BCI3005: Digital Watermarking and Steganography

Project Report

Audio Steganography,

and Spectrogram Analysis

Submitted By

Rohan Dahiya (16BCI0031)

Akash Atkare (16BCI0162)

Under the Guidance of

Dr. Thenmozhi T

Declaration

I the undersigned solemnly declare that the project report AUDIO STEGANOGRAPHY, AND SPECTROGRAM ANALYSIS is based on my own work carried out during the course of our study under the supervision of Dr. Dr. Thenmozhi T.

I assert the statements made and conclusions drawn are an outcome of my research work. I further certify that:

1. The work contained in the report is original and has been done by me under the general supervision of my supervisor.
2. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.
3. We have followed the guidelines provided by the university in writing the report.
4. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references

Rohan Dahiya 16BCI0031

Akash Atkare 16BCI0162

Table of Contents

Abstract

1. Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video.
2. The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny.
3. Digital audio is sound that has been recorded in, or converted into, digital form. In digital audio, the sound wave of the audio signal is encoded as numerical samples in continuous sequence.
4. Waveform Audio File Format (WAVE, or WAV due to its filename extension) is an audio file format standard, developed by Microsoft and IBM, for storing an audio bitstream on PCs. It is an application of the Resource Interchange File Format (RIFF) bitstream format method for storing data in "chunks".
5. A spectrogram is a visual representation of the spectrum of frequencies of a signal as it varies with time. When applied to an audio signal, spectrograms are sometimes called sonographs, voiceprints, or voicegrams.
6. In this project we implement Audio Steganography on WAV files and analyse the results by plotting spectograms.

Keywords

Steganography; Encryption; Data Hiding; Audio; Spectogram; Analysis;

Introduction

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video. The word steganography combines the Greek words steganos, meaning "covered or concealed", and graphe meaning "writing".

Generally, the hidden messages appear to be (or to be part of) something else: images, articles, shopping lists, or some other cover text. For example, the hidden message may be in invisible ink between the visible lines of a private letter. Some implementations of steganography that lack a shared secret are forms of security through obscurity, and key-dependent steganographic schemes adhere to Kerckhoffs's principle.

The advantage of steganography over cryptography alone is that the intended secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages, no matter how unbreakable they are, arouse interest and may in themselves be incriminating in countries in which encryption is illegal.

Project Objectives

(Problem Statement)

In this project we aim to implement audio steganography using LSB method. Further we analyse the result of this method and try to find any tell tails of tampering. As the main purpose of stegnography is not just to prevent the detection of the a hidden message but to also prevent the detection of the presence of it, thus the analysis should not give away the presence of a hidden message.

