Probabilistically Detecting Steganography within MP3 Files

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

A powerful application named MP3Stegazaurus was recently created by Illinois Institute of Technology (IIT) student Mikhail Zaturenskiy which exploits a not-so-well-known feature present in MP3 files. [8] By overwriting areas that are either skipped or ignored by MP3 playing and decoding applications, MP3Stegazaurus can use any MP3 file to safely store covert information in a manner that is difficult, if not impossible, for the average user to detect. This is especially true since it leaves an MP3 file's audio information untouched.

As a result of research and work which took place during the first half of a student project, the author developed MP3StegDetector; an application which examines the interesting, non-audio-information-containing portions of MP3 files in order to determine and report whether, and where, steganography may be present. In addition to detecting the potential presence of steganography, MP3StegDetector extracts and outputs the information contained within each of the interesting portions of a given MP3 file. If a complete file such as a JPEG image, PDF document, or AES-encrypted file is identified within this extracted information, MP3StegDetector will extract and recompose it.

In order to determine exactly where in MP3 files these interesting portions exist, the author carried out extensive research on the MP3 file's post-encoding format. This research is included in this document along with details of how MP3StegDetector's various scanning functions and features have been implemented.

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, Steganalysis, MP3, Anomaly Detection, Secret Writing, Hidden Information, Blind Detection

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 format is perceptual, taking into account the ability of humans to perceive certain frequencies of sound in order to intelligently compress digital audio information. Both lossy and lossless forms of compression take place during the MP3 encoding process. [2]

During the MP3 encoding process, the original, 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. [2]

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

Because every frame within an MP3 file, regardless of its bit rate and sampling frequency, contains exactly 26 milliseconds worth of audio information, the number of frames that exist within an MP3 file is directly proportional to the duration of all of the audio

information contained within it. [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.

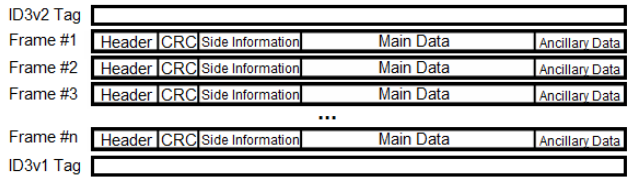

Figure 1. Structure of a complete MP3 file.
Note: ID3 Tag components and CRC portions are optional.

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. While the author's project is dependent upon the values contained within all of the fields present in the header portion of every frame in an MP3 file, the ones that are truly interesting include the Private, Copyright, Original, and Emphasis fields. These fields can safely be overwritten in order to store covert information since they are skipped and ignored by MP3 playing and decoding applications. [8]

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. [8] This means whenever the interesting fields within a frame's header and/or side information portion are overwritten in order to store covert information, a new CRC value must be calculated and used to overwrite the original CRC value for that frame. 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. [6]

2.2 Side Information

The side information portion, which exists in every frame, contains the metadata needed to decode the 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. [5]

The author's project depends upon the information contained within several of the fields of the side information portion of every frame, however the only field that is of true concern is the padding_bits (or private_bits) field. This field contains a number of bits (five bits for MP3 files encoded using the single channel mode and three bits for MP3 files encoded using the dual channel, stereo, or joint stereo mode) and its only purpose is to round the side information portion up to an integer number of bytes. This field can therefore be safely overwritten in order to store covert information as it is skipped and ignored by MP3 playing and decoding applications. [8]

2.3 Main Data

The main data portion present within every frame contains the 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 factors and Huffman Code bits.

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 results in overall MP3 file size decrease. [6]

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. It is skipped and ignored by MP3 playing and decoding applications and can safely be overwritten in order to store covert information. [8] The author's project, therefore, is concerned with the ancillary data portions that may exist within the frames of an MP3 file.

3. MP3STEGAZAURUS

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. MP3STEGDETECTOR

MP3StegDetector is a steganalysis application developed by the author which examines the interesting, non-audio-information-containing portions of MP3 files where covert information can be safely be stored without affecting playback in order to report whether, and where, steganography may exist. In addition to detecting the presence of steganography, MP3StegDetector extracts and outputs the information contained within each of the portions of interest to files which represent the extracted information in human-readable bit and byte formats. If complete files such as JPEG images or PDF documents are identified within this extracted information, MP3StegDetector extracts and recomposes them. In addition to its scanning features, an added feature called the MP3 Frame Viewer provides a user with detailed information contained within the specified frames of an MP3 file.

