[12pt]report

graphicx geometry fancyhdr mdframed setspace tocloft tabularx float algorithm algpseudocode nomencl etoolbox

a4paper, left=1.25in, right=1in, top=0.75in, bottom=1.25in

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI-590018

MajorProjectPhase-II(BIC786)ReportOn

"Secure Data Transmission Framework for Images, Videos, and Text Using Elliptic Curve Cryptography "

Submitted in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Engineering

in

CSE(IoT & Cybersecurity including BlockchainTechnology)

Submitted by

1BI22IC018	DILIP GOWDA JS
1BI22IC027	JAYAKRISHNA REDDY
1BI22IC028	JYOTHIKA SUDARSAN
1BI22IC034	L TEJASWINI REDDY

Under the guidance of

Dr. ShivakumarBR
Professor & Head
Department of CSE(ICB)

BANGALORE INSTITUTE OF TECHNOLOGY

K R Road, V V Pura, Bengaluru-560004

Affiliated to VTU, Belagavi, Approved by AICTE, Accredited by NBA, NAAC

2025-2026

BANGALORE INSTITUTE OF TECHNOLOGY

Department of CSE-ICB
K R Road, V V Pura, Bengaluru-560004.

CERTIFICATE

This is to certify that the project entitled **"Secure Data Transmission Framework for Images, Videos, and Text Using Elliptic Curve Cryptography"** is carried out by **DILIP GOWDA JS (1BI22IC018), JAYAKRISHNA REDDY (1BI22IC027), JYOTHIKA SUDARSAN (1BI22IC028), L TEJASWINI REDDY (1BI22IC034)**, bonafide students of Bangalore Institute of Technology, Bengaluru, in partial fulfillment of the requirements for the award of Bachelor of Engineering in CSE(IOT and Cybersecurity including Blockchain technology). This report fulfills all the requirements of the regulations for the award of the degree. The contents of this report have not been submitted to any other institute or university for the award of any degree or diploma and are not repition of the work carried out by others.

Project Guide:
Dr.Shivakumar B R
Professor & HOD
Dept.CSE(ICB)
BIT, Bengaluru

Principal:
Dr.Shanthala S
BIT, Bengaluru-560004

Project Final Viva Voce Examination

Examiners	Signature with Date
Examiner 1	
Examiner 2	

BANGALORE INSTITUTE OF TECHNOLOGY

Department of CSE-ICB

K R Road, V V Pura, Bengaluru-560004.

DECLARATION

We hereby declare that the work embodied in this project report titled **"Secure Data Transmission Framework for Images, Videos, and Text Using Elliptic Curve Cryptography"** is the result of original work carried out by **DILIP GOWDA JS (1BI22IC018), JAYAKRISHNA REDDY (1BI22IC027), JYOTHIKA SUDARSAN (1BI22IC028),** and **L TEJASWINI REDDY (1BI22IC034)** at Bangalore Institute of Technology, Department of CSE (IoT & Cybersecurity including Blockchain Technology), Bengaluru – 560004, under the guidance of **Dr. Shivakumar B R**. This report has not been submitted, either in part or in full, for the award of any diploma or degree of this or any other university.

.

USN	Name	Student Signature with Date
1BI22IC018	DILIP GOWDA JS	
1BI22IC027	JAYAKRISHNA REDDY	
1BI22IC028	JYOTHIKA SUDARSAN	
1BI22IC034	L TEJASWINI REDDY	

ACKNOWLEDGEMENT

First and foremost, we would like to extend our deepest gratitude to our project guide, **Dr. Shivakumar B R**, Professor & Head, Department of CSE(ICB), BIT, Bengaluru. His unwavering technical expertise, insightful guidance, and moral encouragement have been pivotal in helping us successfully complete **Phase 2** of our project. His mentorship has significantly contributed to our academic and personal growth.

We are very grateful to **Rajya Vokkaligara Sangha** and **Dr.Shanthala S**, Principal of BIT, for fostering an environment of academic excellence and innovation.Our heartfelt thanks to the faculty and staff of the Department of CSE (ICB), whose support and resources have played a crucial role in facilitating this project.

Special acknowledgment to our Project Coordinator **Dr. Anupama K C**, Assistant Professor, Department of CSE (ICB), Bengaluru for her invaluable advice and consistent support that helped shape this project at various stages. On a personal note, we owe our heartfelt gratitude to our parents for their unconditional love, sacrifices, and unwavering belief in our capabilities. Their encouragement has been the cornerstone for our perseverance and success. Lastly, we extend our sincere thanks to our friends, peers, and all those who contributed directly or indirectly to the successful completion of this project. Your camaraderie, insights, and encouragement have been a source of motivation throughout this journey.

1BI22IC018	DILIP GOWDA JS
1BI22IC027	JAYAKRISHNA REDDY
1BI22IC028	JYOTHIKA SUDARSAN
1BI22IC034	L TEJASWINI REDDY

Abstract

Secure data transmission is essential in today's digital world, where large amounts of sensitive information, including images, videos, and textual data, are exchanged across networks. Conventional cryptographic methods often struggle to balance security, performance, and efficiency, particularly when dealing with multimedia content. To address these concerns, this project presents a secure and efficient data transmission framework using elliptic curve cryptography. The focus of the work is on comparing two popular ECC algorithms: Curve25519 and secp256r1. The goal is to analyze their performance and demonstrate that Curve25519 offers better encryption and decryption efficiency while maintaining strong security.

The proposed framework enables the encryption and decryption of various data types, such as images, videos, and text, using Python-based cryptographic libraries. It emphasizes real-time performance by measuring critical parameters like encryption time, decryption time, key generation speed, and resource utilization for different file sizes and formats. Through experimental analysis, the project evaluates the practical advantages of Curve25519, highlighting its speed and computational benefits over secp256r1. This analysis is particularly relevant in scenarios where secure, fast, and lightweight encryption is needed, such as in secure messaging, cloud-based storage, media transmission, healthcare, and government applications.

Overall, the project contributes to the development of secure communication systems by implementing and validating a framework that supports high-performance encryption using elliptic curve cryptography. By providing a detailed comparison between Curve25519 and secp256r1, the project aims to guide future implementations toward more efficient cryptographic practices .

Contents

List of Figures

Nomenclature

AES Advanced Encryption Standard

ECC Elliptic Curve Cryptography

Chapter 1

Introduction

Information security has become a critical aspect of modern communication systems due to the rapid growth of digital data transmission over public networks. Protecting sensitive information such as text, images, and videos from unauthorized access is a major challenge in today's interconnected world. This chapter provides an introduction to the proposed system, highlighting the need, objectives, and structure of the project.

1.1 General Overview

In the modern digital era, the transmission of multimedia data such as text, images, and videos over public networks has increased significantly. With this growth, the risk of unauthorized access, data manipulation, and information leakage has also risen. Ensuring the confidentiality, integrity, and security of digital data has therefore become a major concern in information security systems. Traditional cryptographic techniques provide data protection; however, they often involve higher computational costs and make encrypted data easily identifiable to attackers.

Elliptic Curve Cryptography (ECC) has emerged as an efficient public key cryptographic technique that provides strong security with smaller key sizes compared to conventional algorithms. ECC is widely preferred in modern security applications due to its reduced computation time and lower resource requirements. Among various elliptic curves, Curve25519 and secp256r1 are commonly used curves that offer high levels of security, but their performance characteristics differ.

In addition to cryptography, steganography plays a vital role in enhancing data

security by concealing the existence of encrypted information within multimedia files. By hiding encrypted data inside images or videos, steganography reduces the chances of detection by unauthorized users. The combination of cryptography and steganography provides a dual-layer security mechanism, ensuring both data protection and data invisibility.

This project focuses on securing text, image, and video data using Elliptic Curve Cryptography combined with steganography techniques. The project implements and compares two elliptic curves, Curve25519 and secp256r1, based on encryption time, decryption time, and key size. Furthermore, encrypted data is embedded using text-in-image, image-in-image, and video-in-video steganography techniques. The proposed system is implemented as a web-based application using Python and Flask, providing a secure and efficient framework for multimedia data protection.

1.2 Motivation

With the increasing use of digital media and online communication, sensitive information such as personal messages, confidential images, and private videos is frequently transmitted over open networks. This widespread data exchange makes information vulnerable to interception, unauthorized access, and security breaches. Conventional encryption techniques provide security, but encrypted data is often easily recognizable, which may attract attackers.

Elliptic Curve Cryptography offers a secure and efficient alternative to traditional public key cryptographic systems by providing strong security with smaller key sizes and reduced computational overhead. Among different elliptic curves, Curve25519 is designed for high performance, while secp256r1 is a widely adopted standard curve. Comparing these curves helps in understanding their practical efficiency in real-time applications.

Furthermore, enhancing security by hiding encrypted data using steganography adds an additional layer of protection. By embedding encrypted text, images, or videos within cover media, the existence of secret data itself is concealed. The motivation behind this project is to design a secure, efficient, and reliable system that combines Elliptic Curve Cryptography with steganography and evaluates the performance of

Curve25519 and secp256r1 for protecting multimedia data.

1.3 Objectives

The main objectives of the proposed project are as follows:

I. To design and implement a secure system for protecting text, image, and video data using Elliptic Curve Cryptography (ECC).

II. To compare the performance of Curve25519 and secp256r1 elliptic curves based on encryption time, decryption time, and key size.

III. To enhance data security by integrating steganography techniques such as text-in-image, image-in-image, and video-in-video for hiding encrypted data.

IV. To develop a Flask-based web application that enables secure key generation, encryption, steganographic embedding, extraction, and decryption of multimedia data.

