Video Steganography Using Least Significant Bit (LSB) Insertion for Secure Communication In UDP

Senarathne D G S K , Lakshan M H P , Fonseka W F I S

May 2025

Video Steganography Using Least Significant Bit (LSB) Insertion for Secure Communication In UDP

A thesis

Presented to the

Department of Information and Communication Technology

Faculty of Technology

University of Sri Jayewardenepura

In partial fulfilment

Of the Requirements for the

Bachelor of Information and Communication Technology Honors Degree

By

Senarathne D G S K

Lakshan M H P

Fonseka W F I S

May 2025

Declaration

Declaration by the Candidate

We affirm that we conducted the research presented in this thesis at the Faculty of Technology, University of Sri Jayewardenepura, Sri Lanka. This thesis does not include any content that has been previously published or authored by someone else. The document has not been presented for any academic qualification at this university or any other educational institution.

Senarathne D G S K
ICT – 20934

Lakshan M H P
ICT – 20879

Fonseka W F I S
ICT - 20903

Certification of the above statement by the Supervisor

I hereby certify that I have supervised this thesis.

Mr. Bathiya Seneviratne
Lecturer, Faculty of Technology
University of Sri Jayewardenepura
bathiyasenevirathna@gmail.com
Supervisor

Mr Surath AyodhyaKahandawala
Lecturer, Faculty of Technology
University of Sri Jayewardenepura
surathayodhya@sjp.ac.lk
Co-supervisor

Dedication

This work is dedicated to the culmination of our collective journey, reflecting the unwavering dedication, teamwork, and growth that shaped this research. Our project, titled **"Video Steganography Using Least Significant Bit (LSB) Insertion for Secure Communication,"** represents the blend of passion, creativity, and hard work that brought our vision into reality.

We are profoundly grateful for the lessons learned, the challenges overcome, and the support received throughout this journey. Special thanks to our families and friends for their constant encouragement and belief in us. Their support has been the foundation of our persistence and success.

We also extend our heartfelt appreciation to our mentors, especially our internal supervisor **Mr. Bathiya Seneviratne**, and our external supervisor **Mr. Surath Kahandawala**, for their invaluable guidance and wisdom, which were instrumental in completing this research.

May this accomplishment serve as a reminder that perseverance, collaboration, and a shared passion for knowledge can lead to meaningful success. Here’s to embracing challenges, fostering growth, and exploring the endless possibilities that the future holds.

Acknowledgement

Completing this research project has been a journey marked by challenges, learning, and personal growth. We would like to extend our deepest gratitude to the following individuals and groups, without whom this achievement would not have been possible.

First and foremost, we express our heartfelt appreciation to our Internal Supervisor, Mr. Bathiya Seneviratne, for his steadfast guidance, mentorship, and expertise throughout this research journey. Your insightful feedback, patience, and encouragement have been crucial in shaping this project and fostering our academic development.

We are equally grateful to our External Supervisor, Mr. Surath Kahandawala, for his invaluable support and expert advice. Your contributions have significantly enriched the quality and scope of our research.

We would also like to acknowledge the support of the Faculty of Technology, University of Sri Jayewardenepura, for providing the resources and environment necessary to complete this project successfully.

To our families and friends, your unwavering support, love, and belief in us have been our greatest sources of motivation. Your sacrifices and encouragement have enabled us to reach this milestone, and we are deeply thankful for your presence throughout this journey.

Special thanks to our dedicated team members for their collaboration and commitment:

- D.G. Supun Kalana
- M.H.P. Lakshan
- W.F.I.S. Fonseka

Your combined efforts and shared dedication have made this accomplishment possible.

Lastly, we would like to express our sincere gratitude to everyone who, in their unique ways, supported, encouraged, and believed in us throughout this research journey. Your contributions, whether small or significant, have left a lasting impact on both our academic and personal growth.

Thank you all for being part of this journey and helping us reach this milestone. Your support has truly been the driving force behind our success.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

LSB	Least Significant Bit
UDP	User Datagram Protocol
AVI	Audio Video Interleave
VP8	VideoCodec VP8
H.264 (AVC)	Advanced Video Coding
PNG	Portable Network Graphics
MP4	MPEG-4 Part 14
PSNR	Peak Signal-to-Noise Ratio
GUI	Graphical User Interface
OpenCV	Open Source Computer Vision Library

NumPy	Numerical Python
WebRTC	Web Real-Time Communication
RGB	Red Green Blue
DCT	Discrete Cosine Transform
DWT	Discrete Wavelet Transform
ASCII	American Standard Code for Information Interchange
EOF	End of File (used as an end marker in binary strings)
HLSB	Hash-based Least Significant Bit
PIL	Python Imaging Library
WSGI	Web Server Gateway Interface
ORM	Object Relational Mapper
HTTP	Hypertext Transfer Protocol

Abstract

In the modern age, Secure communication over public networks is very important. This research focus improving the security of data transmission in videos by hiding data in the Least Significant Bit (LSB) method for communication through User Datagram Protocol (UDP). The project aims to embed plain text and images within video frames with no impact on quality and security. Unlike how data is usually handled in traditional systems, this system uses various improved techniques to ensure data integrity and detectability.

This proposed system uses several Python libraries and OpenCV to manipulate video frames using AVI, VP8, and H.264 (AVC) formats for experimentation. To ensure that communication is robust and secure communication, introduced an additional layer to the UDP protocol. This improvement includes segmenting data, sequence numbering, data redundancy, hashing, and a passkey-based encoding and decoding technique. The modified UDP layer significantly improves data integrity and reliability during transmission over public networks.

Key findings show that the proposed system meets the requirements for secure and undetectable data embedding while maintaining the quality of the video. According to the experimental results indicate that encoding and decoding using a passkey not only secures the hidden data but also makes the exchange of information appear unremarkable. This solution is particularly effective for secure messaging during public live streaming.

This research makes the way for future improvements in video steganography by introducing a secure, reliable, and codec-flexible system for data hiding and transmission over the network. This applications can be uses for secure messaging, covert data embedding during live streams and video watermarking.

Chapter 1: Introduction

1.1 Overview of the Proposed Study

Digital communication is important in this era. The need for secure and covert data transmission has become extremely important. Steganography, that is called the science of hiding information within ordinary data, has become important for guaranteeing privacy and confidentiality when communicating in public channels (Rani S, 2022). Video steganography exploits the high data capacity and temporal variability of video files to conceal confidential data inside them in a manner imperceptible to the human eye(Cheddad, 2010)). In this study, an examination is conducted on developing an effective and secure video steganography system using Least Significant Bit (LSB) embedding for transmission over the unreliable User Datagram Protocol (UDP). The diagram below illustrates the process of embedding secret data into a video stream using Least Significant Bit (LSB) insertion for secure communication over the UDP protocol.

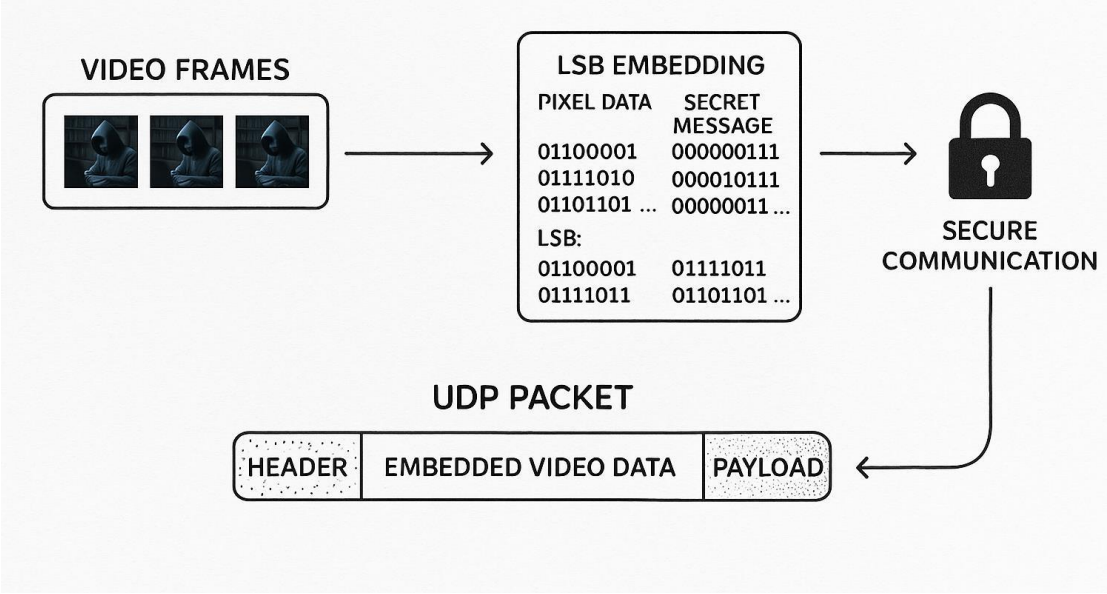


Figure 1: LSB-based video steganography in UDP communication

1.2 Project Scope

The scope of this research is to design and implement an advanced security-enhanced video steganography system that utilizes LSB insertion to securely embed data within video frames, verifying both data confidentiality and reliable transmission over User Datagram Protocol (UDP).

1.3 Background and Motivation

With the increasing capacity of sensitive information exchanged over public networks, the traditional cryptographic methods are often not suitable for scenarios requiring undetectable communication. Video steganography offers an improved solution by embedding secret data within video content while concealing the existence of the communication itself (Rani, 2022). UDP, widely used for real-time applications due to its low latency, poses significant challenges for reliable data transmission, as it lacks built-in mechanisms for error correction and packet ordering. This research is motivated by the need to overcome these challenges and enable secure, robust, enhanced, and covert data transmission in practical, real-world environments such as public live streaming (Johnson, 2023).

1.4 Rationale and Justification

The growing need for secure and reliable covert communication channels in the digital age, especially over public networks where traditional encryption alone is insufficient, underscores the rationale for this research. Existing video steganography methods, particularly those utilizing LSB insertion, face challenges due to the unreliable nature of UDP transmission, including packet loss, out-of-order delivery, and limited error detection (Gupta & Sharma, 2020) .This research seeks to address these limitations by introducing redundant data embedding, sequence numbering, and encryption, thereby improving both the security and reliability of hidden data transmission. The project targets a critical gap in current steganographic solutions by focusing on UDP, aiming to provide a practical and robust framework for secure communication in scenarios such as public live streaming and digital forensics (Islam, et al., 2022).

1.5 Research Problem

Existing video steganography methods, particularly those based on LSB insertion, that is also face several limitations when applied to UDP-based transmission. These include vulnerability to packet loss, out of order delivery, limited error detection, and restricted data hiding capacity (Gupta & Sharma, 2020).

The main research question addressed in this research is:

Can reliability issues, security concerns, limited error detection, and restricted data hiding capacity in UDP-based video steganography be mitigated by developing an advanced improved method to enhance its performance?

