Blowfish Enhanced Randomized Audio Steganography

AMITH G (B191002EC) | R MALAVIKA (B190855EC)

CRYPTOGRAPHY

1

Theoretical Background

A brief explanation about audio steganography and blowfish algorithm

Audio Steganography

- Audio steganography is the art of discreetly embedding and transmitting secret messages into digital audio signals by subtly altering the signal.
- There are various techniques to embed the message onto the audio like substitution techniques, echo hiding, spread spectrum, and phase coding.

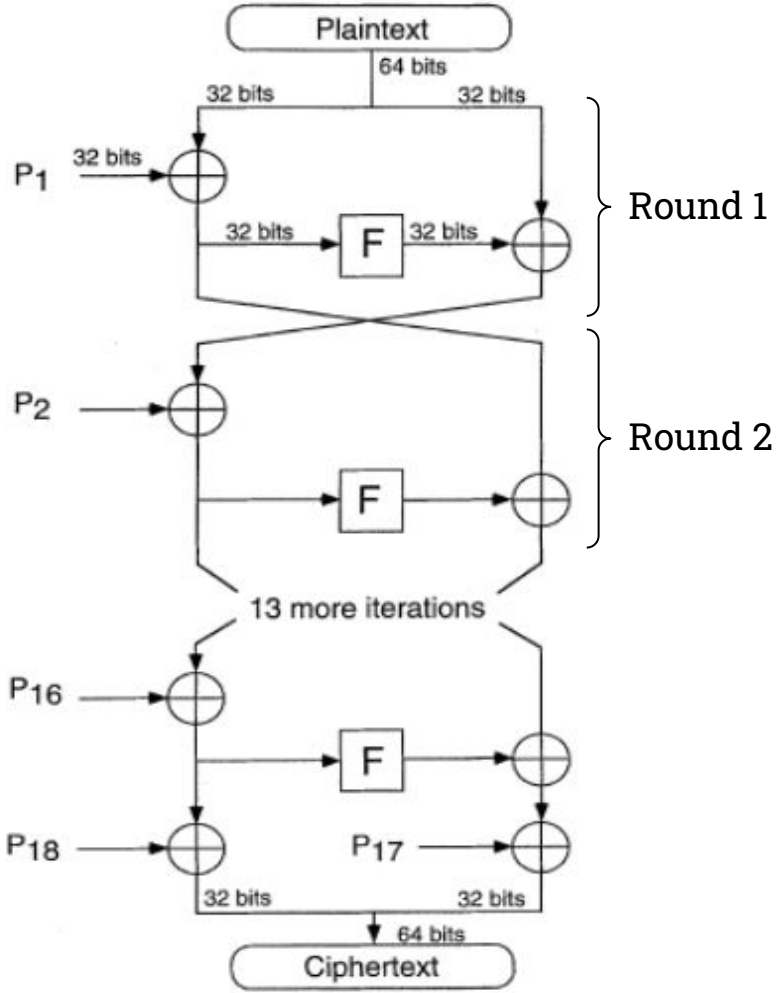
Key Characteristics of a steganographic algorithm

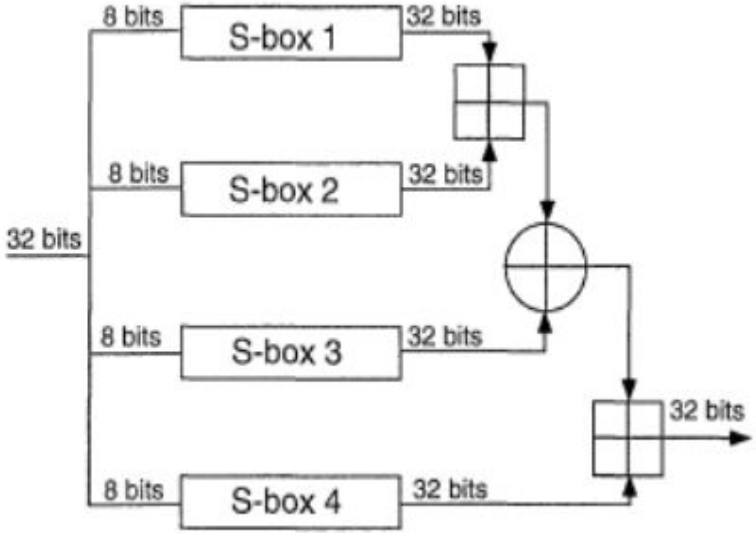
- **Robustness** : measures the ability of secret message to withstand against attacks
- **Capacity** : means the amount of secret information that can be embedded within the host message
- **Transparency** : evaluates the audible distortion due to signal modifications like message embedding