Litrature Review

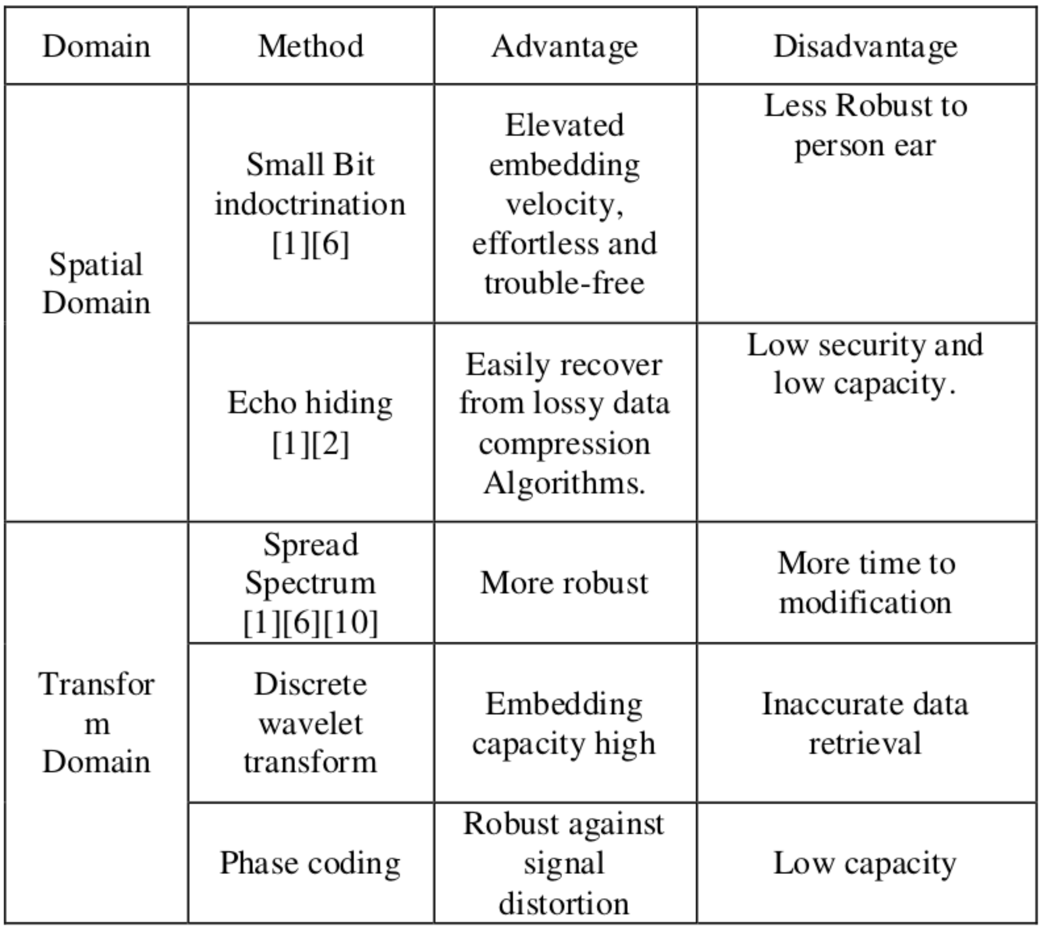
A comparison of various Audio Steganography techniques was made in the paper by Kiran et.al.[K01] These were classified into Spatial domain and Transform Domain. Those in the Spatial domain covered were as follows:

Low Bit Encoding: It is also called least significant bit (LSB). LSB substitution algorithm is an uncomplicated type of substitution algorithm. In which LSBs of the cover image is changed according to the secret message. LSB is the most primitive technique used for data hiding. Low bit encoding is embeds the secret information into the least significant bit of the audio file. This technique is easy for information hiding and alteration in LSB of audio file have to be conceded out in such a method that the eminence of the auditory file is not compromised.

Echo Hiding: This manner purpose a little resonance to the swarm indication and after that embeds information in it. After modification of the echo in the carrier file, the stego gesture is obliged to hold the similar geometric characteristics. For hiding data there are three parameters manipulated: initial amplitudes, the offsets (impediment) and the decompose rate (used for the out of earshot echo). The outcome is impossible to differentiate for delay up to 1ms between original signaland echo. Information could be hidden invisibly as amplitude and decompose rates can be set to values which are below capable of being heard entrance of human ear. The disadvantages of echo hiding method are less embedding speed and safety.

