Secure IoT Medical Data Transmission Using Elliptic Curve Cryptography and Video Steganography with Firefly Optimization

Software Design Document

submitted by

ALAN SHAIJU KURIAN

Reg. No. MAC21CD003

ANNA ANN MATHEW

Reg. No. MAC21CS012

KEVIN PAULSON

Reg. No. MAC21CD031

KIRAN PAULSON

Reg. No. MAC21CD032

to

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

in partial fulfilment of the requirements for the award of the Degree

of

BACHELOR OF TECHNOLOGY (HONOURS)

in

COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science & Engineering

Mar Athanasius College of Engineering (Autonomous)

Kothamangalam, Kerala, India 686 666

MARCH 2025

Secure IoT Medical Data Transmission Using Elliptic Curve Cryptography and Video Steganography with Firefly Optimization

Software Design Document

submitted by

ALAN SHAIJU KURIAN

Reg. No. MAC21CD003

ANNA ANN MATHEW

Reg. No. MAC21CS012

KEVIN PAULSON

Reg. No. MAC21CD031

KIRAN PAULSON

Reg. No. MAC21CD032

to

APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

in partial fulfilment of the requirements for the award of the Degree

of

BACHELOR OF TECHNOLOGY (HONOURS)

in

COMPUTER SCIENCE AND ENGINEERING

Department of Computer Science & Engineering

Mar Athanasius College of Engineering (Autonomous)

Kothamangalam, Kerala, India 686 666

MARCH 2025

DECLARATION

We, the undersigned, hereby declare that the project report ”Secure IoT Medical Data Transmission Using Elliptic Curve Cryptography and Video Steganography with Firefly Optimization”, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology (Honours) of the APJ Abdul Kalam Technological University, Kerala, is a bonafide work done by us under the supervision of Prof. Aswathy M V and Prof. Shabiya M I. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to the ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not previously formed the basis for the award of any degree, diploma, or similar title of any other University.

Place: Kothamangalam

Date:

Anna Ann Mathew (MAC21CS012)
Alan Shaiju Kurian (MAC21CD003)
Kevin Paulson (MAC21CD031)
Kiran Paulson (MAC21CD032)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAR ATHANASIUS COLLEGE OF ENGINEERING

(AUTONOMOUS)

KOTHAMANGALAM

CERTIFICATE

This is to certify that the report entitled **Secure IoT Medical Data Transmission Using Elliptic Curve Cryptography and Video Steganography with Firefly Optimization** *submitted by* **Mr. ALAN SHAIJU KURIAN (MAC21CD003), Ms. ANNA ANN MATHEW (MAC21CS012), Mr. KEVIN PAULSON (MAC21CD031), Mr. KIRAN PAULSON (MAC21CD032)** *towards partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer Science and Engineering (Honours) from APJ Abdul Kalam Technological University for March 2025 is a bonafide record of the project carried out by them under our supervision and guidance.*

Prof. Aswathy M V	**Prof. Shabiya M I**	**Prof. Joby George**
Project Guide	Project Coordinator	Head of the Department

Internal Examiner(s)	External Examiner(s)
Date:	Dept. Seal

ACKNOWLEDGEMENT

First and foremost, we sincerely thank the 'God Almighty' for his grace for the successful and timely completion of the project.

We express our sincere gratitude and thanks to ***Dr. Bos Mathew Jos****, Principal and* ***Prof. Joby George****, Head of the Department for providing the necessary facilities and their encouragement and support.*

We owe special thanks to our project guide ***Prof. Aswathy M V*** *and project coordinator* ***Prof. Shabiya M I*** *for their corrections, suggestions and sincere efforts to co-ordinate the project under a tight schedule.*

We express our sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in guiding and correcting us in conducting this project.

Finally, we would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given to us by our dear friends during the preparation of the project and also during the presentation without which this work would have been all the more difficult to accomplish.

Table of Contents

Chapter 1
Overview

1.1 Scope

The scope of this project encompasses the design and implementation of a proof-of-concept security framework for protecting sensitive medical data within IoT healthcare systems. This project integrates Elliptic Galois Cryptography (EGC) with the secp256r1 curve, Matrix XOR steganography, and Adaptive Firefly optimization to encrypt and conceal synthetic medical datasets—comprising patient vital signs such as heart rate, blood pressure, and body temperature—within H.264-encoded MP4 video streams. The system employs a chaotic neural network (CNN)-enhanced EGC protocol to ensure robust encryption with minimal computational overhead, leveraging lightweight efficiency suitable for resource-constrained IoT devices. The encrypted data is then embedded into MP4 frames using Matrix XOR steganography, with embedding locations optimized by the Adaptive Firefly algorithm to balance efficiency and resilience while minimizing visual distortion.