Blowfish Algorithm

Significantly faster than DES and provides a good encryption rate with no effective cryptanalysis technique found to date

Block Size	: 64-bits
Key Size	: 32-bits to 448-bits (variable size)
Number of subkeys	: 18 [P-array]
Number of rounds	: 16
Number of S-Boxes	: 4

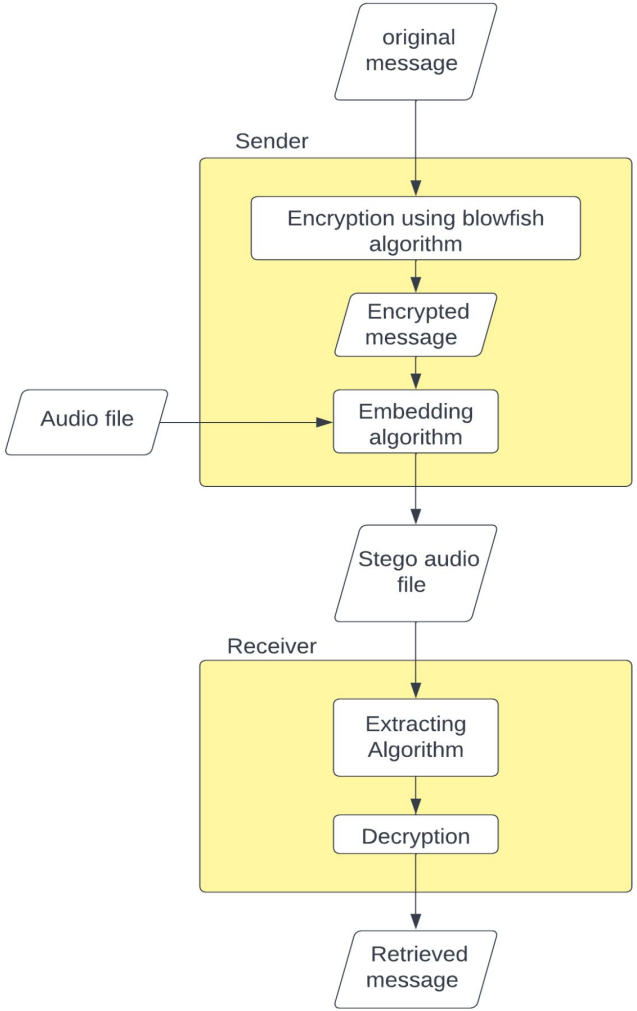

Plaintext
64 bits
32 bits
32 bits
P1
32 bits
32 bits
F
32 bits
Round 1
P2
F
Round 2
13 more iterations
P16
F
P18
P17
32 bits
32 bits
64 bits
Ciphertext


8 bits
S-box 1
32 bits
8 bits
S-box 2
32 bits
32 bits
8 bits
S-box 3
32 bits
32 bits
8 bits
S-box 4
32 bits

2

Randomized Substitution Algorithm

A modified substitution technique for audio steganography


original message
Sender
Encryption using blowfish algorithm
Encrypted message
Audio file
Embedding algorithm
Stego audio file
Receiver
Extracting Algorithm
Decryption
Retrieved message

Blowfish Encryption

The input data is encrypted using the Blowfish Algorithm with parameters:

- Block size : **8**
- Key size : **16** bytes

The Blowfish encryption was done using the Python library *PyCryptoDome*

Embedding Algorithm

- The embedding algorithm is performed to embed the message bits in a randomized manner.

k = size of the message file

Si = Byte location to be modified

Ti = sum of digits of Si

Qi = bit position to be embedded

Step 1 : $S_i = k + 60$

Step 2 : Find T_i

Step 3 : $Q_i = T_i \% d$

Step 4 : Encode every bit similarly

for j=1 to k-1

$S_{i+j} = S_i + Q_i$

T_{i+j}

$Q_{i+j} = T_{i+j} \% d$

$S_i = S_{i+j}$

$Q_i = Q_{i+j}$

end

Embedding Algorithm

- Assuming message file is of size 120 bytes.
- Giving space for the header of the wav file (first 60 bytes), encode the first bit of the message in **120+60 = 180th byte.**
- To get the position of the bit to be changed, get the sum of digits of the byte **(1 + 8 + 0 = 9)**.
- Now get the value of **9 % 8 (maximum bit depth) = 1**
- Change the 1st bit from the end of the 180th byte to the message bit.