1.6 Proposed Solution

To overcome these challenges, this research proposes a novel modern video steganography technique that combines redundant data embedding and data fragmentation with sequence numbering. By embedding data redundantly across multiple frames and assigning sequence numbers to each fragment. That method ensures reliable reconstruction of hidden data, even in the presence of packet loss or out of order delivery inherent in UDP transmission. The integration of encryption further improves the security of the embedded data, making unauthorized extraction exceedingly difficult. This hybrid approach significantly improves the robustness, reliability, and error resilience of video based steganographic communication. Below illustration shows the flow of the proposed solution.

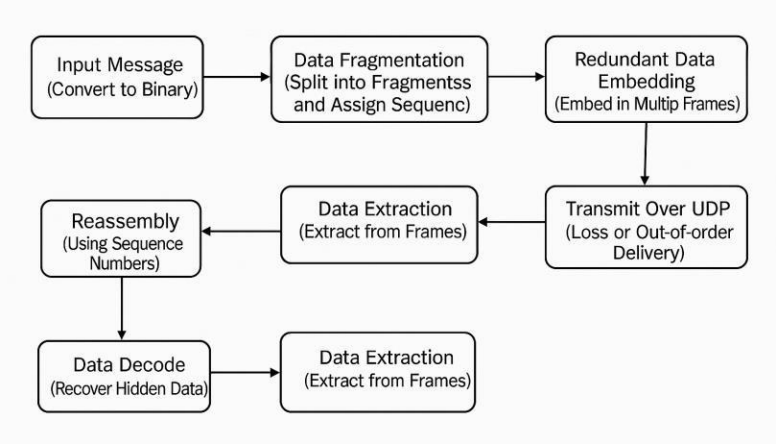


Figure 2 : Proposed solution

1.7 Research Objectives

- To develop an efficient Lease Significant Bit based video steganography method for embedding data within video frames transmitted over User Datagram Protocol (UDP).

- To improve the security of hidden data by integrating encryption or additional security techniques to prevent unauthorized access during transmission over the network.

- To improve steganography based data communication over UDP transmission by addressing packet loss, error detection, and integrity checks.

- To conduct a comprehensive literature review on existing LSB-based video steganography techniques, Systems, and identify their limitations.
- To systematically evaluate the proposed algorithm using performance metrics such as robustness, imperceptibility, and transmission reliability.

- To develop an application for LSB insertion and extraction, the practical viability of the improved technique in secure communication scenarios.

1.6 Conceptual, Theoretical, and Practical Significance

1.6.1 Conceptual Significance:

This research improves the conceptual understanding of secure communication by integrating steganography and modern network protocols such as UDP. It bridges the gap between data hiding techniques and real world network limitations. That offers an integrated view of how covert data can be embedded and transmitted reliably in dynamic digital environments (Singh and Kumar, 2023).

1.6.2 Theoretical Significance:

The project contributes to the theoretical foundation of video steganography by proposing and validating a hybrid approach that combines LSB insertion, data fragmentation, sequence numbering, and encryption. This approach not only contains the limitations of existing spatial domain methods. Also introduces new techniques for error correction and security, thereby enriching the body of knowledge in multimedia security and covert communication (Katzenbeisser & Petitcolas, 2000)

1.6.3 Practical Significance:

The research provides a flexible and robust framework for secure messaging and live streaming that can be done practically. By leveraging widely available tools such as Python, OpenCV, and improved UDP protocols, the system is accessible and adaptable for real-world applications. The integration of user friendly interfaces and codec flexibility ensures that the solution can be adopted in various professional and academic settings. Supporting secure data exchange without compromising video quality or raising suspicion (Islam, et al., 2022) (Thakur, et al., 2019)

1.7 Expected Outcomes/Anticipated Results

The expected outcomes of this project are threefold:

1. **Enhanced Security and Reliability:**

   The proposed system is anticipated to deliver secure and undetectable data embedding in video frames, with improved reliability over UDP due to redundant embedding and sequencing. This will offer reliable data recovery even in the presence of packet loss or network disturbances, making the system suitable for real time and public network environments.

2. **Advancement of Theoretical Understanding:**

   The research will contribute new insight to the field of steganographic integration into network protocols. That demonstrates how hybrid technologies can overcome the limitations of unreliable channels of transmission. This will improve the theoretical understanding of secure, covert communication and will inspire subsequent advances in the field.

3. **Practical Contributions to Society:**

   By providing a codec flexible and user friendly platform, the project will make secure messaging and data embedding in live video streams possible, to the benefit of applications across domains of cybersecurity, digital forensics, and online communication. The anticipated results include minimal impact on video quality, high data integrity, and a framework that can be readily adapted for various real world applications, thereby contributing to the betterment of digital privacy and security for individuals and organizations.

Chapter 2: Literature Review

Steganography on video, the act of concealing information in video files, is now well known as a safe and covert means of communication in the modern era. Among various approaches, Least Significant Bit (LSB) insertion is generally favored because of the ease of use and the capacity to insert data with little noticeable distortion. However, the application of insecure protocols such as UDP for the transmission of steganographic data introduces new challenges, including packet loss, out-of-order arrival, limited error detection, and security vulnerabilities (Yadav & Kumar, 2018). This literature review presents the background of LSB-based video steganography, advantages and limitations, and newer developments in improving security and reliability, especially for UDP transmission (Ghosal, et al., 2020).

2.1 Introduction to Steganography and LSB Techniques

LSB insertion is embedding secret data into the least significant bits of pixel intensities within video frames. It is well liked for its high payload and imperceptibility because slight changes in pixel intensities are usually not perceivable by the human vision system. Different researchers have proposed adjustments to the primary LSB method:

- (Dasgupta, et al., 2012)put forward a technique by which confidential data is divided and distributed amongst the RGB color channels, increasing payload but not necessarily visual quality. They created their HLSB method, which accommodated increased values of PSNR compared to standard LSB, thereby suggesting better-quality video.
- (Bhole & Patel, 2014) utilized the first frame as an index to control data allocation in subsequent frames, but the method remained vulnerable due to the utilization of the spatial domain, which is not robust against compression and processing of signals

- (Paul, et al., 2014) proposed the embedding of data within frames selected based on histogram variation and scene movement, enhancing security using random allocation of sites of embedding.

Despite these improvements, spatial domain LSB techniques are usually susceptible to common attacks, compression, and noise that limit their power.

2.2 Review of LSB-Based Video Steganography Techniques

LSB (Least Significant Bit) steganography is a simple yet common technique of concealing information in multimedia media. In video steganography, it works by modifying the least significant bits of pixel values of video frames to hide secret data, sometimes without resulting perceptual changes in visual quality. Different approaches have, over the years, been developed to improve LSB's security, robustness, and imperceptibility.

2.2.1 Traditional LSB-Based Techniques

These are the simplest and earliest LSB steganography applications. They aim for embedding bits in pixel value of video frames directly, primarily linearly or sequentially. There has been some early foundation research on LSB-based steganography for digital media:

- (Dasgupta, et al., 2012) suggested a method named Hash-Based LSB (HLSB), which enhances traditional LSB steganography through distributing secret message bits over the RGB channel of video pixels by using a hash function. This randomized (yet deterministic) distribution increases security and reduces predictability, rendering it more difficult for unauthorized parties to recover hidden information. Using all three-color channels also increases payload size. The approach is

not built on the assumption of reliable transmission, and therefore, it contains no error recovery methods built into it, thereby susceptible to information loss when implemented over unreliable transport like UDP

- (Paul, et al., 2014) proposed an enhancement to LSB embedding in which frames from a video are selected based on histogram change and scene cuts. The technique embeds non-uniformly by embedding data only in the frames where there is significant motion or visual noise, thereby making concealed content less recognizable and harder to extract through steganalysis. Though this technique improves security and stealth by avoiding deterministic embedding patterns, it is also based on ordered, stable transmission. As a result, it lacks error correction or data sequencing capability, which makes it an impediment to application within UDP-based communication**.**

2.2.2 Advanced LSB Techniques

These techniques represent improvements over the traditional methods. They include levels of structure or logic to enhance security, imperceptibility, or patterns of distribution.

- (Paul, et al., 2014) Expanding on their earlier work, Paul et al. developed a more dynamic and adaptive embedding approach that analyzes scene complexity, motion levels, and histogram variations to select optimal frames for steganographic embedding. This method enhances security by focusing on high-entropy or fast-changing scenes. In these areas, the subtle changes caused by data embedding are less likely to be detected by human observers or steganalysis tools.

- (Bhole & Patel, 2014) proposed an LSB-based steganography method that utilizes the first frame of a video as an index reference to determine where secret information is to be embedded across other frames. This approach adds a level of structure during the embedding. Even though this

indexing method introduces a layer of concealment and enhances data placement organization more than blind embedding, the technique is still fundamentally spatial domain-based. As such, it remains highly vulnerable to compression artifacts, resizing, and signal processing and lacks any means of recovering data if some frames are lost, which is a very critical failure in UDP-based streaming environments.

2.2.3 Hybrid and Secure Approaches

To overcome the limitations of traditional LSB-based steganography, particularly its vulnerability to unauthorized access and transmission loss, researchers have proposed hybrid techniques to combine steganography with cryptographic algorithms and data integrity mechanisms. One such example is the work by (Hossain, et al., 2019), who introduced a model that blends LSB embedding with XOR operations and AES (Advanced Encryption Standard) encryption. This technique ensures that even if the embedded data is discovered, it cannot be interpreted without the encryption key. Additionally, the use of XOR operations provides an extra layer of Concealment, making pattern detection more difficult. However, the system still relies on static pixel selection, embedding bits in a fixed manner for making it predictable to advanced steganalysis. More significantly, it lacks sequencing or redundancy features which are required for maintaining the integrity of hidden data during UDP-based communication, where packet loss and reordering are common.

Similarly, (Alanzy, et al., 2023) proposed the use of hybrid encryption algorithm methods such as combining symmetric (e.g., AES) and asymmetric (e.g., RSA) encryption with LSB steganography to improve message confidentiality and protection against interception. This approach is particularly effective in hiding sensitive data in static media like images, whose transmission environment does not

change. However, the study does not address real time constraints or performance issues in streaming environments. For instance, it does not address how the computational overhead of hybrid encryption would impact video encoding and decoding during live UDP-based transmissions.

As a result, while these methods significantly enhance security and they are not well-suited for high-throughput or real-time video steganography. In such cases, efficiency, synchronization, and packet-level reliability become critical challenges that the research aims to address directly.

2.3 Comparative Analysis of Steganographic Methods

LSB is the most popular technique due to its simplicity and high capacity. Alternative methods such as DCT, DWT, and hybrid approaches offer improved robustness at the cost of increased complexity. Frequency domain methods are less susceptible to compression and noise, but they may reduce payload capacity and increase computational requirements.

Some researchers have moved beyond spatial domain LSB into frequency domain techniques:

- (Dalal & Juneja, 2020) evaluated biorthogonal wavelets for video steganography using DWT, yielding better robustness against compression but increasing computational complexity.
- (Banik, 2019) reviewed advances in DCT/DWT-based techniques, emphasizing resilience to media transformation at the cost of real-time processing speed.