A pre-processing verification phase ensures the integrity of five MP4 video files and the synthetic dataset, mitigating potential inconsistencies prior to encryption and embedding. The steganography process balances embedding capacity and video quality, evaluated through Peak Signal-to-Noise Ratio (PSNR) metrics. A structured file system organizes raw data, encrypted outputs, original videos, and stego-videos, enhancing scalability and traceability. This framework simulates a realistic IoT healthcare scenario, ensuring data confidentiality and covertness while remaining accessible only to authorized recipients.

The scope is intentionally limited to a controlled environment, focusing on core functionalities—encryption, embedding, and optimization—without extending to real-time IoT device integration or receiver-side decryption workflows. The ECC implementation is streamlined, leveraging secp256r1 for efficiency rather than full protocol complexity, and steganography is confined to MP4 carriers. Challenges such as video quality degradation or variable data loads are acknowledged but not comprehensively resolved here. Designed for adaptability, the system supports future enhancements to address evolving IoT healthcare needs. This project lays a foundation for secure, scalable medical data transmission, with potential applications in patient privacy and regulatory compliance, such as HIPAA.

1.2 Purpose

The purpose of this project is to develop a software-based security solution that addresses the critical challenge of safeguarding sensitive medical data in IoT healthcare environments. By integrating Elliptic Galois Cryptography (EGC) with the secp256r1 curve, Matrix XOR steganography, and the Adaptive Firefly Optimization Algorithm, this project aims to encrypt and conceal patient vital signs—such as heart rate, blood pressure, and temperature—within H.264-encoded MP4 video streams. The system employs a CNN-enhanced EGC protocol to deliver robust, efficient encryption tailored for lightweight IoT security. The encrypted data is embedded with minimal visual impact, optimized by the Firefly algorithm, ensuring both security and usability for healthcare stakeholders, including providers, IoT developers, and privacy advocates.

This dual-layer approach combines encryption and steganography to achieve confidentiality and covertness, protecting data across vulnerable IoT networks while maintaining video functionality for applications like telemedicine. The project seeks to demonstrate the practical synergy of these techniques, offering a scalable tool that balances security with computational efficiency. Beyond implementation, it aims to highlight the growing need for privacy in IoT healthcare, where patient trust and compliance with standards like HIPAA are paramount.

By providing a working proof-of-concept, this initiative intends to contribute to the discourse on IoT security, offering a discreet, effective method to protect medical data without disrupting workflows. It seeks to empower stakeholders with a practical approach to preserving patient privacy, fostering innovation in secure digital health solutions. Ultimately, the purpose is to establish a foundation for future IoT healthcare systems that ensure both data security and patient dignity, advancing the field through practical application and theoretical validation of hybrid security methodologies.

1.3 Intended Audience

This system design document targets a diverse audience engaged in the development, assessment, and application of secure IoT healthcare solutions, reflecting the interdisciplinary nature of this project:

- Academic advisors and faculty overseeing this honours project will utilize this document to evaluate its technical design, alignment with IoT security principles, and contribution to healthcare privacy. It provides a comprehensive overview of the EGC, Matrix XOR steganography, and Adaptive Firefly optimization integration, supported by empirical metrics like PSNR, facilitating rigorous academic assessment.

- Engineers and developers focused on IoT, particularly in healthcare, will find this document a valuable resource for implementing lightweight security solutions. It details the use of secp256r1 and H.264 video embedding, offering practical guidance for resource-constrained environments and multimedia applications.

- IT personnel in medical settings, such as hospital administrators or telemedicine specialists, will benefit from understanding how this system ensures data confidentiality and covertness. The document outlines compliance potential with standards like HIPAA, making it relevant for those managing sensitive patient data in IoT ecosystems.

- Researchers exploring IoT security and privacy will find this document a case study in hybrid security frameworks. It bridges cryptography and steganography, offering insights into resilience against attacks and quality preservation, fostering further investigation and policy development.

- Undergraduate and graduate students in computer science, cybersecurity, or biomedical engineering can use this document as an educational tool or project reference. It synthesizes advanced concepts into a practical IoT application, providing a clear example of theory-to-practice translation.

1.4 References

- Mousavi, S. K., Ghaffari, A., Besharat, S. (2021), *Security of Internet of Things Based on Cryptographic Algorithms: A Survey*, Proceedings of the International Conference on IoT and Security
- Azeez, N. A. (2020), *Chaotic Cryptography and Multimedia Security*, Presentation at the International Symposium on Multimedia Security
- Jebrane, A., Belkasmi, M., El Hajji, S. (2022), *Elliptic Curve Cryptography with Machine Learning*, Journal of Cryptographic Engineering
- Singh Rathore, P., Kumar, A., Kaur, J. (2023), *A Novel Trust-Based Security and Privacy Model for Internet of Vehicles Using Encryption and Steganography*, IEEE Transactions on Vehicular Technology
- Khayyat, M., Alotaibi, A., Alharbi, M. (2022), *Blockchain Enabled Optimal Hopfield Chaotic Neural Network Based Secure Encryption Technique for Industrial Internet of Things Environment*, IEEE Internet of Things Journal
- Kumar, V., Sharma, S., Gupta, R. (2021), *Efficient Three Layer Secured Adaptive Video Steganography Method Using Chaotic Dynamic Systems*, Proceedings of the IEEE International Conference on Signal Processing and Communications
- Nokhwal, S., Kumar, N., Singh, P. (2020), *Secure Information Embedding in Images with Hybrid Firefly Algorithm*, International Journal of Information Security

