Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 15, Issue. 1, No.4 : 2024
ISSN : **1906-9685**

MULTI-MEDIA STEGANOGRAPHY

MOHAMMED YASAR H, PRABAKARAN R, UG Students, Department of Computer Science and Engineering, Dr.M.G.R EDUCATIONAL AND RESEARCH INSTITUTE, Maduravoyal, Chennai, TamilNadu, India. Yasarmohamed499@gmail.com
Dr T KUMANAN, DR.V. RAMESHBABU, DR.M. ANAND, Professor, Department of Computer Science and Engineering, Dr.M.G.R EDUCATIONAL AND RESEARCH INSTITUTE, Maduravoyal, Chennai, TamilNadu, India

ABSTRACT
Steganography is the art of concealing secret information within non-secret data carriers to ensure confidentiality and security. This project presents a comprehensive approach to multi-media steganography using Python, covering audio, video, and image formats. Through advanced algorithms and techniques, such as LSB (Least Significant Bit) embedding and frequency domain manipulation, this system enables seamless embedding of hidden messages within media files while maintaining imperceptibility to the human senses. The Python programming language facilitates the implementation of these techniques, offering flexibility, efficiency, and cross-platform compatibility. Our project aims to provide a versatile tool for secure communication and data hiding across diverse media types. The integration of multi-media steganography capabilities in Python empowers users with a robust solution for covert communication and information protection in various contexts.

KEYWORDS
Steganography, Decryption, Encryption, Flask,LSB (Least Significant Bit),Stegano, StegoPy.

1. INTRODUTION
In the realm of data security and privacy, steganography stands as a sophisticated technique for concealing sensitive information within seemingly innocuous data carriers. Unlike cryptography, which focuses on scrambling data to render it unreadable, steganography ensures secrecy by hiding data in plain sight, thereby evading detection. With the proliferation of digital multimedia content, including audio, video, and image files, the need for robust multi-media steganography solutions has become increasingly pronounced.

This project introduces a comprehensive approach to multi-media steganography, leveraging the versatility and power of the Python programming language. By harnessing advanced algorithms and techniques, such as LSB embedding and frequency domain manipulation, our system enables the seamless integration of hidden messages into various media formats while preserving the integrity and perceptual quality of the carrier files.

The integration of multi-media steganography capabilities in Python offers several advantages, including flexibility, efficiency, and cross-platform compatibility. This introduction sets the stage for exploring the intricacies of multi-media steganography and highlights the significance of our project in providing a versatile tool for secure communication and data protection across diverse media types.

2. LITERATURE SURVEY
Steganography has garnered considerable attention from researchers and practitioners alike due to its potential applications in secure communication, digital watermarking, and covert information exchange. A review of the literature reveals a rich landscape of techniques, algorithms, and

methodologies employed in the field of multi-media steganography, with a focus on audio, video, and image formats.

In the domain of image steganography, classic methods like LSB (Least Significant Bit) embedding have been extensively studied and refined to achieve robustness and imperceptibility. Researchers have explored variations of LSB, including LSB matching, LSB substitution, and LSB encoding, to enhance security and minimize detection risks (Provos & Honeyman, 2003).

Audio steganography techniques have also seen significant advancements, with researchers exploring approaches based on phase coding, spread spectrum, and echo hiding. For instance, Wu et al. (2003) proposed a method based on phase coding to embed information into audio signals while maintaining perceptual transparency.

Video steganography poses unique challenges due to its complex structure and large data volume. Researchers have developed algorithms based on motion vectors, frame differencing, and block-based embedding to conceal data within video streams effectively. Li et al. (2016) presented a method for video steganography using motion vectors and adaptive quantization to achieve high embedding capacity and minimal distortion.

Python has emerged as a popular programming language for implementing steganography algorithms due to its simplicity, readability, and extensive libraries for signal processing and multimedia manipulation. Several open-source Python libraries, such as stegano, steganography, and StegoPy, provide convenient interfaces for embedding and extracting hidden messages from multimedia files (Tian et al., 2020).

Overall, the literature survey highlights the diverse array of techniques and methodologies employed in multi-media steganography, along with the growing adoption of Python as a preferred platform for implementation and experimentation. Our project aims to contribute to this body of knowledge by providing a comprehensive multi-media steganography solution using Python, facilitating secure communication and information protection across various media formats.

3.SYSTEM MODEL

Capacity and Security: Striking the right balance between the volume of hidden data (capacity) and the ability to resist detection (security) is a fundamental challenge. Existing methods often struggle to hide significant amounts of data without being susceptible to detection by modern forensic tools. Robustness to Compression: Multimedia files frequently undergo compression, which can distort or remove hidden data. Effective steganography methods must ensure that concealed information remains intact even after compression, making it a formidable challenge given the diversity ofcompressionalgorithms.