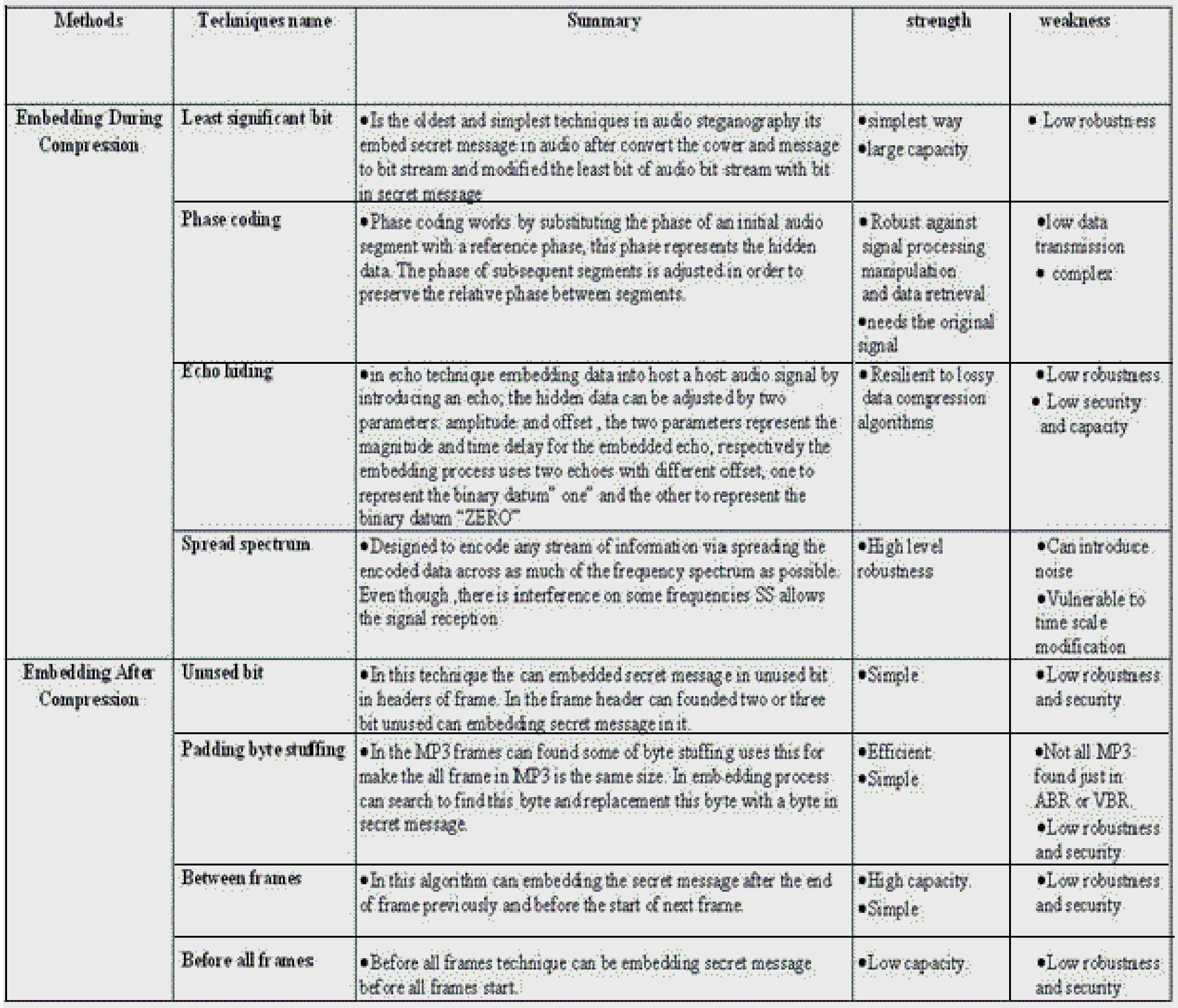
The comarison of these techniques has been summarised in the following table taken from the paper, as is:

TABLE 1 COMPARISON VARIOUS AUDIO STEGANOGRAPHY TECHNIQUES



Domain	Method	Advantage	Disadvantage
Spatial Domain	Small Bit indoctrination [1][6]	Elevated embedding velocity, effortless and trouble-free	Less Robust to person ear
	Echo hiding [1][2]	Easily recover from lossy data compression Algorithms.	Low security and low capacity.
Transfor m Domain	Spread Spectrum [1][6][10]	More robust	More time to modification
	Discrete wavelet transform	Embedding capacity high	Inaccurate data retrieval
	Phase coding	Robust against signal distortion	Low capacity

A similiar comparison was made in the paper by Mohammed et. al. [M01] , the tabulation of their results is as follows:



Methods	Techniques name	Summary	strength	weakness
Embedding During Compression	Least significant bit	•Is the oldest and simplest techniques in audio steganography its embed secret message in audio after convert the cover and message to bit stream and modified the least bit of audio bit -stream with bit in secret message	•simplest way •large capacity	• Low robustness
	Phase coding	•Phase coding works by substituting the phase of an initial audio segment with a reference phase, this phase represents the hidden data. The phase of subsequent segments is adjusted in order to preserve the relative phase between segments.	• Robust against signal processing manipulation and data retrieval •needs the original signal	•low data transmission • complex
	Echo hiding	•in echo technique embedding data into host a host audio signal by introducing an echo; the hidden data can be adjusted by two parameters: amplitude and offset , the two parameters represent the magnitude and time delay for the embedded echo, respectively the embedding process uses two echoes with different offset, one to represent the binary datum" one" and the other to represent the binary datum "ZERO"	• Resilient to lossy data compression algorithms	•Low robustness • Low security and capacity
	Spread spectrum	•Designed to encode any stream of information via spreading the encoded data across as much of the frequency spectrum as possible. Even though ,there is interference on some frequencies SS allows the signal reception	•High level robustness	•Can introduce noise •Vulnerable to time scale modification
Embedding After Compression	Unused bit	•In this technique the can embedded secret message in unused bit in headers of frame. In the frame header can founded two or three bit unused can embedding secret message in it.	•Simple	•Low robustness and security
	Padding byte stuffing	•In the MP3 frames can found some of byte stuffing uses this for make the all frame in MP3 is the same size. In emb edding process can search to find this byte and replacement this byte with a byte in secret message.	•Efficient •Simple	•Not all MP3 found just in ABR or VBR. •Low robustness and security
	Between frames	•In this algorithm can embedding the secret message after the end of frame previously and before the start of next frame.	•High capacity. •Simple	•Low robustness and security
	Before all frames	•Before all frames technique can be embedding secret message before all frames start.	•Low capacity	•Low robustness and security