Embedding Algorithm

To get the next embedding position,

- Add the current byte position and the current bit-change position **(180 + 1) = 181**
- Now change the **((1+8+1)%8) = 2**nd bit of the 181st byte to the message bit.
- The first embedding position can be seen as the symmetric key for the embedding process.

LSB reverse end embedding

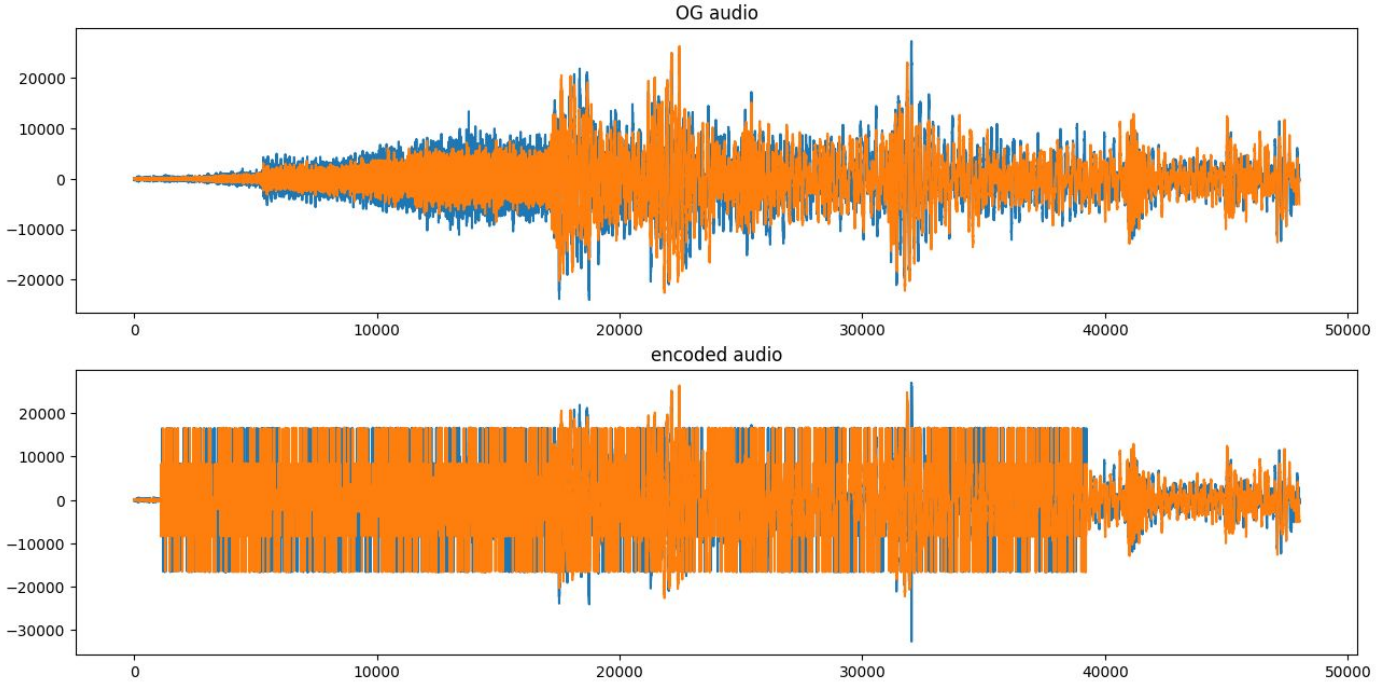
- The symmetric key for the embedding process can be encrypted using BlowFish and secured in the end of same audio file using LSB encoding.
- The encrypted key is embedded in the LSBs of the audio file from the end in a reverse order. To indicate the end of embedding, we set the second LSB as 1 once the embedding process is over.

Decoding

- The first step in decoding is to obtain the symmetric key used for encoding. It can be obtained using LSB extraction from the end of the received audio file until second LSB bit is 1.
- This encrypted key can be decrypted using BlowFish decryption, and the actual message cipher can be extracted by following the same steps used for encoding and reading the bits at the calculated positions.