1.4 Problem Statement

The increasing use of digital communication has resulted in the frequent transmission of sensitive text, image, and video data over open networks. Although cryptographic techniques are used to protect such data, encrypted information is often easily detectable and may attract malicious attacks. Moreover, traditional encryption techniques involve higher computational overhead and larger key sizes, making them less efficient for multimedia data security.

Hence, there is a need for a secure and efficient system that provides strong encryption with reduced computation time while also concealing the existence of the encrypted data. The problem addressed in this project is to design a Flask-based application that uses Elliptic Curve Cryptography along with steganography techniques to securely protect multimedia data and to analyze the performance of Curve25519 and secp256r1 in terms of encryption time, decryption time, and key size.

1.5 organisation of report

The report is structured to provide a detailed overview of the proposed Secure Multimedia Encryption and Steganography System. Each chapter focuses on specific aspects of the project, as summarized below:

- **Chapter 1: Introduction**
  Introduces the background, motivation, objectives, and problem statement of the project, establishing the foundation for the research work.
- **Chapter 2: Literature Survey**
  Reviews existing research on secure data transmission, encryption techniques, and applications of Elliptic Curve Cryptography (ECC) in multimedia security, highlighting strengths, limitations, and gaps.
- **Chapter 3: Project Planning**
  Covers Work Breakdown Structure (WBS), timeline development, and Cost Breakdown Structure (CBS), detailing how the project activities are organized and scheduled.
- **Chapter 4: Requirements Collection**
  Specifies hardware and software requirements, along with tools such as Python, Flask, cryptography libraries, OpenCV, PIL, and SQLite database.
- **Chapter 5: System Design**
  Describes the existing and proposed system, system architecture, workflow, algorithms used, and modeling diagrams including use case, sequence, and DFDs.
- **Chapter 6: Implementation**
  Explains the development of input handling, key generation, encryption, steganography, decryption, user authentication, and user interface design. Screenshots and pseudocode are included to demonstrate functionality.
- **Chapter 7: Results and Discussion**
  Presents performance analysis, encryption/decryption times, key size

comparison, steganography quality, and graphical observations of system efficiency.

- **Chapter 8: Conclusion and Future Work**

  Summarizes the outcomes, potential applications, and suggests future enhancements for the proposed system.

Chapter 2

Literature Survey

The literature survey provides a comprehensive overview of existing research and developments related to secure data transmission, encryption techniques, and the application of elliptic curve cryptography (ECC) in modern communication systems. With the rapid growth of digital data exchange, ensuring security, confidentiality, and data integrity has become a primary concern, especially in mobile and cloud-based environments. Traditional encryption methods like RSA and AES, while effective, often face limitations in terms of computational complexity, energy efficiency, and scalability on resource-constrained devices. ECC has emerged as a strong alternative due to its smaller key size, faster computation, and robust security. This section reviews past studies and technologies that have contributed to the evolution of secure transmission systems, highlights their strengths and limitations, and identifies the gaps that the proposed ECC-Based Secure Transmission System aims to address.

Bernstein, D., et al. [**1**] introduced Curve25519, a high-performance elliptic curve specifically designed for secure key exchange. The research emphasizes the curve's resistance to timing and side-channel attacks, which are common vulnerabilities in cryptographic implementations. Curve25519 achieves high-speed computations without compromising security, making it ideal for constrained environments such as mobile devices and Internet-of-Things (IoT) systems. The authors conducted comprehensive analyses comparing Curve25519 with traditional ECC curves and demonstrated notable improvements in computational efficiency, particularly for establishing shared secrets using Elliptic Curve Diffie-Hellman (ECDH). While the

work provides a strong foundation for secure key exchange, it primarily focuses on key establishment rather than full multimedia encryption or real-time data transmission, leaving opportunities for integration with multimedia frameworks.

Kumar, R., and Singh, A. [2] conducted a detailed performance analysis of ECC algorithms applied to multimedia encryption, focusing on images and video files. The study compared the widely used secp256r1 and Curve25519 curves by evaluating encryption and decryption times, computational load, memory consumption, and overall system efficiency. Their results showed that ECC provides robust security with significantly better performance than conventional RSA encryption, especially for larger multimedia files. Notably, Curve25519 consistently outperformed secp256r1 in speed and resource utilization, making it suitable for real-time encryption applications. The authors also discussed practical challenges such as latency in video encryption, memory overhead, and maintaining data integrity across multiple media types. While their study confirms ECC's effectiveness for multimedia security, it does not address integration with steganography techniques or interactive user interfaces, which are essential for a complete secure transmission system.

Chen, L., and Li, H. [3] investigated the comparative performance and security characteristics of ECC and RSA in wireless data transmission scenarios. Their study highlighted that ECC achieves comparable security levels with significantly smaller key sizes, resulting in faster encryption and decryption, which is advantageous for bandwidth- and resource-constrained environments like mobile networks and IoT devices. Detailed benchmarking experiments were conducted to analyze encryption latency, key generation times, and computational overhead for various payload sizes, including real-time text and image transmissions. The study concluded that ECC not only reduces computational costs but also offers stronger security per bit of key size. However, the research primarily emphasizes key-level efficiency and does not explore integration with multimedia or steganographic embedding, leaving a gap for secure, covert data transmission frameworks.

Zhao, X., et al. [4] developed a secure cloud storage framework leveraging ECC to protect stored and transmitted multimedia data. The system employed both secp256r1 and Curve25519 for file encryption and key management, demonstrating that Curve25519 enables faster key generation and encryption without compromising security. The authors also addressed challenges in secure key sharing and management, providing a practical solution for protecting sensitive multimedia content in cloud environments. Additionally, the research examined the impact of ECC on storage efficiency and bandwidth utilization. While highly effective for cloud-based applications, the study did not explore real-time encryption or interactive web-based platforms, highlighting the need for user-friendly, real-time multimedia encryption systems.

Patel, N., and Mehta, P. [5] focused on securing healthcare data using ECC, including patient records, medical images, and real-time video streams in telemedicine applications. Their implementation utilized ECC-based encryption to maintain privacy and integrity, while achieving faster performance than symmetric key alternatives. Curve25519 was highlighted for its resistance to side-channel attacks and efficient computation. The study also emphasized the importance of real-time processing in critical applications, demonstrating the feasibility of ECC for performance-sensitive environments. However, the work is domain-specific to healthcare and does not extend to general multimedia encryption or the integration of steganographic data hiding.

Wang, Y., and Zhao, L. [6] investigated the optimization and implementation of ECC for resource-constrained devices, such as IoT sensors, embedded systems, and mobile platforms. Their study analyzed the performance of Curve25519 and secp256r1 with respect to encryption/decryption speed, memory usage, and resistance to side-channel attacks. Results confirmed that Curve25519 consistently outperforms secp256r1 in speed, computational efficiency, and security robustness, making it highly suitable for lightweight encryption in real-time applications. The paper also discussed implementation strategies for efficient ECC arithmetic, memory optimization, and power consumption reduction, which are critical for

battery-operated and low-resource devices. Despite its extensive focus on optimization, the research does not consider the integration of ECC with web-based interfaces, multimedia embedding, or steganography techniques, which are addressed in the proposed system.

Overall, these studies highlight the **advantages of ECC**, particularly Curve25519, for secure data transmission and real-time multimedia applications. Key trends include high-speed encryption/decryption, small key sizes, low computational overhead, and suitability for constrained devices. However, gaps remain in the integration of ECC with steganography, building interactive user interfaces, and evaluating real-time multimedia frameworks, which the proposed project addresses by combining ECC, AES encryption, and steganographic embedding within a unified Python Flask-based system. The proposed system leverages these findings to achieve secure, efficient, and covert multimedia transmission across text, image, and video data.

Chapter 3

Project Planning

Effective project planning ensures systematic execution of the Hate Speech Detection System by defining clear phases, resource allocation, scheduling, and effort estimation. The project follows an Agile development approach with iterative testing, model validation, and modular deployment.
The planning process emphasizes secure data handling, a scalable multimodal machine learning architecture, and a user-friendly interface design.

3.1 Work Breakdown Structure (WBS)

The Work Breakdown Structure (WBS), as shown in Table 3.1, organizes the proposed system into manageable tasks to ensure structured progress throughout the development lifecycle. It divides the project into logical phases covering requirement analysis, system design, implementation, testing, and documentation of the secure multimedia encryption system. Each task is assigned a specific timeline and responsible team member to improve accountability and track progress efficiently. The WBS also helps identify dependencies between tasks, enabling better resource allocation and risk management. By breaking down the project into smaller components, it ensures that every aspect of the system is systematically developed, tested, and integrated for a cohesive final product.

WBS Level	Description
1.0	Secure Multimedia Encryption and Steganography System Development
1.1	Requirement Analysis and System Specification
1.2	Backend Environment Setup and Flask Application Configuration
1.3	ECC Key Generation Module Implementation
1.4	AES Encryption and Decryption Module Development
1.5	Steganography Module for Text, Image, and Video
1.6	User Interface Design and Integration
1.7	System Testing, Performance Analysis, and Optimization
1.8	Documentation and Final Report Preparation

Table 3.1: Work Breakdown Structure (WBS) for Secure Multimedia Encryption and Steganography System

Figure 3.1 illustrates the effort distribution across requirement analysis, ECC key generation, AES encryption, steganography implementation for text, image, and video, user interface development, system testing, and documentation of the secure multimedia encryption system.