2.4 Security and Reliability Enhancements

Recent research has focused on addressing the security and reliability shortcomings of LSB-based steganography, particularly for transmission over UDP:

- Hybrid Algorithms: (Hossain, et al., 2024) combined LSB with XOR operations, spiral pixel selection, and AES encryption to enhance both security and imperceptibility. Their system leverages random frame selection and advanced encryption to make unauthorized detection and extraction more difficult. However, limitations remain, such as static pixel selection and restricted data capacity per frame.
- Error Detection and Correction: Studies emphasize the need for error correction and integrity verification mechanisms, especially when transmitting over UDP, which lacks built-in reliability. Techniques such as redundant embedding, data fragmentation, and sequence numbering have been suggested to mitigate packet loss and out-of-order delivery.

2.5 Limitations of Existing Systems

Despite the extensive exploration of LSB-based video steganography techniques, the existing literature consistently reveals several inherent limitations that hinder their effectiveness, particularly when deployed in real-world and networked environments.

Firstly, one of the most significant drawbacks is the vulnerability of spatial domain LSB techniques to steganalysis and signal distortion. Since traditional LSB methods directly manipulate pixel values, any post-processing operations such as video compression (e.g., MPEG or H.264), resizing, or format conversion can easily corrupt the embedded data. Additionally, these alterations are more detectable using

statistical steganalysis techniques that analyze patterns in pixel values, rendering the method susceptible to attacks that aim to uncover or destroy hidden content (Dasgupta, et al., 2012).

Secondly, many existing systems provide only basic data concealment without incorporating robust security features. A large number of LSB implementations simply embed the raw message into pixel bits without any encryption. This means that even if the presence of hidden data is not immediately obvious, once detected, it can be easily extracted and interpreted, compromising the confidentiality of the communication. The lack of authentication or integrity-checking mechanisms further exacerbates this issue, making the system vulnerable to data tampering (Hossain, et al., 2024).

Thirdly, there is a widespread assumption in prior research that the transmission environment is reliable, typically modeled around protocols such as TCP. As a result, the implications of using UDP, a connectionless protocol known for its packet loss, lack of guaranteed delivery, and absence of built-in sequencing or error correction, are often ignored. This limits the applicability of many proposed techniques in real-time or live streaming scenarios where UDP is preferred due to its low latency.

Finally, the data hiding capacity of many LSB systems is constrained due to static pixel selection and single-frame embedding strategies. These approaches fail to utilize the full potential of a video stream's temporal and spatial redundancy, resulting in limited payload sizes. In modern applications that require the transmission of larger amounts of sensitive data, such as video messages, documents, or cryptographic keys. This limitation significantly restricts the practical utility of traditional systems (Dalal & Juneja, 2020)

The table below summarizes these key limitations:

Table 1 : Key Limitations of Existing Systems

Limitation	Description
Susceptibility to Attacks	Spatial LSB methods are vulnerable to steganalysis, compression, and distortion.
Lack of Encryption	Most methods do not apply cryptographic protections, exposing data to theft.
UDP Challenges	Techniques often assume TCP-like reliability, ignoring real-world packet loss.
Low Capacity	Embedding is limited by static, inefficient data distribution methods.

2.6 Research Gap and Motivation for the Proposed Work

While prior work has attempted to address some of the individual shortcomings in LSB-based steganography, there remains a notable gap in the literature when it comes to comprehensive solutions that address security, reliability, and scalability in tandem, especially for real-time data transmission over UDP. Existing methods often enhance either stealth or capacity but rarely incorporate mechanisms to maintain message integrity or prevent data loss in unreliable network conditions (Bhole & Patel, 2014) (Alanzy, et al., 2023)

There is a critical need for an approach that integrates multiple strategies:

Redundant Data Embedding: This ensures that even if certain video frames (or packets) are lost during transmission, duplicate fragments embedded in other frames can be used to successfully reconstruct the original message.

Data Fragmentation and Sequencing: By breaking the secret message into numbered fragments, the system can reorder the pieces correctly after transmission, mitigating the unordered nature of UDP communication.

Encryption: Protecting the embedded content with a passkey or cryptographic algorithm ensures that even if unauthorized access occurs, the hidden information remains unintelligible (Hossain, et al., 2024).

The proposed research seeks to address these requirements by designing a robust, secure, and scalable LSB-based video steganography framework that operates effectively over UDP. This involves integrating not only technical mechanisms like redundancy and sequencing but also developing a user-centric platform that demonstrates real-world applicability, featuring user roles, a GUI, live streaming, and hidden messaging capabilities.

The following table summarizes the research gap and how this work responds to it:

Table 2 : Research gaps and proposed solutions

Challenge	Status in Existing Literature	Proposed Solution
UDP reliability (packet loss, disorder)	Rarely addressed	Introduced fragmentation, redundancy, and sequencing
Real-time secure transmission	Minimal focus	Implemented passkey-based encryption and live streaming support
Scalability for high-resolution video	Often overlooked	Developed codec-flexible platform using Python and OpenCV
Practical integration and usability	Limited	Created a web-based application with user roles and messaging features

2.7 Conclusion

In conclusion, although LSB-based video steganography has proven to be a useful and widely adopted method for hiding data due to its simplicity and minimal impact on visual quality, it suffers from several unresolved challenges and especially in the context of modern, real-time digital communication. The shortcomings related to data security, transmission reliability, processing robustness, and practical scalability significantly limit the deployment of existing systems in live environments using UDP.

Recent research has acknowledged the value of hybrid techniques, combining LSB with encryption or selective embedding, yet a truly integrated framework that addresses all critical requirements, stealth, reliability, security, and real-time applicability, is still lacking. The research presented in this project fills this critical gap by offering a comprehensive solution that combines advanced LSB embedding, redundancy, fragmentation, and passkey-based encryption within a user-friendly application. This not only advances the theoretical understanding of video steganography over unreliable networks but also contributes a practical, deployable system for secure communication in sensitive or adversarial contexts.

Chapter 3: Methodology

This chapter details the methodology used in the development of a video steganography system based on the Least Significant Bit (LSB) insertion technique. That improved with redundancy, sequence numbering techniques, and secure passkey insertion. The research aims to provide an effective, reliable, secure, and imperceptible method for embedding information within digital video frames. Also, ensuring data integrity, particularly in the context of UDP transmission, where data loss and corruption are common. The methodology is designed to be transparent, reproducible, and consistent with the research objectives.

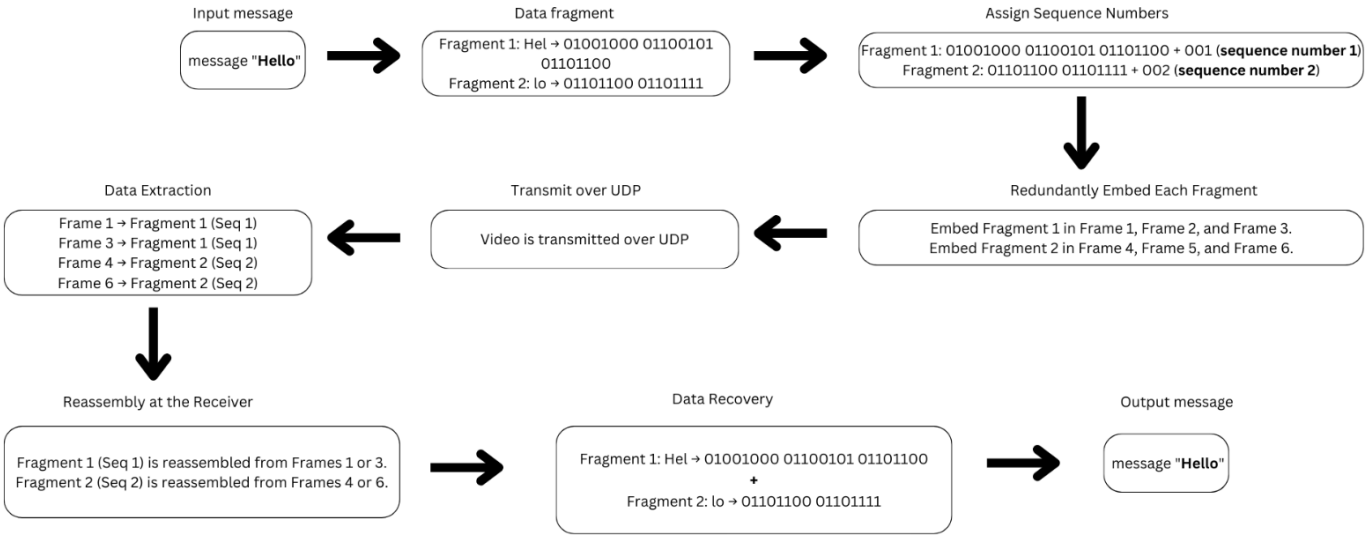


Figure 3 : Proposed System Architecture

3.1 Research Design

This study employs a quantitative experimental research design to examine the effectiveness of the least significant bit (LSB) insertion technique in video steganography for secure communication over UDP. The main objective is to develop a method for inserting and extracting secret messages within digital video files while preserving visual quality and ensuring data integrity assurance.

In this experiment, LSB insertion is applied to each frame of a video to conceal binary-encoded text messages. The study evaluates the impact of this embedding process on video quality and message accuracy by varying factors such as message length, bit rate, video format, and video capacity. The independent variables include the size of the hidden message and the resolution of the host video. The dependent variables are metrics such as Peak Signal-to-Noise Ratio (PSNR) and Bit Rate.

A set of test videos will be used with control parameters such as codec type, frame rate and color depth kept unchanged to ensure reproducible results. The proposed design allows testing of LSB steganography under reproducible and measurable conditions, thus suitable for the assessment of the method's robustness and practicability.

3.2 Tools and Technologies

This research uses a variety of tools and technologies. That's for implementing, testing, and evaluating the proposed video steganography method based on Least Significant Bit (LSB) insertion. The following tables show the primary tools and technologies used throughout the study of the research:

- **Programming Language**

Table 3 : Selected Programming Language and Justification

Language	Version	Purpose
Python	3.9	Extensive libraries, simple syntax, suitable for video/image processing and rapid prototyping

- **Libraries and Frameworks**

Table 4 : Libraries and Frameworks Used in Implementation

Library/Framework	Purpose
Flask	Lightweight Python web framework for API and app development
OpenCV	Video frame extraction, image processing, stego video reconstruction
NumPy	Efficient numerical computation, pixel-level operations
Tkinter	GUI interface for file selection, message input/output
WebRTC	Real-time video/audio streaming for live steganography
Werkzeug	WSGI support for web server integration
Matplotlib/Seaborn	Visual analysis, e.g., plotting PSNR graphs

❖ **Tools**

Table 5 : Tools Employed for Platform Development

Tool	Purpose
Apache Web Server	Hosts the web interface and handles HTTP requests
Bootstrap	Responsive front-end framework for UI
MySQL	Database management for users, video metadata, messages

❖ **Algorithms and Techniques**

Table 6 : Algorithms and Techniques Applied in Steganography

Algorithm/Technique	Purpose
Least Significant Bit (LSB)	Core embedding method to hide bits in pixel data
Polynomial Equation	Determines pixel/frame positions for embedding
Stego Key	Enhances security by controlling access to embedded data

❖ **Video and Data Processing**

Table 7 : Video and Data Processing Tasks Overview

Task	Description
Frame Extraction & Reconstruction	Extracts video frames for embedding and reassembles stego video
Binary Conversion	Converts text message to binary before embedding

❖ **Evaluation Metrics**

Table 8 : Evaluation Metrics for System Performance

Metric	Description
Embedding Capacity	Number of bits embedded without visible degradation
Processing Time	Time required for embedding and extraction
Bit Rate Impact	Measures change in bit rate after embedding

3.3 System Architecture and Technology Stack

The proposed system for secure communication through video steganography is structured as a centralized web application. Its main goal is to hide secret messages within video and image files. The system is built with several key parts: a user authentication system, dashboards for different user types (streamers/admins and seekers/viewers), a system for encoding and streaming videos, a chat system that can encode images, and tools for managing access and history.