Currently, MP3StegDetector is a Java application which can be used from both a Graphical User Interface and an operating system's command line interface (assuming the Java Runtime Environment is installed and appropriately configured). Single and dual channel MP3 files that use constant or variable bitrates are supported as are MP3 files that use any combination of bitrate and sampling frequency values. For testing purposes, the author had access to a training set consisting of 200 clean, unaltered MP3 files retrieved from various sources including Microsoft, Amazon, iTunes, Audacity, GoldWave, FreeCorder, BladeEnc, and Hulkshare.

4.1 Unused Header Bit Scanning

The four fields of interest present within the header portion of a frame include the one-bit Private field, the one-bit Copyright field, the one-bit Original field, and the two-bit Emphasis field. After examining the frames within several single channel and dual channel MP3 files available in his training set, the author observed the values of these four fields remained the same throughout every frame of a clean, unaltered MP3 file. The Unused Header Bit scanning function of MP3StegDetector therefore examines the values of these four fields for every frame within an MP3 file and tracks the number of value changes that occur from one frame to the next.

The scanning process begins by locating the first frame within the MP3 file to be scanned. Once located, several fields within the frame's header portion are examined in order to determine the frame's characteristics. These characteristics include channel mode, bitrate, and sampling frequency as well as whether the frame has its Padding field value set to 1. These characteristics are used to determine the frame's length in bytes. Next, the values of the frame's Private, Copyright, Original, and Emphasis fields are located and stored so they can be compared later with the values of the same fields in the frame that follows. In order to locate the following frame, a number of bytes equal to the current frame's length is skipped and a check is performed to ensure the next valid frame has been accurately located. Again, several fields in this frame's header portion are examined in order to determine its characteristics and determine its length in bytes. The values of this frame's Private, Copyright, Original, and Emphasis fields are then stored separate from those of the previous frame.

The stored values of these two frames' Private, Copyright, Original, and Emphasis fields are then compared. If the values do not match, then a counter that keeps track of the number of value changes for a particular field is incremented by 1. Once the comparison of the field values for these two frames has completed, the next valid MP3 frame is located by skipping ahead a number of bytes equal to the length of the frame that was last examined. This process repeats until the last valid frame in the MP3 file has been reached.

If zero header portion field value changes are counted in a particular MP3 file, it is determined the probability of steganography being present within the header portions of the MP3 file's frames is equal to zero. If one or more header portion field value changes are counted, a percentage value is derived by dividing the total number of header portion field value changes by the total number of interesting header portion field bits present in the MP3 file (this second value is found by multiplying the total number of frames in the MP3 file by five). This anomalous header portion bit percentage value is then interpreted by a user in order to determine whether steganography exists within the header portions of the MP3 file.

Additionally, by examining the bit and byte representations of the information extracted from the interesting header portion fields of an MP3 file's frames, a user is able to independently determine whether patterns or signatures not properly identified by MP3StegDetector may be present.

4.2 Side Information Padding Bit Scanning

The side information portion of a frame contains a Padding (padding_bits, private_bits) field which, according to Martin Ruckert, serves no purpose other than to round the side information portion up to an integer number of bytes. In single channel MP3 files, the Padding field is five bits in length while in dual channel MP3 files, it is three bits in length. After examining the frames within several single channel and dual channel MP3 files available in his training set, the author observed the value of the Padding field was equal to zero in every frame of a clean, unaltered MP3 file. The Side Information Padding Bit scanning function of MP3StegDetector therefore examines the value of the Padding field within every frame of an MP3 file and tracks the number of bits which have non-zero values.

