STEGOWAV

Steganography

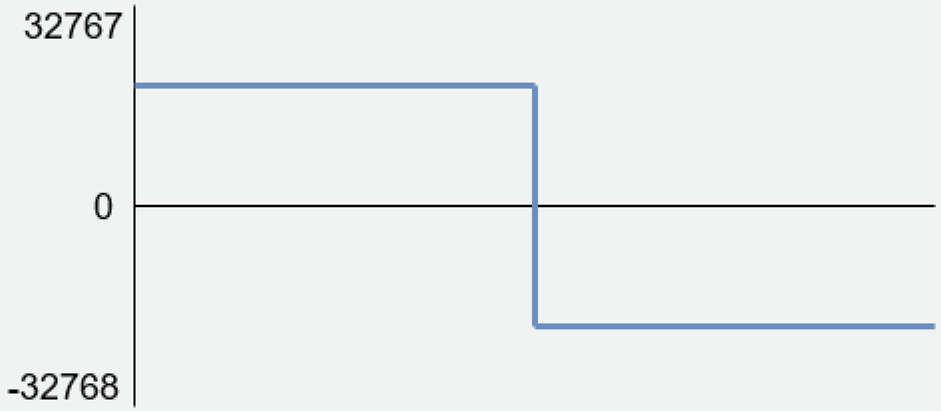
Project of Group 3

Introduction

- Aim: Develop prototype of a steganographic WAV encoder
- Hide data in WAV files
- Make use of Python libraries

Basics – WAV file

- How is data stored in a WAV file?
- Example: Square wave
- `16383` → `0b11111111111111` → encode `01` → `0b11111111111101`



→

[
16383, 16386, 16382, 16385, 16384,
16384, 16383, 16383, 16385, 16385,
-16383, -16384, -16382, -16385, -16383,
-16384, -16383, -16385, -16385, -16384
]

= amplitude

Parser, Flags

- Implemented:
  - WAV Parser
  - Implemented information encoding/decoding in WAV file using LSBs
  - Created unit tests for parsing, encoding, decoding
  - Flags for a command line program to encode/decode messages:

```
$ ./stegowav.py input.wav -e "my secret" -o encoded.wav
```

```
$ ./stegowav.py encoded.wav -d
my secret
```

Encryption

- Motivation: providing an additional security layer
- Data Encryption and Decryption
- Use of a specialized library
- Integration into the existing Encoding and Decoding

Hamming distance

- Motivation: Preserve encoded data and prevent data loss from corruption
- Generate hamming code from WAV file bytes
    - Original file, Converted file
    - Compare, find and correct errors

Test audio files

- Create a test dataset
- Collect different WAV files
- Encode text of different length into the files

- Goals:
    - Test which files are best for hiding text
    - Test how much text you can hide

Testsetup A/B Test

- Scripting audio degradation test
  - Compare unmodified with (un-)modified samples
  - User input to indentify audio degradation
  - Generate Csv test report
- Analyze confusion over samples

STEGOWAV

Thank you for your attention!