StegIbiza: Steganography in Club Music Implemented in Python

Krzysztof Szczypiorski
Warsaw University of Technology, Warsaw, Poland
Crypotmage SA, Wroclaw, Poland
Email: ksz@tele.pw.edu.pl

Wojciech Zydecki
Warsaw University of Technology, Warsaw, Poland
Email: w.zydecki@gmail.com

Abstract—This paper introduces the implementation of steganography method called StegIbiza, which uses tempo modulation as hidden message carrier. With the use of Python scripting language, a bit string was encoded and decoded using WAV and MP3 files. Once the message was hidden into a music files, an internet radio was created to evaluate broadcast possibilities. No dedicated music or signal processing equipment was used in this StegIbiza implementation.

I. INTRODUCTION
Steganography was used by human kind through out the history in many ways. We find first known examples in ancient Greece [1] and, as technology and culture progressed, new methods of steganography were developed. In the age of Internet major attention is paid to applications of steganography in image [2] and network [3] as it is currently best platform for covert communication, which pushes the construction of new image and network steganography methods. Recent years have shown usage growth of internet radio and music streaming services [4], which provides plenty of room for new audio steganography methods.

In this paper, an implementation of new music steganography method called StegIbiza (Steganographic Ibiza) [5] is introduced and used to share hidden information over an internet radio and other internet services. Implementation was done in Python programming language and was later used on pack of MP3 music files which were added to the internet radio station. The whole process was done using regular PC with out any special sound or signal processing equipment. The idea behind StegIbiza is to encode a message by increasing or decreasing the music tempo by small percent to keep it inaudible to humans. Music tempo is measured in beats per minute (bpm) and needs to be constant for the StegIbiza to work.

The paper is structured as follows: Section II briefly describes the art of steganography. Section III focuses on the idea of StegIbiza method and how it can be used. Section IV presents the limits which were discovered during the implementation process. Section V explains and presents the Python implementation of StegIbiza. Section VI shows usage of the implementation on different songs. Section VII presents possible broadcast sources like internet radio and other services. Section VIII concludes our work and describes our future efforts.
noticed the difference. With a change of 3% and above all testers have detected the StegIbiza.

IV. IMPLEMENTATION REVELATIONS

Before implementing StegIbiza in Python, a research was made in order to find best solution and tools for the task. Most important part of the StegIbiza implementation was accurate tempo measurement and error free message decoding. For this three libraries were tested:

- Aubio [9]: free and opensource library written in C to label music and sounds.
- SoundTouch [10]: opensource audio processing library for modifying tempo, pitch and playback rates of audio streams and files.
- madmom [11]: audio signal processing library written in Python.

In the test, a song was divided into parts where \( \Phi \in \{10, 15, 20\} \) seconds. Tempo of each part was measured and compared with the original one. Finally “madmom” was chosen due to providing best method of measuring tempo value in a song file. In comparison to other libraries, “madmom” had a constant error value, while measurements with “Aubio” and “SoundTouch” presented errors within \(+/−1\%\) range. In addition to “madmom” for tempo reading, “SoundTouch” library is used for tempo modulation and brought to Python using a wrapper “pysoundtouch”.

When it came to measurement accuracy, none of the libraries could correctly detect tempo change that was lower than 2.75%. At first this seemed like an big issue, because the goal was to read \(+/−1\%\) tempo changes, so that StegIbiza stays inaudible. However in the end there is no need for accurate change measurement, as long as the results allow us to determine if the tempo was increased or decreased. Measurement results from “madmom” are presented as an array of BPM values and beat strength. Example measurement results of two files is presented in Figure 2. First array on the left contains results of unchanged tempo measurement of first 10 seconds of the song. Array on the right presents tempo measurement results from 10 till 20 seconds of the same song with its tempo increased by 1%. When we compare the results we can notice small changes in some values and identify the increase.

The process of implementation has also shown that lowest possible value of \( \Phi = 10 \) seconds. It turned out that for decent tempo measurement result, one needs at least 9 seconds of a song, which would make constant \( \Phi = 9 \) seconds, however when we increase or decrease the tempo, we also change the length of the \( \Phi \) parts. When reading the encoded value with \( \Phi = 9 \) we might read a small part of the previous or next song part with different tempo. This will lead to an error and will most likely end in reading wrong value. To avoid that we cut 5% of the beginning and end of each song part while reading, hence \( \Phi = 10 = 9 + 10 \times 5\% + 10 \times 5\% \). The implementation is presented in Figure 3.

For \( \Phi = 10 \), with average song time of 3 minutes we can only hide 17 bits - 3 \times 60/10 − 1 (we subtract 1 bit as it is used for original tempo reference). By the cost of songs data capacity, we can increase the \( \Phi \) value to 15 or 20 seconds which will lower the probability of an error and StegIbiza detection.
StegIbiza starts

Input files

Read input parameters

Convert message into bit string

Create temporary directories

Open 1st music file

Slice music into pieces

Process song pieces

Output file

Files remain?

yes

Open next file

no

StegIbiza stops

Fig. 4. Flowchart of StegIbiza implementation.

