PRELIMINARY REPORT

Implementation of Steganography Techniques

Nithish Ravikkumar

Contents

PRELIMINARY REPORT

1 Introduction

1.1 Background

Information hiding is a field of study that encompasses techniques to conceal data within other, non-secret data. Its primary goal is to ensure that the secret information is not revealed. Within this field, two major disciplines are steganography and cryptography.

Cryptography is the practice of securing communication by converting a message into an unreadable format. The existence of the communication is public, but its *content* is hidden from unauthorized parties. An encrypted message, while unreadable, clearly indicates that a secret communication is taking place.

Steganography is the art and science of hiding a message, image, or file within another message, image, or file. Its core goal is to conceal the very *existence* of the secret communication, making the output (the "stego-signal") appear innocuous. The word itself is derived from the Greek words *steganos* (meaning "covered" or "concealed") and *graphein* (meaning "writing").

1.2 Objectives

The primary goal of this project is to implement, compare, and analyze various steganography techniques for digital audio and image signals. The study focuses on evaluating each method based on three fundamental, often conflicting, metrics: data capacity, imperceptibility, and robustness against common signal processing manipulations.

The key objectives set forth for this project are as follows:

- Implement time-domain steganography techniques, specifically 1-bit and 2-bit Least Significant Bit (LSB) substitution for both audio and image carriers.
- Implement frequency-domain steganography techniques, including a Discrete Cosine Transform (DCT) based method and Phase Coding.
- Develop a framework to systematically analyze the robustness of each implemented method against common signal processing attacks, such as lossy compression and noise addition.
- Quantify and compare the performance of each technique based on calculated metrics for capacity (in KB), imperceptibility (PSNR), and robustness (BER).

2 Background

2.1 Representation of Signals

The foundation of digital steganography lies in the discrete representation of continuous, real-world signals. This project considers two primary signal types: images and audio, each with a distinct mathematical model.

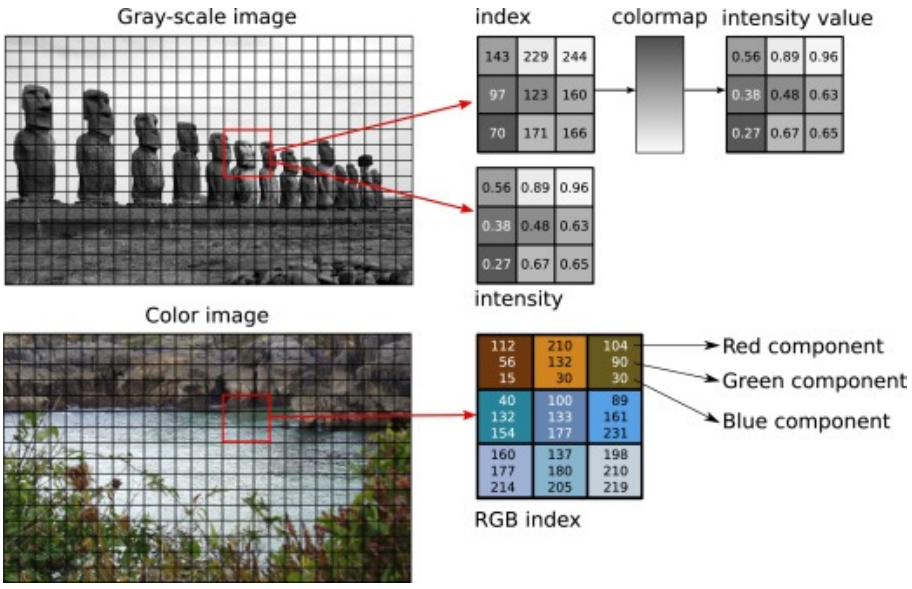

Figure 1: Image pixelated representation

2.1.1 Images as 2D Discrete Signals

A digital image is a discrete representation of a continuous two-dimensional light intensity function. The process of converting this continuous scene into a digital format involves two key steps:

- **Spatial Sampling:** The continuous image plane is partitioned into a finite grid. The light intensity is measured within each grid cell, and this cell is represented by a single element called a **pixel**. The density of this grid determines the image's spatial resolution.
- **Amplitude Quantization:** The measured intensity or color information at each pixel is mapped to a discrete value from a finite set of levels. For standard color images, the **RGB color model** is employed, where each pixel is a vector of three intensity values corresponding to the Red, Green, and Blue channels. Each channel is typically quantized using 8 bits, providing $2^8 = 256$ distinct levels (from 0 to 255).