- Ghosh, S., Roy, P., Chowdhury, A. (2023), *A Comparative Analysis of Chaos Theory Based Medical Image Steganography to Enhance Data Security*, Journal of Medical Imaging and Health Informatics

Chapter 2
List of Abbreviations

ECC	Elliptic Curve Cryptography
IoT	Internet of Things
PSNR	Peak Signal-to-Noise Ratio
Stego	Steganography

Chapter 3
Conceptual Model for Software Design Descriptions

This section outlines the conceptual framework guiding the software design descriptions (SDD) for this project, which integrates Elliptic Galois Cryptography (EGC), Matrix XOR steganography, and Adaptive Firefly optimization to secure pre-recorded IoT medical data in a healthcare context. The SDD serves as a blueprint that bridges theoretical foundations and practical implementation, ensuring the system achieves confidentiality, covertness, and efficiency using a Python-based approach. By defining the software's role within its broader context, its evolution through the development life cycle, and its verification and validation processes, this section establishes a structured methodology for designing a proof-of-concept that balances security with usability. The conceptual model is crafted to meet the needs of stakeholders, including developers, healthcare professionals, and academic evaluators, while accommodating the constraints and opportunities of IoT environments using pre-recorded data.

3.1 Software Design in Context

The software design for this project is situated within the domain of IoT healthcare systems, where the secure handling of sensitive medical data is paramount, particularly when transmitted across networks susceptible to interception or unauthorized access. Specifically, it focuses on protecting pre-recorded IoT readings—such as patient vital signs including heart rate, blood pressure, and body temperature—derived from real-world sensor data, which serve as a realistic proxy for live IoT feeds in this proof-of-concept. The design addresses the dual imperatives of ensuring data confidentiality and maintaining the practical utility of multimedia carriers, utilizing H.264-encoded MP4 video streams as the medium for data concealment. This choice reflects a balance between widespread compatibility in healthcare applications (e.g., telemedicine, patient monitoring archives) and the technical feasibility of embedding encrypted data discreetly.

At its core, the design leverages EGC with the secp256r1 curve, a lightweight cryptographic method optimized for resource-constrained IoT environments, ensuring that encryption remains efficient without taxing computational resources. This encrypted data is then embedded into MP4 video frames using Matrix XOR steganography, a technique that manipulates pixel values via bitwise operations to hide information imperceptibly. The Adaptive Firefly optimization algorithm further refines this process by dynamically selecting embedding locations within the video frames, minimizing visual distortion—quantified through metrics like Peak Signal-to-Noise Ratio (PSNR)—while maximizing the capacity to conceal data. This optimization ensures that the resulting stego-videos remain visually indistinguishable from their originals, preserving their functionality for downstream uses such as clinical review or secure data sharing.

The broader context of this design includes its interaction with an ecosystem comprising pre-recorded IoT datasets, a Python-based development environment, and potential end-users such as healthcare providers or IT administrators. It operates under constraints typical of IoT systems, including limited processing power, memory, and energy availability, even though the current implementation relies on pre-recorded rather than real-time data. The design is also influenced by external factors, such as the need to comply with privacy regulations like HIPAA, which dictate stringent requirements for data protection in healthcare settings. As a proof-of-concept, the software prioritizes core functionalities—encryption, embedding, and optimization—over full-scale deployment features like real-time IoT integration or receiver-side decryption, providing a focused yet extensible foundation. This contextual framework ensures that the design not only meets technical objectives but also aligns with practical healthcare needs, offering a scalable solution that can adapt to future enhancements, such as integration with live IoT devices or expanded media formats.

3.2 Software Design Descriptions within the life Cycle

The SDD for this project evolves through distinct phases of the software development life cycle, from conceptualization to implementation and evaluation, reflecting a systematic approach to achieving the project's goals using Python. It provides a detailed representation of the system's architecture, algorithms, and interfaces, serving as a guide for development with Python files and a reference for stakeholders assessing its progress. The life cycle is tailored to a proof-of-concept using pre-recorded IoT readings, emphasizing design, simulation, and testing within a controlled environment, with provisions for future iteration based on findings.

3.2.1 Influences on SDD Preparation

Several factors influence the preparation of the SDD for this project. The technical complexity of integrating EGC, Matrix XOR steganography, and Adaptive Firefly optimization within a Python environment requires a clear delineation of each component's functionality and interaction, driving the need for precise algorithmic descriptions and modular design across Python files. Resource constraints typical of IoT scenarios necessitate a focus on lightweight solutions, such as the secp256r1 curve, shaping the encryption module's specifications. The choice of H.264 MP4 video as the carrier medium influences the steganography design, requiring optimization strategies to maintain video quality, as measured by PSNR, within a Python-based implementation.

