MP3 Steganography Techniques

Yalinne Castelan Guzman
School of Applied Technology
Illinois Institute of Technology
201 East Loop Road
Wheaton, IL 60189
+1 (708) 244-8819
ycastela@hawk.iit.edu

Ben Khodja
School of Applied Technology
Illinois Institute of Technology
201 East Loop Road
Wheaton, IL 60189
+1 (630) 815-9149
bkhodja@hawk.iit.edu

ABSTRACT
The MP3 file is largely use because is a convenient way to store audio to be replay. MP3 files are widely use by electronics to play music, or audio of a video. Although it has been not greatly exploded there are various types of steganography techniques exist which allow for the hiding of information within MP3 files of any type. One of these techniques involves hiding information during the MP3 encoding process as is the case with Fabien Petitcolas' MP3Stego application. MP3Stego accepts as inputs a non-compressed, Pulse-Code Modulated (PCM) audio file (such as one formatted in the Waveform Audio File Format commonly known as a WAV or .wav file) as well as the information that is to be hidden. Another technique involves hiding information after the MP3 encoding process has taken place within the areas of an MP3 file known to be skipped or ignored by MP3 playing and decoding applications. This is the technique used by Illinois Institute of Technology (IIT) student Mikhail Zaturenskiy's MP3Stegazaurus application which accepts as inputs an MP3 file as well as the information that is to be hidden.

The authors have developed a technique which allows for the hiding of information within the audio information-containing portions of an MP3 file in a way that has not been done before. Unlike the technique used by MP3Stego which involves hiding information within the audio information-containing portions of an MP3 file during the encoding process, this technique involves hiding it after the encoding process has already taken place. This paper details the research carried out by the authors as well as how the application that makes use of their technique works.

Categories and Subject Descriptors
H.5.5 [Sound and Music Computing], E.3 [DATA ENCRYPTION]

General Terms
Algorithms, Measurement, Documentation, Experimentation, Security, Theory, Verification

Keywords
Steganography, MP3, Secret Writing, Hidden Information

RIIT'14, October 15–18, 2014, Atlanta, Georgia, USA.

ACM 978-1-4503-2711-4/14/10…$15.00.
http://dx.doi.org/10.1145/2656434.2656438

1.INTRODUCTION
The MPEG-1, Layer 3 (MP3)audio encoding is based in psychoacoustics which means that it considers the perceptive behavior of the human ear. Recording equipment like microphones store more audio data than we can actually listen. MP3 will take advantage of this situation using compression. Both lossy and lossless forms of compression take place during the MP3 encoding process. [3]

During the MP3 encoding process, the original, WAV file which is uncompressed audio signal is broken into independent portions called frames which contain audio information that is a fraction of a second in duration; much like the many frames of a film strip which compose a motion picture. First, the original audio signal is analyzed and broken into subbands using algorithms to calculate and determine the best distribution of bits for the pieces of audio signal which fall within the spectrum of frequencies determined to be perceivable by humans. Taking the encoding bitrate (the number of bits per second devoted to storing the audio information) into account, the maximum number of bits that can be included in each frame is calculated which ultimately determines how much of the original audio information is to be kept and how much of it is to be removed.

Using mathematical models of human psychoacoustics included within the MP3 encoding format, the frequency spread of the signal in each frame is analyzed. This process determines the frequencies within each frame that are to be rendered with the most precision as they will be perceivable. It also determines which frequencies are to be rendered with less precision by being given a fewer number of bits as they will not be perceived well thereby ridding each frame of audio information likely to be unperceivable. [3]

The collection of frames is then run through Huffman coding which acts like a traditional lossless form of compression. It compresses the redundant data found throughout the collection of frames and allows for the storage of that same data within a smaller amount of space (20% less space on average when compared to an MP3 file not compressed with Huffman coding). [6]

Lastly, the collection of frames is assembled into a complete MP3 file where each frame is prepended with a header portion, an optional CRC portion, and a side information portion. The header and side information portions contain metadata about the audio information contained within the frame as well as metadata about the frame itself.

2.MP3 FILE FORMAT

The MP3 file is divided into frames, this is the WAV uncompressed file was broken down into 26 milliseconds worth of audio information. This means that the number of frames will depend on the duration of the audio. [6] The size in bytes of each frame is dependent upon the bitrate and sampling frequency specified and used by the MP3 encoder during the encoding process. As an example, an MP3 file encoded using a specified constant bitrate of 128kbps and sampling frequency of 44,100 Hz will have a frame size of 417 bytes and sometimes 418 bytes when padding must be applied to maintain the exact average bitrate throughout the entire MP3 file. An MP3 file encoded using a specified constant bitrate of 192kbps and sampling frequency of 44,100 Hz will have a frame size of 626 bytes.Optionally, each frame may contain a 2-byte CRC value and the entire MP3 file may begin with an ID3v2 tag or end with an ID3v1 tag. The following figure illustrates the components that make up a complete MP3 file as well as how they are organized.