The core technologies and tools used for building this system include:

- **Flask:** This is a lightweight web framework in Python that handles the backend logic and how different parts of the application communicate.
- **Flask-SQLAlchemy:** This tool helps the application interact with databases by mapping programming code to database operations.
- **Python:** The entire system relies heavily on Python and its libraries, especially for working with video frames.
- **OpenCV:** This Open Source Computer Vision Library is essential for changing video frames to embed and extract hidden messages.
- **UDP Protocol Enhancements:** The system improves the User Datagram Protocol (UDP) for better security and reliability. This is done by dividing data into segments, assigning sequence numbers, adding redundant data, using hashing, and applying a passkey for encoding and decoding.

This setup allows for embedding messages into video files (specifically KVI format) and images in chat. A key feature is redundancy, where each message part is embedded into three different video frames. This

ensures that the message can still be recovered even if some frames are lost or corrupted during transmission.

3.4 Reproducibility & Transparency

The methodology used in this research is designed to be clear and allow others to reproduce the work. This means the steps and tools are well-defined, so other researchers can understand, verify, and build upon the findings. The study achieves this by:

- Detailed Methodology: The research clearly explains how the video steganography system was developed. This includes specific techniques like Least Significant Bit (LSB) insertion, along with improvements like redundancy and secure passkey insertion.
- Clear Objectives: The methods used directly support the research goals. These goals include creating an efficient LSB-based video steganography method, improving data security with encryption, making UDP-based data communication more reliable by handling packet loss, reviewing existing techniques, and evaluating the new algorithm's performance.
- Open-Source Tools: The use of widely available and open-source tools such as Python and OpenCV makes the research more accessible and easier for other researchers to reproduce the results.
- Practical Framework: The research offers a robust and practical framework for secure messaging and live streaming. This makes the system adaptable for various real-world uses.

Chapter 4: System Implementation

4.1 Overview of Implementation Approach

The "System Implementation" chapter focuses on putting the theoretical concepts discussed in earlier parts of this document into a practical application. This section explains the detailed steps taken to build the proposed system for secure communication using video steganography. The implementation process included developing different parts of the system, combining various technologies, and ensuring that the system functions correctly.

The development of the secure video steganography system followed a well-organized approach. This made sure that all the theoretical ideas were successfully transformed into a fully working and robust application. This section provides a high-level summary of the entire implementation process, highlighting the main stages and the key components used. The system was mainly built using Python as the programming language, utilizing its rich collection of libraries for processing multimedia and managing network communication. Key technologies that were integrated into the system include OpenCV for handling video frames, Flask for developing the web application, and Flask-SQL Alchemy for managing the database. Additionally, the communication aspect was enhanced by using a modified User Datagram Protocol (UDP) to ensure data is transmitted securely and reliably.

4.2 Main Components of the Implementation

Our proposed system consists of three core implementation components:

1. Image Steganography Implementation

A Python desktop tool that uses the Least Significant Bit (LSB) method to hide and retrieve secret text messages within image pixels using a simple GUI

2. Video Steganography Implementation

An LSB-based video tool that hides messages across multiple video frames with redundancy, enhancing robustness against frame loss or corruption.

3. Platform Implementation

Combines both image and video steganography modules into a single user-friendly platform for streamlined encoding and decoding of secret data.

4.2.1 Image Steganography Implementation

The image steganography tool allows users to embed secret messages into PNG images and later extract them, all within a user-friendly desktop application environment. And also, this is a Python-based desktop application with a graphical user interface built using Tkinter. It allows users to:

I. Key Steps in Implementation

1. Image Selection and Loading

- Users select a PNG image through a file picker.
- The image is loaded using Pillow's Image.open() function and converted to RGB mode if necessary.

2. **Message Preparation**
    - The secret message is converted into a binary string, with each character represented by 8 bits.
    - A unique end-of-message marker is appended to the binary string to signal the end of the embedded data.
3. **Message Embedding (Encoding)**
    - The Least Significant Bit (LSB) of each pixel's red, green, and blue channels is replaced with bits from the binary message.
    - This process is repeated until all message bits are embedded.
    - The modified image is saved as a new PNG file to avoid compression artifacts that could corrupt the hidden data.
4. **Message Extraction (Decoding)**
    - The encoded image is loaded.
    - The LSB of each pixel's color channels is extracted and combined into a binary string.
    - The binary string is split into 8-bit chunks, each converted back to an ASCII character.
    - The extraction stops when the end-of-message marker is detected, ensuring accurate retrieval of the secret message.

II. Technologies used

Below table shows the technologies used in image steganography implementation

Table 9 : Technologies used in image steganography

Component	Description
Python	Programming language
Tkinter	GUI framework
Pillow	For image processing
LSB	The steganography technique used

III. Code explanation

Encoding Process (Embedding Text into Image)

Function: encode_message_in_image(img_path, message, output_path)

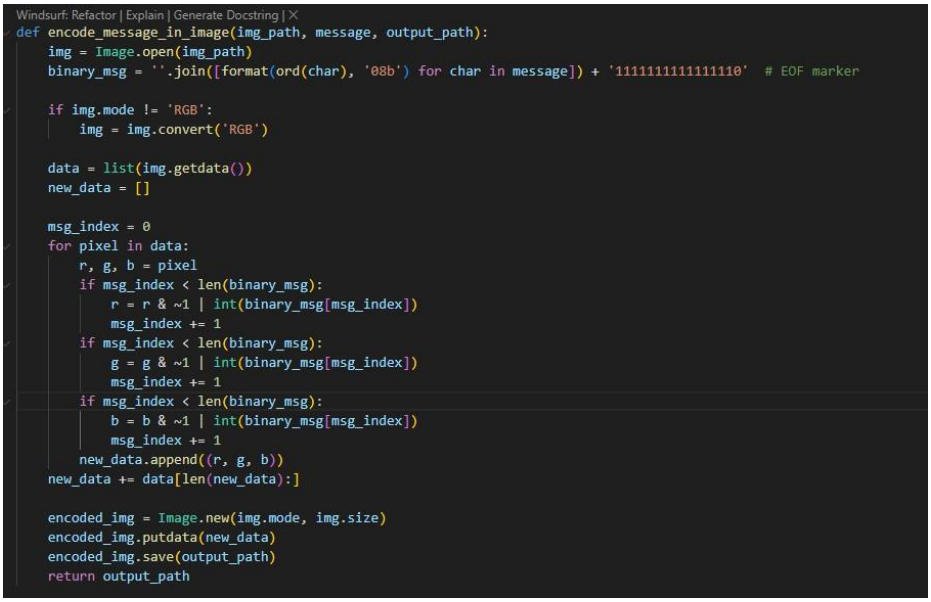
Steps:

1. **Image Loading**:
    - Opens the selected image using PIL.Image.open().
    - Converts it to RGB mode if it's not already in RGB.
2. **Message Conversion to Binary**:
    - The message is converted into a binary string (8 bits per character).
    - A special end marker "1111111111111110" is added to indicate the end of the message.

3. **Pixel Manipulation**:
    - Each pixel's R, G, and B values are modified slightly to embed bits of the message.
    - The Least Significant Bit (LSB) of each color channel is replaced with one bit from the binary message.
4. **New Image Creation**:
    - A new image is created using the modified pixel data.
    - The output is saved as a new image (typically PNG format to avoid compression artifacts)

Purpose: To invisibly hide secret text messages inside an image using binary manipulation, which can later be extracted accurately.

Below figure shows the encoding process when embedding text into an image



```
Windsurf: Refactor | Explain | Generate Docstring | X
def encode_message_in_image(img_path, message, output_path):
    img = Image.open(img_path)
    binary_msg = ''.join([format(ord(char), '08b') for char in message]) + '1111111111111110'  # EOF marker

    if img.mode != 'RGB':
        img = img.convert('RGB')

    data = list(img.getdata())
    new_data = []

    msg_index = 0
    for pixel in data:
        r, g, b = pixel
        if msg_index < len(binary_msg):
            r = r & ~1 | int(binary_msg[msg_index])
            msg_index += 1
        if msg_index < len(binary_msg):
            g = g & ~1 | int(binary_msg[msg_index])
            msg_index += 1
        if msg_index < len(binary_msg):
            b = b & ~1 | int(binary_msg[msg_index])
            msg_index += 1
        new_data.append((r, g, b))
    new_data += data[len(new_data):]

    encoded_img = Image.new(img.mode, img.size)
    encoded_img.putdata(new_data)
    encoded_img.save(output_path)
    return output_path
```

Figure 4:Encoding Process

Decoding Process (Extracting Hidden Message)

Function: decode_message_from_image(img_path)

Steps:

1. **Image Reading**:
    - The image is loaded using PIL.Image.open().
2. **Bit Extraction**:
    - Each pixel's R, G, and B values are read.
    - The LSB of each channel is extracted and combined to form a long binary string.
3. **Binary to Text Conversion**:
    - The binary string is divided into 8-bit chunks.
    - Each chunk is converted back to its ASCII character.
4. **End Marker Detection**:
    - The extraction process continues until the unique EOF marker (ÿþ or 1111111111111110) is detected.
    - The message is returned, excluding the marker.

Purpose: - To accurately retrieve the hidden text from an image that was previously encoded using the same tool.

This Figure shows the code of the decoding process when Extract the hidden message

Figure 5 : Decoding Process (Extracting Hidden Message)

IV. 4.2.1.4 GUI Components (Tkinter)

- Select Image: File picker to load an image.
- Text Box: For entering the message to encode.
- Encode and Save: Button to start the encoding process and save the output.
- Decode Message: Button to extract and display the hidden message.
- Output Text Area: Shows the decoded message.

