SF LAB II

Experiment 2

Submitted by

Aman Borkar

22CS06014

MTECH (CSE)

Under the supervision of

Dr. MANORANJAN SATPATHY

School of Electrical Science

INDIAN INSTITUTE OF TECHNOLOGY, BHUBANESWAR 2022

Introduction

Audio steganography is the process of hiding data within an audio file. The hidden data can be in the form of text, images, or even other audio files. The goal of audio steganography is to conceal the existence of the hidden data so that it is not detected by anyone other than the intended recipient.

There are various techniques used in audio steganography, such as Least Significant Bit (LSB) insertion, frequency domain techniques, and echo hiding. In LSB insertion, the least significant bit of each audio sample is replaced with a bit from the hidden data. Frequency domain techniques involve manipulating the frequency components of the audio signal, while echo hiding involves adding a second, slightly delayed version of the audio signal to the original.

It is important to note that audio steganography is not a secure method of hiding data, as the hidden data can be easily extracted by someone who knows what to look for. Additionally, the hidden data can be corrupted during the process of hiding and retrieving the data.

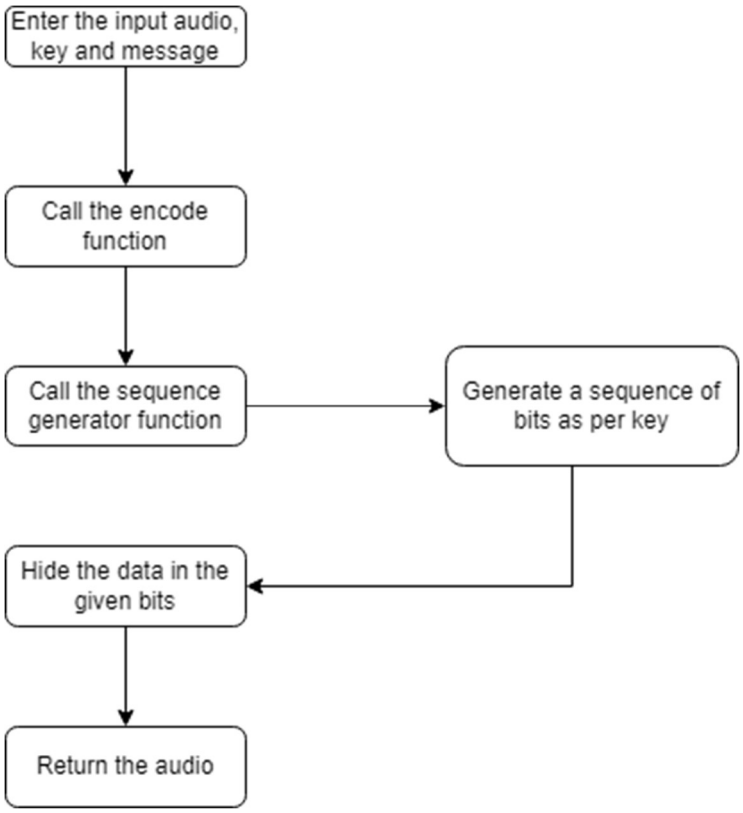
Algorithm:

This code uses the wave package in python to perform audio steganography, which is the process of hiding data within an audio file. The code is written in such a way that it takes an audio file path, a key, and a message as input and it uses the given key to hide the message in the audio file.

Here is a step-by-step breakdown of the algorithm used in the code:

1. The script imports the wave and math packages.
2. A function called encode() is defined, which takes four arguments: a byte array of the audio file, a key, the message to be hidden, and the output audio file path. This function first converts the message to a bit array and then uses the key to generate the sequence of positions where the message will be hidden in the audio file. The function then iterates over the bit array and hides each bit in the LSB of the corresponding byte in the audio file.
3. The function run_encode() is defined, which takes four arguments: an audio file path, a key, the message, and an output audio file path. This function first opens the audio file in read mode, reads the frames of the audio file, and converts them to a byte array. It then calls the encode() function and passes the byte array, key, message, and output audio file path as arguments. The modified byte array is then written to the output audio file.
4. A function called run_decode() is defined, which takes two arguments: an audio file path and a key. This function opens the audio file in read mode, reads the frames of the audio file, and converts them to a byte array. It uses the key to generate the sequence of positions where the message is hidden in the audio file. The function then iterates over the sequence of positions and extracts the LSB of each byte, which is the hidden message.
5. The script prompts the user to enter the audio file path, key, and message. It then calls the run_encode() function and passes the user-provided inputs as arguments. The script then prints the status of the encoding process and the decoded message.

Flowchart



Output

The code can be accessed at:

https://colab.research.google.com/drive/1LfyH6YCVlP5XFNmhwvsBBrSD80gO5Zup?usp=sharing