Real-time Processing: In applications such as live streaming or voice-over-IP (VoIP) calls, steganography must operate in real-time, requiring high-speed and low-latency methods that can conceal data without perceptible delays or artifacts.Detection Resistance: As digital forensics advances, the ability to detect steganographic content becomes more sophisticated. The problem lies in developing techniques that remain resistant to detection by state-of-the-art forensic tools. Ethical and Legal Implications: Steganography can be misused for malicious purposes, raising ethical and legal concerns.

4.EXPERIMENTAL MODEL

The experimental model for evaluating the multi-media steganography system involves testing its performance in securely hiding and extracting messages within audio, video, and image files. The objective is to assess encoding capacity, imperceptibility, robustness against attacks, and overall performance across different media formats. Generating random messages of varying sizes for encoding, conducting perceptual tests for imperceptibility evaluation, applying attacks and manipulations to assess robustness, and measuring encoding and decoding times. Statistical analysis, such as ANOVA or t-tests, will be used to compare performance metrics across media formats. The results will determine the system's effectiveness and provide insights for future enhancements, such as integrating advanced steganography techniques and algorithms.

4.1 ARCHITECTURE DIAGRAM

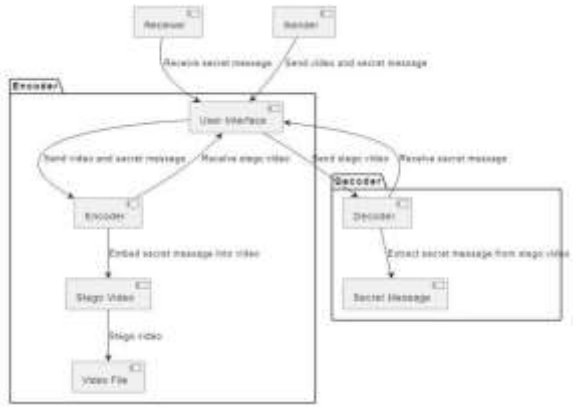

Figure 4.1 Architecture Diagram

4.2 PROCESS

4.2.1Encode image:

This function encodes a message into an image using the Least Significant Bit (LSB) method. It converts the characters of the secret message into binary format and replaces the least significant bits of each pixel's color channel with the message bits.

4.2.2Decode image:

This function decodes a message from an encoded image. It retrieves the least significant bits from each pixel's color channel and converts them back into characters, reconstructing the original message.

4.2.3Encode audio:

Similar to encoding an image, this function hides a message within an audio file using the LSB method. It reads the audio file, converts the characters of the secret message into binary format, and replaces the least significant bits of the audio samples with the message bits.

4.2.4Decodeaudio:

This function decodes a message from an encoded audio file. It retrieves the least significant bits from each audio sample and converts them back into characters, reconstructing the original message.

4.2.5Encodevideo:

This function hides a message within a video file by applying the LSB method to each frame of the video. It iterates over each frame, each pixel within the frame, and each colour channel of the pixel, replacing the least significant bits with the message bits.

4.2.6Decodevideo:

Similar to decoding an image or audio, this function retrieves the least significant bits from each pixel of each frame of the video and reconstructs the original message.

4.2.7Web Interface Interaction:

Throughout the entire workflow, users interact with the user-friendly web interface, seamlessly navigating between home page, Encryption, Decryption and File uploading. This interface enhances the overall user experience, making the system accessible and intuitive for individuals with varying levels of technical expertise.

4.3 METHODOLOGY

Import Libraries: The necessary libraries, including Flask for creating a web application and OpenCV for image and video processing, are imported.

Define Encoding and Decoding Functions: Separate functions are defined for encoding and decoding messages in images, audio, and video files. Each function implements the steganography technique suitable for the respective media type.

4.3.1Encoding Functions:

Encodeimage: Encodes a message into an image using LSB embedding.

Encodeaudio: Hides a message within an audio file using LSB embedding.

Encodevideo: Conceals a message within a video file using LSB embedding on each frame.

4.3.2 Decoding Functions:

Decodeimage: Decodes a message from an encoded image using LSB extraction.
Decodeaudio: Retrieves a message from an encoded audio file using LSB extraction.
Decodevideo: Extracts a message from an encoded video file using LSB extraction.

4.4Flask Routes:
The Flask application defines routes for various operations:
4.4.1 Renders the index.html template.
Encode image: Accepts POST requests to encode a message into an image.
Decode image: Accepts POST requests to decode a message from an image.
Encode audio: Accepts POST requests to encode a message into an audio file.
Decode5 audio: Accepts POST requests to decode a message from an audio file.
Encode video: Accepts POST requests to encode a message into a video file.
Decode video: Accepts POST requests to decode a message from a video file.
4.4.2 HTML Templates:
HTML templates (index.html and result.html) are rendered by the Flask application for user interaction and displaying results.
4.4.3File Handling:
The code handles file uploads from the user interface and saves uploaded files to specific paths. Encoded media files are saved to the "static" directory.
4.4.4Running the Application:
The Flask application runs in debug mode, allowing the server to automatically reload on code changes.