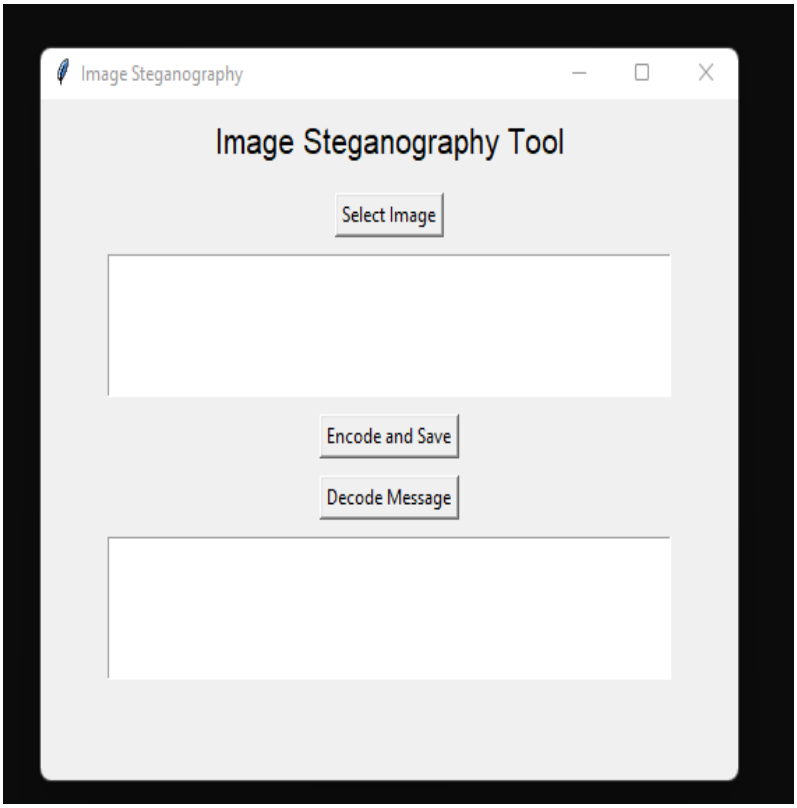


Figure 6 : Image steganography GUI

4.2.2 Video Steganography Implementation

The video steganography tool extends the LSB technique to video files, specifically supporting the KVI format. This tool allows for the embedding and extraction of secret messages within video frames, with built-in redundancy for increased reliability.

I. Key Steps in Implementation

1 . Message Preprocessing

- The secret message is converted to a binary string using ASCII encoding.
- The length of the binary message is converted to a 32-bit binary string and prepended to the message.
- The combined binary message is divided into 10-bit fragments, each padded with zeros if necessary.
- Each fragment is combined with a 32-bit sequence number to ensure correct ordering during decoding.

2 . Message Embedding (Encoding)

- Frame Selection and Redundancy
    - Each 10-bit message fragment is embedded into three different frames for redundancy.

        example:

        - Fragment 1 is embedded into Frame 1, Frame 10, and Frame 20.
        - Fragment 2 is embedded into Frame 2, Frame 11, and Frame 21.
        - This pattern continues for all fragments.

- By embedding each fragment into 3 frames, the system ensures that even if some frames are dropped or corrupted, the message can still be recovered from the remaining frames.

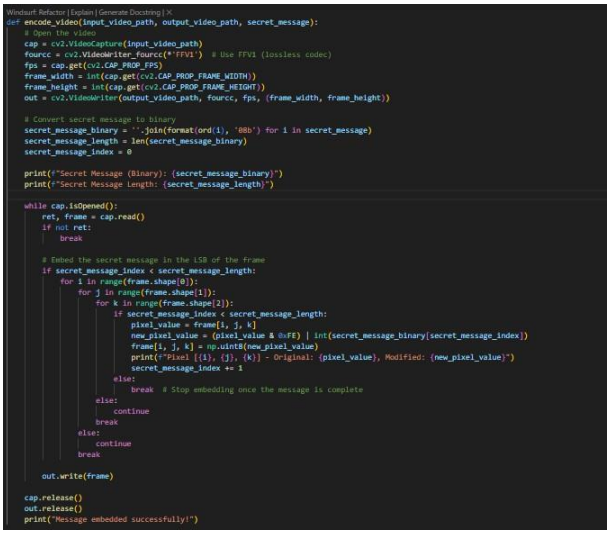
- Pixel Modification
  - For each selected frame, the LSB of each pixel's color channel is modified to store the binary data.
  - This process continues until all message fragments are embedded.
  - The modified frames are written back into the output video file using OpenCV's video writer.

    Example :

    - If the pixel value is 10101010 (170 in decimal) and the next bit of the fragment is 1, the new pixel value becomes 10101011 (171 in decimal).

II. Code explanation

Encoding Process (Embedding Text into Video)



```
def encode_video(input_video_path, output_video_path, secret_message):
    # Open the video
    cap = cv2.VideoCapture(input_video_path)
    fourcc = cv2.VideoWriter_fourcc(*'FFV1')  # Use FFV1 (lossless codec)
    fps = cap.get(cv2.CAP_PROP_FPS)
    frame_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    frame_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    out = cv2.VideoWriter(output_video_path, fourcc, fps, (frame_width, frame_height))

    # Convert secret message to binary
    secret_message_binary = ''.join(format(ord(i), '08b') for i in secret_message)
    secret_message_length = len(secret_message_binary)
    secret_message_index = 0

    print(f"Secret Message (Binary): {secret_message_binary}")
    print(f"Secret Message Length: {secret_message_length}")

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Embed the secret message in the LSB of the frame
        if secret_message_index < secret_message_length:
            for i in range(frame.shape[0]):
                for j in range(frame.shape[1]):
                    for k in range(frame.shape[2]):
                        if secret_message_index < secret_message_length:
                            pixel_value = frame[i, j, k]
                            new_pixel_value = (pixel_value & 0xFE) | int(secret_message_binary[secret_message_index])
                            frame[i, j, k] = np.uint8(new_pixel_value)
                            print(f"Pixel [{i}, {j}, {k}] - Original: {pixel_value}, Modified: {new_pixel_value}")
                            secret_message_index += 1
                        else:
                            break  # Stop embedding once the message is complete
                    else:
                        continue
                    break
                else:
                    continue
                break

        out.write(frame)

    cap.release()
    out.release()
    print("Message embedded successfully!")
```

Figure 7 : Encoding Process (Embedding Text into Video)

Steps

1: Convert the Secret Message to Binary

The message "Hello" is converted to binary using ASCII encoding:

H > 01001000
e > 01100101
l > 01101100
l > 01101100
o > 01101111

The full binary representation of "Hello" is: 01001000 01100101 01101100 01101100 01101111

2: Add the Message Length

The length of the binary message (40 bits) is converted to a 32-bit binary string and prepended to the message.

40 in 32-bit binary: 00000000000000000000000000101000

The full binary message (length + secret message) becomes:

00000000000000000000000000101000 01001000 01100101 01101100 01101100 01101111

Total length of the full binary message: **72 bits**.

3: Divide the Binary Message into 10-Bit Fragments

The binary message is divided into 10-bit fragments. If a fragment is less than 10 bits, it is padded with zeros.

Fragment 1: 0000000000 (first 10 bits of the 32-bit length)

Fragment 2: 0000000000 (next 10 bits of the 32-bit length)

Fragment 3: 0000000010 (next 10 bits of the 32-bit length)

Fragment 4: 1000010010 (first 10 bits of the secret message)

Fragment 5: 0001100101 (next 10 bits of the secret message)

Fragment 6: 0110110001 (next 10 bits of the secret message)

Fragment 7: 1011000110 (next 10 bits of the secret message)

Fragment 8: 1111 (last 10 bits of the secret message, padded with zeros)

4: Add a 32-Bit Sequence Number to Each Fragment

Each 10-bit fragment is combined with a 32-bit sequence number. The sequence number starts at 1 and increments for each fragment.

Fragment 1:	0000000000 +	00000000000000000000000000000001	(Sequence number 1)
Fragment 2:	0000000000 +	00000000000000000000000000000010	(Sequence number 2)
Fragment 3:	0000000010 +	00000000000000000000000000000011	(Sequence number 3)
Fragment 4:	1000010010 +	00000000000000000000000000000100	(Sequence number 4)
Fragment 5:	0001100101 +	00000000000000000000000000000101	(Sequence number 5)
Fragment 6:	0110110001 +	00000000000000000000000000000110	(Sequence number 6)
Fragment 7:	1011000110 +	00000000000000000000000000000111	(Sequence number 7)

5: Final Binary Structure

The final binary structure for each fragment is: 10-bit message fragment + 32-bit sequence number

Fragment 1: 0000000000 00000000000000000000000000000001

Fragment 2: 0000000000 00000000000000000000000000000010

Fragment 3: 0000000010 00000000000000000000000000000011

Fragment 4: 1000010010 00000000000000000000000000000100

Fragment 5: 0001100101 00000000000000000000000000000101

Fragment 6: 0110110001 00000000000000000000000000000110

Fragment 7: 1011000110 00000000000000000000000000000111

Fragment 8: 1011000110 00000000000000000000000000001000

Decoding Process (Embedding Text into video)

The decoding process extracts the embedded message from the video frames and reconstructs the original secret message.

Decoding Steps:

1. **Extract Fragments:**
    - For each frame, extract the 42-bit fragment (10-bit message fragment + 32-bit sequence number) from the LSBs of the pixel values.
2. **Sequence Number Verification:**
    - Use the 32-bit sequence number to ensure the correct order of fragments.
    - If multiple copies of a fragment are found (due to redundancy), the system uses the first valid fragment.
3. **Reconstruct the Message:**
    - Combine the 10-bit message fragments in the correct order using the sequence numbers.
    - Remove the 32-bit length prefix and convert the remaining binary data back to the original message.

4. User Input Validation:

- Check if the user-provided length matches the extracted length. If not, display an error message

```
Windsurf: Refactor | Explain | Generate Docstring | X
def decode_video(input_video_path, secret_message_length):
    cap = cv2.VideoCapture(input_video_path)
    secret_message_binary = ''
    secret_message_index = 0

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Extract the secret message from the LSB of the frame
        for i in range(frame.shape[0]):
            for j in range(frame.shape[1]):
                for k in range(frame.shape[2]):
                    if secret_message_index < secret_message_length:
                        lsb = str(frame[i, j, k] & 1)
                        secret_message_binary += lsb
                        print(f"Pixel [{i}, {j}, {k}] - LSB: {lsb}")
                        secret_message_index += 1
                    else:
                        break  # Stop extracting once the message is complete
                else:
                    continue
                break
            else:
                continue
            break

    cap.release()

    print(f"Extracted Binary Message: {secret_message_binary}")
    print(f"Extracted Message Length: {len(secret_message_binary)}")

    # Convert binary message to string
    secret_message = ''
    for i in range(0, len(secret_message_binary), 8):
        byte = secret_message_binary[i:i+8]
        if byte:  # Ensure the byte is not empty
            secret_message += chr(int(byte, 2))

    print(f"Decoded Message: {secret_message}")
    return secret_message
```

Figure 8:Decoding Process (Embedding Text into video)

Fragments Extract Process

Steps :

1 : Extract Fragments from Video Frames

Frame 1	0000000000	00000000000000000000000000000001 (Fragment 1)
Frame 2	0000000000	00000000000000000000000000000010 (Fragment 2)
Frame 3	0000000010	00000000000000000000000000000011 (Fragment 3)
Frame 4	1000010010	00000000000000000000000000000100 (Fragment 4)
Frame 5	0001100101	00000000000000000000000000000101 (Fragment 5)
Frame 6	0110110001	00000000000000000000000000000110 (Fragment 6)
Frame 7	1011000110	00000000000000000000000000000111 (Fragment 7)
Frame 8	1111	00000000000000000000000000001000 (Fragment 8)

2: Separate Message Fragments and Sequence Numbers

For each extracted 42-bit fragment, separate the 10-bit message fragment and the 32-bit sequence number.