The scanning process begins by locating the first frame within the MP3 file to be scanned and calculating its length in bytes by using the same techniques implemented by the Unused Header Bit Scanning function. Once located, the value of the Padding field within the frame's side information portion is examined. If the value of this field is found not to be zero, the number of bits within it which have non-zero values is counted. In order to locate

the following frame, a number of bytes equal to the current frame's length is skipped and a check is performed to ensure the next valid frame has been accurately located. Again, this frame's length in bytes is determined by using the same technique implemented by the Unused Header Bit Scanning method. The value of this frame's Padding field is located and examined in order to count the number of bits with non-zero values. This process repeats until the last valid frame in the MP3 file has been reached.

If zero bits within the Padding fields of an MP3 file's frames are found to have non-zero values, it is determined the probability of steganography being present within the side information portions of the MP3 file's frames is equal to zero. If one or more bits within the Padding field of an MP3 file's frames were found to have non-zero values, a percentage value is derived by dividing the total number of bits found to have non-zero values by the total number of side information portion Padding field bits present in the MP3 file (this second value is found by multiplying the total number of frames in the MP3 file by five for a single channel MP3 file or by three for a dual channel MP3 file). This anomalous side information Padding field bit percentage value is then interpreted by a user in order to determine whether steganography exists within the side information portions of the MP3 file.

Additionally, as with the Unused Header Bit scanning function, by examining the bit and byte representations of the information extracted from the side information portion Padding field of an MP3 file's frames, a user is able to independently determine whether patterns or signatures not properly identified by MP3StegDetector may be present.

4.3 Empty Frame Scanning

Some MP3 files contain a number of "empty frames" which are usually located toward the beginning and/or the end of an MP3 file. [8] Empty frames are just like regular frames except their main data portions contain no audio information. Because of this, the main data portions of empty frames can safely be overwritten in order to store covert information. In order to determine whether a particular frame can be considered empty, one needs to look only at the big_values and table_select fields present within its side information portion. [8] If the value for each of these fields is equal to zero, then it can be determined the frame's main data portion contains no audio information. It is important to keep in mind that because of the bit reservoir feature the MP3 encoding format makes use of, the beginning of the information within an empty frame's main data portion may actually exist in one or several of the preceding frames' main data portions.

After examining the empty frames present within several single channel and dual channel MP3 files available in his training set, the author determined that MP3 encoding applications treat empty frames differently. Some MP3 insert differing types of information into empty frames while others insert no information at all. Because of this, no clear and unique patterns could be used as a basis for determining whether anomalous information exists within the main data portion of an empty frame. However, the information within the main data portions of empty frames is still extracted by MP3StegDetector for independent examination by a user.

The extraction process begins by locating the first frame within the MP3 file to be scanned and calculating its length in bytes by using the same techniques implemented by the previously mentioned scanning functions. The value of every frame's big_values and table_select fields present in its side information portion is examined in order to determine whether its main data portion contains audio information. If it is determined the main data portion of a frame contains no audio information, the value of the frame's main_data_begin field present in its side information portion is examined in order to determine where in the bit reservoir the frame's main data portion begins. If the value of the main_data_begin field is zero, then the frame's main data portion begins right after its own side information portion. If instead the value of the main_data_begin field is greater than zero, then the frame's main data portion begins within the main data portion of one of the preceding frames.

A non-zero main_data_begin field value is a negative offset representing the beginning location of a frame's main data portion, and it does not account for the sizes of previous frames' header, optional CRC, and side information portions. As an example, if the value of a frame's main_data_begin field is 500, then its main data portion begins 500 bytes ahead of its own header portion, not including the sizes of the previous frames' header portion (always four bytes), optional CRC portion (always two bytes, if it exists), and side information portion (17 bytes for single channel MP3 files, 32 bytes for dual channel MP3 files). [5]

After locating the beginning location of an empty frame's main data portion, the length in bits of the empty frame's main data portion is determined by examining each of the part2_3_length fields contained within its side information portion. All of the information contained within the empty frame's main data portion is then extracted. The next frame within the MP3 file is then located and the process mentioned above is repeated in order to determine whether its main data portion contains audio information. If it is determined the frame does not contain audio information, the beginning location and length in bits of the frame's main data portion are determined in order to extract the information contained within. This process repeats until the last valid frame in the MP3 file has been reached.

