Security & Forensic Lab-2 Report

Friday, 19.02.2021

P Rahul Bharadwaj, 17CS02003

Problem Statement

To write a program that hides the secret message in a cover audio (wav file) in such a way that it minimizes the quality of the original voice does not get distorted after the message embedding is done.

Approach:

LSB Algorithm:

The idea behind LSB embedding is that if we change the last bit value of an audio sample, there won't be much change in quality of the audio.This approach has the advantage that it is the simplest one to understand, easy to implement and results in stego-audio file that contains the embedded data as hidden.

The disadvantage of Least Significant Bit is that it is vulnerable to steganalysis and is not secure at all. So as to make it more secure, the least significant bit algorithm is modified to work in a different way. This proposed approach follows hiding any form of digital data into uncompressed digital audio files in a random manner. It uses two intermediates to convey the secret data. An uncompressed audio file acting as a carrier file holding the secret data inside the LSBs of its audio samples and an attachment.txt file which will be used while recovering a secret message.

Software Requirements

- **Python**
- **Wave library**
- **Numpy**
- **Simpleaudio (**Installation : pip install simpleaudio**)**

Encryption Algorithm

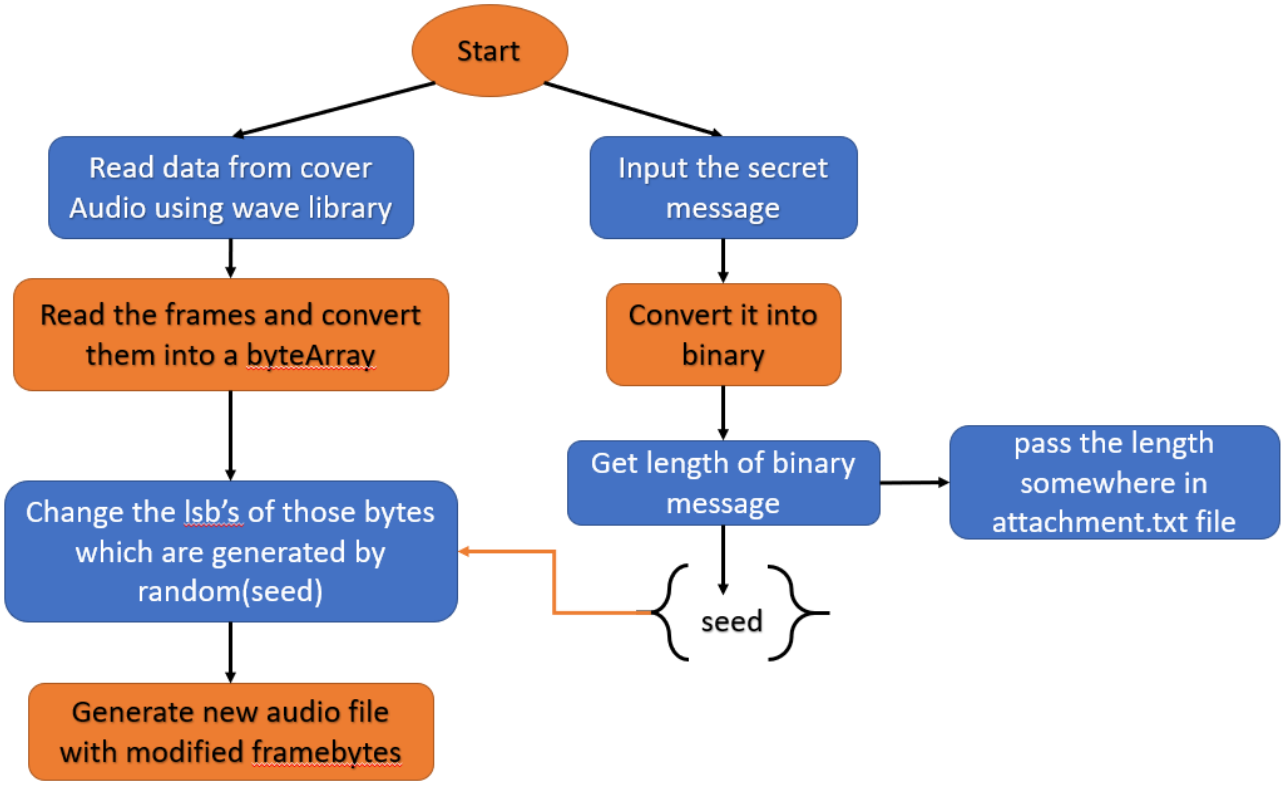

Fig.1 Encryption Algo for hiding the message

To hide secret text messages we have to read data from a cover audio file and convert them into a byte array. Get the secret message to be hidden and convert it into binary.Now fetch the length of the binary message and use it as a seed to generate a random byte sequence. Now replace the lsb's of those bytes which are generated randomly using a seed with a bit of secret message. Once all the message bits are hidden then write the modified frame bits into a new audio file named embedded.wav file.The final output comprises two intermediates. The first one is the carrier audio file which houses the secret data into its randomly selected samples; while, the second one is an attachment.txt file which consists information about seed.Both, the carrier audio file and the text file are to be sent to the receiver, not necessarily at the same time, however, they should be both present when the receiver is uncovering the secret data.

