



**RAJALAKSHMI  
ENGINEERING COLLEGE**

An AUTONOMOUS Institution  
Affiliated to ANNA UNIVERSITY, Chennai

**ENABLING A SECURE COMMUNICATION SYSTEM  
IN LEGACY APPLICATION AND PREVENTING PHISHING  
AND MIMT ATTACKS**

Submitted by:

**GURUPRASATH P(221801014)**

**HARSHINI M D(221801017)**

**S MADHUVANTHIY(221801031)**

**AD19541 SOFTWARE ENGINEERING METHODOLOGY**

**Department of Artificial Intelligence and Data Science**

**Rajalakshmi Engineering College, Thandalam**

**2024-2025**

## **BONAFIDE CERTIFICATE**

Certified that this project report “**ENABLING SECURE COMMUNICATION SYSTEM IN LEGACY APPLICATION AND PREVENTING PHISHING AND MIMT ATTACKS**” is the Bonafide work of “**GURUPRASATH P (221801014), HARSHINI M D (221801017),MADHUVANTHIY S (221801031)**”who carried out the project work under my supervision.

Submitted for the Practical Examination held on \_\_\_\_\_

**SIGNATURE**

**Dr.J.M.GNANASEKAR**  
Professor & Head,  
Dept. of Artificial Intelligence and Data Science,  
Rajalakshmi Engineering College,  
Thandalam, Chennai-602105

**SIGNATURE**

**Dr.MANORANJINI J**  
Associate Professor,  
Dept. of Artificial Intelligence and Data Science,  
Rajalakshmi Engineering College,  
Thandalam, Chennai-602105

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## TABLE OF CONTENTS

S.NO	CHAPTER	PAGE NO.
<b>1</b>	<b>INTRODUCTION</b>	
	1.1 GENERAL	1
	1.2 NEED FOR THE STUDY	2
	1.3 OVERVIEW OF THE PROJECT	2
	1.4 OBJECTIVES OF THE STUDY	3
<b>2</b>	<b>REVIEW OF LITERATURE</b>	
	2.1 INTRODUCTION	4
<b>3</b>	<b>SYSTEM OVERVIEW</b>	
	3.1 EXISTING SYSTEM	5
	3.2 PROPOSED SYSTEM	6
	3.3 FEASIBILITY STUDY	7
<b>4</b>	<b>SYSTEM REQUIREMENTS</b>	
	4.1 HARDWARE REQUIREMENTS	8
	4.2 SOFTWARE REQUIREMENTS	9
<b>5</b>	<b>SYSTEM DESIGN</b>	
	5.1 SYSTEM ARCHITECTURE	10
	5.2 MODULE DESCRIPTION	12
	5.2.1 USER AUTHENTICATION MODULE	12
	5.2.2 SECURE DATA ROUTING MODULE	12
	5.2.3 ENCRYPTION AND DECRYPTION MODULE	12

<b>6</b>	<b>UML DIAGRAM</b>	
	6.1 DATA FLOW DIAGRAM	13
	6.2 USE CASE DIAGRAM	14
	6.3 SEQUENCE DIAGRAM	14
	6.4 CLASS DIAGRAM	15
<b>7</b>	<b>SOFTWARE DEVELOPMENT LIFECYCLE</b>	
	7.1 WATERFALL MODEL	16
<b>8</b>	<b>PROGRAM CODE AND OUPUTS</b>	
	8.1 SAMPLE CODE	17
	8.2 OUTPUT SCREENSHOTS	19
<b>9</b>	<b>TESTING</b>	
	9.1 UNIT TESTING	24
	9.2 TESTING SCREENSHOTS	25
<b>10</b>	<b>RESULTS AND DISCUSSION</b>	
	10.1 RESULTS	26
	10.2 DISCUSSION	27
<b>11</b>	<b>CONCLUSION AND FUTURE ENHANCEMENT</b>	
	11.1 CONCLUSION	28
	11.2 FUTURE ENHANCEMENT	29
	REFERENCES	30

## LIST OF FIGURES

Figure No	Figure Name	Page No
1.	System Architecture	10
2.	Agile Model	16
3.	Key Generation	19
4.	Encrypting the Document	20
5.	Decrypting the Document	20
6.	DFD Level 1	21
7.	Use Case Diagram	22
8.	Sequence Diagram	22
9.	Class Diagram	23
10.	Unit Testing	25