Thus, a color image is mathematically modeled as a 3D tensor of size (height $\times$ width $\times$ channels), denoted as $I(x, y, c)$, where (x, y) are the spatial coordinates and c is the color channel. Other color spaces, such as YCbCr, separate the image into a luminance (brightness) component and two chrominance (color) components, a representation that is critical for compression algorithms like JPEG.

2.1.2 Audio as 1D Discrete-Time Signals

Digital audio is a 1D discrete-time signal that represents a continuous analog sound wave. The conversion process is known as **Pulse-Code Modulation (PCM)** and is governed by the principles of sampling and quantization.

- **Sampling:** The amplitude of the analog waveform is measured at discrete, uniform time intervals. The frequency at which these measurements are taken is the **sampling rate**, f_s. For CD-quality audio, a sampling rate of 44,100 Hz is used to capture the full range of human hearing (up to 20 kHz).

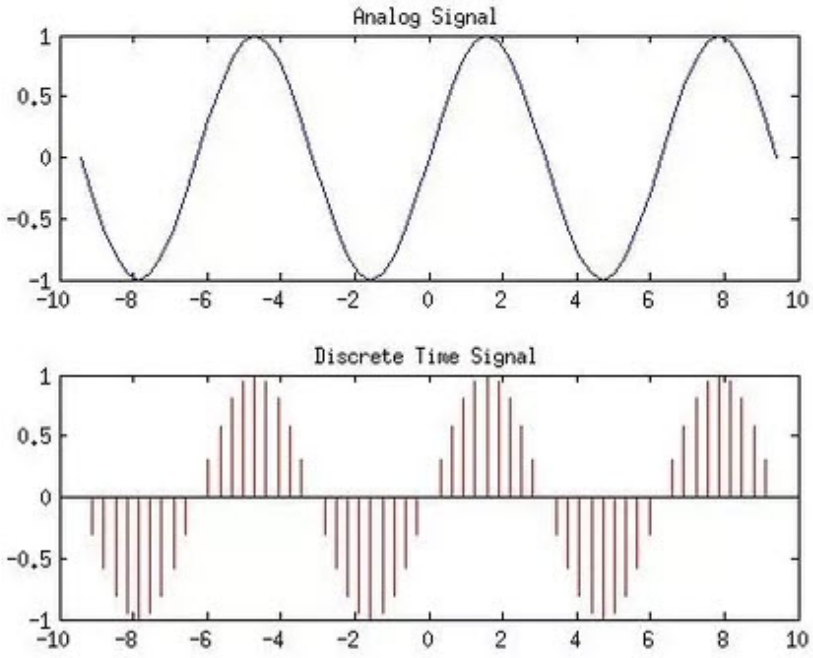

Figure 2: Pulse Modulation

- **Quantization:** Each captured sample's continuous amplitude is approximated by a discrete value from a finite set of levels. The number of bits used to represent each level is the **bit depth**. A higher bit depth provides a more accurate representation of the sample's amplitude, which directly impacts the audio's fidelity. This is quantified by the **Signal-to-Quantization-Noise Ratio (SQNR)**, which for a linear PCM system is approximately:

$$\text{SQNR (dB)} \approx 6.02 \cdot N + 1.76 \tag{1}$$

  where N is the bit depth. For 16-bit audio, this yields a high dynamic range and an SQNR of about 98 dB.

A stereo audio file consists of two such 1D signals (channels), while a mono file consists of one.

2.2 Signal Domains for Steganography

The domain in which data is embedded is a critical design choice that dictates the trade-offs between capacity, imperceptibility, and robustness.