V. STEGIIBIZA PYTHON IMPLEMENTATION

The implementation takes WAV or MP3 type music file and slices it into $\Phi = 10$ second parts by default. First part remains untouched as it is used as a reference for original tempo value. The tempo of the remaining parts is modified accordingly to bits of the hidden message. Time of the last part of the song file is always lower than $\Phi$ therefore it remains untouched as it is very likely to produce errors when decoding a message. After the encoding all parts are put back together into one file. For long messages it is possible to provide several music files for increased storage capacity. Figure 4 presents simplified flowchart of StegIbiza python implementation.

When decoding a message, the file is split into $\Phi - 10\% \Phi$ parts. Here we cut the beginning and end by 5% of $\Phi$ of each part in order to eliminate overlapping with neighboring parts. Once the file is divided, we start reading the message by comparing the tempo of each file with the first one and qualify it if has increased or decreased tempo. Same as before, time of the last part of the song file is always lower than $\Phi$ therefore it remains untouched.

In order to classify the measured sample tempo as $+1\%$ or $-1\%$ a simple classifier is used. In order to obtain attributes of the sample, each tempo value from the reference song slice is compared with each tempo value of the analyzed sample and saved as difference in %. Then all obtained attributes with mean value larger than 4% are discarded. Remaining features are summed and classified as $+1\%$ if the sum is larger than 0 or as $-1\%$ otherwise.

VI. IMPLEMENTATION RESULTS

The implementation was tested on 10 different songs obtained from "Free Music Archive" [12]. The test results are presented in Table I. The bit string used in test is composed of 21 bits, therefore requires $21 \times 10s = 3\text{min} & 30s$ for the message to be fully encoded. Some songs used in the test were shorter than the time required for the whole message to be encoded and the missing bits are represented as "x" in the table.

Songs numbered from 1 till 6 are categorized as electronic music and they have constant tempo. Form the results we see that songs from 1 to 5 are error free. Song number 6 presents interesting result as it is inverted from the original bit string. This was caused by 5 seconds silence in the beginning of the song, which led to bad tempo reference sample and caused reverted results.

Songs numbered 7 and 8 are classified as instrumental indie-rock. Those songs produce some errors when decoding hidden message due to small tempo changes in some parts.

Songs 9 and 10 do not have constant tempo and produce large number of errors. In this case there is no way to use StegIbiza in current form. The only possibility here would be to compare each part of the song with equivalent part from the original song version.

VII. STEGIIBIZA BROADCAST APPLICATIONS

The easiest approach to broadcast information using StegIbiza is to simply upload the music files to a public website as a playlist and allow the receiver to download them and decode the message. Any other random person to download the playlist or any song in it, would not recognize any tempo changes by simply listening to it.

Another simple way is to put the songs with hidden message on popular video service like YouTube [23]. Then the receiver can extract song file from the video and decode the hidden message.

Slightly harder solution is to setup an internet radio using SouthCast [24] or other tools. Then we can setup automatic broadcast of previously prepared playlist with a hidden message. Each song should be separated by a proper separator like few seconds of silence or some other sound mark that will indicate new song. Later to extract message from each song, the stream can be easily recorded and split into song files using the separators.

There is always a possibility of using StegIbiza in regular radio, however messages are most likely to be unreadable due to bad quality recordings caused by bad reception and interferences.

Besides hiding messages in music, idea of tempo changing from StegIbiza can be used to control variety of equipment in clubs or music festivals. Lighting, screens and many others devices can be wirelessly controlled by reacting to music tempo changes.
TABLE I
RESULTS OF MESSAGE ENCODING AND DECODING USING STEGIIBIZA IMPLEMENTATION IN PYTHON

| # | Original Bit String | Decoded Bit String | #Errors |
|---|---------------------|--------------------|---------|
| 0 | 11011011111111001111 | X                  |         |
| 1 | 3:30                | 10011111100111100011 | 0       |
| 2 | 3:14                | 11011011111001111001 | 0       |
| 3 | 5:57                | 11011011111001110011 | 0       |
| 4 | 2:51                | 11011011111001111111 | 0       |
| 5 | 2:23                | 11011011111001111111 | 0       |
| 6 | 6:21                | 01001000011001100010 | 21      |
| 7 | 2:20                | 11011011111011111111 | 3       |
| 8 | 2:23                | 11110111100111111111 | 3       |
| 9 | 2:40                | 11010100001110000000 | 9       |
| 10| 3:05                | 11011100010111111111 | 6       |

VIII. CONCLUSION

This article presented Stegiibiza implementation and application on different music tracks. It showed that Stegiibiza can be applied using regular personal computer without any special music or signal processing hardware. Implementation has also revealed its constrains in terms of tempo change limits and song capacity for data. The implementation was tested on 10 different songs and proven to work with 1% tempo change, with 1 bit saved in each 10 seconds of the song. Article has also presented simple ways to secretly share information hidden by Stegiibiza using internet services.