4.4 Ancillary Data Bit Scanning

The length of a frame's ancillary data portion varies depending on how many bits are left over in the last incomplete byte of its main data portion. The main purpose of these bits is to round a frame's main data portion up to an integer number of bytes which means the ancillary data portion is usually one to seven bits in length, though in some cases it may be longer. It is possible for a frame not to have an ancillary data portion at all, as is the case when there exists a full eight bits in the last byte of a frame's main data portion. After examining the ancillary data portions present within several single channel and dual channel MP3 files available in his training set, the author observed MP3 encoding applications set the values of the bits within ancillary data portions to a repeating 00, a repeating 11, a repeating 01, or a repeating 10 pattern. The Ancillary Data Bit scanning function of MP3StegDetector therefore examines the values of all of the bits extracted from the ancillary data portion of each frame (if one exists) within an MP3 file and tracks the number of occurrences where a particular bit does not follow one of the four observed patterns.

The scanning process begins by locating the first frame within the MP3 file to be scanned and calculating its length in bytes by using the same techniques implemented by the previously mentioned

scanning functions. Next, the beginning location of this frame's main data portion is determined in addition to the beginning location of the following frame's main data portion by examining the main_data_begin field within each frame's side information portion. The values of each of the first frame's part2_3_length fields are examined in order to determine the length in bits of only its main data portion. The number of ancillary bits that exist in-between these two frames' main data portions is then calculated by doing the following: First, the number of bits in the first frame's main data portion is added to the value of the second frame's main_data_begin field (which must first be multiplied by eight since its value represents an integer number of bytes). Then, the length in bits of the first frame is determined. Finally, the number of bits found during the first step is subtracted from the number of bits found during the second step. Using this number of ancillary data portion bits now known to exist in the first frame, a length of information which begins directly after the end of the first frame's main data portion is extracted and examined.

In order to locate and examine the ancillary data portion bits of the frame that follows (if they exist), a number of bytes equal to the current frame's length are skipped and a check is performed to ensure the next valid frame has been accurately located. Again, this frame's length in bytes is determined by using the same technique implemented by the Unused Header Bit Scanning method. This process repeats until the last valid frame in the MP3 file has been reached.

If zero bits within the ancillary data portion of every frame are found to not follow one of the four patterns observed in clean, unaltered MP3 files, it is determined the probability of steganography being present within the ancillary data portions of the MP3 file's frames is equal to zero. If one or more bits within the ancillary data portion of an MP3 file's frames are found to not follow one of the four patterns observed in clean, unaltered MP3 files, a percentage value is derived by dividing the total number of bits found to not follow one of the four patterns observed in clean, unaltered MP3 files by the total number of bits present in the ancillary data portions of the MP3 file. This anomalous ancillary data portion bit percentage value is then interpreted by a user in order to determine whether steganography exists within the ancillary data portions of the MP3 file.

Additionally, as with the previously mentioned scanning functions, by examining the bit and byte representations of the information that is extracted from the ancillary data portions of an MP3 file's frames, a user is able to independently determine whether patterns or signatures not properly identified by MP3StegDetector may be present.

4.5 MP3 Frame Viewer

The MP3 Frame Viewer feature of MP3StegDetector allows for the retrieval and viewing of detailed information regarding any user-specified frames within an MP3 file. The following information retrieved from a frame's header portion is displayed: MPEG audio version, layer value, Protection field state, bitrate value, sampling frequency value, Padding field state, channel mode, mode extension, emphasis type, Private field state, Copyright field state, Original field state, reported length value in bytes, and actual length value in bytes. The following information retrieved from a frame's side information portion is displayed: main_data_begin field value in bytes and Padding field value in bit format, as well as the values of the part2_3_length, big_values, and table_select fields of all of the granule and channel portions contained within a frame's main data portion. Finally, the length in bytes of the main data portion is displayed along with the information contained within in human-readable bit and byte formats.

Although the author did not intend to include this feature in MP3StegDetector, it was decided it should be after realizing it could be a useful tool for independently detecting steganography and other anomalies at a low level, as well as be useful in other MP3-related applications.

