Information hiding using IDv3 tag in MP3 file

Saad H. Abid*

ABSTRUCT

As we all know that digital information hiding (or steganography) is used to embed data (information) into digital media for the purpose of identification, annotation, and copyright. In other words it is the operation of hiding certain information into a specific cover media such as pictures and audio files or even in a simple text files. In this paper a new approach in hiding information into an MP3 file was proposed, simply instead of using the traditional method of hiding into the actual audio data we will hide our secret messages into the IDv3 tag which is a tag used to store specific information about the MP3 file.

The IDv3 tag has gone into a several changes to reach its current version (version 3) containing several interesting features that will be used as a cover for our secret message, first of all a simple overview for the versions and releases of the ID tag is illustrated and display the features of the IDv3 tag which we will use as a cover media.

*AL-Mansour University College, Computer science department

1. Introduction

The audio format MPEG layer I, layer II and layer III (MP3) has no native way of saving information about the contents, except for some simple yes/no parameters like "private", "copyrighted" and "original home" (meaning this is the original file and not a copy). A solution to this problem was introduced with the program "Studio3" by Eric Kemp alias NamkraD in 1996. By adding a small chunk of extra data in the end of the file one could get the MP3 file to carry information about the audio and not just the audio itself[1].

The placement of the tag, as the data was called, was probably chosen as there was little chance that it should disturb decoders. In order to make it easy to detect a fixed size of 128 bytes was chosen. The tag has the following layout:

Song title	30 characters
Artist	30 characters
Album	30 characters
Year	4 characters
Comment	30 characters
Genre	1 byte

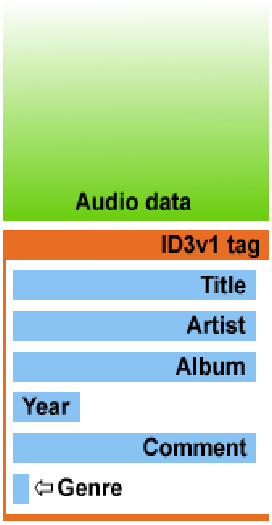

Fig. 1
Internal layout of an ID3v1 tagged file.

As all artists doesn't have a 30 character name it is said that if there is some bytes left after the information is entered in the field, those bytes should be filled with the binary value 0. You might also think that you cannot write that much in the genre field, being one byte big, but it is cleverer than that. The byte value you enter in the genre field corresponds to a value in a predefined list. The list that Eric Kemp created had 80 entries, ranging from 0 to 79 [4].

If you sum the size of all these fields we see that 30+30+30+4+30+1 equals 125 bytes and not 128 bytes. The missing three bytes can be found at the very beginning of the tag, before the song title. These three bytes are always "TAG" and is the identification that this is indeed an ID3 tag. The easiest way to find an ID3v1/1.1 tag is to look for the word "TAG" 128 bytes from the end of a file.

2 .ID3v1.1Tag

ID3 v1.1 tag still had some obvious limitations and drawbacks, though. It supported only a few fields of information, and those were limited to 30 characters, making it impossible to correctly describe "The Hitchhiker's Guide to the Galaxy from BBC Radio" as well as "P.I. Tchaikovsky's Nutcracker Suite Op. 71 a, Ouverture miniature danses caractéristiques by The New Philharmonic Orchestra, London, conducted by Laurence Siegel". Since the position of the ID3 v1.1 tag is at the end of the audio file it will also be the last thing to arrive when the file is being streamed. The fix size of 128 bytes also makes it impossible to extend further. That's why was thought that a new ID3 tag would be appropriate.

3 .ID3v2Tag

ID3v2 is a new tagging system that lets you put enriching and relevant information about your audio files within them. In more down to earth terms, ID3v2 is a chunk of data pretended to the binary audio data. Each ID3v2 tag holds one or more smaller chunks of information, called frames. These frames can contain any kind of information and data you could think of such as title, album, performer, website, lyrics, equalizer presets, and pictures etc. Figure (2) is an example of how the layout of a typical ID3v2 tagged audio file may look like.