Stakeholder requirements also play a significant role. Academic evaluators demand rigorous documentation and empirical validation, while IoT developers and healthcare professionals prioritize practicality and compliance with privacy standards. The use of pre-recorded IoT readings, simulating real sensor outputs, influences the SDD by providing a standardized input for testing, ensuring consistency across development phases. Additionally, the project's proof-of-concept nature limits the scope to core functionalities using pre-recorded data, deferring real-time integration and receiver-side processes, which simplifies the SDD's preparation but requires explicit boundaries to be defined.

3.2.2 Influences on Software Life Cycle Products

The SDD impacts various products throughout the software life cycle, shaping their form and utility within a Python-based framework. During the requirements analysis phase, it defines the need for encryption strength, embedding capacity, and video quality preservation, resulting in a detailed specification document. In the design phase, the SDD produces architectural diagrams, Python pseudocode for EGC encryption, Matrix XOR embedding, and Firefly optimization, as well as file structure models, providing a tangible foundation for implementation. These design artifacts directly influence the implementation phase, guiding the development of Python files that execute the encryption, embedding, and optimization processes using pre-recorded IoT readings.

In the testing phase, the SDD informs the creation of test cases that evaluate performance metrics—embedding efficiency, PSNR, and time complexity—producing evaluation reports that assess results against baseline expectations. The structured file system (raw data, encrypted outputs, original videos, stego-videos) emerges as a life cycle product, enhancing traceability and scalability within the Python environment. These products collectively ensure that the software meets its objectives, while also serving as deliverables for academic review and potential future enhancements, such as real-time IoT integration.

3.3 Design verification and Design Role in Validation

Design verification ensures that the software is constructed accurately, confirming that each component—EGC encryption, Matrix XOR steganography, and Adaptive Firefly optimization—performs as specified within the SDD in the Python implementation. This process entails a multi-faceted approach to testing. Unit testing targets individual Python modules: for instance, verifying that the EGC module, implemented with the secp256r1 curve, generates valid cipher text from pre-recorded IoT readings, ensuring cryptographic integrity through checks against known plaintext-ciphertext pairs. Similarly, the Matrix XOR steganography module is tested to confirm that it embeds data into MP4 frames without detectable errors, using bitwise comparison to validate the embedding process. The Adaptive Firefly optimization module is assessed to ensure it selects optimal embedding locations, minimizing distortion by comparing PSNR values before and after embedding against a threshold (e.g., ¿30 dB for acceptable quality). Integration testing evaluates the end-to-end workflow across Python files, from encrypting pre-recorded data to producing stego-videos, ensuring seamless component interaction and data flow consistency. Performance metrics—embedding efficiency (bits per pixel), PSNR, and execution time—are quantified using Python scripts, benchmarked against predefined standards to confirm technical precision.

The design's role in validation extends beyond correctness to ensure the system fulfills its intended purpose of securing IoT medical data discreetly and efficiently in a healthcare context. Validation leverages pre-recorded IoT readings to simulate realistic scenarios, such as transmitting patient vitals securely for clinical review. This involves embedding encrypted data into MP4 videos and assessing the stego-videos' imperceptibility through both quantitative metrics (e.g., PSNR, structural similarity index) and qualitative visual inspection by stakeholders, ensuring no perceptible artifacts compromise usability. Validation also tests the system's ability to maintain confidentiality, confirming that embedded data cannot be extracted without the correct decryption key, potentially through simulated attack scenarios (e.g., statistical analysis of pixel distributions). Usability is validated by ensuring stego-videos remain functional for healthcare applications, such as playback in standard video players, mimicking telemedicine or archival use cases. The process incorporates stakeholder feedback—e.g., from healthcare professionals on video quality or developers on implementation feasibility—to refine the design. By aligning verification and validation with the SDD, this project establishes a robust proof-of-concept, demonstrating technical reliability and practical relevance for secure IoT data transmission within a Python-based framework.

Chapter 4
Design Description Information Content

4.1 Introduction

This section delineates the information content of the Software Design Description (SDD) for this project, which integrates Elliptic Galois Cryptography (EGC), Matrix XOR steganography, and Adaptive Firefly optimization to secure pre-recorded IoT medical data within H.264 MP4 video streams. The SDD encapsulates the system's architecture, components, and implementation details, providing a comprehensive guide for stakeholders to understand, develop, and evaluate the design. Drawing from the base paper's hybrid security framework, this project adapts EGC for encryption, Matrix XOR for embedding, and Adaptive Firefly for optimization, implemented entirely in Python to process pre-recorded patient vital signs such as heart rate, blood pressure, and temperature. This section outlines the identification, stakeholders, views, viewpoints, elements, overlays, rationale, and languages of the design, ensuring alignment with the proof-of-concept's objectives of confidentiality, covertness, and efficiency in an IoT healthcare context.