## **ABSTRACT**

In response to cybersecurity challenges in critical sectors like finance, healthcare, and government, this project offers a cybersecurity solution for aging infrastructures. Legacy systems, often lacking modern security features, are increasingly vulnerable to cyberattacks, yet upgrading them is costly and complex. This project introduces a flexible framework to fortify legacy systems against contemporary threats with minimal disruption. Employing a client-server architecture with 4096-bit RSA encryption and public-private key management, it ensures secure data transmission and automated key distribution. Digital signature verification helps mitigate human error—one of the biggest vulnerabilities—by preserving data integrity and authenticity. The framework's cloud-based Software as a Service (SaaS) model provides scalability, enabling organizations to adjust security measures without a full system overhaul. Key components include automated encryption, user-friendly key management, and scalability testing to meet varying organizational demands. In addition to protecting against unauthorized access and phishing, the solution ensures message authenticity, closing critical security gaps in legacy systems. Combining flexibility, scalability, and resilience, this approach empowers sectors dependent on legacy infrastructure to adopt robust cybersecurity practices. By securing data within legacy environments, the project supports the broader adoption of adaptable, data-driven cybersecurity measures tailored to the specific needs of aging systems.

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 GENERAL**

The unprecedented growth of digital communication has revolutionized information sharing but has also heightened the risk of cyber threats. Many older or legacy systems—still widely used in critical infrastructure sectors like finance, healthcare, and government—lack modern security features, making them vulnerable to cyberattacks. Unlike contemporary systems designed with robust cybersecurity frameworks, these legacy systems were typically created with limited defenses against sophisticated threats such as phishing and Man-in-the-Middle (MITM) attacks. Phishing, in particular, exploits human vulnerabilities to access sensitive information, while MITM attacks intercept and manipulate data between communication channels.

Project-R is an innovative solution created to address the security needs of legacy systems. Unlike conventional methods that often require expensive software upgrades or hardware changes, Project-R is a non-invasive, cost-effective solution that does not necessitate structural changes to the network. Instead, it integrates at the application layer, leveraging a client-server architecture to enable secure, automated data transmission. By embedding public-private key encryption with a 4096-bit RSA standard, Project-R prevents unauthorized access and phishing attacks, ensuring both data integrity and confidentiality. Its approach ensures seamless, secure communication without disrupting the existing setup, providing an effective, reliable security solution for legacy systems that require up-to-date protection against evolving cyber threats.

## **1.2 NEED FOR THE STUDY**

The Legacy systems are extensively used in industries where upgrading or replacing systems is costly, time-intensive, and complex due to dependencies on older technology. However, these systems are increasingly vulnerable to cyber threats, with traditional cybersecurity measures proving insufficient to prevent modern attacks. Most legacy infrastructures lack encryption capabilities or rely on obsolete security measures, exposing them to significant risk. Phishing and MITM attacks are particularly concerning as they exploit both human and technological weaknesses to steal sensitive information or compromise data integrity.

Given these risks, a specialized solution is necessary to safeguard these systems without requiring extensive modifications. Project-R addresses this need by introducing an end-to-end encryption solution that incorporates an automated public-private key infrastructure. This setup minimizes the need for user intervention, which can often be a point of failure in security. The system's application layer integration ensures compatibility with existing setups, preserving the integrity of the network layer while providing robust protection against unauthorized access. Thus, this study responds to a critical cybersecurity gap, presenting a solution that strengthens legacy systems and prevents vulnerabilities inherent in outdated security models.

## **1.3 OVERVIEW OF THE PROJECT**

Project-R is designed as a comprehensive cybersecurity framework that secures communication within legacy systems through automated 4096-bit public-private key encryption. At its core, the project leverages a client-server architecture to facilitate secure data transmission without impacting the underlying network infrastructure. The server generates public keys and securely distributes them to the clients, while the corresponding private keys remain protected on the client-side, ensuring only authorized entities can decrypt transmitted information. This structure achieves end-to-end encryption, rendering intercepted data unreadable and protecting communication from unauthorized parties.



To make the system scalable, Project-R employs a cloud-based Software as a Service (SaaS) model, providing subscription access to encryption and decryption services. This approach enables the system to adapt to varying workloads, making it accessible to organizations of different sizes. Additionally, Project-R includes integrity checks through digital signatures to prevent phishing attacks, ensuring the authenticity of each communication. By embedding at the application layer, Project-R provides a versatile, secure communication solution that is compatible with legacy systems while ensuring robust, low-intervention security measures suitable for high-stakes environments.

