters in .303 seconds. This implies a theoretical limit of 330033 character per second. This calculation doesn't account for the fact that the encoded message was almost entirely comprised of padding zero's, however longer messages would not take significant time to convert to decimal representations and would also decrease the amount of time spent padding. Decoding this message however took 1.11 seconds.

5.2 Robustness

This channel as presently constituted is decently robust. It hides the data in a way that is completely uncommon for audio steganography by adding an entire channel. This also means that the data is interspersed throughout the entire wavform if at regular intervals. There has been little work done to obfuscate or encrypt the data before encoding. It remains to be seen whether compression or conversion to other formats will alter or destroy the encoded data.

6. PREVENTION AND DETECTION

This channel is hard to prevent because it is incredible hard to detect. Most detection schemes rely upon analysis of the wav data looking for the statistical signature of binary data. This channel won't be detected that way because each ascci value is being represented as a 16-bit integer rather than directly as bits or event bytes. The statistical shape of the data won't give it away as text. Additionally other prevention techniques such as random bit shifting add noticeable noise to an audio stream and are thus rarely if ever used. While "a signal-to-noise ratio (SNR) above 20dB guarantees for a reasonable audio quality"[1] there is future work that would effectively eliminate this as a viable prevention method.

7. CONCLUSION

Steganography is by no means a new field and while it's use in audio may be a somewhat recent event there are still several well researched and established schemes in existence for encoding data. This paper presents a new, novel approach to encoding data within wav audio files using stereo audio channels. This method has been shown through both experimentation and research to effectively transmit data unbeknownst to end users. The data is virtually undetectable without doing a thorough cryptographic analysis of the resulting raw audio. This does indeed have weaknesses as do all covert channels but there should be great optimism that proposed future work could go along way to fixing those weaknesses.

8. FUTURE WORK

This channel has numerous areas where future work could expand on and improve this channel. The most obvious area of work is to make this channel work for more than just ascii data. The ability to encode binary data would greatly increase the utility of this channel. Additionally, some work into obfuscating or encrypting the data before encoding would make it even less likely to be detected. Like most of the other encoding schemes mentioned in related work this channel is vulnerable to random bit shifting. While this isn't guaranteed to destroy a message there is a great method available to mitigate this attack that there simply wasn't enough time to implement. By analyzing the mean values across each frame and sample it would be possible to create a seperate alphabet for each sample in the new channel and encode the data using values that lie closer to those of the other channels. This would make the encoded data look that much more like the wavforms of the rest of the audio but also likely result in audio that played close enough to the original to blend in. One could also map each ascii value to a range of values based on the sample frame that way even random bit shifting wouldn't alter the message[1].

9. REFERENCES

[1] M. Nutzinger, Real-time Attacks on Audio Steganograhy. Journal of Information Hiding and Multimedia Signal Processing. Vol. 3, no. 1, January 2012.

[2] Gunjan Nehru, Puja Dhar, A Detailed look of Audio Steganography Techniques using LSB and Genetic

Algorithm Approach. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 2, January 2012

[3] L. Silvestro and G. Baribault, "Waveform Audio File Format MIME Sub-type Registration." [Online]. Available: http://tools.ietf.org/html/draft-ema-vpim-wav-00

[4] "WebP Container Specification – WebP – Google Developers." [Online]. Available: https://developers.google.com/speed/webp/docs/riff_container?csw=1