4.2 SDD Identification

The Software Design Description (SDD) for this project, which focuses on securing pre-recorded IoT medical data through a hybrid approach of encryption and steganography, will be formally released following the completion of validation and verification processes. A prototype showcasing the integration of Elliptic Galois Cryptography (EGC), Matrix XOR steganography, and Adaptive Firefly optimization within H.264 MP4 video streams is scheduled for demonstration in the coming months, providing stakeholders with a practical preview of the system's capabilities in a Python-based environment. Key sections of the SDD, including scope, purpose, and contextual details, will be accessible under an "Overview" section, offering comprehensive insights into the project's objectives of ensuring confidentiality, covertness, and efficiency in an IoT healthcare setting. Additionally, a glossary containing definitions, acronyms, and abbreviations pertinent to the secure transmission of medical data will be included in a dedicated "Definitions, Acronyms, and Abbreviations" section, enhancing clarity and understanding throughout the document.

4.3 Design Stakeholders and Their Concerns

The stakeholders for this project include the development team responsible for its implementation, as well as faculty members offering guidance and oversight. Additional stakeholders encompass IoT developers, healthcare IT professionals, security researchers, and academic peers, each contributing unique perspectives. A primary concern shared across these groups is the system's quality, encompassing its ability to deliver robust security, maintain video integrity, and operate efficiently within the constraints of an IoT healthcare context. Ensuring high quality is critical, reflecting the collective aim of producing a dependable solution that effectively encrypts and conceals pre-recorded medical data using EGC, Matrix XOR steganography, and Adaptive Firefly optimization, while meeting stakeholder expectations for usability, compliance, and technical excellence.

4.4 Design views

Design views serve as critical lenses through which stakeholders can examine specific aspects of the system's design and address their respective concerns related to securing pre-recorded IoT medical data within an IoT healthcare context.

Each identified concern is represented by at least one design view to ensure the SDD's completeness, tailored to this project's focus on integrating Elliptic Galois Cryptography (EGC), Matrix XOR steganography, and Adaptive Firefly optimization. For instance, concerns about data confidentiality and covertness are addressed through a security-focused logical view, while video quality preservation and processing efficiency are evaluated via a performance view. This document includes several design views, each customized to highlight the system's unique processes and outcomes:

- Context View: Illustrates the system's interactions with its environment, depicting how pre-recorded IoT readings (e.g., heart rate, blood pressure, temperature) are processed and embedded into H.264 MP4 video streams, defining the system's boundaries and external interfaces.
- Composition View: Details the breakdown of the system into Python-based components—EGC encryption, Matrix XOR steganography, and Adaptive Firefly optimization modules—alongside the file system managing raw IoT data, encrypted outputs, and stego-videos, clarifying their structural roles.
- Logical View: Describes the functional logic of encrypting medical data with EGC using the secp256r1 curve, embedding it into video frames via Matrix XOR, and optimizing embedding locations with the Firefly algorithm, providing a clear process flow.
- Dependency View: Highlights interdependencies among the encryption, steganography, and optimization components, showing how encrypted data flows into the embedding process and is refined by optimization, ensuring seamless integration.
- Information View: Focuses on data structures, such as the format of pre-recorded IoT readings (e.g., CSV files) and the H.264 video frames, detailing storage and access mechanisms within the Python environment.
- Interaction View: Outlines the sequence of operations—from reading IoT data, encrypting it with EGC, embedding it into MP4 frames, to optimiz-

ing with Firefly—emphasizing the step-by-step transformation into stego-videos.

- Evaluates system metrics specific to this project, such as Peak Signal-to-Noise Ratio (PSNR) for video quality, embedding efficiency for data concealment capacity, and execution time in Python, comparing pre- and post-embedding states.

These views provide diverse insights into the system's design, enabling stakeholders to assess its effectiveness in achieving confidentiality, covertness, and efficiency while preserving video usability for healthcare applications like telemedicine or secure data archiving.

4.5 Design Viewpoints

This document outlines several design viewpoints, each offering a unique perspective on the system's architecture and behavior:

- Context Viewpoint: Describes relationships and interactions between the system, pre-recorded IoT data, and H.264 MP4 videos, using diagrams like context and block diagrams to define boundaries and external interfaces.
- Composition Viewpoint: Details how the system is segmented into Python modules (e.g., encryption, steganography, optimization), supporting resource planning and implementation, depicted via component diagrams.
- Logical Viewpoint: Outlines the logical structure and interactions of EGC, Matrix XOR, and Firefly optimization processes, aiding development and reuse, represented through flowcharts or pseudocode.
- Dependency Viewpoint: Maps component dependencies and execution order, providing clarity on data processing sequences.
- Information Viewpoint: Defines data types, storage (e.g., IoT readings, stego-videos), and access methods, detailing file system organization.

- Interface Viewpoint: Details external and internal interfaces, facilitating designers, programmers, and testers in understanding and utilizing the system.
- Interaction Viewpoint: Describes the operational sequence and rationale for encryption, embedding, and optimization, setting the stage for performance analysis.
- Performance Viewpoint: Focuses on system efficiency and quality metrics (e.g., PSNR, embedding efficiency), supported by quantitative data.