#### **1.4 OBJECTIVES OF THE STUDY**

The main objective of Project-R is to develop a scalable, secure communication system specifically designed for legacy systems that lack modern cybersecurity features. To achieve this, the study focuses on the following key objectives:

**1. System Scalability and Compatibility:** The solution must be compatible with legacy infrastructures, enabling integration without the need for extensive network modifications. Through its cloud-based, subscription model, Project-R can scale according to organizational needs.

**2. Automation of Encryption and Decryption:** By automating encryption and decryption processes, Project-R reduces the reliance on user input, which often leads to errors in security-sensitive environments. The server manages the creation and distribution of public keys, while private keys are managed on the client-side for secure decryption, minimizing user error.

**3. Phishing Prevention through Integrity Checks:** Project-R incorporates digital signature-based integrity checks, which authenticate data origin and ensure that the message remains unaltered during transit. These checks prevent phishing attacks by enabling clients to verify the authenticity of each communication, thereby protecting sensitive information from unauthorized access and manipulation. These objectives are strategically designed to offer a comprehensive solution that strengthens legacy systems, facilitating secure, resilient communication that meets modern cybersecurity demands.

## **CHAPTER 2**

### **REVIEW OF LITERATURE**

#### **2.1 LITERATURE REVIEW**

Phishing detection has seen significant advancements with the adoption of machine learning (ML), deep learning (DL), and hybrid approaches, addressing the limitations of traditional techniques. Researchers have explored methods such as client-side solutions that offer lightweight, real-time protection by analyzing user behavior and website interactions. Graph mining techniques identify suspicious relationships between entities like URLs and IPs, while visual similarity analysis detects phishing websites mimicking legitimate ones. Deep learning models analyze URL and HTML content to recognize evolving phishing patterns, showcasing high accuracy but requiring substantial computational resources.

Hybrid models and ensemble techniques, which combine algorithms like decision trees, SVMs, and neural networks, provide enhanced robustness and reduced false positives. These approaches capture diverse phishing patterns and improve detection accuracy by leveraging the strengths of multiple classifiers. Real-time systems tailored for enterprise networks and financial institutions further refine detection by analyzing network traffic and web server logs to identify anomalies and phishing attempts. Despite these advancements, challenges such as computational demands, frequent updates, and resource-intensive deployments remain barriers to widespread adoption.

Overall, these studies emphasize the need for adaptive systems that integrate advanced algorithms and continuous model training to combat evolving phishing strategies. By addressing challenges like false positives, latency, and resource constraints, these solutions can enhance cybersecurity in various domains. This foundation supports the development of secure communication platforms like Project-R, advancing the fight against phishing with innovative detection methods.

## **CHAPTER 3**

### **SYSTEM OVERVIEW**

#### **3.1 EXISTING SYSTEM**

Current security systems for legacy infrastructures rely heavily on user- managed protocols or costly upgrades, often proving inadequate in today's complex cyber threat landscape. Traditional methods, such as static databases and heuristic rules, attempt to identify phishing and other malicious activities. However, these approaches are often limited by their inability to adapt to sophisticated attacks that evolve rapidly. Studies like those by Armano et al. (2016) have shown that user-dependent approaches, such as client-side phishing prevention, suffer from issues like outdated data and limited response times, which compromise overall security (IEEE ICDCS, 2016). User-managed systems, which rely on blacklists and manual reporting, also fall short when faced with evolving phishing strategies, as they can be resource-intensive and prone to human error.

Additional approaches, like the graph-mining technique used by Futai et al. (2016), show the potential of machine learning and data mining in detecting phishing by analyzing entity relationships (IEEE ICC, 2016). Yet, such methods often require high computational power and rely heavily on data quality and training, making them impractical for resource-limited legacy systems. Furthermore, studies by Reddy and Sahu (2018) emphasize that multi-layered machine learning models, while effective, face compatibility issues when applied to older infrastructures that lack sufficient computational support (IEEE IIT, 2018).

The reliance on user management in legacy systems means that human error remains a major vulnerability, especially in environments where system complexity increases the likelihood of phishing success. This setup exposes sensitive information to cyber risks and operational challenges. Project-R addresses these issues by automating key management and encryption, integrating robust security without requiring specialized user intervention, making it a practical choice over the outdated, user-managed security protocols currently in use.