2.2.1 Time/Spatial Domain

This domain represents the signal in its most direct form: a sequence of sample amplitudes over time (for audio) or a grid of pixel intensities over space (for images). Steganography in this domain, exemplified by the LSB substitution method, involves directly modifying these individual data values. The modification to a value at a specific coordinate, $I(x, y, c)$, is independent of its neighbors.

While this approach is computationally simple and typically offers the highest data-hiding capacity, it is inherently fragile. The embedded data, often statistically independent of the

host signal, manifests as high-frequency noise. Signal processing operations such as lossy compression or filtering, which are designed to remove or reduce such noise, will invariably destroy the hidden information.

2.2.2 Frequency (Transform) Domain

This approach operates on a more abstract representation of the signal. A mathematical transform is applied to convert the signal from the time domain into the frequency domain. In this representation, the signal is described as a weighted sum of basis functions (e.g., cosine or sine waves of different frequencies), and these weights are known as the transform coefficients.

Steganography is performed by modifying these coefficients. This method is built upon the principle of **Perceptual Irrelevance**, which leverages characteristics of the Human Auditory System (HAS) and Human Visual System (HVS). For instance, the HVS is less sensitive to changes in high-frequency components (e.g., fine textures) than in low-frequency components (e.g., smooth areas). Similarly, the HAS is less sensitive to modifications of frequencies that are either very high or are "masked" by louder, adjacent frequency components. By embedding data in these perceptually less significant coefficients, the modification is more likely to be imperceptible. A single coefficient modification results in a small, distributed change across the entire signal frame in the time/spatial domain, making the embedded data inherently more resilient to localized noise and compression algorithms.

2.3 Mathematical Transforms in Signal Processing

Transform-domain steganography relies on converting a signal from its native time/spatial domain into a frequency representation. This allows for the manipulation of coefficients that correspond to specific spectral characteristics of the signal, which is often more robust and less perceptible than direct time-domain modification. The two key transforms utilized in this project are the Discrete Cosine Transform (DCT) and the Fast Fourier Transform (FFT).

2.3.1 Discrete Cosine Transform (DCT)

The DCT is a real-valued transform that expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. For signal processing, the most common variant is the DCT-Type II, which is the basis for standards like JPEG compression. It is defined as:

$$X_k = \alpha(k) \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi(2n+1)k}{2N}\right) \quad \text{for } k = 0, 1, \ldots, N-1 \tag{2}$$

where:

- x_n is the n-th sample of the input signal frame.
- N is the total number of samples in the frame.
- X_k is the k-th DCT coefficient, representing the amplitude of the k-th frequency component.
- The cosine term represents the **basis functions** of the transform. The input signal x_n is projected onto these cosine waves of varying frequencies.

- $\alpha(k)$ is an orthonormality scaling factor defined as:

$$\alpha(k) = \begin{cases} \sqrt{1/N} & \text{if } k = 0 \\ \sqrt{2/N} & \text{if } k > 0 \end{cases}$$

The primary advantage of the DCT is its exceptional **energy compaction** property. For highly correlated signals, such as natural images and audio, the DCT concentrates most of the signal's energy into a few low-frequency coefficients (those with a small index k). This property is exploited in steganography by embedding data into the mid-frequency coefficients, which are less perceptually significant than the low-frequency ones but more robust to quantization than the high-frequency ones.

2.3.2 Fast Fourier Transform (FFT)

The FFT is not a transform, but a highly efficient algorithm for computing the **Discrete Fourier Transform (DFT)**. The DFT is a fundamental transform that converts a finite, discrete-time signal into its complex-valued frequency-domain representation. The DFT is defined as:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-j\frac{2\pi}{N}kn} \quad \text{for } k = 0, 1, \ldots, N-1 \tag{3}$$

where:

- x_n is the n-th sample of the input signal frame.
- N is the number of samples.
- X_k is the k-th DFT coefficient, which is a **complex number**.
- j is the imaginary unit, $\sqrt{-1}$.
- The term $e^{-j\frac{2\pi}{N}kn}$ is the complex exponential basis function. Using Euler's formula, $e^{j\theta} = \cos(\theta) + j\sin(\theta)$, it can be seen that this term represents both cosine and sine components, which is why the output is complex.