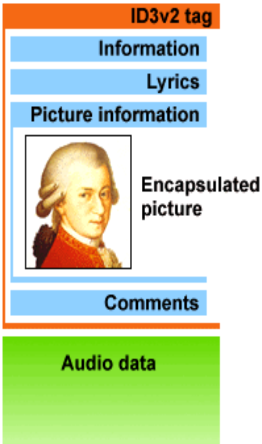

Fig. 2 layout of a typical ID3v2 tag

One of the design goals was that the ID3v2 should be very flexible and expandable. It is very easy to add new functions to the ID3v2 tag, because, just like in HTML, all parsers will ignore any information they don't recognize. Since each frame can be 16MB and the entire tag can be 256MB you'll probably never again be in the same situation as when you tried to write a useful comment in the old ID3 being limited to 30 characters.

Speaking of characters, the ID3v2 supports Unicode so even if you use the arabic character set you'll be able to write in your native language. You can also include in which language you're writing so that one file might contain e.g. the same lyrics but in different languages.

Even though the tag supports a lot of byte consuming capabilities like inline pictures and even the possibility to include any other file which is a very suitable place to hide a secret information inside it, ID3v2 still tries to use the bytes as efficient as possibly. If you convert an ID3v1 tag to an ID3v2 tag it is even likely that the new tag will be smaller[4].

4 A quick look at Information hiding (steganography)

Information (or data) hiding embeds data into digital media for the purpose of identification, annotation, and copyright. cryptography protects the content of messages while steganography conceals their very existence[1].

Or it is “Hiding a secret message within a larger one in such a way that others can not discern the presence or contents of the hidden message” Several constraints affect this process:

- The quantity of data to be hidden
- The need for invariance of these data under distortion of the cover signal
- The degree to which the data must be immune to interception modification, or removal by a third party.

The information-hiding problem space was characterized by a trade-off between robustness and bandwidth. By constraining the degree of cover-signal degradation, an information-hiding method can operate with either high embedded data rate or high resistance to modification, but not both. As one increases, the other must decrease [1].

5 What dose the tag contains that can be used as a potential host for data?

- The tag supports unicode (secret messages written in any language can be used such as arabic).
- It has several new text fields such as composer, conductor, media type, BPM (Bits per Minutes), copyright message, etc. and the possibility to design your own as you see it fit.
- Can contain lyrics as well as music-synced lyrics (karaoke) in almost any language.
- Is able to contain volume, balance, equalizer and reverb settings.
- Is able to contain images and just about any file you want to include.

- **Supports enciphered information, linked information and web-links.**

5.1 Structure:

TAG ID3v2 consists of header **and** frames **that the information resides in it the following will give a detailed explanation of this tag [5]:**

Header (10 Bytes)

Bytes	Content
0-2	TAG identifier. It contains of string "ID3"
3-4	TAG version. Can be e.g. 03 00
5	Flags
6-9	Size of TAG

Flags Byte has this structure (in bits): (abc00000) where the first 3 bits indicate Unsynchronization, Extended header and Experimental indicator. Flags normally don't have special meaning, can be set to 00.
Size of TAG is encoded into 4 Bytes. But not to be so easy, the most significant bit in each Byte is set to 0 and ignored. Only remaining 7 bits are used. The reason is to avoid mismatch with audio frame header which has the first synchronization Byte FF).

Example:-
TAG length 257 is encoded as 00 00 02 01.
Size of TAG doesn't contain header itself so total length of previous TAG is 257 + 10 Bytes.

5.2 Frames

Each frame is used for storing one information - e.g. Artist or Album.
Frame consists of header and body[2].

Header(10 Bytes)

Bytes	Content
0-3	Frame identifier
4-7	Size
8-9	Flags

Frame identifier consists of four characters. There are many predefined values you can use. But e.g. current version of WinAmp displays only these ones:

Iden.	Description
TRCK	Track number
TENC	Encoded By
WXXX	URL
TCOP	Frame identifier
TOPE	Original Artist
TCOM	Composer
TCON	Genre
COMM	Comments
TYER	Year
TALB	Album
TPE1	Artist
TIT2	Song name

You can freely define you own frame, e.g. with identifier "CUNT" but you cant expect that some player will be able to display that. Anyway you can store any information into MP3 file without limits.

Size is stored from the most significant Byte to the least. It doesn't include frame header, so total length of frame is Size + 10. Warning: After the header always one Byte with value 00 follows and then begins frame body. Size has to include this Byte.