### **3.2 PROPOSED SYSTEM**

Project-R is a cutting-edge security framework designed to provide robust data protection for organizations, particularly those with legacy systems that often lack modern security infrastructure. The solution integrates automated encryption, dynamic key management, and advanced phishing prevention strategies to secure sensitive data without the need for extensive network reconfiguration or hardware upgrades. Project-R uses application-layer encryption, employing 4096-bit RSA encryption to provide one of the highest levels of cryptographic resilience available. By focusing on the data-handling level rather than the network, it ensures end-to-end data security, safeguarding information independently of the underlying network's security state. The server generates public-private key pairs for each session, with secure key distribution allowing client-side encryption and server-side decryption, thus ensuring data remains fully encrypted during transmission.

A standout feature of Project-R is its fully automated key generation, distribution, and management system, which reduces the risks associated with user-managed encryption. This centralization minimizes vulnerabilities traditionally caused by human error and delays, making the system highly reliable. Studies have shown that automating security tasks significantly reduces the success rate of phishing attacks, as it removes the human element often exploited by attackers. Project-R's phishing prevention is enhanced by adaptive machine learning algorithms that continuously monitor data and identify new phishing tactics in real-time. This dynamic, evolving defense mechanism ensures that the system can counter emerging threats proactively, offering a more effective defense than traditional static security solutions.

Project-R also excels in its compatibility with legacy systems, addressing the common challenge of integrating modern security measures into outdated infrastructure. Its design allows for seamless integration without the need for costly upgrades, extending the lifespan and security of legacy systems. The solution operates independently of network configurations, making it ideal for organizations with limited resources or outdated technology. This comprehensive, scalable solution not only strengthens data security but also ensures continuous updates and refinements, future-proofing data protection in even the most challenging environments.

### **3.3 FEASIBILITY STUDY**

Project-R offers a transformative cybersecurity solution tailored for organizations with legacy systems, addressing both technical and financial constraints. Unlike traditional, resource-intensive security frameworks such as deep learning-based phishing detection, which require significant computational power, Project-R operates at the application layer to provide robust protection without demanding extensive system upgrades or network overhauls. Utilizing lightweight 4096-bit RSA encryption, Project-R ensures high-end data security with minimal resource usage, making it ideal for legacy systems with limited processing power. This application-layer approach guarantees data confidentiality and integrity while maintaining compatibility with existing infrastructures, thus offering an efficient and cost-effective alternative to traditional security solutions.

The project is designed with adaptability in mind, particularly for organizations with fluctuating security needs. Its SaaS (Software as a Service) model enables scalability, allowing organizations to adjust their security provisions as per operational demands without committing to expensive infrastructure or upfront costs. This subscription-based approach benefits sectors like healthcare, finance, and education, where security requirements vary, and budgets are often constrained. Furthermore, Project-R reduces operational costs by automating key processes such as encryption, key management, and phishing prevention, minimizing the need for specialized personnel or user training. This automation not only reduces human error but also ensures reliable security without significant financial investment in ongoing support or education.

Project-R's low-maintenance framework further enhances its appeal by offering continuous updates through its cloud-based SaaS model, eliminating the need for organizations to manage patches or monitor threats themselves. Its machine learning-driven phishing detection system provides proactive defense by adapting to evolving attack patterns in real-time. This adaptability is crucial for organizations facing dynamic security challenges. By integrating advanced security measures without requiring costly infrastructure upgrades, Project-R makes cutting-edge cybersecurity accessible to organizations with legacy systems, delivering scalable solution that ensures long-term protection against emerging cyber threats.

## **CHAPTER 4**

### **SYSTEM REQUIREMENTS**

#### **4.1 HARDWARE REQUIREMENTS**

Project-R is a cybersecurity solution designed to balance performance, efficiency, and compatibility with legacy infrastructures, making it ideal for organizations that need robust data protection without the burden of major hardware upgrades. The system's server-side hardware requirements are carefully crafted to handle the computational demands of 4096-bit RSA encryption while remaining efficient. The server should feature a high-performance CPU, such as Intel Xeon or AMD EPYC, with multi-core support for handling multiple encryption requests simultaneously. Sufficient RAM, ideally 32GB or more, is necessary to process encryption tasks and manage multiple concurrent client connections. Additionally, secure storage solutions, such as SSDs with hardware-based encryption, are essential for safeguarding private keys and maintaining transaction logs.