A direct computation of the DFT has a complexity of $O(N^2)$, whereas the FFT algorithms reduce this to $O(N \log N)$, making it computationally feasible for practical applications.

The critical feature of the FFT for steganography is its complex-valued output. Each coefficient X_k can be represented in polar coordinates by its **magnitude**, A_k, and its **phase**, ϕ_k:

$$X_k = A_k e^{j\phi_k} \tag{4}$$

where:

- $A_k = |X_k|$ represents the amplitude or strength of the k-th frequency component.
- $\phi_k = \arg(X_k)$ represents the phase offset of the k-th frequency component.

This explicit separation of magnitude and phase is essential for the phase coding technique implemented in this project, which relies on modulating the latter while preserving the former.

The Cooley-Tukey Algorithm

The specific FFT algorithm implemented in the NumPy library used in this project, is the **Cooley-Tukey algorithm**. It is a highly efficient "divide and conquer" algorithm that recursively breaks down a DFT of any composite size $N = N_1 N_2$ into many smaller DFTs of sizes N_1 and N_2.

The most common implementation, and the one relevant to this project, is the **radix-2 decimation-in-time** version. This variant requires the signal frame size N to be a power of two (e.g., 1024, 2048), which aligns with the parameters chosen in `script_phase_coding.py`. The algorithm proceeds as follows:

- **Decimation:** The input signal frame, x_n, is first decimated (split) into two sub-sequences of length $N/2$: one containing the even-indexed samples and one containing the odd-indexed samples.
- **Recursion:** The algorithm then recursively calls itself to compute the $N/2$-point DFT of each of these smaller sub-sequences.
- **Combination:** The results of the two smaller DFTs are then combined through a series of complex multiplications with terms known as "twiddle factors" ($e^{-j\frac{2\pi}{N}k}$) and additions to yield the full N-point DFT of the original signal.

This recursive decomposition is what reduces the computational complexity from the inefficient $O(N^2)$ of a direct DFT computation to the much more efficient $O(N \log N)$. This makes the transform practical for the frame-based analysis performed in this project.

3 Implementation

This section provides a detailed account of the project's implementation, outlining the development workflow, file structure and the specific algorithms for each steganography technique.

3.1 Project Flow

The project was developed in an incremental manner, starting with fundamental techniques and progressively building towards more complex methods and a comprehensive analysis framework. The file structure, as detailed below, reflects this workflow.

Phase 1: LSB Implementation LSB substitution is the first method tested out, on various carrier and payload combinations.

- `script_image.py`: Initial implementation of text-in-image LSB.
- `script_img2img.py`: Extended LSB to handle image-in-image payloads.
- `script_audio2audio.py`, `script_img2audio.py`: Adapted the LSB logic for audio carriers, handling both audio and image payloads.
- `script_img2audio_2b.py`: Enhanced the audio LSB method to use 2 bits per sample, doubling the capacity.

Phase 2: Frequency-Domain Implementations. This focused on more robust transform-domain techniques, which offer higher resistance to compression induced noise and errors.

- `script_dct_txt2audio.py`: Implemented a DCT-based method for hiding text in audio.
- `dct_idct_algo.py`: Manual implementation of the DCT and IDCT algorithms, separate from optimized library functions.
- `script_phase_coding.py`: Implemented the Phase Coding technique, leveraging the Fast Fourier Transform (FFT) for high-capacity, robust embedding.

Phase 3: Utility and Analysis Framework. The final phase involved creating supporting utilities and the framework for performance evaluation.

- `create_audio.py`: A utility script to generate sample WAV audio files for testing.
- `subtract_image.py`: A crucial analysis tool to create "difference maps" between original and stego-images, visually demonstrating the impact of embedding.
- `image_compress.py`: A pre-processing utility to compress large image payloads to a target file size, enabling them to fit within the capacity limits of certain methods.
- `robust_analysis.py`: Performance evaluation framework which contains functions to apply various "attacks" (e.g., JPEG compression, noise) to stego-signals and calculate performance metrics like the Bit Error Rate (BER).