3.2 Timeline Development

The complete project timeline from April to November 2025 is illustrated in Figure 3.2. It outlines all major phases of the Secure Multimedia Encryption and Steganography System, including requirement analysis, backend and Flask setup, ECC key generation, AES encryption module development, steganography implementation for text, image, and video, user interface design, system testing, and final deployment. The timeline ensures systematic progress, timely completion of

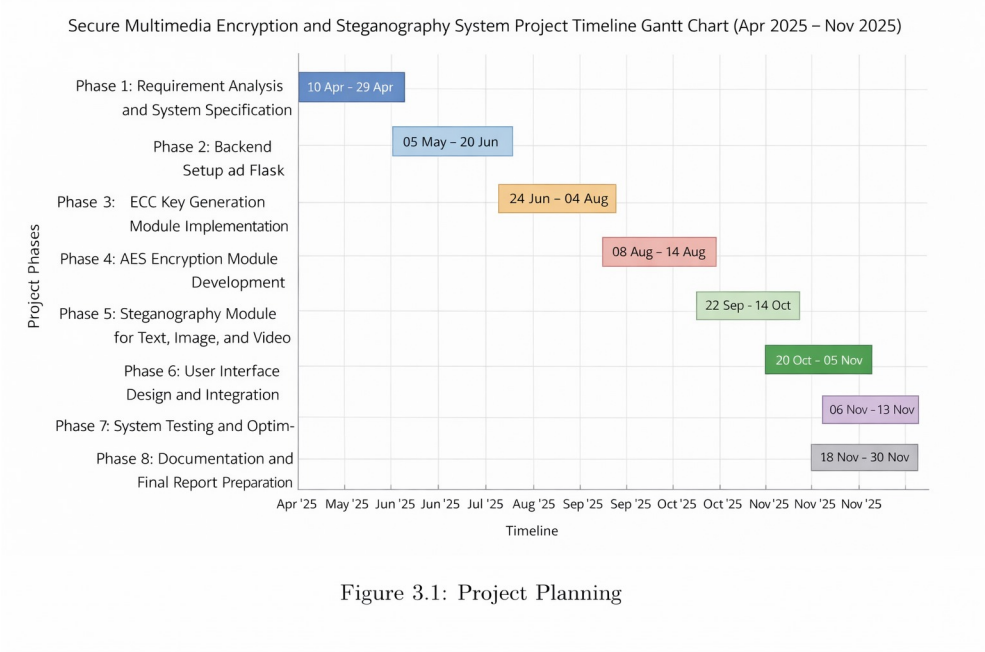

Figure 3.1: Project Planning

Figure 3.1: Effort Distribution of Secure Multimedia Encryption System

milestones, and effective coordination among encryption, steganography, and security validation tasks throughout the development lifecycle.

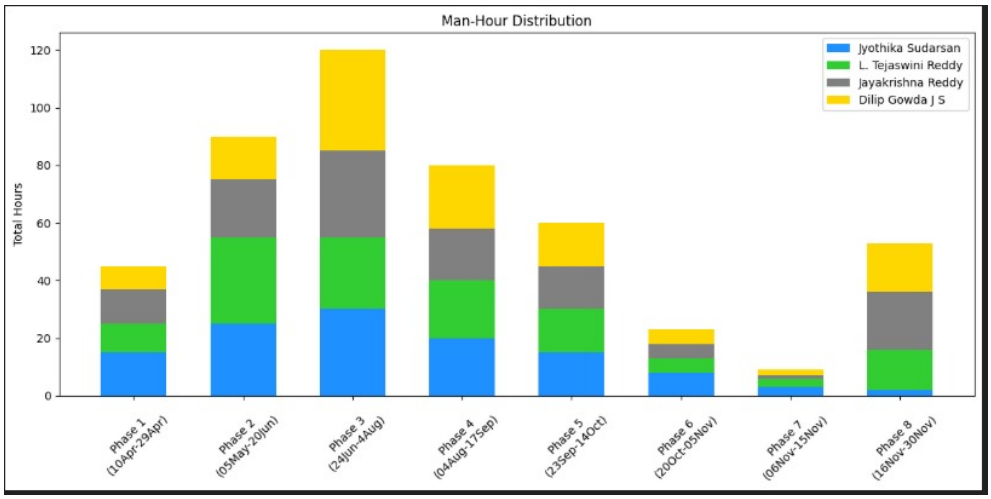

Figure 3.2: Project Timeline from April to November 2025

3.3 Cost Breakdown Structure (CBS)

The Cost Breakdown Structure (CBS) illustrates the distribution of academic effort across different phases of the project rather than direct financial expenditure. As the

work was carried out in an academic setting, the primary costs correspond to human effort, time investment, software usage, and documentation activities. Figure 3.4 presents the percentage-wise allocation of effort across all development phases of the Secure Multimedia Encryption and Steganography System.

Among all phases, Backend and Flask Setup (Phase 2) accounts for the largest share at 22%, representing the maximum amount of time spent during the project timeline. This phase required considerable effort for setting up the backend architecture, configuring the Flask environment, integrating the database, and ensuring reliable data flow between encryption, steganography, and user interface modules through iterative development and testing.

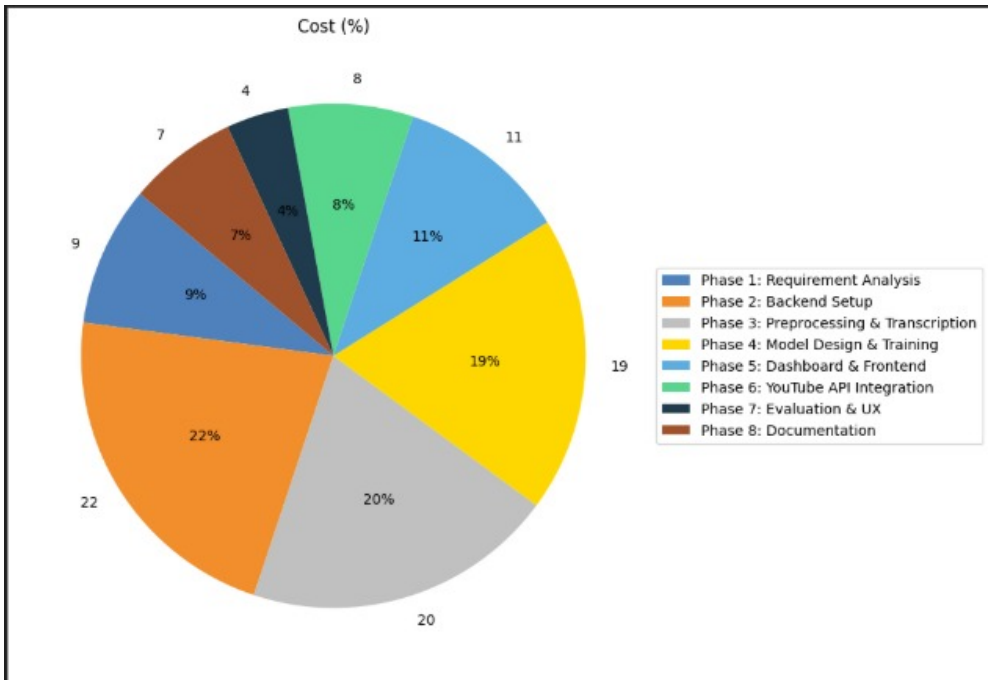

Figure 3.3: Cost Breakdown Structure

ECC Key Generation and AES Encryption Module Development (Phase 3) contributes 20% of the total project time. A significant portion of the schedule was spent on implementing key generation using Curve25519 and secp256r1, deriving AES keys, and integrating encryption and decryption functionalities to ensure secure multimedia processing.

Steganography Module Implementation (Phase 4) represents 19% of the overall

project time. This phase involved embedding encrypted data into text, image, and video cover files, designing extraction methods, and testing the robustness and imperceptibility of hidden data across different media types.

User Interface and Frontend Development (Phase 5) accounts for 11% of the total time spent. This duration was utilized for designing a user-friendly web interface, integrating frontend components with backend services, and providing seamless file upload, encryption, and decryption operations.

Testing and System Integration (Phase 6) required 8% of the project time, focusing on validating the correct operation of encryption, steganography, data storage, and retrieval, as well as identifying and resolving implementation issues.

Requirement Analysis and Planning (Phase 1) accounts for 9% of the total time invested and includes early-stage activities such as problem definition, requirement gathering, feasibility analysis, and module-level planning.

The remaining project time is distributed between System Evaluation and Performance Optimization (Phase 7) at 4%, involving encryption and decryption performance analysis, steganography quality assessment, and system fine-tuning, and Documentation (Phase 8) at 7%, covering the preparation of technical documentation, diagrams, report writing, and final project submission materials.

Chapter 4

Requirements Collection

The proposed system is designed to provide secure communication for text, image, and video data using Elliptic Curve Cryptography combined with steganography. It ensures both data confidentiality and invisibility while maintaining efficient performance. The system is implemented as a web-based application using Python and Flask, supporting key generation, encryption, embedding, extraction, and decryption. This chapter specifies the software and hardware requirements needed to develop and execute the project successfully.

4.1 Software and Hardware Requirements

This subsection lists the essential software tools and hardware components required for the successful development and execution of the proposed system. The specifications ensure smooth implementation and efficient performance of the Flask-based ECC and steganography application.

4.1.1 Software Requirements

A software requirement is a description of the software applications that will be used or expected to be used to implement the proposed system. These requirements briefly describe the features and functionality necessary for creating a secure Flask-based ECC and steganography application. Requirements typically define the

user's expectations from the software product.