Various free steganography tools are available online, we have tested some of them, a comparison of a few of them, that support audio files as a cover, is as follows.

Program	Audio files	Other support	Notes
Anubis	?[*clarification needed*]	*Data being appended to the end of file*	*Open Source*[*clarification needed*]
DarkCryptTC	WAV	*EXE*, *DLL*, *NTFS* *streams*	*RSD mode (RNG-based random data distribution), AES encryption supported*
DeepSound	Audio CD, APE tag, FLAC, MP3, WAV, WMA	-	*AES 256-bit encryption*
MP3Stego	MP3	-	*Open source*
OpenPuff	MP3, WAV	-	*Open source, 256-bit multi-encryption, Carrier chains, Multi-*

			layered obfuscation
S-Tools	WAV	*Unused floppy disk space*	-
Steghide	WAV, AU	-	*Open source (GNU General Public License)*

Proposed Methodology

We implimented a LSB based steganography method to hide datain a WAV file. Then we generate a spectogram of the same. We then analyse the result of this method and try to find any tell tails of tampering. As the main purpose of stegnography is not just to prevent the detection of the a hidden message but to also prevent the detection of the presence of it, thus the spectrogram should not give away the presence of a hidden message.

Further we aply the same analysis to some common tools used for stegnography.

Results

1. Upon generating the spectogram of the cover file after adding in the secret using the LSB method,the presence of secret was easily given away.
2. There was a sudden stop in the presence of an underlying noise componenet indicationg that the noise was artificially injected and thus wasn’t actully noise. Refer to the second spectrogram from the top on the next page.
3. Further, the secret itself can be just easily compromised just as its presence was detected.
4. Using Xiao Steganography, we see that there are various encryptionalgorithms available as well as hashing, however the presence of a secret is clearly seen in thespectrogram thus the tool has failed its task.
5. Using steghide however, the presence was not detected. Thus this tool successfully hid the data.