4 Methodology

This section provides a detailed description of the steganography techniques implemented in this project.

4.1 LSB Manipulation

Least Significant Bit (LSB) substitution is a technique that operates on the premise that any digital signal, image or audio, has a noise floor of data that can be altered without being noticed. The goal is to make changes so small that they are indistinguishable from this inherent noise. This is achieved by modifying the last, or least significant bit (LSB) of the numbers that represent the signal.

For example, in a high-resolution digital photograph, if you change the color of a single pixel from a dark grey with a value of 150 to a slightly different dark grey with a value of 151, the change is visually imperceptible. LSB steganography exploits the lack of precision in perceiving color intensity and changes, which allows for minor differences to go unnoticed.

1. **Payload format:** The payload (Image, audio or text) is converted to a binary bitstream.
2. **Header:** To reliably extract the hidden file, the decoder needs to know exactly when to stop reading bits. Instead of using a special "end" character that could accidentally appear in the payload, a more robust system was designed. The total length of the payload bitstream is calculated and stored as a 32-bit integer. This number is embedded at the very beginning of the data stream.
3. **Data embedding:** The program then iterates through the carrier signal's data points (the samples of a WAV file or the individual R, G, and B values of an image's pixels). For each bit in the combined header-plus-payload stream, it modifies one of these data points.
4. **Decomposition and Retrieval:** The decoding process is the reverse of the encoding. The decoder acts first reads enough data to extract and reconstruct the 32-bit header. Once it knows the exact size of the hidden file, it continues to extract the LSBs from the subsequent data points until it has reassembled the entire payload bitstream, which is then converted back into the original file.

4.1.1 1-bit LSB

The 1-bit LSB variant offers the highest degree of imperceptibility among LSB methods by embedding a single data bit into the least significant bit of a carrier signal's quantized value.

- **Algorithmic Process:** The embedding process for a single carrier sample, S, and a message bit, $b \in \{0, 1\}$, can be described mathematically. The LSB of the sample is first cleared (set to zero), and then the message bit is embedded using a bitwise OR operation. The resulting stego-sample, S', is calculated as:

$$S' = (S \wedge \neg 1) \vee b \tag{5}$$

  where $\wedge$ denotes a bitwise AND, $\vee$ denotes a bitwise OR, and $\neg 1$ is the bitmask that clears the LSB (e.g., `...11111110`). This operation guarantees that the modification introduces a maximum absolute error of 1, as the original value is only ever changed if its LSB differs from the message bit. The data-hiding capacity is exactly one bit per carrier data point (e.g., per audio sample or per pixel channel).

4.1.2 2-bit LSB

This method was implemented in `script_img2audio_2b.py` to address the capacity limitations of 1-bit LSB. It doubles the data-hiding capacity by embedding two bits of the secret message into the two least significant bits of each carrier data point.

- **Algorithmic Process:** The embedding process is analogous to the 1-bit case but uses a different bitmask to manipulate two bits simultaneously. For a sample S and a 2-bit message chunk B (represented as an integer in the range $[0, 3]$), the new sample value S' is calculated as:
$$S' = (S \wedge \neg 3) \vee B \tag{6}$$
The mask $\neg 3$ (e.g., `...11111100`) clears the two least significant bits of the sample, which are then replaced by the 2-bit message chunk B.

- **Capacity vs. Distortion Trade-off:** While this doubles the capacity to two bits per carrier data point, it also increases the potential for perceptual distortion. The modification can alter the original sample value by an absolute maximum of 3, introducing a larger error term compared to the 1-bit variant. This exemplifies the fundamental trade-off between data-hiding capacity and the imperceptibility of the stego-signal.

4.2 DCT

This technique moves the embedding process into the frequency domain to achieve greater robustness against signal processing modifications. The approach is to hide data not in the signal's raw amplitude values, but within its spectral structure, which is revealed by the Discrete Cosine Transform (DCT).