Encryption System Results

```
amith  audio-stegano-blowfish  main ≡ ~3  3.10.7  0ms  py .\main.py
──────────── Sender ────────────
Message:  Location of the secret safe house: 11.268501570832086, 75.79267392030299

Blowfish encrypted message:  b"\x96&`\xaa\xdbL\xa9g\xe3)X\xc0'\xa1O\xd8Ev\x94\xb7{^\xd3\xf4F\x99\x0b\x95\xe6\x04\xc4hL\xb3\xac\xfc
\xf1\x8fA\xb8\xdd\xf3\xa1\xa0>\x17\xcd\x1d\xc6 \xd5\xe3\xfcO\xb9\xa8\x11\xda'\x04\xc3\xfdz(W\xed\xa7\xc0\xdb_\x89%\x19_\xc8\xbe\xb
2\xaf\x85\xf2w{\xfb}u\xdd\xbe\xb3"

Symmetric key of encoder:  148

Encrypted symmetric key of encoder:  b'^\xe6\x86\xc8\xf1\xd4o\x11\xb6-\x89\x98\x92\x02\xf2F'


──────────── Receiver ────────────
Recovered encrypted symmetric key of encoder:  bytearray(b'^\xe6\x86\xc8\xf1\xd4o\x11\xb6-\x89\x98\x92\x02\xf2F')

Decrypted symmetric key of encoder:  148

Recovered message from audio:  bytearray(b"\x96&`\xaa\xdbL\xa9g\xe3)X\xc0\'\xa1O\xd8Ev\x94\xb7{^\xd3\xf4F\x99\x0b\x95\xe6\x04\xc4h
L\xb3\xac\xfc\xf1\x8fA\xb8\xdd\xf3\xa1\xa0>\x17\xcd\x1d\xc6 \xd5\xe3\xfcO\xb9\xa8\x11\xda\'\x04\xc3\xfdz(W\xed\xa7\xc0\xdb_\x89%\x
19_\xc8\xbe\xb2\xaf\x85\xf2w{\xfb}u\xdd\xbe\xb3")

Decrypted message:  Location of the secret safe house: 11.268501570832086, 75.79267392030299
```



Noisy encoded audio obtained after encoding (Low SNR)

Solutions

To increase the SNR of the encoded audio, we can apply two methods:

- Enhanced encoding algorithm
- Change maximum bit depth

3

Enhanced Algorithms

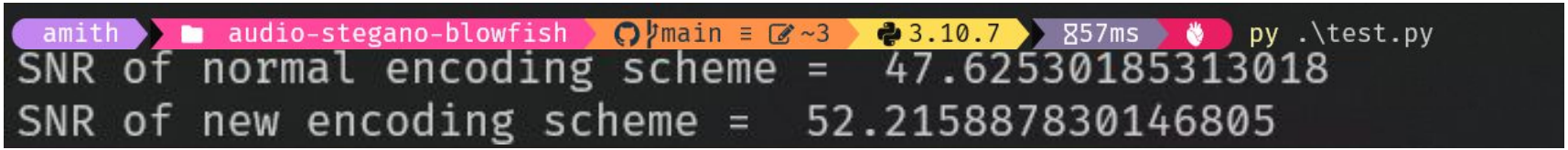
Enhanced encoding technique and bit depth variations are discussed

Enhanced Encoding Algorithm

The difference between original sample and modified sample are reduced by bit modification. This maintains the transparency of the encoded signal.

Original value	: 001**0**1111 $(47)_{10}$
Bit position to change	: 5
New value (old method)	: 001**1**1111 $(63)_{10}$
Difference	: 16
New value (new method)	: 001**1**0000 $(48)_{10}$
New difference	: 1