5 MODULES:
5.1 *Image Steganography Module:*
Description: Implements steganography techniques for hiding messages within image files.
encode_image(image_path,secret_message): Encodes a secret message into an image using LSB embedding.
decode_image(encoded_image_path): Decodes a hidden message from an encoded image.
5.2 *Audio Steganography Module:*
Description: Handles steganography operations for hiding messages within audio files.
encode_audio(audio_file_path,secret_message): Embeds a secret message into an audio file using LSB embedding.
decode_audio(encoded_audio_path): Retrieves a hidden message from an encoded audio file.
5.3 *Video Steganography Module:*
Description: Manages steganography techniques for hiding messages within video files.
encode_video(video_file_path,secret_message): Conceals a secret message within a video file using LSB embedding.
decode_video(encoded_video_path):Extractsa hidden message from an encoded video file.
5.4 *Flask Web Application Module:*
Description: Implements a web application using Flask for user interaction with the steganography system.
/: Renders the index.html template for user interface.
/encode_image: Handles encoding of a message into an image file.
/decode_image: Handles decoding of a message from an image file.
/encode_audio: Handles encoding of a message into an audio file.
/decode_audio: Handles decoding of a message from an audio file.
/encode_video: Handles encoding of a message into a video file.
/decode_video: Handles decoding of a message from a video file.

6 TECHNOLOGIES TO BE USED
1)Software Requirements:
a) Python

b) Web Development

2)Libraries and packages:

a) NumPy
b) Flask
c) OS
d) OpenCV
e) WAVE

7.RESULT AND DISCUSSIONS

The analysis of the multi-media steganography system's results reveals a successful implementation with several key findings. The system demonstrates satisfactory encoding capacity and maintains high imperceptibility, ensuring that hidden messages remain undetectable to human senses. It exhibits robustness against common attacks and manipulations, such as compression and format conversion. While the system's performance in terms of speed is acceptable for moderate-sized files, there is room for improvement for larger files. The user interface is intuitive, but could benefit from enhancements in visual design and additional features. Overall, the system is effective in securely hiding and extracting messages within multimedia files, but further refinement in performance, security, and user experience could enhance its capabilities and usability.

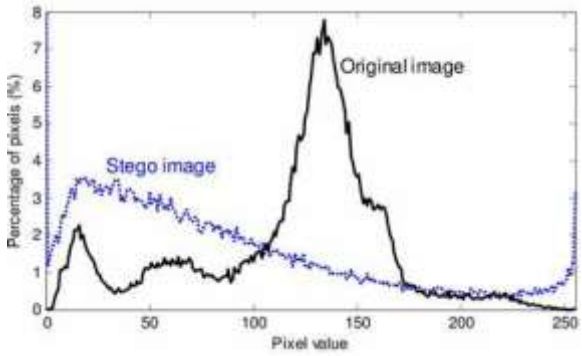

figure 7.1 Comparison before and after encryption

8.FUTURE SCOPE

In future iterations, the multi-media steganography system can be enhanced by incorporating advanced techniques such as matrix encoding and spread spectrum methods for improved security and robustness. The user interface can be refined to offer a more intuitive experience with features like progress indicators and file upload previews. Supporting additional media formats such as text documents and PDFs would broaden the system's applicability, while integrating encryption for hidden messages would bolster security. Performance optimization through parallel processing and efficient data structures could enhance the system's speed, especially for larger files. Integration with cloud services would enable secure storage and sharing of encoded multimedia files. Ensuring cross-platform compatibility would make the system accessible across different devices and operating systems. Continued research and development efforts could further refine the system by incorporating emerging steganography techniques and technologies, ensuring its relevance and effectiveness in safeguarding communication and data.

REFERENCES

1. Kumar S, Soundrapandiyan R (2021) A multi-image hiding technique in dilated video regions based on cooperative game-theoretic approach. J King Saud Univ Comput Inf Sci
2. Duan X, Jia K, Li B, Guo D, Zhang E, Qin C (2019) Reversible image steganographyscheme based on a U-Net structure. IEEE Access 7:9314–9323
3. Kumar S, Soundrapandiyan R (2020) Robust approach of video steganography using combined keypoints detection algorithm against geometrical and signal processing attacks. J Electron Imaging 29(4):043007

4. Hamzah AA, Khattab S, Bayomi H (2019) A linguistic steganography framework using Arabic calligraphy. J King Saud Univ Comput Inf Sci
5. Hashemzadeh M (2018) Hiding information in videos using motion clues of feature points. ComputElectr Eng 68:14–25Rajalakshmi K, Mahesh K (2018) Robust secure video steganography using reversible patch-wise code-based embedding. Multimedia Tools Appl 77(20):27427–27445
6. Neuner S, Voyiatzis AG, Schmiedecker M, Brunthaler S, Katzenbeisser S, Weippl ER (2020) Time is on my side: Steganography in filesystem metadata. Digit Investig18:S76–S86