These viewpoints align with stakeholder needs, loosely reflecting the base paper's emphasis on security and optimization, adapted to this Python-based IoT context.

4.6 Design Elements

Design elements include all items within the design views, such as entities, relationships, attributes, and constraints, each detailed in their corresponding viewpoints in subsequent sections of the SDD. Key elements encompass the EGC encryption process with secp256r1, the Matrix XOR steganography mechanism for embedding data into H.264 frames, and the Adaptive Firefly optimization algorithm for location selection. Additional elements include the input/output file structures and verification scripts, all implemented in Python. These elements form the foundational building blocks of the system, ensuring clarity and coherence in translating the design into a functional proof-of-concept for secure medical data transmission.

4.7 Design Overlays

Design overlays provide supplementary information layered onto existing design views, enhancing their depth and clarity. They are employed when additional context or precision is needed to address stakeholder concerns. Examples include:

- Security Overlay: Adds details on encryption strength and steganographic concealment across logical and interaction views.
- Performance Overlay: Integrates PSNR and execution time data into performance and dependency views, clarifying efficiency trade-offs.
- Error Handling Overlay: Annotates potential failure points (e.g., video format errors) in composition and interaction views, bolstering robustness.

These overlays ensure stakeholders gain a thorough understanding of critical design aspects, supporting evaluation and refinement.

4.8 Design Rationale

The decision to adopt a modular, Python-based approach was driven by the need to efficiently integrate encryption, steganography, and optimization functionalities within a resource-constrained IoT context. EGC with secp256r1 was selected for its lightweight cryptographic strength, Matrix XOR steganography for its effective data concealment in H.264 frames, and Adaptive Firefly optimization for its dynamic efficiency, aligning with the base paper's hybrid methodology. Python was chosen for its flexibility and rich ecosystem, enabling seamless processing of pre-recorded IoT data and video streams. This modular design ensures maintainability and scalability, allowing independent refinement of components while supporting the proof-of-concept's focus on security and quality in healthcare applications.

4.9 Design Languages

Python-like pseudocode and Unified Modeling Language (UML) diagrams serve as the primary design languages in this SDD. Pseudocode outlines algorithmic steps for EGC, Matrix XOR, and Firefly optimization, bridging design and implementation. UML diagrams, including flowcharts, component diagrams, and sequence diagrams, depict system architecture, interactions, and data flows, emphasizing both static and dynamic aspects. Natural language (English) provides descriptive clarity, while mathematical notation (e.g., PSNR equations) ensures technical precision. These languages facilitate effective communication and comprehension across stakeholders, aligning with the project's technical and educational goals.

Chapter 5
Design Viewpoints

5.1 Introduction

This section elaborates on the design viewpoints that frame the Software Design Description (SDD) for this project, which secures pre-recorded IoT medical data using Elliptic Galois Cryptography (EGC), Matrix XOR steganography, and Adaptive Firefly optimization within H.264 MP4 video streams. Each viewpoint provides a distinct perspective on the system's architecture and behavior, addressing stakeholder concerns such as security, efficiency, and usability in an IoT healthcare context. Drawing from the base paper's hybrid security approach, this project implements these techniques in Python to process patient vital signs (e.g., heart rate, blood pressure, temperature), ensuring confidentiality and covertness as a proof-of-concept. The viewpoints detailed here—Context, Composition, Logical, and Interface—build upon the design views outlined in Section 4, offering a structured framework to describe design concerns and elements, facilitating development, evaluation, and potential future enhancements.

5.2 Context Viewpoint

The Context Viewpoint defines the system's interactions with its external environment, establishing its boundaries and relationships within the IoT healthcare domain.

5.2.1 Design Concerns

- System Boundaries: Clearly delineating the scope of the system, including inputs (pre-recorded IoT medical data) and outputs (stego-videos), while excluding real-time IoT integration or receiver-side decryption as per the proof-of-concept focus.
- External Interactions: Ensuring compatibility with pre-recorded IoT data formats and H.264 MP4 video streams, as well as potential downstream use by healthcare providers for applications like telemedicine or secure archiving.
- Environmental Constraints: Addressing IoT resource limitations (e.g., computational power, memory), even in a simulated context, to ensure the design remains lightweight and efficient.
- Stakeholder Accessibility: Providing a clear overview for stakeholders (e.g., healthcare IT professionals, developers) to understand how the system fits into a broader IoT ecosystem.

5.2.2 Design Elements

- Input Data Entity: Pre-recorded IoT readings (e.g., CSV files containing vital signs), serving as the raw data source for encryption and embedding.
- Output Video Entity: H.264 MP4 stego-videos containing encrypted medical data, designed for compatibility with standard video playback tools.
- System Boundary: A conceptual perimeter encompassing the Python-based encryption, steganography, and optimization processes, excluding external IoT devices or network transmission protocols.
- External Interface Relationship: Defines the interaction between the system and external entities, such as loading IoT data files and exporting stego-videos, depicted via context or block diagrams.