Enhanced Encoding Algorithm Result



Bit Depth Variation Method

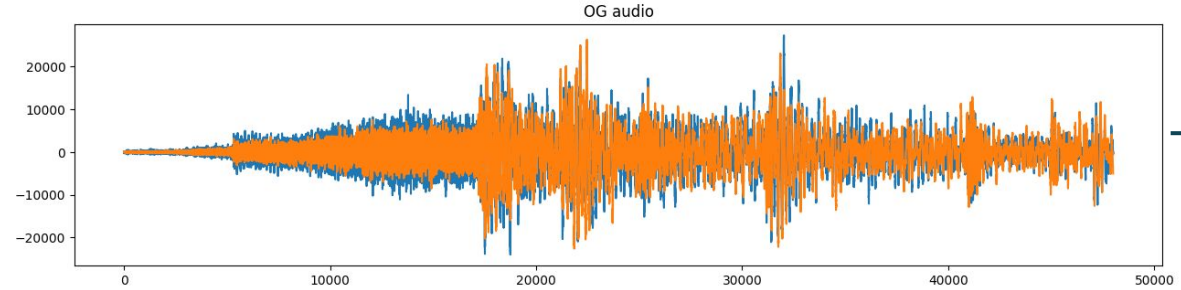
- The maximum number of bits that can be modified is reduced by varying the bit depth.
- This will modify only the lower bits, thus causing lesser changes in the encoded audio. Hence there is lesser distortion and higher SNR values.
- However, as the maximum bit depth decreases, there will be a compromise in the security of the system as the number of possible encoding positions decreases.

Bit Depth Variation Method

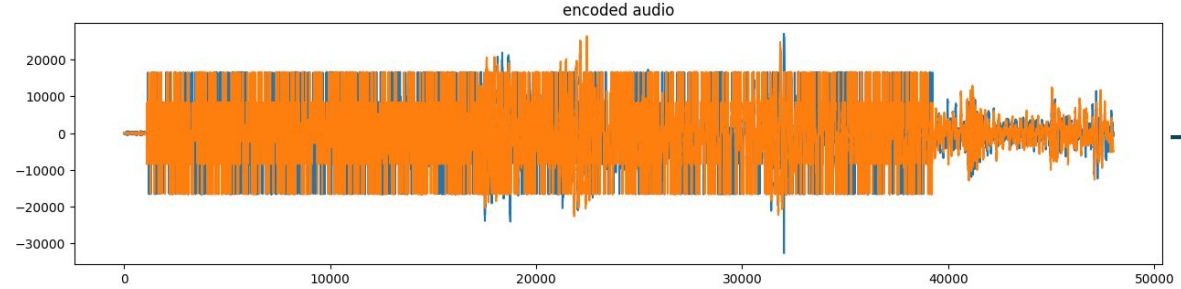
- Capacity of the scheme can be calculated as:
  (60+len) + [(len-1)*bd] <= audio size
  where len = length of the message bits
  bd = bit depth

  maximum length = (audio size - 60 + bd)/(1+bd)

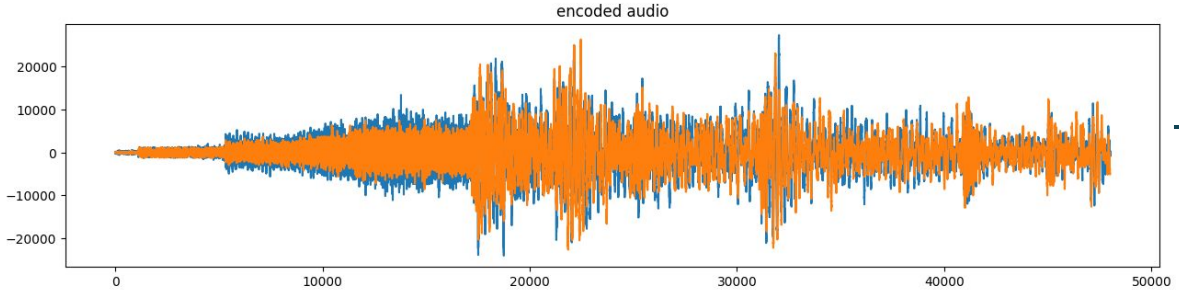
- It can be seen that as the bit depth decreases the capacity of the audio increases. But this comes as a tradeoff with security