4.2.1 DCT

The DCT separates a signal frame into a set of coefficients, where each coefficient, X_k, represents the energy of a specific frequency component (basis function). Due to the DCT's energy compaction property, the low-frequency coefficients (small k) contain most of the signal's perceptually significant information. High-frequency coefficients (large k) represent fine details and are often aggressively quantized or discarded by lossy compression algorithms.

Therefore, the embedding strategy in this project targets the **mid-frequency coefficients**. This region offers an optimal balance: the coefficients are less critical to the overall perceptual quality of the signal than the low-frequency components, yet they are more likely to survive compression than the high-frequency components.

4.2.2 Implementation Algorithm in `script_dct_txt2audio.py`

The implementation of this method was an iterative process that revealed and solved a critical challenge related to data precision in transform-domain manipulation.

1. **Framing:** The continuous audio signal, being non-stationary, is first segmented into a series of non-overlapping, quasi-stationary frames. A frame size of $N = 1024$ samples was selected for this implementation.

2. **Frequency Transformation:** The DCT is applied to each time-domain frame, converting it into a vector of 1024 frequency coefficients.

3. **Quantization**

   - An initial implementation attempt involved embedding data by directly modifying the Least Significant Bit of a selected mid-frequency coefficient. This approach failed, resulting in a completely corrupted message upon decoding (BER near 50%).
   - The root cause was identified as a **loss of precision**. The DCT and its inverse (IDCT) are floating-point operations. The minuscule change introduced by the LSB flip was completely subsumed by the numerical noise and rounding errors inherent in the inverse transform and, most critically, during the final conversion of the reconstructed floating-point audio samples back to the 16-bit integer format required for the WAV file.
   - To overcome this, a more forceful and robust embedding technique, **quantization**, was implemented. This method ensures the embedded data "survives" the round-trip transformation. The process for a selected coefficient X_k and message bit b is as follows:

     (a) The coefficient is quantized by dividing it by a large, predefined **quantization step**, Q_{step} (e.g., 80.0), and rounding to the nearest integer level, L:

     $$L = \text{round}(X_k / Q_{step})$$

     (b) The parity (evenness or oddness) of this integer level L is then forced to match the message bit b. If the parity does not match, the level is adjusted to the nearest integer with the correct parity, creating L':

     $$L' = \begin{cases} L & \text{if } L \pmod 2 = b \\ L+1 & \text{if } L \pmod 2 \neq b \text{ (or } L-1) \end{cases}$$

     (c) The modified coefficient, X'_k, is then de-quantized by multiplying the adjusted level by the quantization step:

     $$X'_k = L' \times Q_{step}$$

     This large, deliberate modification to the coefficient's value is robust enough to be reliably recovered after the inverse transform.

4. **Signal Reconstruction:** The Inverse DCT (IDCT) is applied to each frame of modified coefficients to convert it back into the time domain. The resulting frames are then concatenated in order to reconstruct the full stego-audio signal.

5. **Foundational Study (`dct_idct_algo.py`):** To gain a deeper, fundamental understanding of the transform and the precision issues, a manual from-scratch implementation of the DCT-II formula was coded, separate from the optimized SciPy library functions. This exercise was instrumental in diagnosing the failure of the initial LSB-in-DCT approach.

6. **Data Retrieval (Decoding):** The decoding process must precisely mirror the encoder's parameters. The decoder frames the stego-audio, applies the DCT, and for the same mid-frequency coefficient X'_k, it calculates the quantized level $L' = \text{round}(X'_k / Q_{step})$. The hidden bit is then retrieved by simply checking the parity of this level: $b_{retrieved} = L' \pmod 2$.

4.3 Phase Coding

Phase coding is a frequency-domain steganography technique that operates on the principle that the human auditory system (HAS) is highly sensitive to changes in the magnitude (amplitude) of a signal's frequency components but is relatively insensitive to alterations in their phase. This perceptual characteristic allows for the embedding of data through phase modulation with minimal audible distortion.