- **Programming Environment:** Anaconda IDE or any Python 3.x compatible IDE
- **Operating System:** Windows 7, 8, 10 or Linux distributions
- **Software Tool:** Python 3.8 or above
- **Execution Platform:** Web browser to access the Flask application

4.1.2 Hardware Requirements

Hardware requirements are the specifications for the physical components of the system that will be used to implement the proposed application. These components ensure smooth execution of encryption, decryption, and steganography processes.

- **Processor:** Intel Core i5 or equivalent
- **RAM:** 8 GB or more
- **Hard Disk:** 500 GB or higher
- **Graphics:** Integrated or dedicated GPU (optional)

4.2 Software Tools

The necessary software tools used for the development of this project are listed below. These tools provide the environment and functionalities required for encryption, decryption, steganography, and the web-based user interface in the Flask application.

- **Python Programming Language** (Version 3.8 or above)
- **Flask Web Framework** (Version 2.x)
- **Cryptography Library**

- **OpenCV and PIL Libraries**
- **SQLite Database**

4.2.1 Python Programming Language

Python is a high-level, general-purpose programming language that is widely used in web development, cryptography, and image and video processing. Python's simple syntax and extensive libraries make it ideal for implementing the encryption and steganography algorithms of this project. Python version 3.8 or above is used in this project.

The advantages of using Python in this project are:

- Powerful and easy-to-use language
- Free and open source
- Large library support for cryptography, image, and video processing
- High development efficiency
- Easy integration with Flask for web-based applications

4.2.2 Flask Web Framework

Flask is a lightweight Python web framework used to develop the user interface and handle HTTP requests in this project. It allows seamless integration of the encryption, steganography, and database modules.

The advantages of using Flask are:

- Lightweight and flexible framework
- Easy integration with Python libraries
- Supports rapid web application development
- Provides a secure platform for user authentication and file management

4.2.3 Cryptography Library

The Cryptography library in Python provides modules for ECC, AES, key generation, and secure cryptographic operations. This library ensures robust encryption and decryption in the system.

The advantages are:

- Provides strong security with standard algorithms
- Supports ECC curves such as Curve25519 and secp256r1
- Easy integration with Python applications
- Maintains secure key management for the project

4.2.4 OpenCV and PIL Libraries

OpenCV and PIL libraries are used for image and video processing required in steganography. They allow embedding and extraction of secret data in multimedia files.

The advantages are:

- Supports multiple image and video formats
- Efficient and fast processing of multimedia data
- Provides functions for pixel-level manipulation
- Compatible with Python for seamless integration

4.2.5 SQLite Database

SQLite is a lightweight, file-based database used to store user credentials, cryptographic keys, and operation logs. It ensures persistent and reliable storage for the Flask application.

The advantages are:

- Simple and easy to set up
- Lightweight and fast for small to medium projects
- Provides secure storage of keys and logs
- Integrates easily with Python applications

Chapter 5

System Design

The system design for the ECC-based secure transmission framework combines public-key cryptography with steganography to securely transfer images, videos, and text. The application is implemented as a local-host web application using Flask, where the sender encrypts and embeds a message into a carrier file, and the receiver extracts and decrypts it. For testing purposes, the receiver's private key is sent via email.

5.1 Existing System

In the existing system, multimedia data such as text, images, and videos are often transmitted over public networks without sufficient protection, making them vulnerable to interception, unauthorized access, and tampering. Traditional encryption techniques such as RSA and DES are used to secure data; however, they require larger key sizes and higher computational resources, which reduce efficiency, especially for multimedia files. Additionally, existing systems do not provide mechanisms to hide encrypted data, making it visible to attackers even if it is encrypted.

Furthermore, most current solutions lack a unified framework that combines cryptography and steganography with a user-friendly web interface, resulting in complex implementation and limited accessibility. This project addresses these limitations by implementing ECC-based encryption along with steganography

techniques in a Flask-based web application, providing both security and ease of use.

5.2 Proposed System

The proposed system provides a secure and efficient framework for transmitting text, images, and videos by combining Elliptic Curve Cryptography (ECC) with steganography techniques. Implemented as a Flask-based web application, it allows users to generate encryption keys, encrypt data, embed it within multimedia files, and securely transmit it to the receiver.

The system uses Curve25519 and secp256r1 for encryption, while text-in-image, image-in-image, and video-in-video steganography techniques ensure the hidden transmission of data. This integrated approach enhances confidentiality, reduces computational overhead, and provides a user-friendly interface for secure multimedia communication.

5.3 System Architecture

5.3.1 System Architecture Description

The system architecture of the proposed secure multimedia encryption system is designed to provide seamless integration of encryption, decryption, and steganography functionalities. As shown in Figure 4.1, the architecture consists of a web interface, a backend implemented using Python and Flask, an ECC encryption and decryption module, a steganography module, and a storage module.

The user interacts with the system through the web interface by uploading text, image, or video files. The data is encrypted using ECC algorithms (Curve25519 or secp256r1) and optionally embedded into cover media using steganography techniques. The encrypted data is securely stored or transmitted, and upon request, it is extracted and decrypted to retrieve the original content. This modular architecture ensures secure storage, confidentiality, and efficient multimedia

processing.

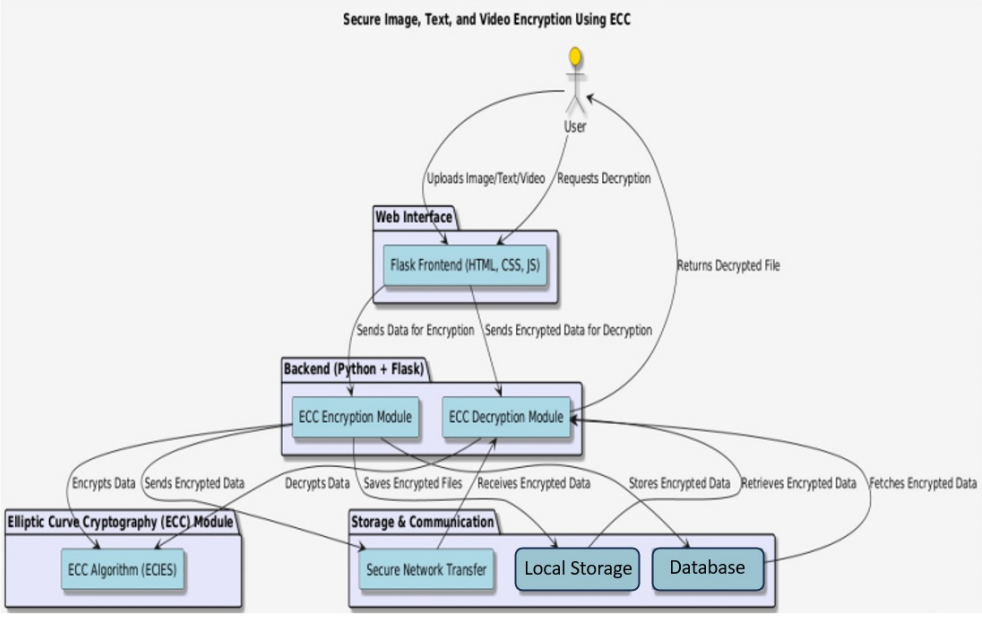

Figure 5.1: System Architecture Diagram

5.4 Workflow of the Proposed System

The workflow of the proposed system provides a step-by-step overview of how multimedia data is securely transmitted using ECC encryption and steganography techniques. The workflow is summarized as follows