Flags in most cases are set to 00 00. But they have this structure (in bits): abc00000 ijk00000

Flag	Description
a	TAG alter preservation
b	File alter preservation
c	Read only
i	Compression
j	Encryption
k	Grouping identity

Example of frame:

```
54 50 45 31 00 00 00 07 00 00 00 53 53 6C 61 79 65 72
 T  P  E  1           7           S  l  a  y  e  r
```

To be even more complicated, some frames can have special structure. Ex:COMM (comments) also contains language version which is not normally displayed[2]:

```
4 4 4 4 0 0 0 0 0 0 0 6 6 6 0 6 6 6 6 6 6 7
3 F D D 0 0 0 C 0 0 0 5 E 7 0 3 F D D 5 E 4
              0
C O M M       C       e n g   c o m m e n t
```

For TCON (genre or style) you can use predefined values as in

TAG v1. Then frame body looks like "(144)Thrash Metal" and the number corresponds with TAG v1. Or you can simply write your own style and then body is "Brutal Black" and is stored like normal frame body.

6 The Proposed approach

The proposed approach consists of exploiting the information stored inside the IDv3 tag as a cover media, since this tag has many multimedia such as still images and text each of which can be a potential host for the hidden information, another approach can be used also by adding a new frame just before the original actual frame we can force the decoder to skip that frame and display the information inside the actual frame. Furthermore we use cryptography to enhance the secrecy of the stored information.

This is not a very difficult way to hide information but yet it is very effective because MP3 files are very widely used and distributed freely around the world, the speed of the system is very high because it dose not involve high computational processes and data frame decoding etc.

In case of hiding information inside a still image contained into the MP3 tag frame we simply follow these algorithmic steps to do the job:

1. locate the MP3 file that we want to embed data in
2. Initialize a pointer to the information in the last address of the MP3 file.
3. Jump as much as the frame size information to locate the token "ID3".
4. Read the 4 byte information that holds the size of the tag.
5. Find the information of the still image information normally it is found in the beginning of the still image stream file where FFD8 is found which it is the starting tag of all jpeg images followed by the tag JFIF.
6. use any of the jpeg information hiding methods to hide the secret information inside the image stream

7. Save all data and close the stream, then you have an ID3 tag with hidden information.

Yet if we wanted to hide into a text tag we can simply write into the genre, title, or the URL field using a coding method such as www.21river-riders33.hos which means for example there are hostile forces riding by the river at coordinates 21 and 33.

Or just by creating another frame of cryptic information that may hold the secret message, this frame will be ignored by the decoders of MP3 players and the steg-analyst wouldn't have the reason to suspect the file[3].

7. The information hiding engine

Once the tag is located into the MP3 stream the data is extracted and an information hiding engine is invoked to process these data and hide the secret information inside it, keeping in mind that the extracted data could be a plain text representing an artist name, genre or song title as in the previous versions of tags or an image representing an album cover or a picture of the singer/band these information is processed according to the types and the method of hiding is selected such as text-in-text of data-in-image. We said data-in-image not text-in-image because the secret data could be in any form that can hold the secret information.

To make the process more secure we use a 128-bit DES algorithm to encrypt the secret data thus even if the secret data is uncoverd in some way it still can hold its contents from being revealed by a third parity, the system is shown in Figure(3) for better comprehension[4].