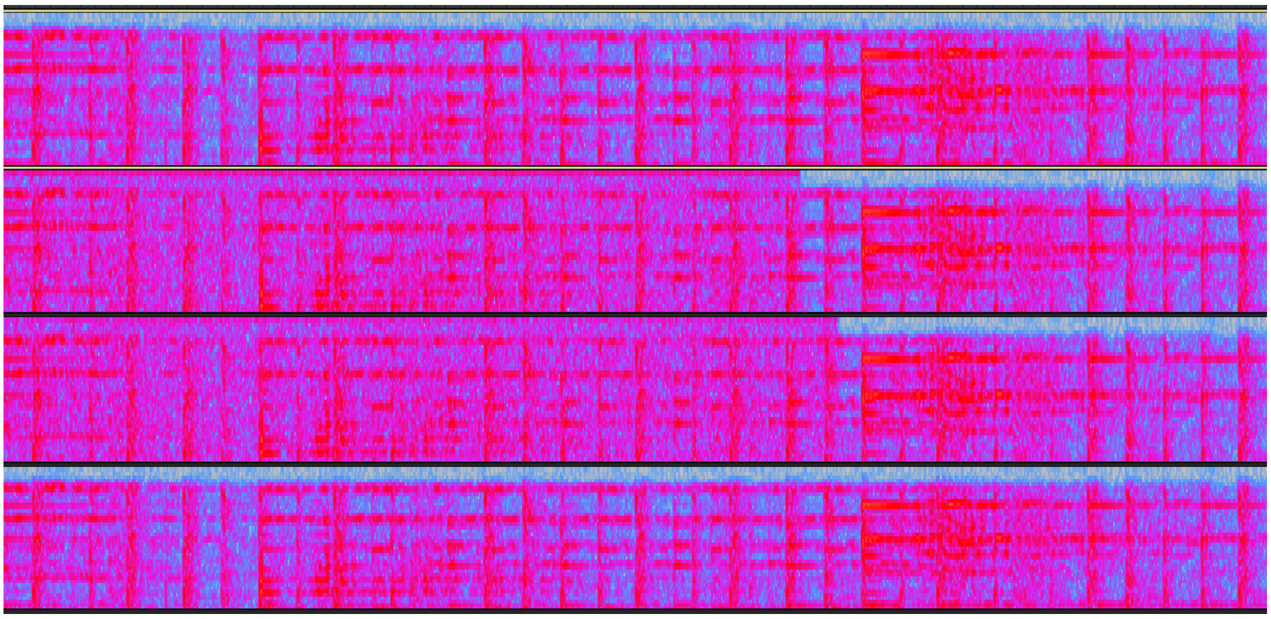


Illustration 1: A portion of the spectrograms, from top to bottom: 1) plain cover, 2)using LSB method, 3)Using Xiao Steganography, 4) Using Steghide

Note that the above spectrograms are not the ones generated from the code, some of those can be found in the appendix, these were genaratedusing a specialised software for clarity on paper.

<u>Conclusion</u>

1. Simple LSB method can be easily detected and decoded. It is thus not fit for steganography.
2. Xiao Steganography may encrypt the data however it also failed our simple test thus it is not viable.
3. Steghide Passed thetest as it uses a graph-theoretic approach to steganography.

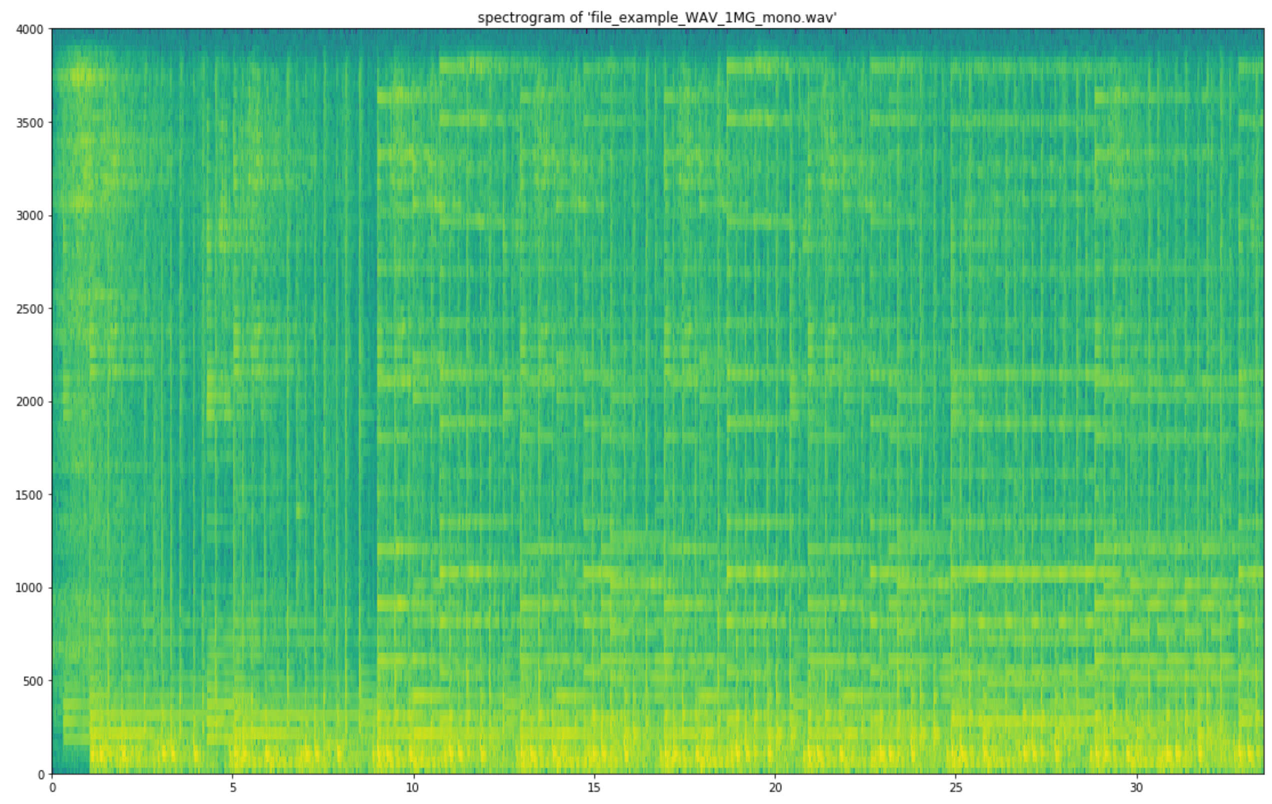
   "At first, the secret data is compressed and encrypted. Then a sequence of postions of pixels in the cover file is created based on a pseudo-random number generator initialized with the passphrase (the secret data will be embedded in the pixels at these positions). Of these positions those that do not need to be changed (because they already contain the correct value by chance) are sorted out. Then a graph-theoretic matching algorithm finds pairs of positions such that exchanging their values has the effect of embedding the corresponding part of the secret data. If the algorithm cannot find any more such