I. The user uploads text, image, or video files through the Flask-based web interface.

II. The system validates the input and retrieves the appropriate ECC encryption keys.

III. The ECC encryption module encrypts the data using Curve25519 or secp256r1.

IV. The steganography module embeds the encrypted data into a cover image or video.

V. The encrypted and embedded files are securely stored or transmitted.

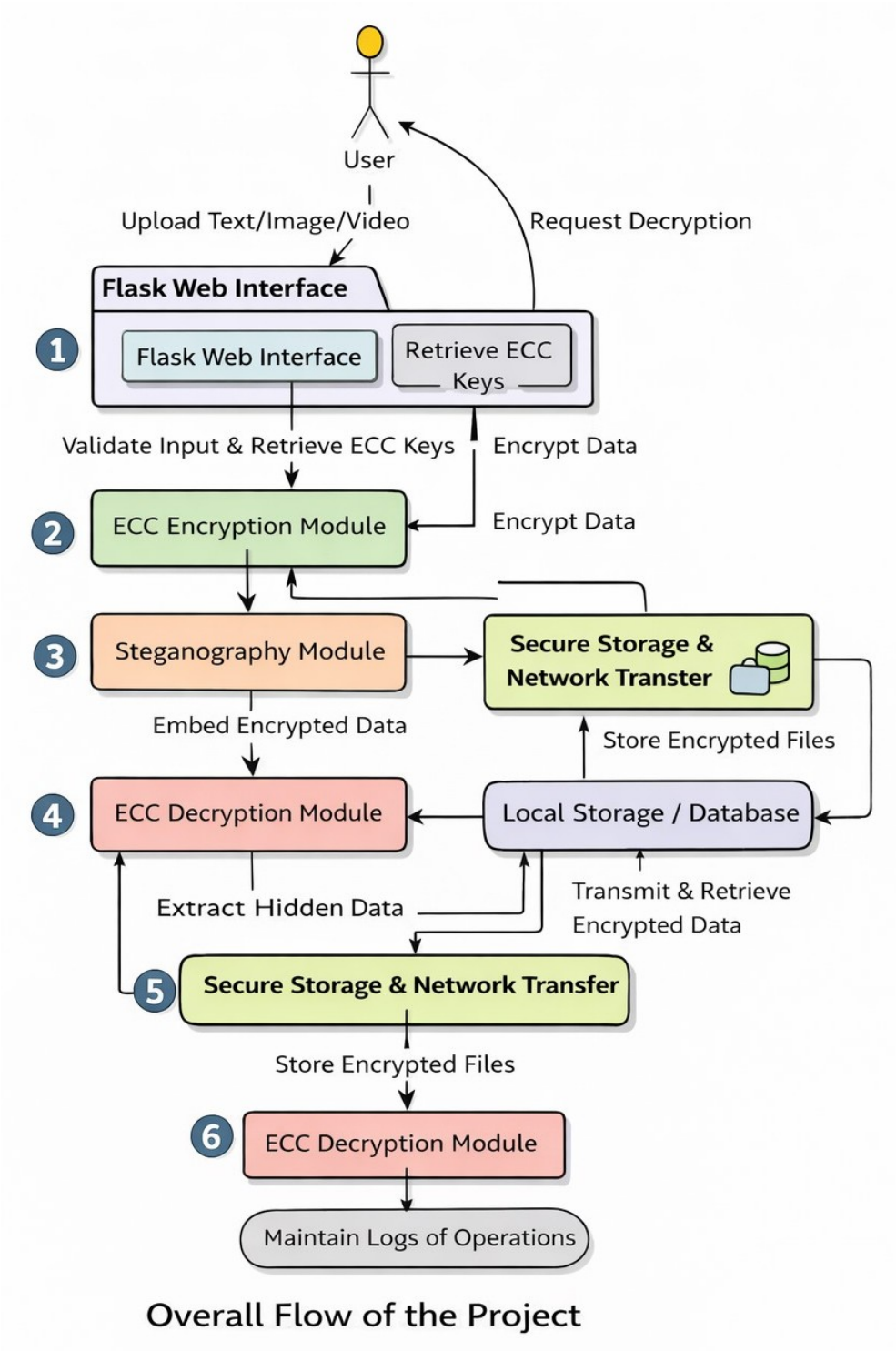

Figure 5.2: Workflow of Proposed System

VI. Upon request, the system extracts the hidden data using steganography techniques.

VII. The ECC decryption module decrypts the extracted data.

VIII. The original content is returned to the user, and logs are maintained for auditing.

5.5 Algorithms Used in the Project

This section describes the algorithms implemented in the proposed system to achieve secure encryption, decryption, and data hiding of multimedia content. The selected algorithms ensure strong security, efficient performance, and reliable integration of cryptography and steganography within the Flask-based application.

5.5.1 Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) is a public key cryptographic technique that provides strong security with smaller key sizes compared to traditional algorithms such as RSA. ECC is based on the mathematical properties of elliptic curves defined over finite fields.

In this project, ECC is used to securely encrypt multimedia data before embedding it using steganography. ECC offers efficient key generation and reduced computational overhead, making it suitable for multimedia applications.

The advantages of ECC are:

- High security with smaller key sizes compared to RSA or DSA.
- Faster encryption and decryption due to reduced computational overhead.
- Efficient key generation and exchange using ECDH.
- Lower power and memory consumption, suitable for resource-constrained devices.
- Strong resistance to known attacks including side-channel and timing attacks.

5.5.2 Curve25519 Algorithm

Curve25519 is a modern elliptic curve designed for high-speed cryptographic operations. It provides strong security with efficient computation and is widely used in secure communication systems.

In this project, Curve25519 is used for encrypting multimedia data. Experimental analysis shows that it achieves lower encryption and decryption time compared to secp256r1.

The advantages are:

- Faster encryption and decryption
- Smaller key size
- Reduced computational overhead
- Suitable for real-time applications

5.5.3 secp256r1 Algorithm

secp256r1 is a standardized elliptic curve defined by NIST and operates over a 256-bit prime field. It provides strong security and is widely supported in cryptographic libraries.

Although secure, secp256r1 requires higher computation time compared to Curve25519, making it slightly less efficient for large multimedia data.

The advantages are:

- Strong standardized security
- Widely supported
- Resistant to cryptographic attacks
- Suitable for compliance-based applications

5.5.4 AES Encryption using ECC-Derived Key

In the proposed system, AES is used for encrypting multimedia data, while the encryption key is generated using ECC through the ECDH mechanism.

Let P be the base point, d_1 and d_2 be private keys:

$$Q_1 = d_1 \times P, \quad Q_2 = d_2 \times P$$

Shared secret:

$$K = d_1 \times Q_2 = d_2 \times Q_1$$

The shared secret is processed using a key derivation function to generate an AES key.

The advantages are:

- Secure key exchange
- Fast encryption for large data
- Reduced computational overhead
- Hybrid security approach

5.5.5 Steganography Algorithm

Steganography conceals secret information within digital media such that its existence is not noticeable. In this project, LSB-based steganography is used to embed encrypted data into images and videos.

The process involves modifying the least significant bits of pixels to store encrypted data while preserving visual quality.

The advantages are:

- Imperceptible data hiding
- Additional security layer

- Supports text, image, and video
- Resistant to unauthorized detection

5.6 System Modelling

System modelling represents the functional behavior and interaction of different components in the proposed system. It helps in visualizing how users interact with the application and how internal modules coordinate to perform secure encryption, steganography, and decryption operations. The system modelling in this project is illustrated using a Use Case Diagram and a Sequence Diagram, which provide a clear understanding of system functionality and execution flow.

5.6.1 Use Case Diagram

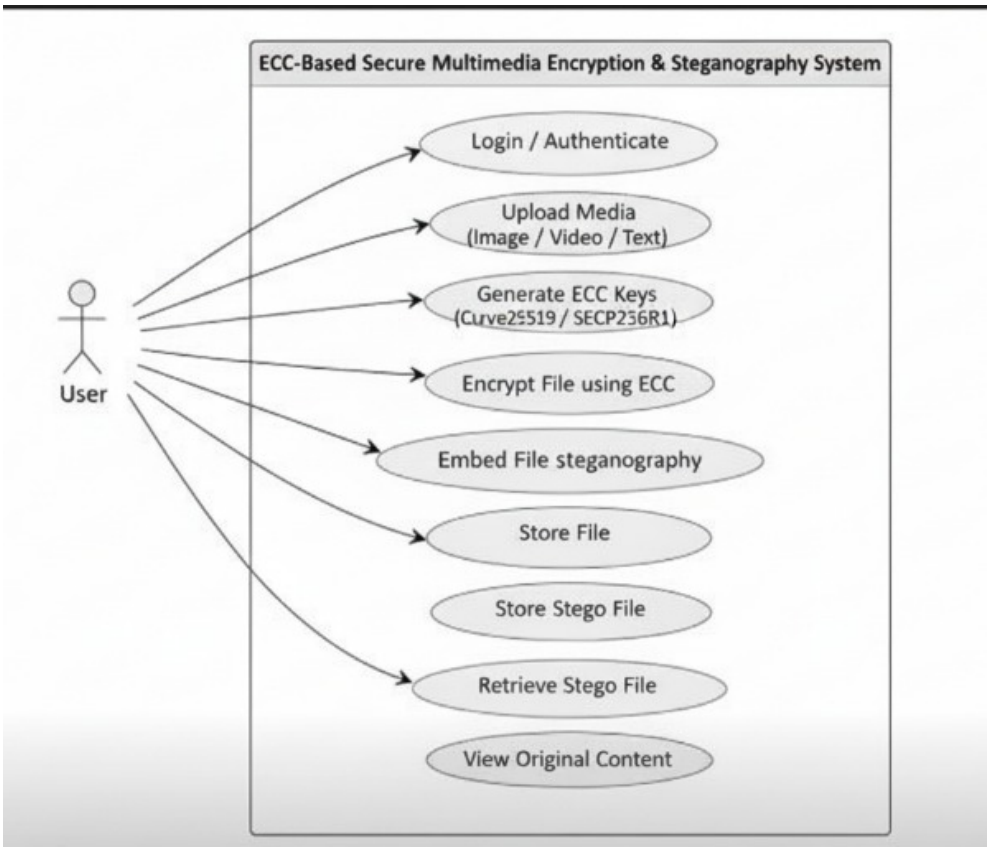

Figure 5.3: Usecase Diagram of Proposed System

The Use Case Diagram illustrates the interaction between the user and the proposed

secure multimedia system. It represents the various functionalities provided by the system, such as key generation, encryption, steganography embedding, decryption, and data extraction. The diagram clearly identifies the user as the primary actor who initiates all operations through the Flask-based web interface.

The use case diagram helps in understanding the overall system behavior by depicting how different actions are performed by the user and processed by the backend modules. It also highlights the access control and flow of operations involved in secure multimedia communication.

5.6.2 Sequence Diagram

The sequence diagram illustrates the step-by-step interaction between the sender and receiver during secure key generation and shared secret establishment using Elliptic Curve Cryptography (ECC). It demonstrates how both parties independently compute the same symmetric key without directly transmitting it.

The sequence of operations is as follows:

I. The sender generates a private key (a) and computes the corresponding public key $A = a \times G$, where G is the base point on the elliptic curve.