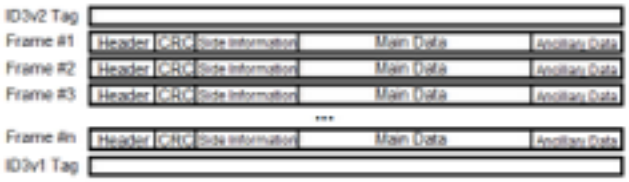

Figure 1. Structure of a complete MP3 file.
Note: ID3 Tag components and CRC portions are optional.

2.1.Frame Header

The first 32 bits (four bytes) of every frame compose the frame's header. This portion contains metadata related to the frame itself including bitrate and sampling frequency values. If the use of CRC protection was specified during the MP3 encoding process, the Protection field within every frame's header portion will contain a value of zero and the 16 bits (two bytes) following every frame's header portion will compose the frame's CRC value. The algorithm used to calculate the CRC value is CRC16. The last 16 bits (two bytes) of the header portion along with the entire side information portion are used as inputs to the CRC16 algorithm. [10] Frames that contain Protection field values that specify the use of CRC values and have missing or incorrect CRC values will be skipped by MP3 playing and decoding applications. [8]

2.2.Side Information

The side information portion, which exists in every frame, contains the metadata needed to decode the encoded audio information contained within every frame's main data portion. The size of the side information portion is dependent upon the channel mode specified during the MP3 encoding process. For MP3 files encoded using the single channel mode, the 17 bytes following a frame's header portion (or CRC value, if one exists) will compose the frame's side information portion. For MP3 files encoded using the dual channel, stereo, or joint stereo mode, the 32 bytes following a frame's header portion (or CRC value, if one exists) will compose the frame's side information portion. [7] In this portion of the frame there is also valuable information about the location of certain bits. For example the "region0_count" and "region1_count" fields hold the lengths in bits of "region0" and "region1." These can be used to determine the beginning and ending locations of a channel's particular regions.

Some other relevant bits for this project are "part2_3_length" field that holds the total length in bits for a particular channel (or granule in the case of a single-channel MP3 file). It can be used to determine the beginning and ending locations of a particular granule or channel. The "table_select" and "count1table_select" fields are used to specify which Huffman Coding tables are to be used in order to decompress the Huffman-encoded data stored within the three regions of a particular channel (or granule in the case of a single-channel MP3 file). These Huffman Coding tables have been pre-defined and are part of the MP3 standard. They appear in the "Table B.7 – Huffman codes for Layer III" section of the standard[4]

2.3.Main Data

In this subsection, we will discuss the components that compose the main data portion of a post-encoding MP3 file's frames as well as how they are structured. The main data portion is where we intend to focus our research in order to store hidden information without greatly impacting the MP3 file's audio quality.

The major components of an MP3 file's main data portion include two granule portions which are each composed of either one or two channel portions depending on whether the MP3 file was encoded in single or dual channel mode. These channel portions are composed of scale factor values followed by Modified Discrete Cosine Transform- (MDCT-) encoded audio subband values which have been compressed using a lossless form of compression known as Huffman Coding.

Included below is a basic explanation of the main data portion. We will add more detail as we get a better understanding of how it is structured and put together. In the final draft of this paper, the length of this subsection should be 1/2 to 1 column in length.

The main data portion present within every frame contains the encoded audio information. The size of a frame's main data portion is dependent upon the values of the bitrate and sampling frequency fields present in the frame's header portion. It consists of two granule components, Granule 0 and Granule 1, which are further divided into left and right channel portions for MP3 files encoded using the dual channel, stereo, or joint stereo mode. Channel portions are composed of scale factor values followed by Modified Discrete Cosine Transform- (MDCT-) encoded audio subband values which have been compressed using a lossless form of compression known as Huffman Coding. For performance reasons, the audio information contained within each channel (scalefactor values and MDCT-encoded frequency subband values) is split into 3 regions named "region0," "region1," and "region2," and each of these regions is compressed individually using different Huffman Coding tables (if necessary) This tables are defined in the MP3 standard.The standard defines 15 large tables for these regions, where each table outputs two frequency samples for a given code word. The tables are designed to compress the "typical" content of the frequency regions as much as possible. [2]

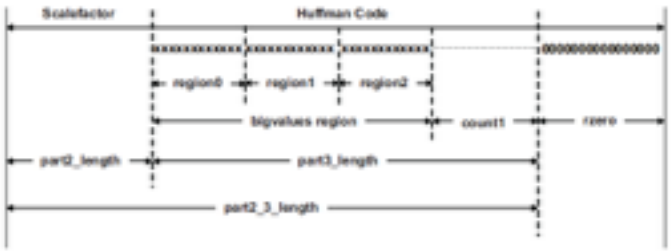

Figure 2.Illustration of main data organization and encoding scheme.[11]

