An Improved Phase Coding Audio Steganography Algorithm

Guang Yang
School of Information
University of California, Berkeley
Berkeley, CA, USA
guangyang19@berkeley.edu
0009-0005-1106-2674

Abstract—As AI technology continues to advance, voice cloning is becoming increasingly easy. Recently, cases of fraud involving audio forgery using AI technology have emerged, making it particularly important to covertly embed information and verify the authenticity and integrity of audio. Digital Audio Water-marking has thus become a crucial tool in this context. This study proposes an improved Phase Coding audio steganography algorithm that dynamically segments the audio signal and embeds information into the phase components of the mid-frequency range. This approach not only enhances the algorithm's resistance to steganalysis but also simplifies the computational process, ensuring the authenticity and integrity of audio both efficiently and securely.

Keywords—*Audio steganography, Phase coding, Anti-steganalysis, Fast Fourier Transform, Data integrity, Bit Error Rate, Digital Audio Watermarking*

I. INTRODUCTION

In recent years, cases of fraud involving AI-generated audio have become increasingly common. For instance, in 2021, some criminals used AI-generated cloned voices to impersonate the relatives or friends of victims in phone scams, leading to significant financial losses [1]. These incidents have exposed security vulnerabilities in current audio communications, particularly in areas like identity verification, digital audio watermarking and the validation of audio authenticity. Consequently, ensuring the authenticity and integrity of audio content has emerged as a critical research challenge that demands immediate attention. [2]

Embedding critical information into audio through audio steganography is a direct solution. Common audio steganography techniques include Least Significant Bit (LSB) embedding, Echo Hiding, Phase Coding, and Spread Spectrum. Among these, Phase Coding has the advantages of minimal impact on audio quality making it a favorable choice for audio steganography. [3]

Phase Coding hides information by modifying the phase components of an audio signal. This technique utilizes Fast Fourier Transform (FFT) to convert the audio signal into the frequency domain, where the secret data is embedded into the frequency domain, where the secret data is embedded into the phase information. The signal is then restored to the time domain using the Inverse Fast Fourier Transform (IFFT). [4]

However, traditional Phase Coding methods have several key drawbacks:

- Low Computational Efficiency: Classic Phase Coding methods require the calculation and maintenance of phase differences, often necessitating two passes over the audio data, which increases the computational load.
- Susceptibility to Detection: The changes introduced to the phase components are often too conspicuous, making traditional Phase Coding algorithms easier to detect, thereby reducing their undetectability. [5]
- Data Integrity : The secret message is typically concentrated in certain parts of the audio, leading to uneven data distribution and challenges in ensuring overall integrity. [6]

This study aims to address the growing threat of AI-generated audio forgeries by improving the traditional Phase Coding algorithm. The goal is to enhance its undetectability, tamper resistance and computational efficiency, thereby safe-guarding the authenticity and reliability of audio communications.

II. ALGORITHM

To address the aforementioned shortcomings , this project introduces an improved Phase Coding algorithm, streamlined into five steps and designed to distribute the embedded information evenly across the entire audio signal.

1. Segmenting the Input Audio

The input signal S is divided into n continuous segments, $S_i (i \in \{1, \ldots, n\})$, with each segment having a length of l.

2. Calculating Amplitude A_i and Phase ϕ_i for Each Segment Using Fast Fourier Transform (FFT), the amplitude A_i and phase ϕ_i for each segment are calculated as follows:

$$A_i = |FFT(S_i)|$$

$$\phi_i = angle\big(FFT(S_i)\big)$$

3. Embedding Information into the Mid-Frequency Phase The binary data d to be embedded (with a length of m, where $< l$) is converted into corresponding phase values $\phi_{data}[i]$:

$$\phi_{data}[i] = \begin{cases} \frac{\pi}{2}, & \text{if } d_i = 0; \\ -\frac{\pi}{2}, & \text{if } d_i = 1 \end{cases}$$

The data is embedded into the mid-frequency range of each segment S_i. Let $seg\_mid = \frac{l}{2}$, then ϕ_{data} is embedded into the mid-frequency portion of ϕ_i:

$$\phi_i'[seg\_mid - (j-1)] = \phi_{data}[j] \quad \text{for } j = 1,2,\dots,\frac{m}{n}$$