II. The receiver generates a private key (b) and computes the corresponding public key $B = b \times G$.

III. The sender and receiver exchange their public keys A and B over an insecure communication channel.

IV. The sender computes the shared secret $S = a \times B$ using its private key and the receiver's public key.

V. The receiver computes the shared secret $S = b \times A$ using its private key and the sender's public key.

VI. Both the sender and receiver independently obtain the same shared secret S, which is mathematically identical.

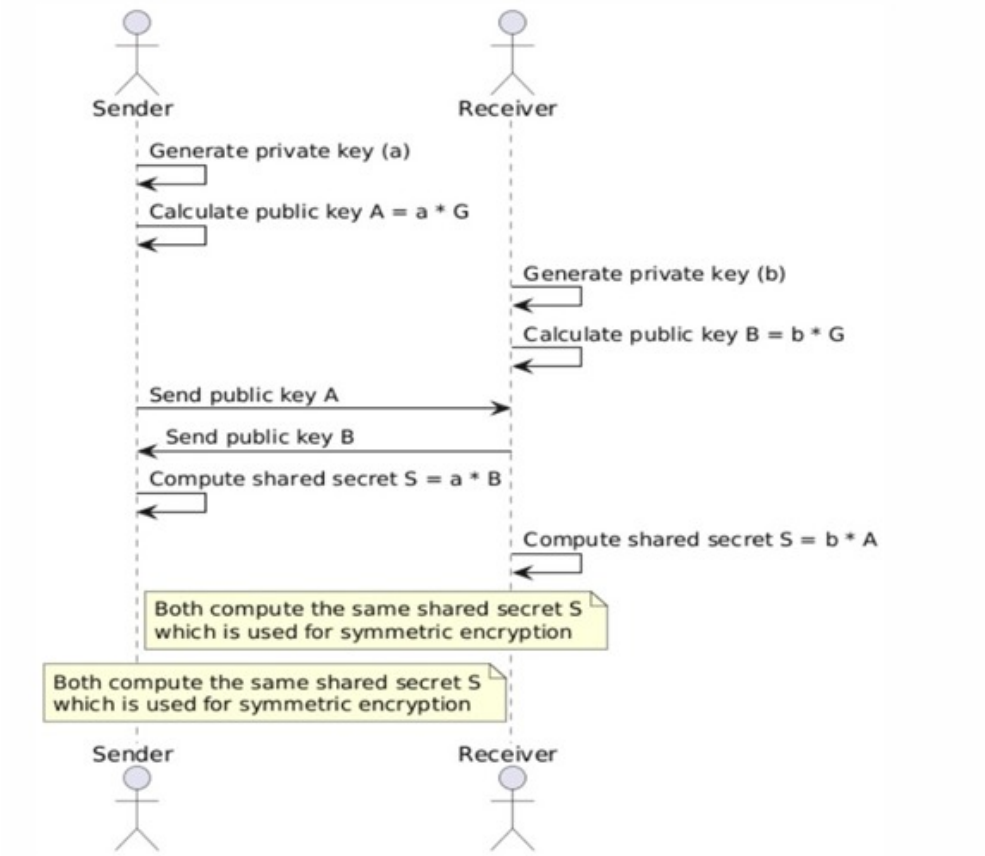

Figure 5.4: Sequence Diagram of the Proposed System

VII. The shared secret S is used as the symmetric key for encrypting and decrypting multimedia data using AES.

This sequence ensures that the encryption key is never transmitted directly, providing strong protection against eavesdropping and unauthorized access.

5.6.3 Data Flow Diagrams (DFD)

The Data Flow Diagrams illustrate the flow of data within the proposed secure multimedia encryption system, highlighting interactions between modules and external entities.

Level 0 DFD

The Level 0 DFD provides a high-level view of the system, showing interactions between the system and external entities such as the User and Storage. It illustrates how user inputs are processed and how encrypted outputs are generated.

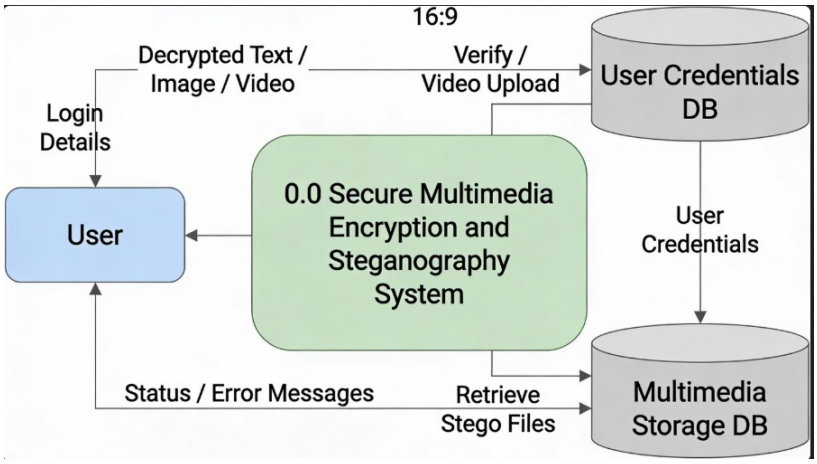

Figure 5.5: Level 0 DFD of the Proposed System

Level 1 DFD

The Level 1 DFD decomposes the system into its main functional modules: User Input Validation, ECC Key Generation and AES Encryption, Steganography Embedding, Decryption Data Extraction, and Storage Retrieval.

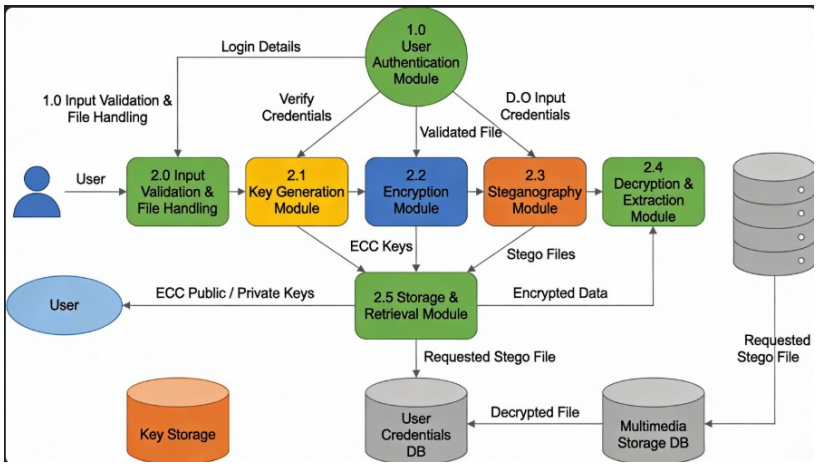

Figure 5.6: Level 1 DFD of the Proposed System

Chapter 6

Implementation

6.1 Input Data Description (Text, Image, and Video Files)

The proposed system supports multiple types of input data, including text files, image files, and video files, to ensure flexible and secure multimedia communication. Text input consists of plain textual content entered by the user or uploaded as a text file. Image inputs include standard formats such as JPEG and PNG, while video inputs include commonly used formats such as MP4 and AVI.

All input files are validated at the application level to ensure supported file formats, size constraints, and data integrity before processing. Once validated, the input data is converted into a suitable binary representation for encryption using ECC and AES algorithms. This unified input handling mechanism allows the system to securely process different multimedia data types using a common encryption and steganography framework.

6.2 Key Generation Module

The key generation module is responsible for generating secure cryptographic keys required for encryption and decryption operations. In the proposed system, Elliptic

Curve Cryptography (ECC) is used to generate public and private key pairs based on Curve25519 and secp256r1 algorithms.

The module generates private keys using cryptographically secure random number generators and computes corresponding public keys using elliptic curve point multiplication. These keys are used to derive shared secret keys through the Elliptic Curve Diffie–Hellman (ECDH) mechanism. The derived shared secret is further processed using a key derivation function to generate a symmetric AES key.

This approach ensures secure key exchange without transmitting the encryption key directly and provides strong protection against brute-force and man-in-the-middle attacks.

6.3 Encryption Module

The encryption module ensures confidentiality of multimedia data by applying hybrid encryption techniques. In this module, ECC is used for secure key exchange, and AES is used for fast encryption of large multimedia data.

Before encryption, the input data is converted into a byte stream. The AES key derived from ECC is then used to encrypt the data using secure encryption modes. The encrypted output is passed to the steganography module for hidden transmission.

6.3.1 Encryption using Curve25519

Curve25519 is used in this project to perform high-speed and efficient ECC-based key generation and encryption. Due to its optimized design and smaller key size, Curve25519 provides faster encryption and decryption with strong security guarantees.

In this implementation, Curve25519 is used to generate public-private key pairs and compute the shared secret key. The derived AES key is then used to encrypt multimedia data. Experimental observations show that Curve25519 offers lower encryption time compared to secp256r1, making it suitable for real-time multimedia

applications.

6.3.2 Encryption using secp256r1

secp256r1 is a standardized elliptic curve defined by NIST and is widely used in secure communication systems. In this project, secp256r1 is implemented as an alternative encryption option to analyze and compare its performance with Curve25519.

Although secp256r1 provides strong security, it requires higher computational effort due to its larger key size. The encrypted data generated using secp256r1 follows the same processing pipeline as Curve25519, allowing fair comparison in terms of encryption time, decryption time, and key size.

6.4 Steganography Module

The steganography module provides an additional layer of security by hiding encrypted data within multimedia files. After encryption, the ciphertext is embedded into cover media using Least Significant Bit (LSB) based steganography techniques.

This module ensures that the visual quality of cover images and videos remains unchanged, making the presence of hidden data imperceptible to attackers.

6.4.1 Text in Image Implementation

In the text-in-image implementation, encrypted text data is embedded into a cover image by modifying the least significant bits of pixel values. The encrypted text is first converted into a binary stream and then embedded sequentially into the image pixels.

This method ensures that the embedded data does not cause noticeable visual distortion while allowing accurate extraction and decryption at the receiver side.

6.4.2 Image in Image Implementation

In image-in-image steganography, an encrypted image is embedded into another image acting as the cover. The encrypted image data is converted into a binary format and embedded into the least significant bits of the cover image pixels.