4.3.1 The Principle of Phase Manipulation and the FFT

Any discrete-time signal can be represented as a summation of sinusoidal components, each defined by its amplitude, frequency, and phase. The phase, ϕ, represents the initial offset of the sinusoid. By modifying the phase of a signal's constituent frequencies while preserving their magnitudes, the fundamental energy distribution of the signal remains unchanged, rendering the modification largely imperceptible.

To access and manipulate these components, the signal is transformed into the frequency domain using the Fast Fourier Transform (FFT). The FFT decomposes a time-domain signal frame into a vector of complex numbers, where each coefficient, C_k, can be expressed in its polar form $C_k = A_k e^{j\phi_k}$. This representation allows for the independent manipulation of the magnitude, A_k, and the phase, ϕ_k.

4.3.2 Implementation Algorithm in `script_phase_coding.py`

The implementation of phase coding in this project follows a systematic, frame-based process of analysis, modification, and synthesis. The specific parameters and design choices were selected to balance capacity, robustness, and imperceptibility.

1. **Parameter Selection:** Specific parameters were chosen for the analysis frames:
    - `frame_size = 2048`: A frame size of 2048 samples was selected to provide sufficient frequency resolution for embedding.
    - `hop_size = 1024`: A 50% overlap between frames was used, which is a standard choice for the subsequent overlap-add reconstruction method.
    - `freq_range_to_modify = (40, 100)`: Data was embedded in the phase of mid-range frequency bins. This range was chosen to avoid the perceptually critical low-frequency components and the less reliable, often low-energy high-frequency components.
2. **Framing and Windowing:** The carrier audio signal is first segmented into frames of 2048 samples, with each frame starting 1024 samples after the previous one. To mitigate spectral leakage artifacts, each frame is multiplied by a **Hanning window function**, which smoothly tapers the frame's amplitudes to zero at its edges.
3. **Embedding Process:** The core of the implementation is an iterative loop that processes each frame:
    - The FFT is applied to the windowed frame to obtain the complex frequency coefficients. These are then converted to their polar representation to separate the magnitude, A_k, from the phase, ϕ_k.

- The payload bitstream is read sequentially. For each frequency bin k within the selected range (40 to 100), the next available bit from the payload, b_k, is embedded by modulating the phase ϕ_k:

$$\phi'_k = \begin{cases} \phi_k & \text{if } b_k = 0 \\ \phi_k + \pi/2 & \text{if } b_k = 1 \end{cases}$$

A phase shift of $+\pi/2$ radians was chosen as it represents a significant, unambiguous change for the decoder. The magnitude component, A_k, is preserved without modification.

4. **Signal Reconstruction:**

   - The modified frequency spectrum is synthesized by reconstructing the complex coefficients, $C'_k = A_k e^{j\phi'_k}$, using the original magnitudes and the newly modulated phases.
   - The Inverse FFT is then applied to convert the frame back into the time domain.
   - The reconstructed frames are combined using an **overlap-add** method. This process, where the overlapping sections of consecutive frames are summed, perfectly reverses the effect of the windowing function, ensuring a smooth transition between frames and yielding a continuous stego-audio signal free of audible artifacts.
   - A final **normalization** step is applied to the entire reconstructed signal to prevent audio clipping before it is saved as a 16-bit integer WAV file.

5. **Data Retrieval (Decoding):** The decoding process must mirror the encoder's parameters precisely. For each frame, it transforms the signal to the frequency domain and examines the phase angle ϕ_k of each frequency bin in the target range. A decision boundary is used to determine the embedded bit:

   - To account for floating-point inaccuracies, a range is checked. If the extracted phase angle falls within a window centered around the shifted value (e.g., between $\pi/4$ and $3\pi/4$), it is decoded as a '1'. Otherwise, it is decoded as a '0'.