1	0000000000
2	0000000000
3	0000000010
4	1000010010
5	0001100101
6	0110110001
7	1011000110
8	1111

3: Reconstruct the Full Binary Message

Combine the 10-bit message fragments in the correct order using the sequence numbers.

0000000000 (Fragment 1) +
0000000000 (Fragment 2) +
0000000010 (Fragment 3) +
1000010010 (Fragment 4) +
0001100101 (Fragment 5) +
0110110001 (Fragment 6) +
1011000110 (Fragment 7) +
1111 (Fragment 8) +

The combined binary message is:

00000000000000000000000000101000 01001000 01100101 01101100 01101100 01101111

4: Extract the Message Length

The first 32 bits of the combined binary message represent the length of the secret message.

First 32 bits: 00000000000000000000000000101000

Convert this to decimal:

00000000000000000000000000101000 (binary) = 40 (decimal)

The extracted length is 40 bits, which matches the user-provided length.

5: Extract the Secret Message

Remove the first 32 bits (length) and extract the remaining binary message

Remaining Binary Message: 0001001000 0110010101 1001101100 1101101111

- Convert the binary message back to ASCII characters:

Table 10 : Converting Table

Binary Fragment	Decimal Value	ASCII Character
01001000	72	H
01100101	101	e
01101100	108	l
01101100	108	l
01101111	111	o

- The decoded message is: "Hello".

6 : Validate User Input

- The user-provided length (40) matches the extracted length (40). Therefore, the decoding process is successful.

7: Output

- The decoded message is displayed to the user:

  The secret message is: Hello

4.2.3 Platform Implementation

The final phase of this project involved the integration of image and video steganography functionalities into a unified, user-friendly platform. The platform supports secure communication through role-based access control, live and recorded video streaming, and multimedia steganography encoding/decoding. Users are assigned specific roles as either **Streamers (Admins)** or **Seekers (Viewers)**, each with distinct permissions and functionalities. This integration ensures practical usability in real-world scenarios where secret communication is required alongside media sharing.

The steganography platform serves as a centralized web application that integrates:

- Image steganography (desktop-based tool).
- Video steganography (frame-based encoding).
- Real-time video streaming.
- Role-based user access (Streamer and Seeker).
- Chat functionality with image-based secret messaging.

The platform is built using the **Flask web framework** in Python and follows a modular architecture to facilitate seamless user interactions, message encoding, live communication, and secure data access.

I. Platform Architecture and Roles

- **User Roles:**
  - **Streamer (Admin):** These users have the capability to encode secret messages into both live and uploaded video streams, as well as images used in chat communications. They also manage stream sessions and control user access permissions.
  - **Seeker (User):** These users are primarily consumers of content. They can subscribe to specific streamers, view both live and recorded videos, and decode any hidden messages embedded within the media.

- **Core Modules**

  The application consists of six core modules:

  1. User Authentication System: Ensures secure, role-based login and registration functionalities.

2. Streamer Dashboard: Provides administrative controls and access to encoding tools for streamers.
3. Seeker Dashboard: Displays available media content and provides decoding tools for viewers.
4. Video Encoding and Streaming System: Supports both live and post-upload steganographic video processing.
5. Image Encoding Chat System: Enables steganographic communication via images in a real-time chat interface.
6. Access and History Management: Maintains records of stream sessions and handles video access permissions.

II. Technologies and Implementation

Table 11: Technologies used in Platform Implementation

Technology	Purpose
Flask	Lightweight web framework used for backend logic and routing
Flask-SQLAlchemy	ORM layer for database models and relationships
Flask-Migrate	Schema migrations and versioning for database evolution
Flask-SocketIO	Real-time WebSocket communication (for live chat and video controls)
eventlet	Asynchronous I/O for handling multiple WebSocket connections
pymysql	MySQL database driver used to connect Flask with a MySQL server
opencv-python	Video frame access, manipulation, and message embedding
numpy	Efficient pixel array manipulation for steganography processing

III. Implementation Phases

The implementation of the platform can be categorized into three major components:

1. User authentication (Both streamer and users)
2. Streamer Panel and Encoding Process(streamer)
3. Seeker Panel and Decoding Process (Seeker)

1. User authentication (Both streamer and users)

Both Streamers and Seekers are authenticated through a role-based login system. Upon successful authentication, users are redirected to different interfaces based on their roles. Streamers are directed to the Admin Panel, while Seekers are taken to the Streaming Wall. User credentials are securely stored using hashed passwords, facilitated by the Werkzeug security library.

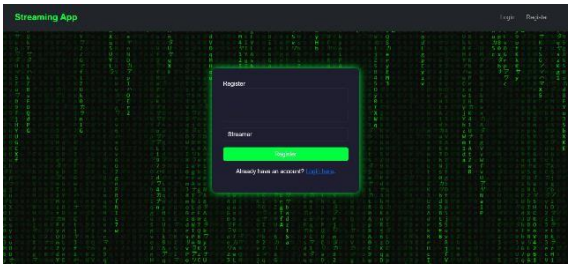


Figure 9:registation from

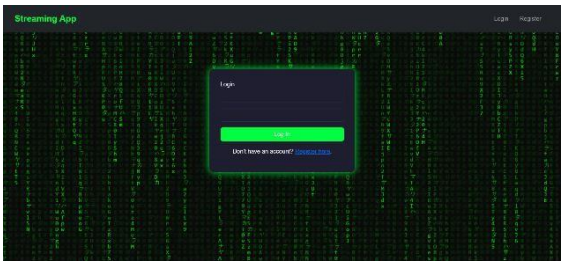


Figure 10:Login from

- Role-based login for Streamers and Seekers

  Upon login, redirection is done based on role:

  - **Streamers** → Admin Panel.
  - **Seekers** → Streaming Wall.

  User registration stores data securely in the database with hashed passwords (using Werkzeug).

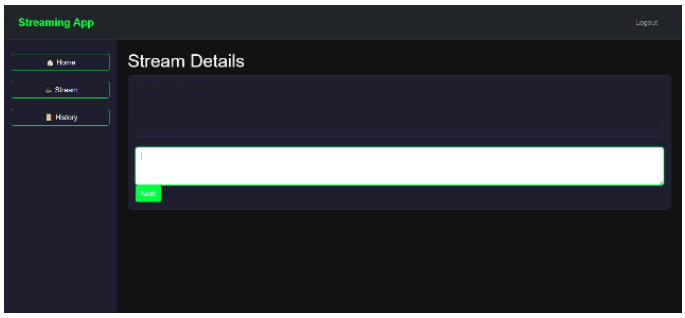


Figure 11:Admin interface

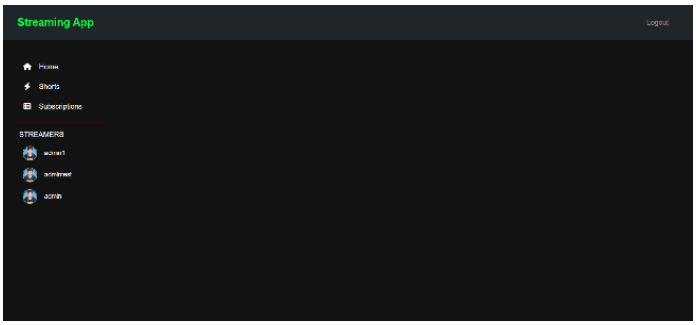


Figure 12:seeker interface

2. Streamer Panel and Encoding Process

The Streamer Panel acts as the administrative interface and includes the following functionalities:

- **Admin Home Page**: Provides an overview of live session statuses, video encoding history, and access control requests.
- **Streaming Modes**:
    - Live Streaming: Implemented using WebRTC (for peer-to-peer communication) and Flask-SocketIO (for backend management). This mode allows Streamers to select specific Seekers for access and enables real-time steganographic encoding using OpenCV.
    - Upload Video Streaming: Validates and converts video files to .mp4 format if necessary. Secret messages are embedded into selected video frames using Least Significant Bit (LSB) encoding techniques. These videos are available for later streaming and decoding.
- **Chat Functionality with Image Encoding**: During both live and uploaded video sessions, a chat interface becomes available. When the "Hide" option is selected, messages are encoded into images using LSB steganography before being transmitted over the chat stream.
- **Stream Video Management**: Streamers can manage video access by deleting past sessions, setting expiration limits (defaulting to 24 hours), and handling viewer access requests.
- **Access Request Handling**: When initiating a stream, Streamers can control user access by selecting specific Seekers. These requests are managed via a dedicated interface for streamlined access control.

3. Seeker Panel and Decoding Process

Seekers interact with the platform through the Seeker Dashboard, which provides access to the following features:

- **Streaming Wall**: A feed of available live and recorded videos. Seekers can view these videos and use the decoding module to extract hidden messages. Chat images containing steganographic messages are also decoded automatically.
- **Streamer Management**: Seekers can subscribe to specific Streamers, submit access requests for restricted videos, and maintain a personalized list of followed content providers.

IV. Data Flow Summary

The platform handles different data types and communication modes as follows:

- **Image Steganography via Chat**: Messages entered by the Streamer are encoded into PNG images upon selecting the "Hide" option. These images are transmitted over WebSockets and decoded by the Seeker upon receipt.
- **Video Steganography**: Messages are segmented into 10-bit pieces and preceded by a 32-bit sequence. These are embedded across selected frames using OpenCV, either in real time or for uploaded content. The Seeker decodes messages using a complementary decoding algorithm.

- **Real-Time Communication**: Flask-SocketIO and eventlet handle chat and streaming session communication, while WebRTC supports peer-to-peer delivery of live video streams. Flask manages all routing, permissions, and access control.

V. Advantages of the Platform

The integrated platform presents several key advantages:

- **Scalability**: The modular Flask-based architecture allows for the extension of features, including additional codecs and encoding techniques.
- **Real-Time Performance**: The use of asynchronous I/O and real-time communication frameworks ensures low-latency streaming and chat functionalities.
- **User-Centric Design**: The platform provides tailored interfaces and experiences for each user role, enhancing usability.
- **Security**: The implementation includes secure password storage, access control mechanisms, and message confidentiality through steganography.
- **Redundancy**: The video steganography system incorporates frame redundancy, improving message recovery in cases of frame loss during transmission.