These elements ensure the system's external context is well-defined, aligning with the base paper's focus on securing IoT data within multimedia carriers.

5.3 Composition Viewpoint

The Composition Viewpoint describes the system's internal structure, breaking it down into modular components and their roles within the Python implementation.

5.3.1 Design Concerns

- Modularity: Ensuring the system is divided into distinct, manageable components (e.g., encryption, steganography, optimization) to support development, testing, and future scalability.
- Resource Allocation: Optimizing component design for efficient use of computational resources, critical for IoT applicability, even in a Python-based simulation.
- Component Integration: Facilitating seamless interaction among components to process pre-recorded IoT data into stego-videos without functional overlap or redundancy.
- Maintainability: Allowing independent updates or refinements to each component, enhancing the proof-of-concept's adaptability for future iterations.

5.3.2 Design Elements

- EGC Encryption Component: A Python module implementing EGC with the secp256r1 curve and chaotic neural networks, responsible for encrypting IoT medical data.
- Matrix XOR Steganography Component: A Python module embedding encrypted data into H.264 MP4 frames using bitwise XOR operations, adapted from the base paper's OM-XOR technique.
- Adaptive Firefly Optimization Component: A Python module optimizing embedding locations to minimize distortion and maximize capacity, leveraging bio-inspired algorithms.

- File System Component: Manages the structured organization of raw IoT data, encrypted outputs, original videos, and stego-videos, ensuring data traceability.
- Component Relationship: Defines interactions, such as the flow of encrypted data from the EGC module to the steganography module, refined by the optimization module, represented in component diagrams.

These elements collectively form a modular architecture, supporting the base paper's emphasis on efficient, integrated security processes.

5.4 Logical Viewpoint

The Logical Viewpoint outlines the functional logic and process flow of the system, detailing how encryption, steganography, and optimization achieve the project's objectives.

5.4.1 Design Concerns

- Functional Accuracy: Ensuring each process (encryption, embedding, optimization) operates correctly to secure IoT data and produce usable stego-videos, validated against expected outcomes.
- Process Efficiency: Minimizing computational overhead in encryption and embedding, critical for IoT applicability, while maintaining high security and quality standards.
- Data Transformation: Guaranteeing that pre-recorded IoT data is transformed into encrypted, embedded form without loss of integrity, aligning with healthcare privacy needs.
- Stakeholder Comprehension: Providing a clear, logical sequence for developers and evaluators to understand and replicate the system's operations.

5.4.2 Design Elements

- EGC Encryption Process: Encrypts IoT data using secp256r1 and chaotic neural networks, producing cipher text with high randomness and security, detailed via pseudocode or flowcharts.
- Matrix XOR Embedding Process: Embeds cipher text into H.264 frames by adjusting pixel values with XOR operations, ensuring imperceptibility, as per the base paper's methodology.
- Firefly Optimization Process: Iteratively selects embedding locations based on distortion metrics (e.g., PSNR), optimizing efficiency and quality, represented algorithmically.
- Data Flow Relationship: Maps the logical sequence from IoT data input, through encryption, embedding, and optimization, to stego-video output, illustrated in flow diagrams.

These elements articulate the system's core logic, ensuring alignment with the base paper's hybrid security and optimization goals.

5.5 Interface Viewpoint

The Interface Viewpoint specifies the system's internal and external interfaces, facilitating interaction among components and with external entities.

5.5.1 Design Concerns

- Internal Compatibility: Ensuring smooth data exchange between encryption, steganography, and optimization modules within the Python environment, avoiding format or processing mismatches.
- External Accessibility: Enabling straightforward input of pre-recorded IoT data and output of stego-videos, usable by stakeholders like healthcare professionals or testers.

- Interface Simplicity: Designing interfaces that are intuitive and efficient, minimizing complexity for developers implementing or extending the system.
- Error Handling: Incorporating mechanisms to manage interface-related issues (e.g., invalid file formats), enhancing system reliability.

5.5.2 Design Elements

- Input Interface: A Python function or script accepting pre-recorded IoT data (e.g., CSV files), parsing vital signs for encryption, with defined input parameters (e.g., file path, data format).
- Output Interface: A Python function exporting H.264 MP4 stego-videos, ensuring compatibility with standard playback tools, specified by output file structure and naming conventions.
- Inter-Module Interface: Data exchange protocols between EGC, Matrix XOR, and Firefly modules (e.g., passing cipher text as a byte array, returning optimized frame coordinates), detailed in interface specifications.
- Error Handling Attribute: Conditions and responses for interface failures (e.g., raising exceptions for corrupted video files), ensuring robust operation, depicted in sequence diagrams.

These elements support seamless interaction, reflecting the base paper's focus on integrating cryptographic and steganographic processes efficiently.