Decryption Algorithm

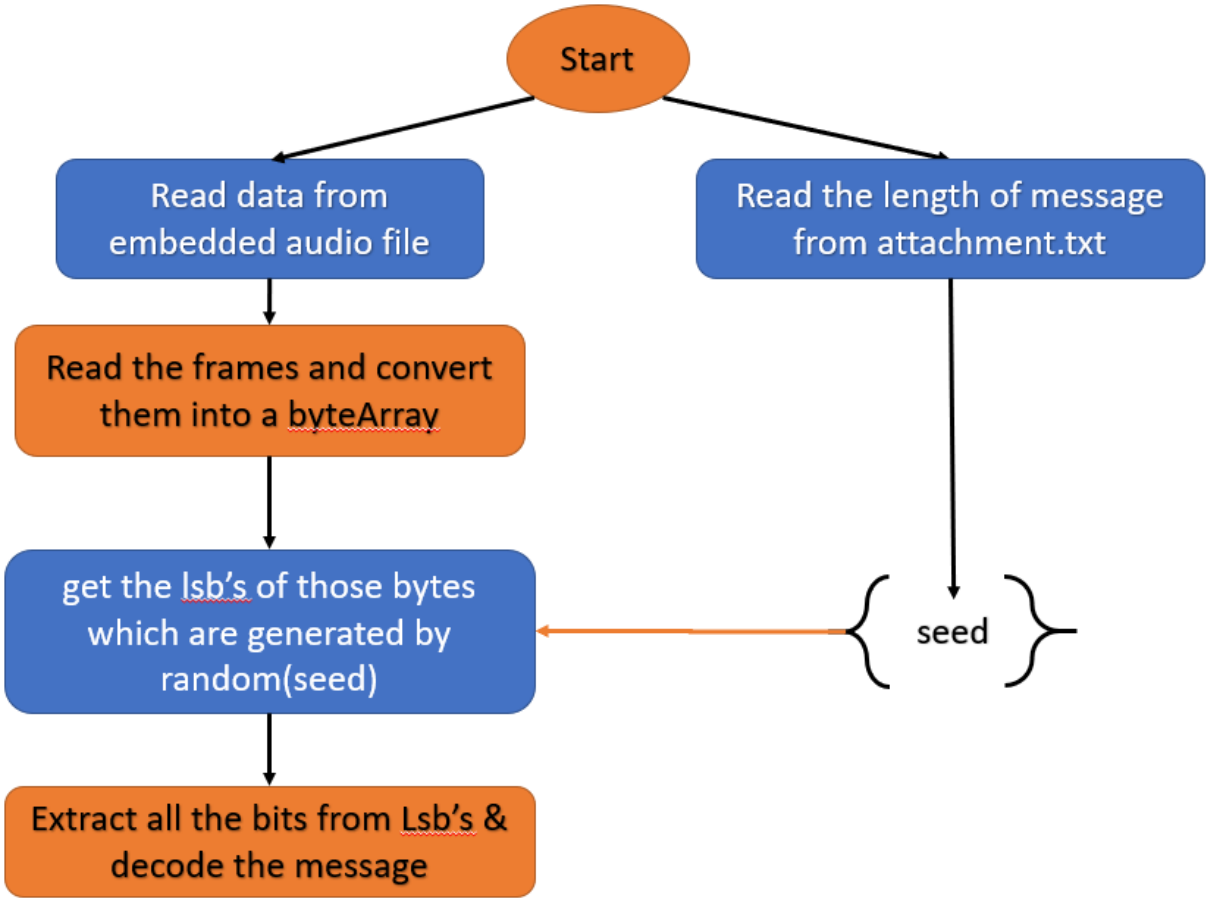

Fig.2 Decryption technique to retrieve hidden message

To recover the secret text message from the stego audio file, we have to read the data from the embedded audio file which has the hidden message.Also get the length of the message from attachment.txt file.This will be used as seed for generating byte sequence. Convert the data into a byte array and extract the lsb's from those bytes which are generated by random(seed).After fetching all the bits we split the array bits into a sequence of 8bits and then convert them into characters by use of ASCII values to recover the hidden message.

Code for embedding & decoding

```
def encode(message):
    audio = wave.open("coverAudio.wav", mode='rb')
    data = bytearray(list(audio.readframes(audio.getnframes())))
    bits=msgToBinary(message)
    length=len(bits)
    print("Message length : ",length)
    writeAttachment(length)
    np.random.seed(len(bits))
    perm = np.random.permutation(len(bits))
    j=0
    for bit in bits:
        data[perm[j]] = (int(data[perm[j]]) & 254) | int(bit)
        j+=1

    dataModified = bytes(data)

    with wave.open('embedded.wav', 'wb') as fd:
        fd.setparams(audio.getparams())
        fd.writeframes(dataModified)
    audio.close()

    print("\nMessage is embedded into embedded.wav file")
```

Fig.3 Encoding Algorithm

```
def decode():
    f = open("attachment.txt", "r")
    lines = f.read().splitlines()
    last_line = lines[-1]
    length = last_line.split("_")[-1]
    #print(length)
    seed =int(length)
    np.random.seed(seed)
    perm = np.random.permutation(seed)
    audioNew = wave.open("embedded.wav", mode='rb')