To maintain the odd symmetry of the Discrete Fourier Transform (DFT), the embedding operation is repeated in the corresponding mid-frequency range:

$$\phi_i'[seg\_mid + j] = -\phi_{data}\left[\frac{m}{n} - j + 1\right]$$

4. Directly Updating the Phase ϕ_i' for Each Segment

For each segment S_i, the phase ϕ_i' is directly updated without propagating phase differences $\triangle \phi_i'$:

$$\phi_i' = updated\ phase\ values$$

5. Reconstructing the Signal

The signal is reconstructed using the Inverse FFT (IFFT) on each segment $A_i \cdot exp(j\phi_i')$, and all segments are then concatenated to form the final audio signal:

$$S' = IFFT\left(A_i \cdot exp(j\phi_i')\right)$$

Key Improvements of the Algorithm:

Enhanced undetectability: By embedding information into the mid-frequency range, the algorithm avoids significant phase changes in the low-frequency range, improving undetectability.

Simplified Computation: The removal of the phase difference propagation step from the traditional algorithm allows for direct phase updates within each segment, simplifying the computational process and reducing potential phase distortion.

Ensured Data Integrity: The algorithm dynamically embeds data into segments of the whole audio signal. If the audio is subjected to clipping or editing, it becomes extremely difficult to recover the original data, thus ensuring the integrity of the embedded information.

Preserved Audio Quality: the improved algorithm enhances resistance to steganalysis while maintaining the perceptual quality of the audio signal.

III. IMPLEMENTATION

Considering that audio often has multiple channels, we will only process the first channel for simplicity in this demonstration. Below is the Python implementation for embedding and extracting using an improved Phase Coding audio steganography technique.

A. Embedding Code

```
import numpy as np
import scipy.io.wavfile as wavfile
def Embedding(input_filename, output_filename, message):
    rate, audio = wavfile.read(input_filename)
    audio = audio[:, 0] if len(audio.shape) > 1 else audio.copy()
    msg_len = 8 * len(message)
    seg_len = int(2 * 2**np.ceil(np.log2(2 * msg_len)))
    seg_num = int(np.ceil(len(audio) / seg_len))
    audio.resize(seg_num * seg_len, refcheck =False)
    msg_bin = np.ravel([[int(y) for y in format(ord(x), ‘08b’)] for x in message])
    msg_pi = np.where(msg_bin == 0, -np.pi /2, np.pi/2)
    segs = np.fft.fft(audio.reshape((seg_num , seg_len)))
    M, P = np.abs(segs), np.angle(segs)
    seg_mid = seg_len // 2
    for i in range(seg_num):
        start = i * len(msg_pi) // seg_num
        end = (i + 1) * len(msg_pi) // seg_num
        P[i, seg_mid - (end - start):seg_mid] = msg_pi[start:end]
        P[i, seg_mid + 1:seg_mid + 1 + (end - start)] = -msg_pi[start:end][::-1]
    audio = np.fft.ifft(M * np.exp(1j * P)).real.ravel().astype(np.int16)
    wavfile.write(output_filename, rate, audio)
```

B. Extracting Code

```
def Extracting(input_filename, msg_len):
    rate, audio = wavfile.read(input_filename)
    seg_len = int(2 * 2**np.ceil(np.log2(2 *
    msg_len)))
    seg_num = int(np.ceil(len(audio) / seg_len))
    seg_mid = seg_len // 2
    extracted_bits = []
    for i in range(seg_num):
        x = np.fft.fft(audio[i * seg_len:(i + 1)
        * seg_len])
        extracted_phase = np.angle(x)
        start = i * msg_len // seg_num
        end = (i + 1) * msg_len // seg_num
    extracted_bits.extend((
    extracted_phase[seg_mid - (end -
    start):seg_mid] < 0).astype(np.int8))
    extracted_bits = np.array(
    extracted_bits[:msg_len])
chars = extracted_bits.reshape((-1, 8)).dot(1 <<
np.arange(8 - 1, -1, -1)).astype(np.uint8)
return ‘’.join(chr(c) for c in chars)
```