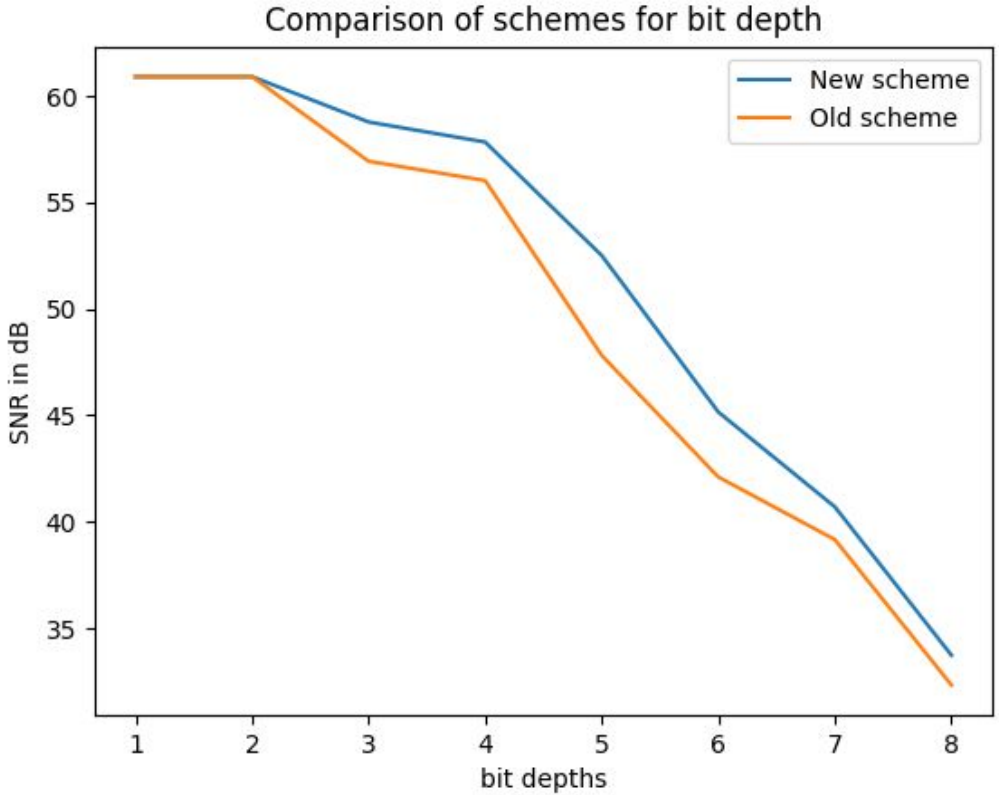
→ Original Audio



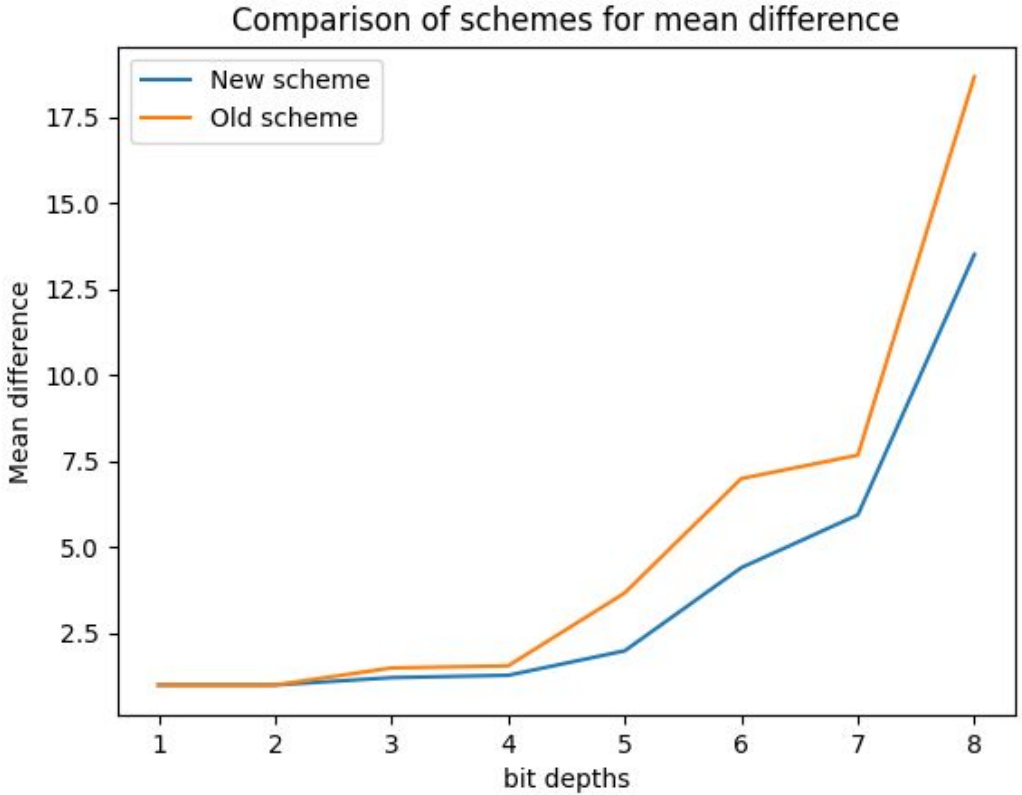
→ Encoded Audio (Bit depth 8)