This technique enables secure transmission of image data while maintaining the quality and integrity of the cover image.

6.4.3 Video in Video Implementation

Video-in-video steganography is implemented by embedding encrypted video data into selected frames of a cover video. The encrypted data is distributed across multiple frames to reduce perceptibility and enhance security.

This approach ensures smooth playback of the cover video while securely hiding encrypted video content within it.

6.5 Decryption and Extraction Module

The decryption and extraction module performs the reverse operations of encryption and steganography. The system first extracts the encrypted data from the stego image or video using the corresponding extraction technique.

Once extracted, the encrypted data is decrypted using the AES key derived through ECC key exchange. The original text, image, or video content is then reconstructed and presented to the user. This module ensures accurate recovery of original data while maintaining data integrity and confidentiality.

6.6 User Authentication and Database Module

The user authentication module ensures secure access to the system by validating registered users through login credentials. User information, encryption keys, and operation logs are stored securely in an SQLite database.

This module prevents unauthorized access, maintains session control, and records system activities for auditing and performance analysis. Database integration also enables persistent storage of encrypted files and metadata.

6.7 Pseudocode for Encryption and Decryption

The pseudocode representation provides a high-level understanding of the encryption and decryption workflow implemented in the proposed system. It abstracts the logic without exposing implementation-specific details.

Encryption Process

- Accept multimedia input data
- Generate ECC public and private keys
- Derive shared secret using ECDH
- Generate AES symmetric key
- Encrypt multimedia data using AES
- Embed encrypted data using steganography
- Store or transmit stego file

Decryption Process

- Receive stego file
- Extract encrypted data using steganography
- Derive AES key using ECC
- Decrypt data using AES
- Recover original multimedia content

6.8 User Interface Design and Implementation

The user interface of the proposed Secure Multimedia Encryption and Steganography System is implemented using the Flask web framework with HTML, CSS, and JavaScript. The interface is designed to be simple, intuitive, and user-friendly, enabling users to securely perform encryption, steganography, and decryption operations for text, image, and video data.

Figure 6.1: Home Page of the Secure Multimedia Data Transmission using ECC

The home page acts as the primary entry point of the application. It provides clear navigation options to access ECC key generation, encryption, steganography, and decryption modules through an organized layout.

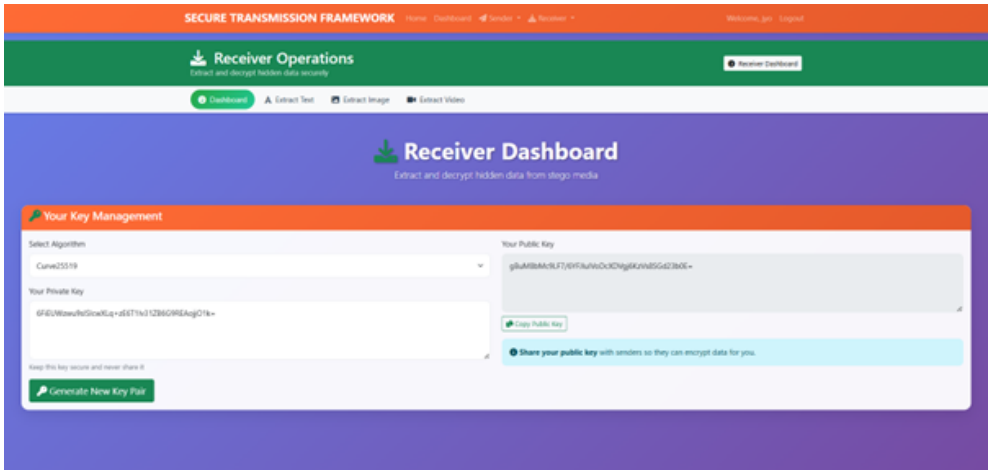

Figure 6.2: ECC Key Generation Interface

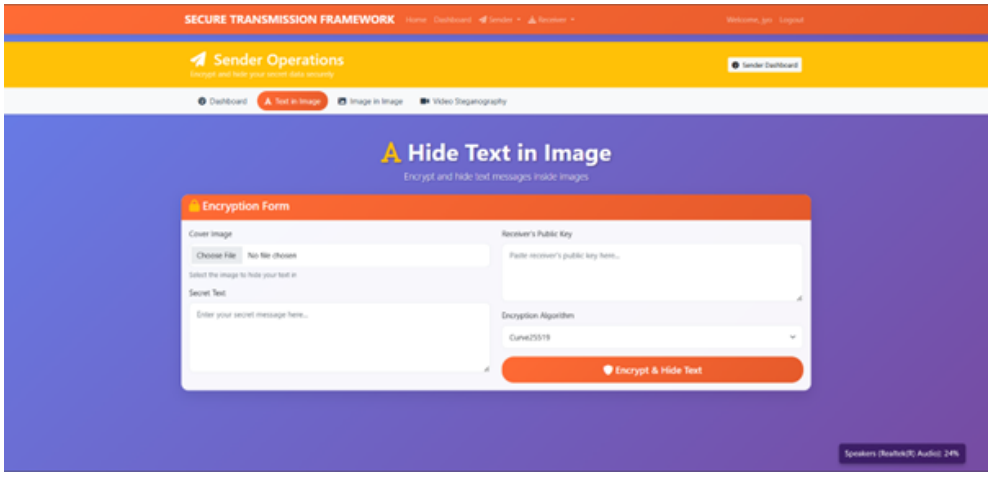

Figure 6.3: Text Encryption Interface

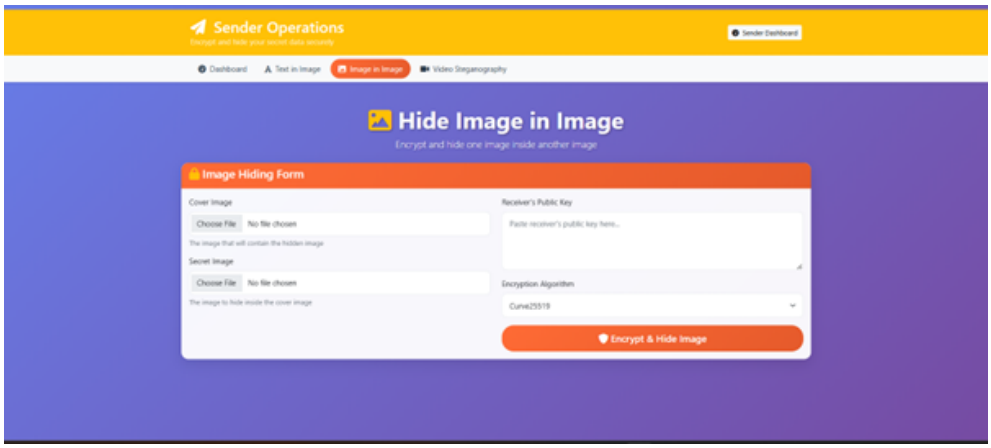

Figure 6.4: Image Steganography Interface

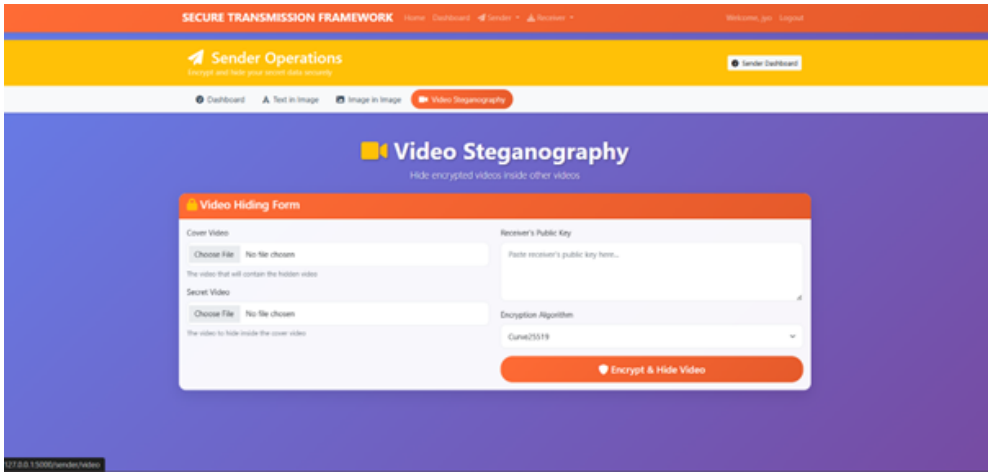

Figure 6.5: Video Steganography Interface

The encryption module secures the input text, image, and video data using Elliptic Curve Cryptography (ECC) for key exchange and Advanced Encryption Standard (AES) for fast multimedia encryption. It ensures confidentiality and efficient processing of all data types.

The steganography module conceals the encrypted data within cover images and videos using Least Significant Bit (LSB) based techniques. It supports text-in-image, image-in-image, and video-in-video embedding while maintaining high visual quality and imperceptibility.