4.3 Evaluation Metrics

Our research establishes several key metrics to evaluate the performance and effectiveness of the proposed video steganography system:

- **Capacity:**

  The system measures the maximum message size that can be embedded within video frames without compromising visual quality or detectability. This is a critical metric, as it reflects the practical utility of the system for real-world applications where larger payloads may be required. Capacity is determined by the number of available frames, frame resolution, and the redundancy strategy employed.

- **Bit Rate:**

  Bit rate is monitored to assess the impact of steganographic embedding on video streaming performance. By embedding data in the least significant bits of pixel values, the system aims to minimize increases in bit rate, ensuring that the stego-video remains suitable for real-time streaming and transmission over UDP.

- **Processing Time:**

  The time taken for both embedding and extraction processes is a crucial practical metric. This includes the time required to segment, encode, and embed the message across multiple frames, as well as the time needed to extract and reconstruct the hidden data. Efficient processing is essential for real-time and scalable applications, especially in live streaming scenarios

4.4 Implementation Limitations

The main limitation identified in our research is “High Resource Usage in Real-Time Processing”.The requirement to process multiple video frames, embed data redundantly across several frames, and manage sequencing and encryption introduces significant computational overhead. This can lead to increased resource consumption, particularly during real-time video streaming or when handling high-resolution videos. As a result, the system may face challenges in resource-constrained environments or when scaling to support large numbers of concurrent users. This limitation is acknowledged as a trade-off for achieving improved reliability and security in UDP-based transmission.

4.5 Testing and Validation

4.5.1 PSNR (Peak Signal to Noise ratio) Validation

Both the original video and the encoded video were analyzed frame by frame using OpenCV. For each corresponding frame, PSNR was calculated using the standard formula:

$$\text{PSNR} = 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right)$$

Figure 13 : PSNR calculation formula

Where:

- MAXIMAX_{I}MAXI is the maximum possible pixel value of the image (255 for 8-bit images),
- MSEMSEMSE is the mean squared error between the original and encoded frames.

A PSNR of ∞ (infinity) occurs when MSE=0MSE = 0MSE=0, indicating – no difference

The test results below show that most video frames have a PSNR value of inf, which indicates an infinite value. Those values show the perfect match between the original and encoded video frames. This typically happens when there is no measurable difference (i.e., zero mean squared error) between two frames. That shows the encoding process kept the visual quality with high precision for those frames.

However, a few frames exhibit finite but extremely high PSNR values, such as 110.05 dB, 108.29 dB, and 107.62 dB. These values are still well above the usual accepted threshold of 40 dB for excellent quality. That means even in frames where some minor differences exist, small that wouldn't notice them with human eyes.

Overall, these results show that the encoded video is almost identical to the original, with little to no visible loss in quality throughout the entire video.

The below figure shows the PSNR between original and encoded videos.

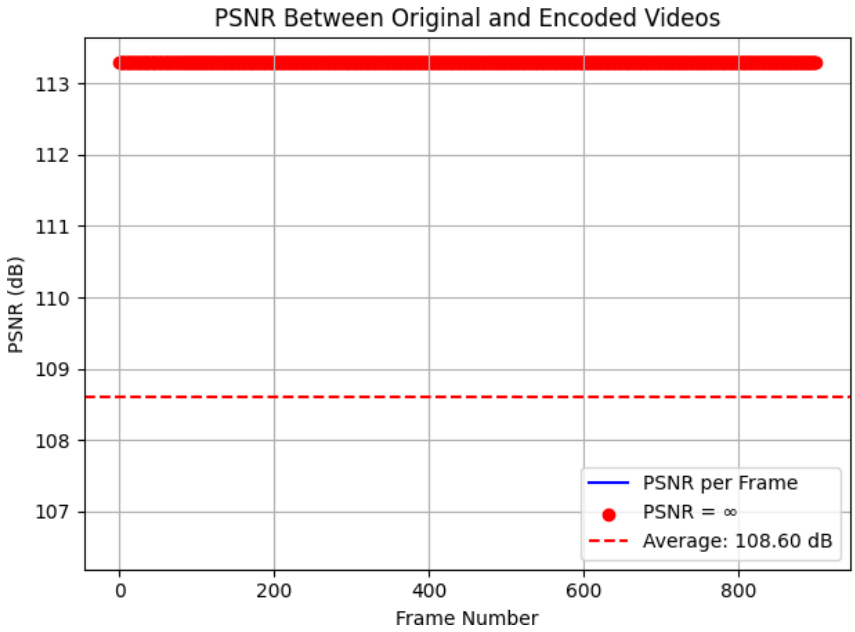


Figure 14 : PSNR between original and encoded videos

4.5.2 SSIM (Structural Similarity Index Measure) testing and validation

This Structural Similarity Index (SSIM) is calculated for check how the encoded video matches to the original video. Below test result chart shows that every frame had a perfect SSIM score. That means encoded frames are exactly the same as the original video's frames without no visible loss in quality.

SSIM value of 1.0000 is the highest possible and that shows brightness, contrast, and details. These all are fully kept as like the original video. This suggests that the encoding process either did not lose any information or used a method that kept all important visual features in the video. In any frame no drop in quality was found and showing that the encoding method is very good at keeping the original video’s details.

These results prove that the encoded video keeps the original quality perfectly , making it suitable for uses where clear and accurate images are very important.

The Below Figure shows the quality of the video after encoding:

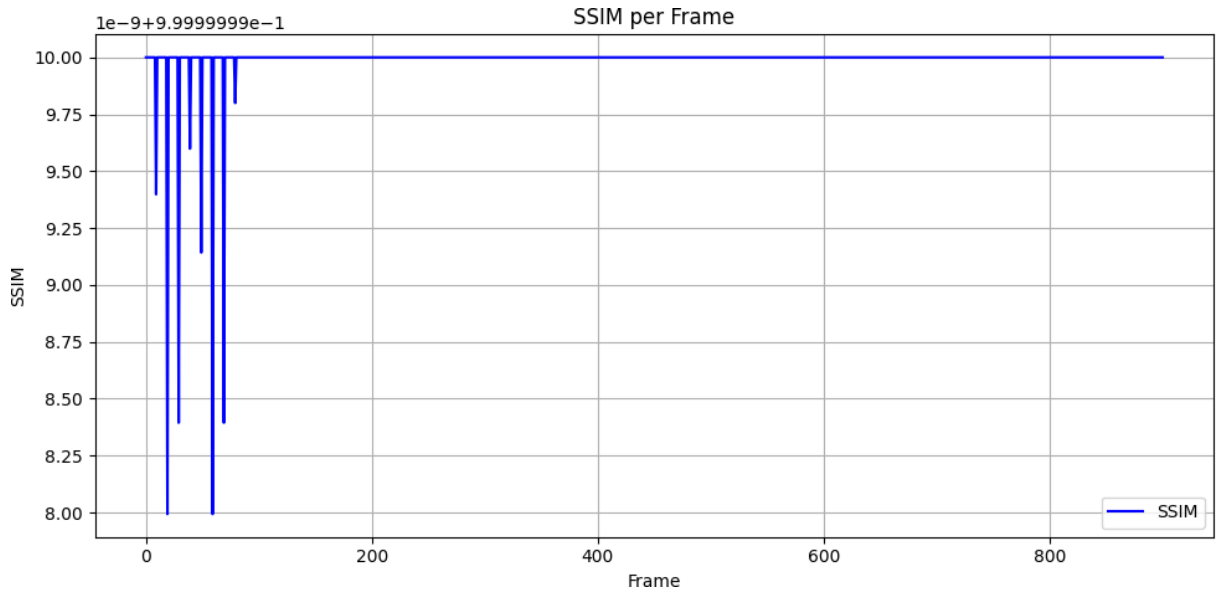


Figure 15 : SSIM calculation

Chapter 5: Results and Discussion

5.1 Results

This research focused on developing a secure, reliable, and efficient method for embedding hidden data within video files using the Least Significant Bit (LSB) insertion technique. And focusing on transmission over the User Datagram Protocol (UDP). After a set of developments and testing, key important findings were gathered. That highlights the strengths and practical usability of our approach.

Effective data encoding and decoding across multiple formats:

Successfully embedded data like text messages and images into different video formats such as AVI, VP8, MP4, and H.264. This LSB insertion technique allowed us to modify the least significant bits of the video frames without causing any noticeable degradation in video quality. When viewers watch the videos cannot detect any visual anomalies to human eyes. That is critical for maintaining the covert nature of communication.

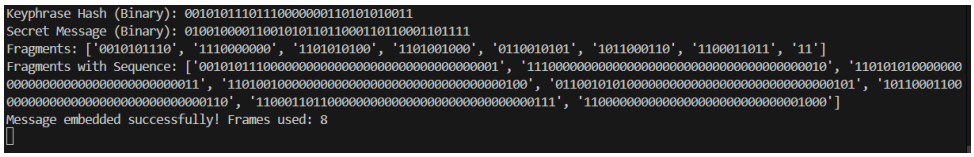


Figure 16:Result of embedding

```
Frame 850 - Fragment: 0000000000, Sequence: 0
Frame 860 - Fragment: 0000000000, Sequence: 0
Frame 870 - Fragment: 0000000000, Sequence: 0
Frame 880 - Fragment: 0000000000, Sequence: 0
Frame 890 - Fragment: 0000000000, Sequence: 0
Frame 900 - Fragment: 0000000000, Sequence: 0
Decoded Message: Hello
```

Figure 17: Result of decoding

- **Robust Security Features:**

  Used a passkey-based encryption mechanism before embedding to improve the confidentiality and integrity of the embedded data with cryptographic hashing for integrity verification. This extra security layer approach guarantees that only authorized users can decode the hidden information in the encoded video by using the correct passkey. Furthermore, any tampering or corruption of the embedded data during transmission could be detected, safeguarding against unauthorized access or data manipulation.

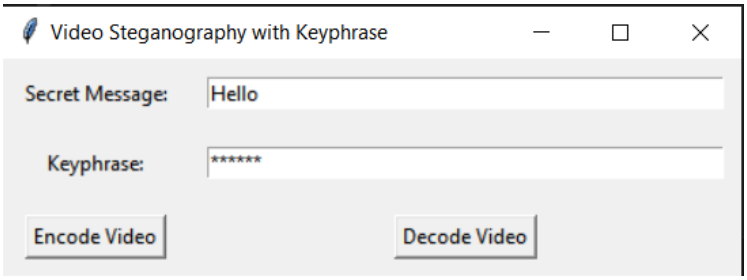


Figure 18: secret message and passkey insertion

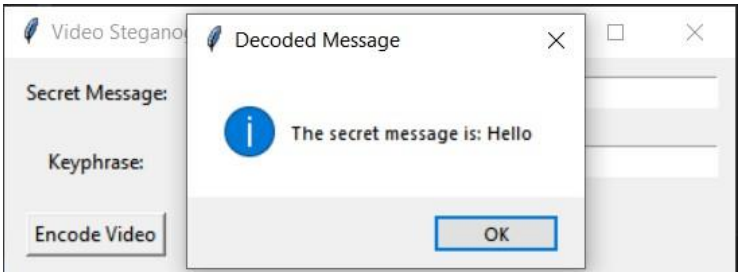


Figure 19: decode message using passkey and embedded video

- **Overcoming UDP's Inherent Limitations:**

  UDP is well known for its speed and low latency but also for its lack of built in reliability technologies such as guaranteed delivery or packet ordering. The designed system addressed these challenges by fragmenting the hidden data into smaller segments then assigning sequence numbers to each fragment and embedding redundant copies across multiple video frames. This approach enabled the receiver to accurately rebuild the original message when some UDP packets were lost, delayed, destroyed, or even received out of order. This significantly improved the robustness of data transmission over an otherwise unreliable channel.