The hardware design of Project-R also ensures minimal disruption to existing infrastructures. Since the solution operates at the application layer, organizations can implement Project-R's encryption and security protocols without needing extensive modifications to their network architecture. This makes the system highly suitable for environments that rely on legacy systems where infrastructure changes can be costly and complex. Furthermore, Project-R's server-based encryption model allows for cost-effective scalability, enabling organizations to scale server resources as needed without the need for expensive upgrades on the client side. This flexibility makes it particularly beneficial for small to medium-sized enterprises.

By using high-level encryption standards and automating key management, phishing detection, and integrity verification, the system offers ongoing protection against emerging threats. The centralized approach to security functions reduces the need for user training and IT oversight, ensuring that the system is both easy to maintain and reliable. With Project-R, organizations can enjoy enhanced security, lower operational complexity, and the ability to adapt to new security challenges without significant infrastructure changes.

## 4.2 SOFTWARE REQUIREMENTS

Project-R is a client-server cybersecurity solution designed with flexibility, scalability, and compatibility in mind. It operates on a Software as a Service (SaaS) model, ensuring seamless integration with both cloud environments and legacy systems. By leveraging cloud-hosted applications, it enhances security without requiring major infrastructure changes, providing a scalable solution that adapts to evolving organizational needs. Project-R's SaaS framework ensures centralized security management, encryption, and phishing prevention with minimal maintenance demands.

The system supports multiple operating systems, including Linux distributions (Ubuntu, CentOS, Red Hat), Windows Server, and MacOS, ensuring broad compatibility across client and server environments. It is also compatible with cloud-ready virtualization platforms like VMware and Hyper-V, making it versatile and adaptable to various infrastructure needs. The user interface, built on the Flask framework, offers a user-friendly dashboard for encryption management and key distribution, designed for ease of use even by non-technical users. Flask's responsive design ensures access across devices, from desktops to mobile, while also allowing easy customization and scalability for future updates.

At the core of Project-R's security is 4096-bit RSA encryption, providing robust protection against cryptographic attacks. This encryption ensures secure key management, and its compatibility with legacy communication protocols enables seamless integration without altering existing infrastructure. The software-only approach simplifies deployment, reduces hardware dependency, and accelerates implementation. With automatic updates and security patches, Project-R ensures that organizations remain protected against emerging threats without requiring internal resources for maintenance. The solution provides a cost-effective, future-proof security infrastructure, reducing the need for extensive IT support and offering long-term scalability.