Chapter 6
Technological Stack

This section outlines the technology stacks employed in this project, which secures pre-recorded IoT medical data through a hybrid approach integrating Elliptic Galois Cryptography (EGC), Matrix XOR steganography, and Adaptive Firefly optimization within H.264 MP4 video streams. The selected technologies form a cohesive framework that supports the design, implementation, and evaluation of the proof-of-concept, ensuring efficiency, security, and usability within an IoT healthcare context. Implemented entirely in Python, the system leverages a combination of programming libraries, cryptographic tools, and multimedia processing utilities to process patient vital signs and embed them discreetly into video files. Each technology stack is detailed below, highlighting its purpose, application, and contribution to achieving the project's objectives of confidentiality, covertness, and quality preservation.

6.0.1 Python

Python serves as the primary programming language and runtime environment for this project, underpinning all aspects of development from data generation to encryption, steganography, and optimization. Its flexibility, extensive library ecosystem, and ease of use make it ideal for rapid prototyping and implementation of the proof-of-concept. The project utilizes Python scripts (e.g., for data generation, encryption, video processing) to orchestrate the workflow, leveraging built-in modules like `os` for directory management and file path handling, `csv` for reading and writing IoT medical data in CSV format, and `random` with `datetime` for generating synthetic patient vital signs. Python's cross-platform compatibility ensures the system can operate on diverse environments, such as Windows, where the code employs Windows-compatible path separators. This stack provides the foundational infrastructure, enabling seamless integration of specialized libraries and custom algorithms tailored to the project's needs.

6.0.2 TinyEC

The TinyEC library is a critical cryptographic stack used to implement Elliptic Galois Cryptography (EGC) with the secp256r1 curve, as seen in the encryption scripts. This lightweight Python library facilitates elliptic curve operations, allowing the generation of private-public key pairs and point multiplication for encrypting pre-recorded IoT medical data. In the code, TinyEC's `registry.get_curve('secp256r1')` retrieves the secp256r1 curve parameters, while `secrets.randbelow()` ensures cryptographically secure random number generation for keys and ephemeral values. The library's efficiency aligns with the base paper's emphasis on lightweight cryptography for IoT, enabling the system to encrypt vital signs (e.g., heart rate, blood pressure) into cipher text with minimal computational overhead. TinyEC supports the project's security goals by providing a robust, standardized cryptographic foundation tailored to resource-constrained environments.

6.0.3 OpenCV (cv2)

OpenCV, accessed via the `cv2` Python module, is the primary stack for video processing and steganographic embedding in this project. It handles the reading, manipulation, and writing of H.264 MP4 video files, as evidenced by the use of `cv2.VideoCapture` to load original videos from the `videos` subdirectory and `cv2.VideoWriter` with the `mp4v` codec to output stego-videos to the `stego_videos` subdirectory. OpenCV's `cv2.cvtColor` converts frames to grayscale for embedding, while its array manipulation capabilities support the Matrix XOR steganography process, embedding encrypted data into pixel values. The library also enables PSNR calculation (`calculate_psnr`) to evaluate video quality post-embedding, a key metric for ensuring imperceptibility as highlighted in the base paper. OpenCV's versatility and performance make it essential for processing multimedia carriers efficiently, aligning with the project's goal of maintaining video usability for healthcare applications.

6.0.4 NumPy

NumPy is employed as a numerical computation stack, enhancing the efficiency of array-based operations in video frame processing and optimization. In the steganography script, `np.mean` calculates block brightness for the Adaptive Firefly algorithm, while `np.mean` and array operations compute Mean Squared Error (MSE) for PSNR evaluation. NumPy's support for multi-dimensional arrays facilitates the manipulation of video frames (e.g., 8x8 blocks) during embedding, optimizing performance over native Python loops. This stack complements OpenCV by providing fast, vectorized operations critical for the Matrix XOR steganography and Firefly optimization processes, ensuring the system meets efficiency requirements within the Python environment. Its role mirrors the base paper's focus on optimized data handling in steganographic techniques.

6.0.5 Standard Python Libraries (os, csv, secrets, math, datetime, random)

A suite of standard Python libraries forms an auxiliary technology stack, supporting foundational tasks across the project:

- **os:** Manages directory structures (`iot_security_dataset/data`, `videos`, `stego_videos`) and file paths, ensuring robust file handling with `os.makedirs` and `os.path.join`.
- **csv:** Reads pre-recorded IoT data from `medical_data.csv` and writes encrypted outputs to `encrypted_medical_data.csv`, structuring vital signs data for processing.
- **secrets:** Provides cryptographically secure random numbers for key generation in EGC, enhancing security over `random`, as seen in `secrets.randbelow`.

- **math:** Supports PSNR computation with logarithmic functions (`math.log10`), ensuring accurate quality metrics.
- **datetime and random:** Generate synthetic IoT data with timestamps and realistic vital sign ranges in the data generation script, simulating real-world inputs.

These libraries collectively enable data management, security, and evaluation, forming a reliable backbone for the project's Python-based implementation.