It is important to note the audio information referenced by a particular frame's header and side information portions may actually exist within the main data portion of one or more preceding frames. It is possible for audio information to not exist at all in the main data portion of the frame in which it is referenced; this is due to a space-saving feature of the MP3 encoding format known as the bit reservoir. The bit reservoir feature allows frames to share unused main data portion space with any number of frames that follow. As an example, a frame may be allocated 500 bytes of main data portion space based on its bitrate and sampling frequency field values. If only 300 of those bytes are used to store audio information referenced by that frame's header and side information portions, the frame that follows will store the audio information referenced by its header and side information portions starting with the first unused byte of the preceding frame's main data portion (the 301st byte in this case). This ensures that no unused space exists within the main data portion of a frame and may result in overall MP3 file size decrease. [8]

2.4.Ancillary Data

Following a frame's main data portion, there may exist a number of bits composing the ancillary data portion. The number of bits within the ancillary data portion usually ranges from one to seven (if it exists) depending on the number of bits left over in the last incomplete byte present in a frame's main data portion. In some cases, the author found the number of bits in a frame's ancillary data portion to exceed seven bits. This was especially true for frames containing a partially empty main data portion, usually found toward the end of an MP3 file. The purpose of ancillary data is mainly to round a frame's main data portion up to an integer number of bytes.

3.EXISTING STEGANOGRAPHY TECHNIQUES AND APPLICATIONS

In this section, we will discuss how various existing MP3 steganography applications work and how the steganography techniques they make use of differ from the one that will be used by the application we plan on developing. In the final draft of this paper, this section should be 1/4 to 1/2 of a column in length.

3.1.MP3Stego

MP3Stego is a tool developed by Fabien Petitcolas. This tool hides information within the main data portions of an MP3 file.

The application takes a .wav file and a file to be hidden. The tool will compress and encrypt the file too hidden first. Then it will inject this file into the MP3 bit stream. The hiding is performed during the"inner_loop" stage of the encoding process which is where quantization of the input takes place.

This tool is different from the tool that we will to attempt to develop because MP3Stego hides the information during the encoding process.

3.2.MP3Stegazaurus

In this subsection, we will detail how Mikhail Zaturenskiy's MP3Stegazaurus application works and how the steganography technique it makes use of differs from the one that will be used by the application we plan on developing. In the final draft of this paper, this subsection should be 1/4 to 1/2 of a column in length.

MP3Stegazaurus is a powerful MP3 steganography application recently developed by Mikhail Zaturenskiy, a former IIT student. It injects covert information into MP3 files by overwriting the fields within the previously mentioned non-audio-information-containing portions of interest that MP3 playing and decoding applications skip or ignore. It can also retrieve previously-injected covert information as long as a user is aware of and is able to specify exactly which method was used to inject it and which file type extension is to be appended to it. Finally, it cleans potential carrier MP3 files of all previously-injected covert information by using a version of the injection method that overwrites the fields within the specified portions of interest with a repeating zero value.

4.[APPLICATION NAME]

In this section, we will describe the application we plan on developing in detail by going over the various methods it will use to hide hidden information within an MP3 file. In the final draft of this paper, this section should be 2 to 4 columns in length.

4.1.Testing

In this subsection, we will discuss the testing process that we will perform on our application. We will take note of the results of this testing and use the results to modify and adjust the hiding methods as needed. In the final draft of this paper, this subsection should be .5 to 1.5 columns in length.

5.CONCLUSIONS

In this section, we will document our findings and what we learned while completing the project. We will also document challenges we faced during the project and how we were able to overcome them. In the final draft of this paper, this section should be .5 to 1 page long.

6.ACKNOWLEDGEMENTS

We will include acknowledgements in this section. In the final draft of this paper, this section should be .25 to .5 columns in length.

7.REFERENCES

1. Bosi, Marina, and Richard E. Goldberg. Introduction to Digital Audio Coding and Standards. Boston: Kluwer Academic, 2003. Print.
2. Edström., Björn. "Blog.bjrn.se." Blog.bjrn.se. N.p., n.d. Web. 25 Nov. 2014.
3. Hacker, Scot. MP3: The Definitive Guide. Sebastopol: O'Reilly Media, 2000. Print.
4. ISO/IEC 11172-3 – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s, 1993.
5. Maciak, Lukasz G., Michael A. Ponniah, and Renu Sharma. MP3 Steganography: Applying Steganography to Music Captioning.

6. Nilsson, Martin, “ID3 tag version 2.3.0”, 1999 ID3, http://www.id3.org/id3v2.3.0
7. Raissi, Rassol. The Theory Behind MP3.
8. Ruckert, Martin. Understanding MP3: Syntax, Semantics, Mathematics, and Algorithms. Wiesbaden: Vieweg, 2005. Print.
9. Supurovic, Predrag, “MPEG Audio Frame Header”, 1999 DataVoyage, http://www.datavoyage.com/mpgscript/mpeghdr.htm
10. Zaturenskiy, Mikhail. “MP3 Files as a Steganography Medium.” Illinois Institute of Technology, 2013.
11. Thiagarajan, Jayaraman Jayaraman., and Andreas Spanias. Analysis of the MPEG-1 Layer III (MP3) Algorithm Using MATLAB. San Rafael, CA: Morgan & Claypool, 2012. Print.