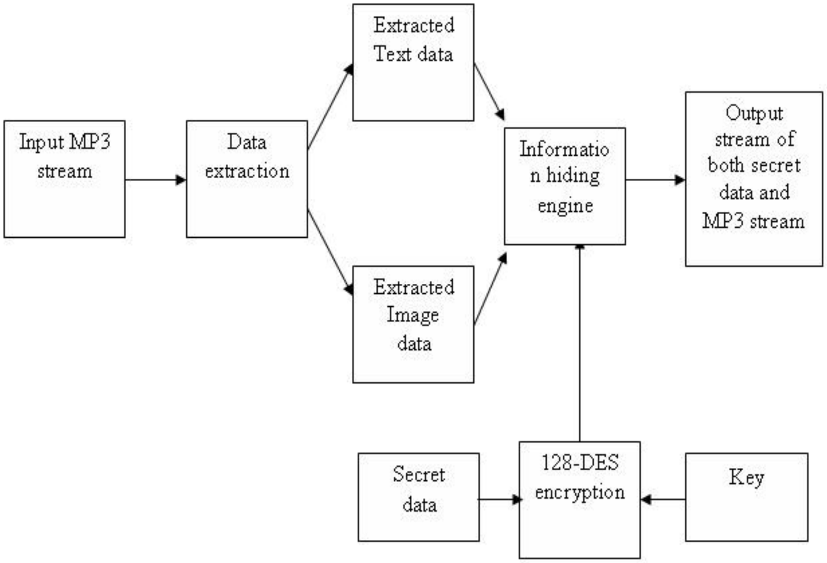

Figure 3
A diagram of the system

8. System components description

Input MP3 stream: is the input song that has the ID3v2 tag and used as the cover media.
Data extraction: is the operation that concern with extracting the data of the ID3v2 tag the extracted data could be an image of the band or the lyrics of the song.
Secret data: is the secret information that we need to hide into the tag data and the system accepts many types of data such as images, sound, text… etc.
128-DES encryption: is a typical data encryption standard algorithm to encrypt the secret data using a specific key before

hiding it into the cover media that acts as a second defensive line against the steganalyst.

Output stream: is the result of combining the cover media with the encrypted secret data.

Information hiding engine: this is the main component of the proposed system that contains the most important operations, the methods of hiding depends on the type of input data hence if the input data is a text then a text-in-text data hiding is performed after the encryption operation and if the input data is an image then one of the known method of image hiding is used and so on.

The output of the proposed system is sown in figure 4 below:

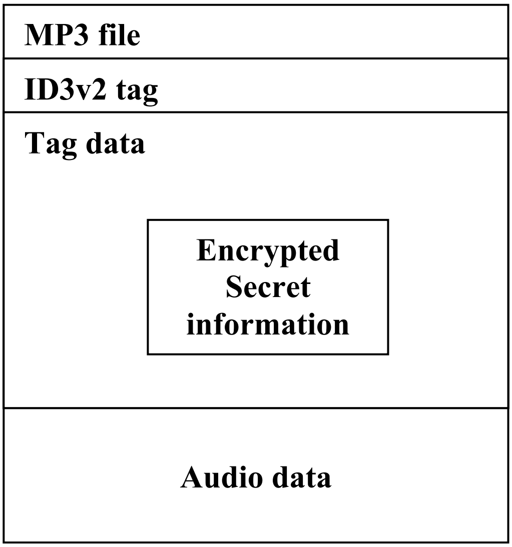

Figure 4
The output of the proposed system

9 Conclusion

This proposed approach exploits the characteristics of the ID3v2 tag to sneak information onto communication channel. Also using numerous of frames and song files to hide a single piece of information makes it chances to reach the destination better than loading the hole information into one MP3 file, thus some of them succeed and others might not get that lucky. On the other hand sending small fragment of information makes it more suspicious and the user might get a little afraid and terminate the current connection.

Refrences

[1] Andreas Pfitzmann "Steganographic Systems, and information hiding methods**" -the fifth international workshop on information hiding June 2004 p19-25**

[2] Kontextové odkazy "MP3 Inside**" a website**

http://www.multiweb.cz/twoinches/MP3inside.htm

[3] Kai S. james "MP3TagID3v2-Data**"**

http://www.sans.org/rr/steg/mp3stego.php

[4] Fraunhofer "Multimedia Security Technologies**"**

http://www.iis.fhg.de/amm/techinf/layer3/index.html

[5] Mark Noto "Hiding Text in MP3 Files**"**

http://mpgedit.org/mpgedit/mpeg_format/MP3Format.html

اخفاء معلومات في تعليمة المحتويات للملف الصوتي MP3

م.م. سعد حميد عبد

المستخلص

ان اخفاء المعلومات انتشر في الاونة الاخيرة على مدى واسع وبمختلف الطرق والاساليب والذي بدوره يستخدم لاخفاء بيانات داخل الاوساط المتعددة وتطبيقاتها لامنية الاوساط المتعددة ، وبعبارة اخرى هو عملية اخفاء معلومة معينة داخل وسط معين كصورة او صوت وا حتى ملف نص بسيط. في هذا البحث سوف نستخدم طريقة جديدة لأخقاء المعلومات داخل الملفات الصوتية من نوع MP3 وبدل استخدام الطريقة التقليدية التي تقوم يأخفاء المعلومات داخل بيانات الصوت نفسها سوف نقوم بأخفاء المعلومات داخل حيز يدعى ب ID3 tag والمخصص للأحتفاظ بمعلومات تخص ماهية الملف الصوتي وليس معلومات الصوت نفسها.

* كلية المنصور الجامعة/قسم علم الحاسبات ونظم المعلومات