4.4 Performance Evaluation Framework

The rationale for this framework is to move beyond anecdotal claims of performance and provide a quantitative, scientific basis for comparing the implemented steganography techniques. The methodology employed is one of empirical stress testing, designed to simulate real-world conditions under which a stego-signal might be altered and to measure the integrity of the hidden data. This was accomplished by creating a suite of attack functions and measuring the outcomes using standard metrics from information theory.

4.4.1 Robustness Attacks

A series of common signal processing corruptions were systematically applied to the stego-signals generated by each steganography method. These "attacks" are designed to test the resilience of the embedded data.

- **Lossy Compression:** This is the most common form of signal degradation. It operates by discarding data that is deemed perceptually redundant, which often includes the high-frequency components where steganographic data may reside.
  - *For Images (JPEG):* The stego-image (saved in a lossless PNG format) was compressed using the JPEG algorithm at various quality levels (e.g., 90, 70, 50). This was implemented using the Python `Pillow` library's `save` method, which allows for precise control over the `quality` parameter.
  - *For Audio (MP3):* The stego-audio (WAV file) was converted to the lossy MP3 format at different bitrates (e.g., 128 kbps, 96 kbps) and then converted back to WAV for analysis. This process simulates a typical audio distribution pipeline.
- **Filtering (Low-Pass):** Filtering is a fundamental signal processing operation that modifies a signal's frequency content. A low-pass filter attenuates high-frequency components.
  - *For Images (Gaussian Blur):* This was implemented by applying a Gaussian blur filter using the `Pillow.ImageFilter.GaussianBlur` function. This operation smooths the image, effectively removing the fine details where data might be hidden.
- **Noise Addition:** This attack simulates the effect of transmitting a signal over a noisy channel, which can introduce random errors.
  - *For Images (Gaussian Noise):* Additive White Gaussian Noise (AWGN) was added to the stego-image. This was implemented using the `scikit-image` library, which adds random values drawn from a Gaussian distribution to each pixel's intensity.

4.4.2 Performance Metrics

To quantify the outcomes of the robustness attacks and the initial quality of the steganography, one primary metric is used.

Bit Error Rate (BER): The BER is the primary metric used to measure the robustness of a steganography technique. It quantifies the integrity of the extracted data by calculating the percentage of bits that were incorrectly recovered after an attack. It is defined as:

$$\text{BER} = \frac{\text{Number of Erroneous Bits}}{\text{Total Number of Transmitted Bits}} \times 100\% \tag{7}$$

A BER of 0% signifies perfect, error-free data recovery. A BER approaching 50% indicates a complete failure of the steganographic channel, where the recovered data is no better than a random guess.

5 Results and Discussion

5.1 Capacity Analysis

The embedding capacity of each steganography method varies based on embedding domain and payload format. LSB offers the highest capacity by altering least significant bits directly in the time domain. DCT and phase coding methods trade some capacity for increased robustness by embedding in frequency components.

5.2 Robustness Analysis (BER Results)

5.2.1 Effect of Lossy Compression

Lossy compression, such as MP3 encoding, distorts frequency components and can severely impact payload recovery. Our DCT method retains moderate robustness under compression, with the BER increasing marginally. Phase coding and LSB methods show higher error rates, with LSB being most vulnerable.

5.2.2 Effect of Noise Addition

Table 1: BER (%) comparisons after Gaussian noise addition

Method	BER (%)
LSB	49.5–50.0
DCT	13.1
Phase Coding	42.9

Gaussian noise introduces random perturbations to the audio waveform, resulting in bit flips and increased BER. The DCT method demonstrates superior robustness, maintaining low BER, whereas LSB and phase coding methods suffer substantial BER elevation.

5.2.3 Effect of Filtering

Table 2: BER (%) comparisons after Low-pass filter attack

Method	BER (%)
LSB	48.5–50.0
DCT	13.5
Phase Coding	41.7

Low-pass filtering attenuates high-frequency components, disrupting embedded frequency domain data and increasing BER. DCT retains notable resilience compared to the other methods.

—

These preliminary results demonstrate that embedding in the frequency domain, specifically via DCT coefficients, offers a balanced tradeoff between payload capacity and robustness to common audio signal distortions.