- **Quantitative Performance Evaluation:**

  Evaluated the video quality after embedding using Peak Signal-to-Noise Ratio (PSNR)metric and SSIM (Structural Similarity Index Measure). The PSNR and SSIM results showed that the video quality remained high. The PSNR values are consistently above in good level, indicating minimal distortion

- **Practical Implementation and User Experience:**

Developed a user friendly application that integrates the entire workflow from data encryption and embedding to transmission over UDP and final extraction. During testing of the designed system highlighted the easy of use and reliability. This confirms its effectiveness for real-world applications such as secure messaging, confidential data transfer, and covert communications.

5.2 Discussion

The outcomes of this study provide strong proof that LSB-based video steganography works well combined with further improvements tailored for UDP transmission. That can serve as a highly effective method for secure and reliable covert communication.

Balancing Stealth and Security:

One of the biggest achievements of the system is its ability to hide data in videos without affecting how they look. The changes made to the least significant bits are so small because of that it's not noticeable for viewers. And also that keeping the video's appearance unchanged. At the same time using a passkey and hashing adds a strong layer of security and makes sure the hidden data stays private and protected. This balance between being hidden and being secure and that especially important for situations where privacy matters.

Mitigating UDP's Reliability Challenges:

UDP provides the low-level guarantee that data will be delivered or arrive in the right order. And it can make it hard to send important information. To fix this we broke the data into smaller parts and added sequence numbers and then included duplicates across different video frames. According to this mechanism even if some packets are lost or come in the wrong order the receiver can still rebuild the full message. That makes this system a good fit for real- time or low-bandwidth situations where UDP is preferred over slower, more reliable options like TCP.

Trade-offs and Limitations:

While the system works well overall but there are some trade-offs such as using extra data for redundancy and splitting messages and this means larger videos are needed to hide the same amount of information. Also adding encryption and data checks takes more processing power for long videos or big messages. In the future this could improve this by finding a better balance between speed and security.

Potential for Future Enhancements:

There are many ways this project could be improved. For example the system could adjust how much data it hides based on how the video is complex or how good the network connection is. And also could add stronger encryption or smarter error-checking methods scuh as supporting more video formats and making the system better for live streaming would help it work in more real-world situations.

Real-World Implications:

This project shows that it's possible to send hidden secure messages over UDP using video. It could be useful for private company communication, military use, or sending information in places where open sharing isn’t safe like journalist. Even when the network isn’t perfect but this system helps make sure that hidden messages still get through safely and secretly.

Chapter 6: Conclusion and Future Work

6.1 Conclusion

Over the of the end of two semesters this project has successfully developed and implemented a video steganography system utilizing Least Significant Bit (LSB) insertion for secure communication over the UDP. The primary objective was to address the critical limitations of existing LSB-based video steganography methods such as vulnerability to data loss, out-of-order delivery, and unauthorized access when applied to unreliable transmission channels such as UDP which is widely used for real-time applications.

The project achieved its objectives by introducing a hybrid approach that combines redundant data embedding, data fragmentation, sequencing, and encryption. This integration ensures that even in the case of packet loss or out-of-order delivery the hidden data can be reliably reconstructed at the receiver's end. The system was implemented using widely available tools and libraries, including Python, OpenCV, and enhanced UDP protocols, resulting in a codec-flexible and user friendly platform. Experimental results shows that the proposed system maintains high video quality (as measured by PSNR), achieves secure and undetectable data embedding, and effectively mitigates the inherent reliability issues of UDP- based transmission.

6.2 Limitations

Although through these achievements several limitations were identified. The primary challenge was the high resource usage during real-time processing especially when handling high resolution videos or large data payloads. This computational overhead is a trade-off for the added security and reliability provided by redundancy and encryption. Additionally , the system's performance may be constrained in resource constrained environments or when scaling to support a large number of concurrent users. Other potential limitations include compatibility with only certain video formats and the need for further optimization to reduce processing time and enhance efficiency. And also there are some limitations discussed under the discussion topic

6.3 Future Work

Looking ahead, there are many ways this system could be improved. One idea is to make the data hiding process smarter by adjusting it based on how the video is complex or how the network connection is good. And also could add stronger encryption, better error-correcting methods, and support for more types of video formats. Using deep learning could help make the hiding and extraction of data even more efficient. Another exciting direction would be to adapt the system for use in messaging apps or live streaming which would make it even more useful in real-world situations.

In short, this project offers a strong and practical way to send hidden messages securely over public networks. It fills a gap in current research and opens the door for future innovations in video steganography in both academic and real-world applications.

Annexure

Research Papers: Relevant articles supporting the project's concepts and methods. https://drive.google.com/drive/folders/1_H7L5sexUwA5A_H98QYWMQeCwro-CBCa?usp=sharing

Documentation: The document update from start to end of the Research

https://docs.google.com/document/d/1wsKeH_TQ5CTimTrh6dkAdCJli474yuTUVd3L-KxOLWA/edit?usp=drivesdk

References

1. Alanzy, A., Alzahrani, A., Alzahrani, B. & Alzahrani, M., 2023. A secure and robust video steganography scheme for covert communication in H.264/AVC. *Multimedia Tools and Applications*, 82(4), p. 14383–14407.

2. Al-Khater WA, Al-Maadeed S, Ahmed AA et al (2020) Comprehensive review of cybercrime detection techniques. IEEE Access 8:137,293–137,311

3. Banik, B., 2019. Exploring recent advances in digital video steganography and future scope. *In: S. C. Satapathy, V. Bhateja and A. Joshi, eds. Intelligent Innovations in Multimedia Data Engineering and Management. IGI Global*, p. 28–47.

4. Bhole, A. & Patel, R., 2014. Steganography over video file using random byte hiding and LSB technique. *International Journal of Computer Applications,* 100(16), pp. 1-5.

5. Balu S, Babu CNK, Amudha K (2019) Secure and efficient data transmission by video steganography in medical imaging system. Clust Comput 22(2):4057–4063

6. Banik BG (2019) Exploring recent advances in digital video steganography and future scope. Intell Innov Multimed Data Eng Manag 88–115

7. Bhawna KS, Singh V et al (2021) Information hiding techniques for cryptography and steganogra phy. In: Singh V, Asari VK, Kumar S (eds) Computational methods and data engineering. Springer Singapore, Singapore, pp 511–527

8. Cheddad, A. a. C. J. a. C. K. a. M. K. P., 2010. Digital image steganography: survey and analysis of current methods. *Signal Processing,* 90(3), p. 727–752..

9. Dalal, M. & Juneja, M., 2020. Evaluation of orthogonal and biorthogonal wavelets for video steganography. *Information Security Journal: A Global Perspective,* 29(1), p. 40–50.

10. Dasgupta, K., Mandal, J. & Dutta, P., 2012. Hash based least significant bit technique for video steganography (HLSB). *International Journal of Security, Privacy and Trust Management (IJSPTM),* 1(2), pp. 1-11.

11. DuanX(2018)Coverless steganography for digital images based on a generative model. Comput Mater Continua 55(3):483–493

12. Duan X, Jia K, Li B et al (2019) Reversible image steganography scheme based on a u-net structure. IEEE Access 7:9314–9323

13. Elharrouss O, Almaadeed N, Al-maadeed S (2020) An image steganography approach based on k-least significant bits (k-lsb). In: 2020 IEEE international conference on informatics, iot, and enabling technologies (ICIoT). IEEE, pp 131–135

14. Ghosal, A., Mondal, S. & Bhowmik, T., 2020. Secure data hiding using video steganography and cryptography technique in compressed video domain. *Multimedia Tools and Applications,* 3–4(2633–2655), p. 79.

15. Gupta, H. & Sharma, S., 2020. UDP-based real-time video steganography with error control. *Wireless Networks,* 26(8), pp. 5899-5912.

16. Hossain, M. et al., 2024. Enhancing Video Steganography Techniques Using Hybrid Algorithms. *Open Access Library Journal,* Volume 11, p. 1–21.

17. Hossain, M., Ullah, A., Khan, N. & Alam, M., 2019. Design and development of a novel symmetric algorithm for enhancing data security in cloud computing. *Journal of Information Security,* 10(3), p. 199–236.

18. Islam, M., Hasan, M., M., A. N. & Rahman, M., 2022. An enhanced LSB video steganography technique for real-time secure communication. *ournal of Information Security and Applications*, Volume 68, p. 103248.

19. Johnson, M. a. M. A., 2023. Secure video communication over UDP: concepts and approaches.. *Computer Networks*, 57(3), pp. 823-835.

20. Katzenbeisser, S. & Petitcolas, F., 2000. Information hiding techniques for steganography and digital watermarking. *Norwood, MA: Artech House.*.

21. Paul, P., Bera, S. & Mandal, J., 2014. Video steganography using LSB technique and pseudo random encoding for secure communication. *International Journal of Computer Applications*, 113(6), pp. 1-6.

22. Rani, S. a. K. S., 2022. A review on video steganography techniques. *International Journal of Advanced Research in Computer and Communication Engineering*, 11(12), p. 10–15.

23. Thakur, P., Sharma, A. & Bhatia, N., 2019. Implementation of secure video steganography using LSB. *International Journal of Recent Technology and Engineering*, 7(6S), p. pp.517–522.

24. Yadav, D. & Kumar, A., 2018. Enhanced secure and robust LSB based video steganography using firefly algorithm. *Multimedia Tools and Applications*, 77(17), p. 23039–23067.

25. Yang, Y. (2019) BASN - Learning Steganography with Binary Attention Mechanism. arXiv:1907.04362 [cs.CV]. Available at: https://arxiv.org/abs/1907.04362 (Accessed: 23 May 2025).

26. Sai Tarun, M.V., Venkateshwara Rao, K., Naga Mahesh, M., Srikanth Reddy, N. and Venkatesh, M. (no date) Digital Video Steganography Using LSB Technique. Department of Electronics and Communication Engineering, Vasireddy Venkatadri Institute of Technology, Nambur, Guntur, Andhra Pradesh, India. Available at: https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/71070208/ca8cc31a-a09e-46f0-96c8-5c6670f87b07/1702126.pdf (Accessed: 23 May 2025).
27. , 100(5), pp. 1419–1428. Available at: http://www.jatit.org/volumes/Vol100No5/18Vol100No5.pdf (Accessed: 23 May 2025).