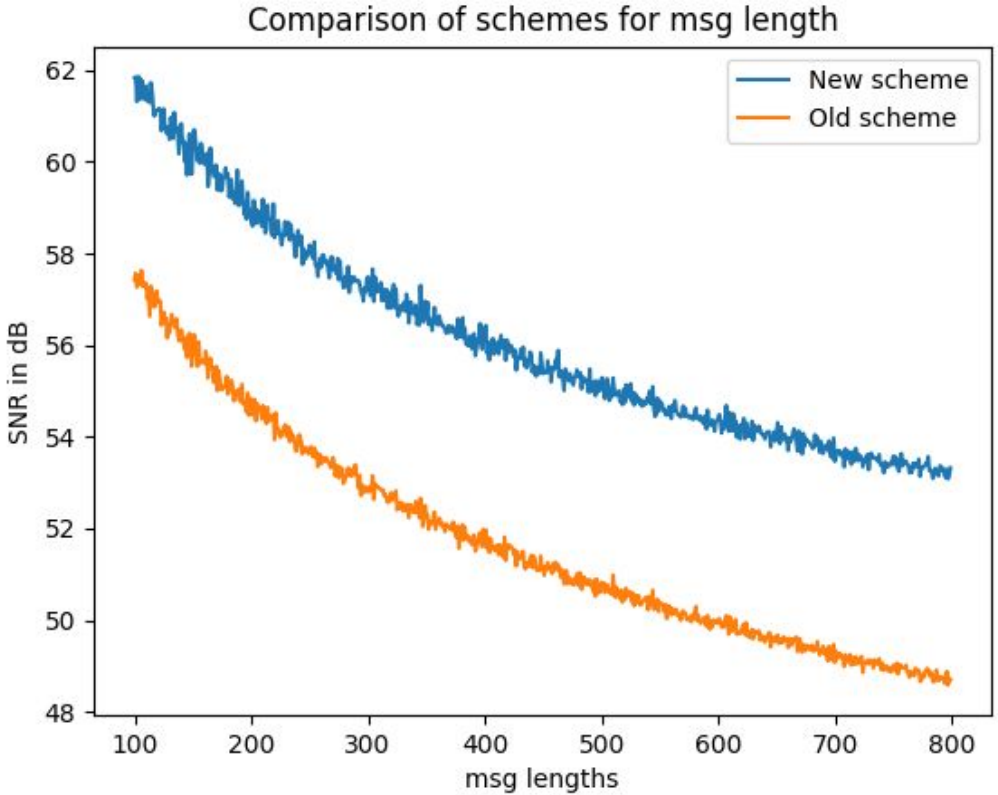
→ Encoded Audio (Bit depth 4)



SNR vs. bit depth plot for both encoding schemes



Mean difference vs. bit depth plot for both encoding schemes



SNR vs. message length plot for both encoding schemes

4

Application

- This method can be utilized for military applications like Confidential communication and secret data storing.
- This method can be used to watermark audio files with some tokens to ensure that the file is not modified.

REFERENCE

[1] F. Hemeida, W. Alexan and S. Mamdouh, "Blowfish–Secured Audio Steganography," 2019 Novel Intelligent and Leading Emerging Sciences Conference (NILES), 2019, pp. 17-20, doi: 10.1109/NILES.2019.8909206.

[2] Rana, Lovey and Saikat Banerjee. “Dual Layer Randomization in Audio Steganography Using Random Byte Position Encoding.” (2013).

[3] M. Zamani, R. B. Ahmad, A. B. A. Manaf and A. M. Zeki, "An approach to improve the robustness of substitution techniques of audio steganography," 2009 2nd IEEE International Conference on Computer Science and Information Technology, 2009, pp. 5-9, doi: 10.1109/ICCSIT.2009.5234863.