C. Verification Code

When verifying the original message and the extracted steganographic message, we should not only check if the extracted message is consistent but also focus on the bit error rate (BER). Therefore, let’s first implement a function to calculate its BER and accuracy.

```
def calculate_accuracy(original_msg,
extracted_msg):
    original_bits = ‘’.join(f‘{ord(c):08b}’ for c
    in original_msg)
    extracted_bits = ‘’.join(f‘{ord(c):08b}’ for
    c in extracted_msg)
    total_bits = len(original_bits)
    incorrect_bits = sum(o != e for o, e in
    zip(original_bits, extracted_bits))
    bit_error_rate = incorrect_bits / total_bits
    message_accuracy = 1 if original_msg ==
    extracted_msg else 0
    return bit_error_rate, message_accuracy
```

In this case, ‘example.wav’ can be any mono or multi-channel audio file. We will embed the word “test” into the file, extract it afterward, and then calculate the bit error rate (BER) and check its accuracy.

```
input_filename = ‘example.wav’
output_filename = ‘embedded.wav’
message = “test”
Embedding(input_filename, output_filename,
message)
extracted_message = Extracting(output_filename, 8
* len(message))
print(f‘Extracted_Message:{extracted_message}’)
ber, accuracy = calculate_accuracy(message,
extracted_message)
print(f‘Bit Error Rate (BER): {ber:.2%}’)
print(f‘Message Accuracy: {"Correct" if accuracy
else "Incorrect"}’)
```

IV. EVALUATION

This study mainly focuses on the undetectability of the improved algorithm and its impact on the bit error rate (BER). Therefore, I primarily use frequency-phase comparison images to analyze the impact of the improved algorithm on undetectability. Additionally, we evaluate the effect of message length on the bit error rate (BER) and the improvements made by embedding messages of different lengths.

A. Undetectability

For undetectability, we primarily examine the changes in frequency-phase before and after information embedding. We plot the frequency-phase comparison charts for the original phase coding algorithm before and after data embedding, and then plot the frequency-phase comparison charts for the improved phase coding algorithm before and after data embedding for comparison.

The improved Phase Coding algorithm embeds information into each segment of the audio, resulting in a more uniform spectral distribution that is harder to detect using spectral analysis tools.

In the phase comparison diagrams below, the blue line represents the frequency domain phase of the original audio, and the red line represents the frequency domain phase after steganography.

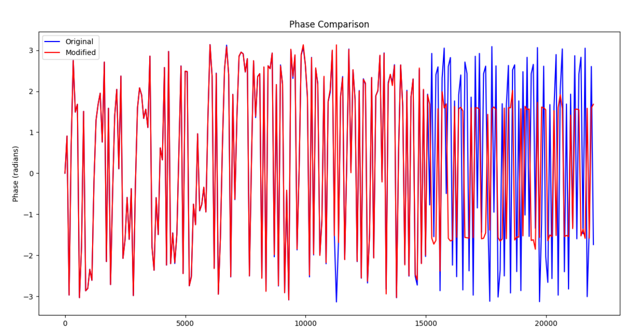

Fig. 1. Phase Comparison of the traditional Phase Coding algorithm

In Figure 1, it is evident that the phase changes in the traditional Phase Coding method are concentrated and pronounced, making them easily detectable by tools like spectral analysis.

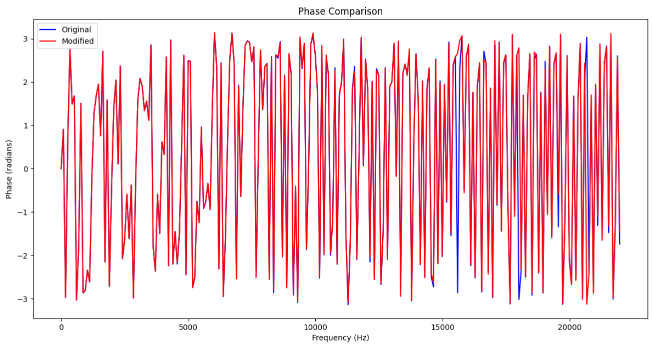

Fig. 2. Phase Comparison of the improved Phase Coding algorithm

In contrast, the improved algorithm distributes the steganographic data more naturally, greatly enhancing its undetectability, as shown in Figure 2.