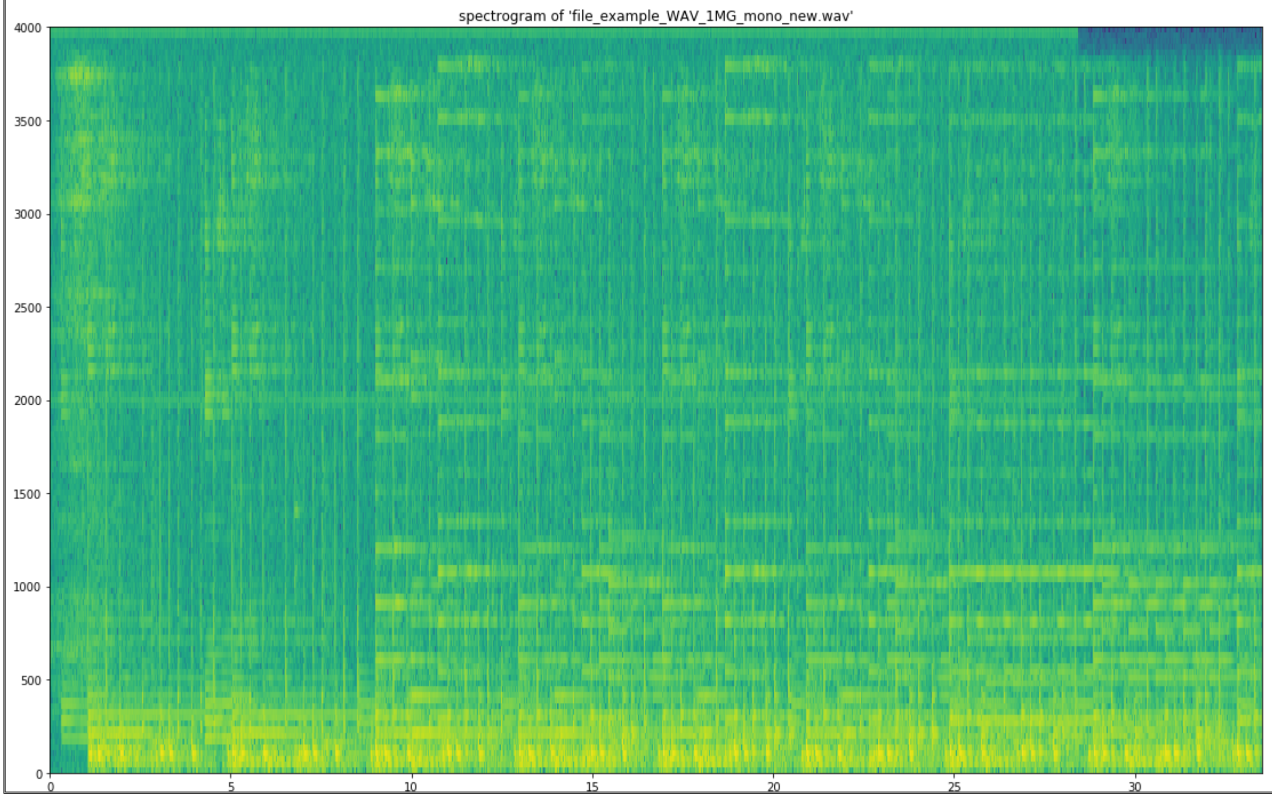
pairs all exchanges are actually performed. The pixels at the remaining positions (the positions that are not part of such a pair) are also modified to contain the embedded data (but this is done by overwriting them, not by exchanging them with other pixels). The fact that (most of) the embedding is done by exchanging pixel values implies that the first-order statistics (i.e. the number of times a color occurs in the picture) is not changed. For audio files the algorithm is the same, except that audio samples are used instead of pixels."