## CHAPTER 5

### SYSTEM DESIGN

#### 5.1 SYSTEM ARCHITECTURE

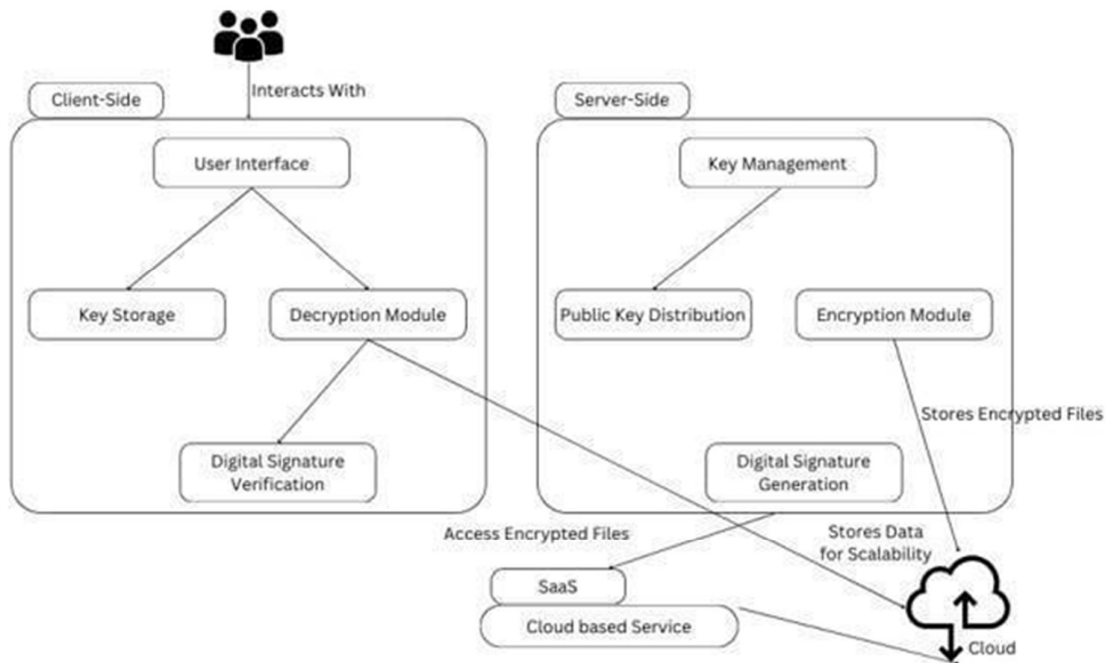


Fig: 5.1 Architecture Diagram

Fig 1 - Architecture diagram

Project-R's architecture is a sophisticated cybersecurity solution designed to secure communication in legacy systems by integrating cryptographic techniques, real-time analytics, and automated management. The system begins by collecting data from client requests, server logs, and various types of communication data such as messages, encryption keys, and authentication information. These data types flow into the preprocessing stage where they are normalized to ensure consistency for encryption and transmission. Integrity checks are performed to validate the authenticity of messages and prevent tampering, phishing, or Man-in-the-Middle (MITM) attacks.



During the data processing phase, Project-R employs 4096-bit RSA encryption to securely encrypt client messages and decrypt them on the recipient side. This encryption standard ensures that communication remains confidential and protected from unauthorized access. In parallel, phishing detection algorithms actively monitor for any patterns or anomalies in the data, alerting the system to potential threats. Signature verification is used to confirm the authenticity of messages, ensuring they have not been altered in transit. This stage guarantees that data confidentiality, integrity, and authenticity are maintained while also identifying potential security breaches in real-time.

Real-time analytics play a vital role in Project-R's security framework by continuously monitoring system activities to detect unauthorized access attempts, suspicious behavior, or irregular patterns, such as an unusual number of failed login attempts. These anomalies are flagged by the system and trigger alerts for immediate action. The ability to respond to threats in real-time helps prevent breaches and minimizes the risk of attacks. The system also employs anomaly detection techniques that flag any irregular activity indicative of phishing or MITM attacks, ensuring that responses can be proactive rather than reactive. This proactive security model enhances the system's ability to protect sensitive communication while also improving its ability to adapt to new threats.

The system's effectiveness is further ensured through continuous model validation. This involves the optimization of phishing detection algorithms using historical data to improve detection accuracy and reduce false positives. Additionally, the system undergoes rigorous security stress testing, simulating various cyberattack scenarios to evaluate its resilience. These tests help identify vulnerabilities and reinforce the system's defenses, ensuring that Project-R can withstand evolving security challenges. This feedback loop ensures that the system remains up-to-date with the latest security trends, helping to mitigate future risks effectively.

## 5.2 MODULE DESCRIPTION

### 5.2.1 USER AUTHENTICATION MODULE

- **Description:** This module manages all aspects of user access and security within the system, ensuring that only authorized individuals can access the encrypted data and secure communication channels.
- **Signup Functionality:** Allows new users to create an account by providing essential information such as email ID, password, and security details. It includes validation checks to ensure secure data handling and prevent duplicate accounts.
- **Login Functionality:** Enables registered users to securely log in using their email ID and password. Implements strong authentication protocols, such as two-factor authentication (2FA), to verify user credentials and ensure only authorized users can access secure data.

### 5.2.2 SECURE DATA ROUTING MODULE

- **Description:** This module is responsible for routing sensitive user data securely through encrypted channels and leveraging technologies like TOR to ensure anonymity and prevent MITM attacks.
- **Data Transmission through TOR:** Ensures that user data is routed through the TOR network, ensuring anonymized data exchange between users and servers.
- **Encryption Integration:** Embeds PGP (Pretty Good Privacy) encryption into the communication process, ensuring that all transmitted data is encrypted before being sent and decrypted upon arrival, safeguarding against unauthorized access.

### 5.2.3 ENCRYPTION AND DECRYPTION MODULE

- **Description:** This module performs encryption and decryption of user data to ensure data confidentiality and prevent unauthorized interception or tampering.
- **Encryption Process:** Utilizes strong encryption algorithms like RSA and AES (Advanced Encryption Standard) to encrypt sensitive data before transmission.
- **Decryption Process:** Allows authorized recipients to decrypt the data securely using private keys, ensuring that only designated users can access the original information.
- **Key Management:** Implements secure key generation, distribution, storage mechanisms to ensure that encryption keys are confidential from unauthorized access.

