HideStream: LSB Steganography Tool Suite

HideStream is a comprehensive tool suite that provides functionality for **embedding**, **extracting**, **and detecting steganographic data** using **Least Significant Bit (LSB) steganography**. It supports file types including **PNG**, **BMP**, **WAV**, **and MP3**. This project includes both a **Graphical User Interface (GUI)** and a **Command-Line Interface (CLI)** for flexible user interaction.

Features

Graphical User Interface (GUI)

- Interactive menus for performing steganography on different file types.
- Integrated real-time console for user feedback.
- Progress bars for long-running tasks.
- File dialogs for user-friendly file selection and output path configuration.

Command-Line Interface (CLI)

- Structured subcommands for automating and scripting operations.
- Supports options for LSB counts, file paths, and more.
- Works seamlessly across supported file types.

Supported Operations and File Types

Sub-Tools

WavSteg:

- o Embeds and extracts hidden data within WAV audio files.
- Supports configurable LSB counts to balance between data capacity and audio quality.

LSBSteg:

- Handles image steganography for PNG and BMP files.
- Maintains image quality while optimizing storage capacity.

MP3Steg:

- Embeds data in MP3 files with minimal quality degradation.
- Extracts hidden data using custom delimiters for separation.

StegDetect:

- Analyzes and visualizes least significant bits in images.
- Scans for hidden data in PNG, BMP, WAV, and MP3 files.

Requirements

- Python 3.8+
- Required Python packages:
 - o Pillow

- o Click
- o wave
- tkinter (built-in with Python)

Install dependencies:

```
pip install pillow click
```

How to Use

GUI

Run the GUI with:

```
python gui.py
```

CLI

Run the CLI with:

```
python cli.py --help
```

Example CLI Commands

- MP3 Steganography:
 - Hide data: python cli.py mp3steg -h -i input.mp3 -s secret.txt -o output.mp3
 - Extract data: python cli.py mp3steg -r -i input.mp3 -o extracted.txt
- Image Steganography:
 - Hide data: python cli.py steglsb -h -i input.png -s secret.txt -o output.png -n
 2
 - Extract data: python cli.py steglsb -r -i input.png -o extracted.txt -n 2
- WAV Steganography:
 - Hide data: python cli.py wavsteg -h -i input.wav -s secret.txt -o output.wav -n
 2
 - Extract data: python cli.py wavsteg -r -i input.wav -o extracted.txt -n 2 -b 1000
- LSB Detection:
 - Detect LSB changes: python cli.py stegdetect -i input.png -n 2

File Structure

```
.

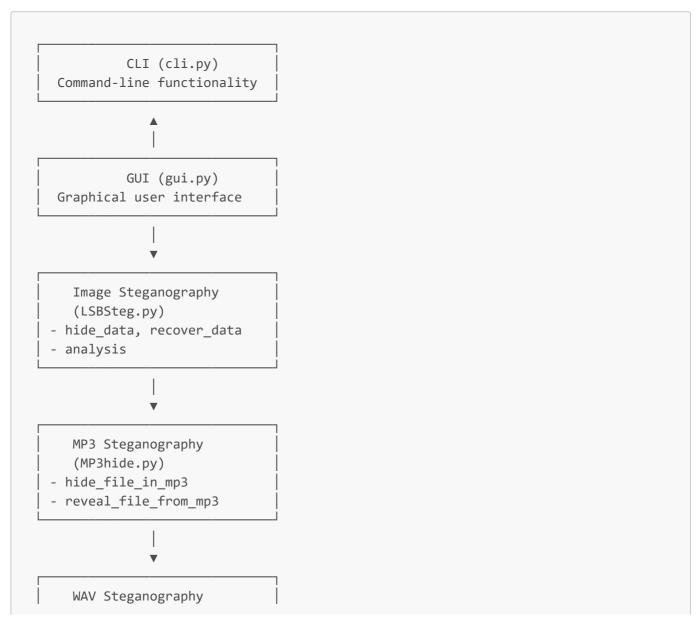
├── gui.py  # GUI implementation
├── cli.py  # Command-line interface
├── LSBSteg.py  # Image steganography module
├── WavSteg.py  # WAV steganography module
├── MP3hide.py  # MP3 steganography module
├── StegDetect.py  # LSB detection module
├── README.md  # Documentation
```

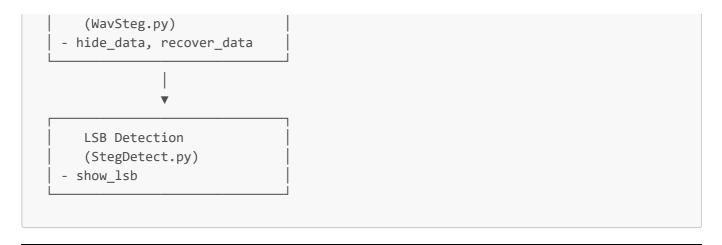
Documentation: Architecture

Overview

HideStream is modular, integrating distinct sub-tools for each supported file type into a unified interface. This architecture ensures scalability and maintainability while allowing extensions for additional file types in the future.

Block Diagram





Security Analysis

Findings

1. Image Steganography:

• Effective for hiding small to medium amounts of data with negligible quality loss.

2. WAV Steganography:

• Works well for lossless formats; vulnerable to lossy compression.

3. MP3 Steganography:

• Simple appending mechanism makes it prone to detection via file inspection.

4. LSB Detection:

• Provides insight into hidden data but cannot decode without parameters.

Strengths

- Modular Architecture: Simplifies development and extension.
- **Usability**: GUI for ease of access; CLI for automation.
- **Versatility**: Supports multiple media types and steganographic operations.

Weaknesses

- Compression Vulnerabilities: Lossy compression in audio can corrupt embedded data.
- **Encryption**: Hidden data is not encrypted, which may expose sensitive information.

Lessons Learned

- Combining steganography with encryption ensures robust data protection.
- A modular structure is ideal for scalability.
- User feedback is crucial for designing intuitive tools.

Acknowledgments

Group 38 Members:

- Muhammad Abdullah (21L-5288)
- Saad Ali (21L-7577)
- Zeeshan Hamid (21L-1876)