In this implementation there is a lot of space for improvements and the future work will focus on lowering the tempo change to lowest possible values and increasing songs data capacity by lowering the minimum time value needed to read the tempo.

ACKNOWLEDGMENT

The authors would like to thank all the people that helped in Stegiibiza research or developed tools that made this implementation possible.

This is an independent publication and has not been authorized, sponsored, or otherwise approved by Google, SouthCast or Free Music Archive.

REFERENCES

[1] Petitcolas, FAP; Anderson RJ; Kuhn MG (1999). “Information Hiding: A survey” (pdf). Proceedings of the IEEE (special issue). 87 (7): 106278. doi:10.1109/5.771065. Retrieved 11th of May 2017.
[2] J. Fridrich, Steganography in Digital Media: Principles, Algorithms, and Applications, Cambridge University Press; 1 edition, December 2009
[3] W. Mazurczyk, S. Wendzel, S. Zander, A. Hounamsadri, K. Szczypiorski, Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures, Wiley-IEEE Press; 1 edition, February 2016
[4] P. Sikka, “Why we’re seeing increased Internet Radio usage”, 7th July 2014, (last visited: 11th of May 2017)
[5] K. Szczypiorski, “StegIibiza: New Method for Information Hiding in Club Music”, 2012 2nd International Conference on Frontiers of Signal Processing, 15-17 Oct. 2016
[6] H. L. Perera, "History of Steganography", http://harenlaks.blogspot.it/2011/04/history-of-steganography.html, 2nd April 2011, (last visited: 11th of May 2017)
[7] N. Provos, P. Honeyman, "Detecting Steganographic Content on the Internet", ISOC NDSS’02, San Diego, CA, February 2002.
[8] H. Kopka and P. W. Daly, A Guide to LATEX, 3rd ed. Harlow, England: Addison-Wesley, 1999.
[9] Aubio library, http://aubio.org (last visited: 11th of May 2017)
[10] SoundTouch Audio Processing Library, http://www.surina.net/soundtouch/index.html (last visited: 11th of May 2017)
[11] S. Bock, F. Korzeniowski, J. Schluter, F. Krebs, G. Widmer, ’madmom: a new Python Audio and Music Signal Processing Library’, ¡Co tu jeszce trzeba wpisa¿, 23 May 2016
[12] Interactive library of legal audio downloads founded by legendary freeform radio station WFMU, http://freemusicarchive.org (last visited: 17th of May 2017)
[13] Broke For Free, "Night Owl", http://freemusicarchive.org/music/BrokeFor Free/Directionless EP/Broke For Free - Directionless EP - _01_Night_Owl Released 18th January 2011, (last visited: 17th of May 2017)
[14] Candlegravity, "Tomie’s Bubbles", http://freemusicarchive.org/music/Candlegravity/2007_Tomies_Bubbles, Released 11th April 2013, (last visited: 17th of May 2017)
[15] LASERS, "Amsterdam", http://freemusicarchive.org/music/LASERS/ LASERS_EP/01_LASERS_-_Amsterdam Released 4th June 2012, (last visited: 17th of May 2017)
[16] Lobo Loco, "Muscle Body Man (ID 534)", http://freemusicarchive.org/music/Lobo_Loco/Phenomenon/Muscle_Body Man_ID_534 Uploaded 28th April 2017, (last visited: 17th of May 2017)
[17] Podington Bear, "Pick Up The Tempo", http://freemusicarchive.org/music/Podington_Bear/PickUpTheTempo, Uploaded 3rd April 2017, (last visited: 17th of May 2017)
[18] The Kyoto Connection, "Hachiko (The Faithful Dog)", http://freemusicarchive.org/music/The Kyoto Connection/Hachiko (The Faithful Dog) Released 27th May 2013, (last visited: 17th of May 2017)
[19] Tours, "Enthusiast", http://freemusicarchive.org/music/Tours/Enthusiast, Tours_-_Enthusiast Released 30th August 2011, (last visited: 17th of May 2017)
[20] Black Ant, "Fater Lee", http://freemusicarchive.org/music/Black_Ant/Free_Beats_Sel_3/Fater_Lee Released 25th January 2010, (last visited: 17th of May 2017)
[21] Free video hosting and streaming service, https://www.youtube.com, (last visited: 20th of May 2017)
[22] Free internet radio broadcasting tools, https://www.shoutcast.com, (last visited: 11th of May 2017)