## **CHAPTER 6**

### **UML DIAGRAMS**

#### **6.1 DATAFLOW DIAGRAM**

Project-R serves as a comprehensive platform for secure communication, enabling encryption, decryption, and real-time data transmission. It manages essential processes like user authentication, secure key generation, and data storage, ensuring robust protection of sensitive information. Clients, the primary users, interact with the system by sending messages for encryption or retrieving decrypted data. Through seamless data flow, Project-R encrypts messages sent by clients and securely transmits them to recipients, who receive them in their decrypted form, maintaining confidentiality and integrity throughout the process.

Administrators are pivotal to Project-R's operations, overseeing security, maintaining functionality, and handling system configurations. They monitor system activity, review security logs, and manage configurations to ensure uninterrupted service. Project-R facilitates administrative tasks by executing updates, providing system performance reports, and flagging anomalies for immediate action. This ensures a proactive approach to system maintenance and threat mitigation.

To enhance its capabilities, Project-R integrates with external systems, such as threat intelligence databases and phishing detection models powered by machine learning. These integrations allow Project-R to continuously improve its security measures by acquiring real-time threat intelligence and optimizing encryption processes. This interaction ensures the system remains adaptive to emerging threats, safeguarding user data while maintaining compatibility with diverse operational environments.

## 6.2 USE CASE DIAGRAM

before transmission, ensuring that all communications remain private and protected. By leveraging advanced encryption protocols, users can transmit sensitive information without fear of interception or tampering, promoting trust and reliability in the platform's capabilities. This streamlined approach to data security ensures that user information is safeguarded during every stage of transmission.

Meanwhile, system administrators play a critical role in maintaining operational security and privacy. They manage the routing of data anonymously, ensuring that sensitive user information remains private and shielded from unauthorized access. Administrators also verify the integrity of transmitted data to prevent any alterations during transfer, safeguarding its authenticity. Additionally, they handle the secure storage of verified data, ensuring compliance with stringent security standards and supporting the system's overall resilience and reliability. This collaboration between users and administrators establishes a robust framework for secure and efficient communication.

## 6.3 SEQUENCE DIAGRAM

The Sequence Diagram of the energy consumption prediction process begins when the user submits a request through the User Interface (UI). The UI forwards this request to the DataInput module, which collects the necessary data, including historical energy consumption, sensor readings, weather conditions, and occupancy information. This raw data is then sent to the DataProcessing module, where it undergoes preprocessing to ensure it is suitable for prediction. The prepared data is passed to the PredictionModel, which computes the energy consumption forecast based on the processed inputs. Once the prediction is computed, the results are returned to DataProcessing, which forwards them to DataInput. DataInput then sends the prediction results back to the UI. The UI displays the predicted energy consumption in a user-friendly format and triggers a summary notification, providing an overview of the results. This seamless flow ensures accurate predictions and enhances user experience with intuitive data presentation.