    newData = bytearray(list(audioNew.readframes(audioNew.getnframes())))
    extractedData = [newData[i] & 1 for i in perm]
    decodedMessage = "".join(chr(int("".join(map(str,extractedData[i:i+8])),2))
                             for i in range(0,len(extractedData),8))
    print("Hidden Message :",decodedMessage)
```

Fig.4 Decoding Algorithm

- You can find the complete code in 17CS02003_Assignment_2.py file attached.

Encoding output

```
PS E:\sem8\sfLab2\lab2> & c:/users/rahulbharadwaj/appdata/local/programs/python/python37/python.exe e:/sem8/sfLab2/lab2/17CS02003_Assignment_2.py

Choose appropriate option:
1.Encode
2.Decode
3.Play Embedded Audio
4.exit

Choice:1

Enter the secret message :  can you test the algorithm?  can you test the algorithm?  can you test the algorithm?  can you test the algorithm?  c
 can you test the algorithm?  can you test the algorithm?  can you test the algorithm?  can you test the algorithm?  can you test the algorithm?
Message length :  2776
```

Decoding output

```
Choose appropriate option:
1.Encode
2.Decode
3.Play Embedded Audio
4.exit

Choice:2
Hidden Message :  can you test the algorithm?  can you test the algorithm?  can you test the algorithm?  can you test the algorithm?  can you test the algorithm?
est the algorithm?  can you test the algorithm?  can you test the algorithm?  can you test the algorithm?  can you test the algorithm?  can you test the algorithm?
```

Conclusion :

In this lab session ,I have learnt how to hide a text message in an audio file and also to recover it using some steganographic technique. I have used an algorithm that intelligently selects the portions of the audio samples to hide the secret message such that the original voice does not get distorted.