B. Bit Error Rate (BER)

For the bit error rate (BER), since the algorithm segments the audio based on the length of the information being embedded, this study focuses on the impact of different information lengths on both algorithms. Therefore, a 12-second audio clip was selected, and data strings with lengths incrementing from 1 to 128 were tested. This approach allows us to observe how the BER is affected by different data lengths, ranging from 8 to 1024 bits, before and after the improvement of the phase coding algorithm.

```
input_filename = ‘example.wav’
output_filename = ‘embedded.wav’
ber_values = []
for n in range(1, 128):
    message = “t” * n
    embed_message(input_filename,
    output_filename, message)
    extracted_message =
    extract_message(output_filename, 8 *
    len(message))
    ber, accuracy = calculate_accuracy(message,
    extracted_message)
    ber_values.append(ber)
```

As shown in Figure 3, the blue line represents the BER for the traditional Phase Coding method, while the red line represents the BER for the improved algorithm. It can be observed that as the amount of data increases, the traditional algorithm becomes increasingly unstable, with the BER rising gradually. In contrast, the enhanced algorithm demonstrates significant improvements, with no noticeable BER until after 768 bits, and even then, the BER remains lower.

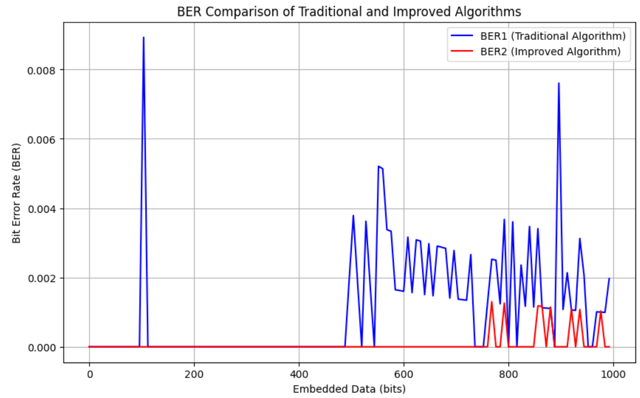

Fig. 3. Bit Error Rate (BER) Comparison of Traditional and Improved Algorithms

This indicates that the improved algorithm is more effective in maintaining data integrity during the embedding and extraction process, reducing the information loss or errors.

V. CONCLUSION

The improved Phase Coding algorithm enhances steganography by increasing stealthiness and resistance to steganalysis, significantly lowering the bit error rate (BER). It also boosts computational efficiency and robustness against interference. With growing concerns over voice cloning and audio forgery, this algorithm provides a vital layer of security, protecting the integrity and authenticity of audio data. Future work will focus on increasing fault tolerance without adding computational complexity, particularly in safeguarding information in complex, real-world audio environments, making it crucial for applications like secure communications and identity verification.

REFERENCES

[1] T. Brewster, “Huge bank fraud uses deep fake voice tech to steal millions,” *Forbes*, 14 Oct 2021. [Online]. Available: https://www.forbes.com/sites/thomasbrewster/2021/10/14/huge-bank-fraud-uses-deep-fake-voice-tech-to-steal-millions/

[2] P. Singh, “A comparative study of audio steganography techniques,” *International Research Journal of Engineering and Technology (IRJET)*, vol. 3, no. 4, pp. 580–585, 2016. [Online]. Available: https://www.irjet.net/archives/V3/i4/IRJET-V3I4117.pdf

[3] F. Djebbar, B. Ayad, K. A. Meraim, and H. Hamam, “Comparative study of digital audio steganography techniques,” *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2012, p. 25, 2012. [Online]. Available: https://doi.org/10.1186/1687-4722-2012-25

[4] D. Oak, “Audio steganography using phase encoding technique,” 2013. [Online]. Available: https://dsp.stackexchange.com/questions/14804/audio-steganography-using-phase-encoding-technique/

[5] W. Zeng, H. Ai, and R. Hu, “A novel steganalysis algorithm of phase coding in audio signal,” in *Sixth International Conference on Advanced Language Processing and Web Information Technology (ALPIT 2007)*, 2007, pp. 261–264. [Online]. Available: https://doi.org/10.1109/ALPIT.2007.41

[6] A. A. Alsabhany, F. Ridzuan, and A. H. Azni, “The adaptive multi-level phase coding method in audio steganography,” *IEEE Access*, vol. 7, pp. 129 291–129 306, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2940640