4.6 Testing

The process of testing MP3StegDetector consisted of gathering clean, unaltered MP3 files from various online sources in order to create a training set consisting of 25 MP3 files retrieved from each source for a total of 200 MP3 files. This training set consisted of MP3 files of single and dual channel mode types making use of varying bitrate and sampling frequency values. Once a complete training set of clean, unaltered MP3 files had been gathered, a copy of it was created. MP3Stegazaurus was then used to inject varying types of test information into the interesting portions of the MP3 files within the training set copy. In the end, the author had access to two training sets consisting of clean, unaltered MP3 files and MP3 files carrying covert information of various known types in varying portions of interest.

Using the training set of carrier MP3 files, the author began to take note of the results produced by MP3StegDetector as each file was scanned in order to see whether it was properly determining values related to the probability of steganography and whether it was properly extracting information from the portions of interest. Once satisfied with the results MP3StegDetector was producing after scanning MP3 files from the carrier training set, the author began to scan the MP3 files from the clean training set in order to take note of the universal patterns observed in the information extracted from the MP3 files' portions of interest. It is these observed patterns that the detecting algorithms implemented by MP3StegDetector's scanning functions use in order to track the number of anomalies present and determine a value related to the probability of steganography within a particular MP3 file's portions of interest.

The author intends to eventually implement statistical methods into MP3StegDetector including regression analysis and/or linear discriminant analysis which may offer a solution for reporting values related to the probability of steganography present within the main data portions of empty frames. Other future functionality includes the ability for MP3StegDetector to detect and scan partially empty frames and custom frames potentially created and inserted by other MP3 steganography tools.

5. CONCLUSIONS

Using the patterns found to exist within the interesting portions of the 200 clean, unaltered MP3 files gathered by the author, a tool named MP3StegDetector was created which scans the interesting portions of MP3 files and reports values related to the probability of steganography present within those interesting portions. The patterns found to exist within the interesting portions of clean, unaltered MP3 files include the following:

Header Portion – The values of the Private, Copyright, Original, and Emphasis fields remain constant throughout every frame.

Side Information Portion – The values of the bits within the Padding field are equal to zero throughout every frame.

Ancillary Data Portion – The values of all of the bits within a frame's ancillary data portion follow a repeating 00, 11, 01, or 10 pattern. This is true for every frame where an ancillary data portion exists.

Empty Frame Main Data Portion – No universal patterns were found to exist. The values of the bits within an empty frame's main data portion may follow a repeating 00, 11, 01, or 10 pattern. These bits may also contain information related to the software or device used to encode and create a particular MP3 file.

6. ACKNOWLEDGEMENTS

The author would like to acknowledge the contributions made to this project by Erfan Setork, Kbrom Tewoldu, and Zach Wagner. The author would also like to thank Professor Bill Lidinsky for encouraging him to take on this project and for providing him with guidance and direction. Finally, the author would like to thank Mikhail Zaturenskiy for his work in developing MP3Stegazaurus and for taking the time to help the author understand how certain aspects of the MP3 encoding and decoding processes work.

7. REFERENCES

[1] Bosi, Marina, and Richard E. Goldberg. Introduction to Digital Audio Coding and Standards. Boston: Kluwer Academic, 2003. Print.

[2] Hacker, Scot. MP3: The Definitive Guide. Sebastopol: O'Reilly Media, 2000. Print.

[3] Maciak, Lukasz G., Michael A. Ponniah, and Renu Sharma. MP3 Steganography: Applying Steganography to Music Captioning.

[4] Nilsson, Martin, "ID3 tag version 2.3.0", 1999 ID3, http://www.id3.org/id3v2.3.0

[5] Raissi, Rassol. The Theory Behind MP3.

[6] Ruckert, Martin. Understanding MP3: Syntax, Semantics, Mathematics, and Algorithms. Wiesbaden: Vieweg, 2005. Print.

[7] Supurovic, Predrag, "MPEG Audio Frame Header", 1999 DataVoyage, http://www.datavoyage.com/mpgscript/mpeghdr.htm

[8] Zaturenskiy, Mikhail. "MP3 Files as a Steganography Medium." Illinois Institute of Technology, 2013.