Source Code

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import os
import sys
import wave
import struct
print("imported libs")


# In[10]:


fileName = 'file_example_WAV_1MG_mono.wav'
outfilename = 'file_example_WAV_1MG_mono_new.wav'
secret=open("secret.txt", "r").read()#"hello world"*2002*2
print("set global constant variables")


# In[11]:


def tobits(x):
    a=[0,0,0,0,0,0,0,0]
    for i in range(0,8):
        a[i]= 1 if x & (1 << i) !=0 else 0
    return a
def toint(bits):
    a=0
    for i in bits:
        a*=2
        a+=i
    return a

def addMsg(data, ordMsg):
    curr=44
    for byte in ordMsg:
        byte=tobits(byte)
        for bitno in range(0,8):
            try:
                data[curr]=int((data[curr] & ~1) | byte[bitno])
            except:
                pass
            curr+=1
    return data


def getMsg(data):
    ordMsg=[]
    temp=[]
    for i in data[44:]:
        temp.append(int((i & 1) ))
        if len(temp)==8:
            ordMsg.append(toint(temp[::-1]))
            temp=[]
    return ordMsg


# In[12]:



print("Adding secret to file")
inFile = wave.open(fileName, 'rb')
inData = inFile.readframes(inFile.getnframes())
inData = [int(byte) for byte in inData]
ordMsg = [ord(x) for x in secret]

outData=addMsg(inData, ordMsg)

outData= bytes(outData)

outFile = wave.open(outfilename, 'wb')
outFile.setparams(inFile.getparams())

for byte in outData:
    pack = struct.pack ('h',byte)
    outFile.writeframes (pack [:1])
outFile.close()
print("done")
```

```
# In[13]:


print("reading secret from file")
file = wave.open(outfilename, 'rb')
Data = file.readframes(file.getnframes())
msg=getMsg(Data)
msg=''.join([chr(i) for i in msg])
msg


# In[15]:


from scipy.io import wavfile
import matplotlib.pyplot as plot

samplingFrequency, signalData = wavfile.read(fileName)
samplingFrequency0, signalData0 = wavfile.read(outfilename)

plot.subplot(111)
plot.specgram(signalData,Fs=samplingFrequency)
plot.xlabel('Time')
plot.ylabel('Frequency')
plot.show()

plot.subplot(111)
plot.specgram(signalData0,Fs=samplingFrequency0)
plot.xlabel('Time')
plot.ylabel('Frequency')

plot.show()


# In[17]:


import os
import wave

import pylab
def graph_spectrogram(wav_file):
    sound_info, frame_rate = get_wav_info(wav_file)
    pylab.figure(num=None, figsize=(19, 12))
    pylab.subplot(111)
    pylab.title('spectrogram of %r' % wav_file)
    pylab.specgram(sound_info, Fs=frame_rate)
    pylab.savefig('spectrogram.png')
def get_wav_info(wav_file):
    wav = wave.open(wav_file, 'r')
    frames = wav.readframes(-1)
    sound_info = pylab.fromstring(frames, 'int16')
    frame_rate = wav.getframerate()
    wav.close()
    return sound_info, frame_rate

graph_spectrogram(fileName)
graph_spectrogram(outfilename)
```





Bibliography

K01: Kiran, Monica Sharma, Satwinder Singh, Recent Advancement in Audio Steganography, 2017

M01: Mohammed Salem Atoum 1 , Subariah Ibrahim 2 , Ghazali Sulong 3 and Ali M-Ahmad 4, MP3 Steganography: Review, 2012