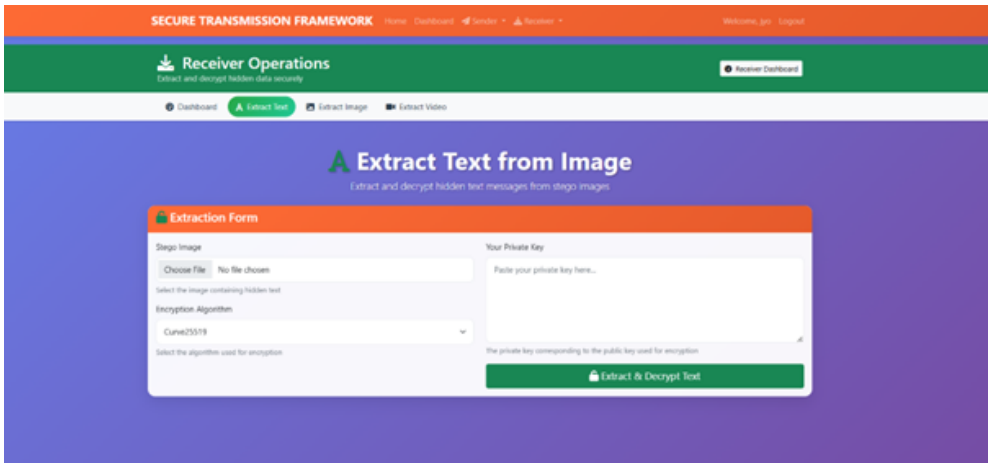

Figure 6.6: Decryption and Data Extraction Interface

The decryption and data extraction interface allows users to retrieve hidden encrypted data from steganographic images or videos and recover the original content. The module extracts the embedded ciphertext using the corresponding steganography technique and decrypts it using the ECC-derived AES key, ensuring accurate and lossless reconstruction of text, image, or video data.

Chapter 7

Results and Discussion

7.1 Encryption Time Analysis

The encryption performance of the proposed system was evaluated for text, image, and video data using both Curve25519 and secp256r1 elliptic curves. Experiments showed that Curve25519 consistently required less time to encrypt data across all multimedia types. For text and small images, the encryption times were almost negligible, demonstrating the suitability of both curves for small payloads. However, Curve25519 offered approximately 30–40% faster encryption than secp256r1, highlighting its computational efficiency. Video encryption required longer processing time due to the larger file size and multiple frames, yet Curve25519 still maintained a faster encryption rate than secp256r1.

Multimedia Type	**Curve25519 (s)**	**secp256r1 (s)**	**Observation**
Text-in-Image	0.0019	0.0033	Curve25519 is faster
Image-in-Image	0.0021	0.0029	Curve25519 is faster
Video-in-Video	2.3228	2.8090	Curve25519 is faster

Table 7.1: Encryption Time Analysis

7.2 Decryption Time Analysis

Decryption performance was also measured to assess the reliability of recovering original data. Similar to encryption, Curve25519 achieved faster decryption times

than secp256r1 across text, image, and video files. The system successfully decrypted all files with 100% accuracy, ensuring no data loss or corruption. Decryption time for videos was higher because of the additional frame-by-frame extraction, but the process remained stable and reliable, confirming the system's robustness for different multimedia types.

Multimedia Type	Curve25519 (s)	secp256r1 (s)	Observation
Text-in-Image	0.0004	0.0007	Curve25519 is faster
Image-in-Image	0.0008	0.0011	Curve25519 is faster
Video-in-Video	4.1410	4.7256	Curve25519 is faster

Table 7.2: Decryption Time Analysis

7.3 Key Size Comparison

Both Curve25519 and secp256r1 use 256-bit keys, providing equivalent levels of cryptographic security. Despite having the same key length, Curve25519 outperforms secp256r1 due to its optimized field arithmetic and efficient computation, which translates into faster encryption and decryption without sacrificing security. This makes Curve25519 more suitable for time-sensitive multimedia applications.

Elliptic Curve	Key Size (bits)	Security Level	Observation
Curve25519	256	High	Optimized arithmetic, faster operations
secp256r1	256	High	Slightly slower due to higher computation

Table 7.3: Key Size Comparison

7.4 Performance Comparison between Curve25519 and secp256r1

The overall performance comparison indicates that Curve25519 provides significant advantages in terms of computational speed while maintaining the same security standards as secp256r1. For small and medium-sized files, the improvement in

processing time is substantial, while for larger files such as videos, Curve25519 continues to show faster performance with slightly reduced latency. Both curves ensure secure key exchange and strong cryptographic protection, but Curve25519's efficiency makes it ideal for real-time or resource-constrained environments.

Parameter	**Curve25519**	**secp256r1**	**Observation**
Encryption Time (average)	Lower	Higher	Curve25519 faster
Decryption Time (average)	Lower	Higher	Curve25519 faster
Computational Overhead	Low	Moderate	Curve25519 more efficient
Security Level	High	High	Both secure

Table 7.4: Performance Comparison between Curve25519 and secp256r1

7.5 Steganography Quality Analysis

The steganography module was tested to ensure that encrypted data could be embedded within multimedia files without noticeable degradation. For text-in-image embedding, encrypted text remained invisible, and extraction achieved perfect accuracy. In image-in-image experiments, the visual quality of the stego-image was almost identical to the original cover image, and the embedded image could be fully recovered. Video-in-video embedding maintained smooth playback, and hidden frames were correctly extracted. These results confirm that the steganography implementation effectively conceals encrypted data while preserving the original media quality.

Steganography Type	**Media**	**Accuracy**	**Observation**
Text-in-Image	Image	100%	No visible distortion
Image-in-Image	Image	100%	Embedded image recovered correctly
Video-in-Video	Video	100%	Smooth playback without frame loss

Table 7.5: Steganography Quality Analysis

7.6 Graphical Analysis and Observations

Visual analysis of performance metrics highlights several key findings. Encryption and decryption times, plotted against different data types, clearly demonstrate the faster execution of Curve25519. Steganography quality metrics such as PSNR and SSIM show minimal degradation, indicating high imperceptibility of hidden data. Overall, the combined ECC, AES, and steganography framework achieves secure, efficient, and covert multimedia transmission. The Flask-based interface further enhances usability by providing real-time feedback on encryption and decryption operations, ensuring a smooth and interactive user experience.

Metric	**Curve25519**	**secp256r1**	**Observation**
Encryption Time Trend	Lower	Higher	Faster for all media types
Decryption Time Trend	Lower	Higher	Faster for all media types
Steganography Imperceptibility	High	High	No noticeable degradation
User Interface Feedback	Real-time	Real-time	Smooth and responsive for both

Table 7.6: Graphical Analysis and Observations

Chapter 8

Conclusion and Future Work

8.1 Conclusion

This project successfully presents a secure multimedia transmission framework that integrates Elliptic Curve Cryptography (ECC), AES symmetric encryption, and steganography within a unified Python Flask–based web application. The proposed system ensures data confidentiality, integrity, and covert communication for different forms of multimedia data, including text, images, and videos.

The core security mechanism of the system is based on the Elliptic Curve Diffie–Hellman (ECDH) key exchange protocol, which enables the sender and receiver to securely generate a shared secret without directly transmitting the encryption key. This shared secret is further used for AES encryption, providing strong protection against unauthorized access. The encrypted data can optionally be embedded into multimedia files using steganography, offering an additional layer of security by concealing the existence of the communication itself.

Experimental evaluation of the system demonstrated 100% accuracy in data recovery across all test cases, confirming the correctness of encryption, decryption, embedding, and extraction processes. Performance analysis showed that Curve25519 consistently outperformed secp256r1 in terms of encryption and decryption time while maintaining equivalent security strength. Furthermore, steganographic embedding resulted in minimal visual distortion in both images and videos, preserving the quality of the cover media.

Although the system performed efficiently for text and image data, video encryption

and steganography required higher processing time due to larger data size and frame complexity. Nevertheless, successful video-in-video encryption and extraction validated the feasibility of secure multimedia transmission in a local environment. Overall, the proposed framework establishes a strong and reliable foundation for secure data exchange by combining lightweight cryptographic techniques with multimedia data hiding methods.

8.2 Future Scope

While the proposed system achieves secure and covert multimedia communication in a local setup, several enhancements can be explored to improve performance and extend its applicability to real-world scenarios. One important direction for future work is performance optimization, particularly for video encryption and steganography. Techniques such as parallel processing, GPU acceleration, or optimized multimedia codecs can be employed to reduce processing latency.

Another promising extension is distributed and cloud-based deployment, enabling secure multimedia transmission over networks instead of a local environment. Integrating the system with cloud platforms can support scalable storage and real-time secure data sharing. Additionally, adaptive steganography techniques can be incorporated to dynamically adjust embedding strategies based on media characteristics, further improving imperceptibility and robustness.

Future research may also focus on integrating post-quantum cryptographic algorithms or advanced elliptic curve variants to enhance resistance against quantum computing attacks. Expanding the framework to support cross-platform applications, including mobile and desktop clients, would make the system more accessible to users. Finally, extending the system for real-time encrypted multimedia streaming and secure video conferencing can transform the proposed framework into a comprehensive multimedia security platform suitable for modern communication environments.

8.3 Applications

The developed Secure Multimedia Encryption and Steganography System has wide-ranging applications in areas where secure and covert communication is critical. It can be employed in military and defense sectors for the safe transmission of classified multimedia data such as encrypted images, maps, or videos. In the healthcare domain, the system ensures the confidentiality of patient medical images, reports, and video consultations, supporting secure telemedicine and remote monitoring. It also facilitates digital watermarking and copyright protection by embedding hidden ownership information within multimedia files. For cloud storage, the framework encrypts multimedia content before uploading, protecting sensitive data from unauthorized access. Additionally, it can be used in forensic investigations to securely transfer and store sensitive evidence files. Furthermore, the system is suitable for integration into private messaging and multimedia sharing platforms, where privacy, authentication, and data integrity are of paramount importance. By combining robust cryptographic techniques with steganography, the system provides a comprehensive solution for data confidentiality and imperceptible information hiding across multiple industries and research applications.