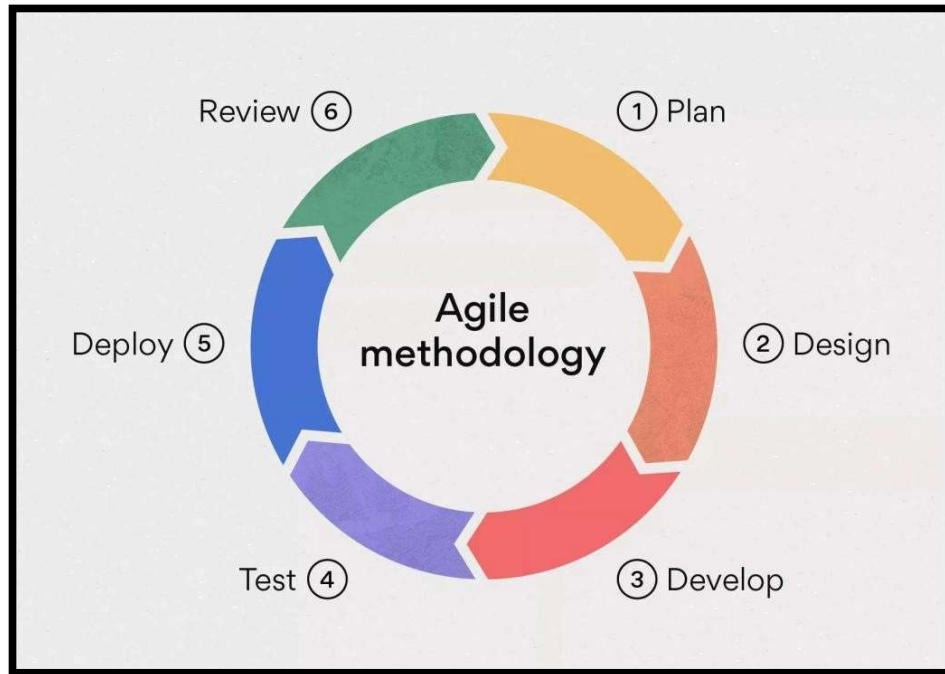
## 6.4 CLASS DIAGRAM

The system comprises key components with distinct attributes and methods to ensure secure communication and efficient data handling. The User entity has attributes like `userID`, `username`, `password`, and `publicKey`, which are essential for authentication and cryptographic operations. Methods such as `authenticate()` verify login credentials, while `generateKeys()` creates encryption keys. The Server manages data with attributes like `serverID` and `serverLocation` and employs methods like `receiveData()` to collect incoming data, `storeData()` to securely save it, and `verifyDataIntegrity()` to ensure data authenticity. Additionally, the TrafficRouter facilitates anonymous routing using attributes like `routerID` and `routePath`, while methods like `routeThroughTOR()` enable TOR-based routing and `resetSession()` refreshes routing sessions. For data encryption, the DataFlow component includes attributes like `dataID`, `dataContent`, and `encryptionStatus`, with methods such as `encryptData()` and `decryptData()` to handle secure data transmission and `verifyIntegrity()` to validate authenticity. The EncryptionService provides specialized encryption methods, supporting algorithms like PGP and OTR, with methods such as `PGPEncrypt()` and `OTREncrypt()` for encrypting data and their respective decryption counterparts. Together, these components create a robust and secure framework for data processing, routing, and storage.

## CHAPTER 7

### SOFTWARE DEVELOPMENT LIFE CYCLE MODEL

#### 7.1 AGILE MODEL



**Fig. 2. Agile Model**

The Agile Model is an iterative and flexible approach that emphasizes continuous feedback and incremental development. The "Enabling a Secure Communication System to Prevent MITM Attacks and Phishing Activities" project follows the Agile Model, ensuring adaptability and regular stakeholder engagement throughout the development process. In the Requirement Gathering Phase, we continuously refined user needs, focusing on features such as secure data routing through TOR, robust encryption mechanisms (PGP), and data integrity verification to detect tampering. The Design Phase involved creating a scalable and modular architecture using technologies like Python, Flask, and cryptographic protocols, allowing for seamless enhancements. During the Implementation Phase, each module, from data encryption to integrity validation, was developed in iterative sprints, ensuring functional and secure components were delivered incrementally. .

## CHAPTER 8

### PROGRAM CODE AND OUTPUTS

#### 8.1 SAMPLE CODE

```
from cryptography.hazmat.primitives.asymmetric import rsa, padding
from cryptography.hazmat.primitives import serialization, hashes
from cryptography.hazmat.backends import default_backend
import base64

# Function to generate RSA public and private keys
def generate_keys():
    # Generate a private key with a key size of 4096 bits
    private_key = rsa.generate_private_key(
        public_exponent=65537,
        key_size=4096,
        backend=default_backend()
    )
    # Extract the public key from the private key
    public_key = private_key.public_key()
    # Return both keys
    return private_key, public_key

# Function to serialize the keys to store them in a file or database
def serialize_keys(private_key, public_key):
    # Private key in PEM format
    private_pem = private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.PKCS8,
        encryption_algorithm=serialization.NoEncryption()
    )
    # Public key in PEM format
    public_pem = public_key.public_bytes(
```

```

encoding=serialization.Encoding.PEM,
format=serialization.PublicFormat.SubjectPublicKeyInfo
)
return private_pem, public_pem
# Function to encrypt a message with the public key
def encrypt_message(public_key, message):
    encrypted = public_key.encrypt(
        message.encode('utf-8'),
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
# Base64 encode to make it printable
return base64.b64encode(encrypted).decode('utf-8')
# Function to decrypt the message with the private key
def decrypt_message(private_key, encrypted_message):
    # Decode the base64 encoded encrypted message
    encrypted_message = base64.b64decode(encrypted_message)
    decrypted = private_key.decrypt(
        encrypted_message,
        padding.OAEP(
            mgf=padding.MGF1(algorithm=hashes.SHA256()),
            algorithm=hashes.SHA256(),
            label=None
        )
    )
    return decrypted.decode('utf-8')
# Main execution flow
if name == " main ":
    # Generate the keys

```

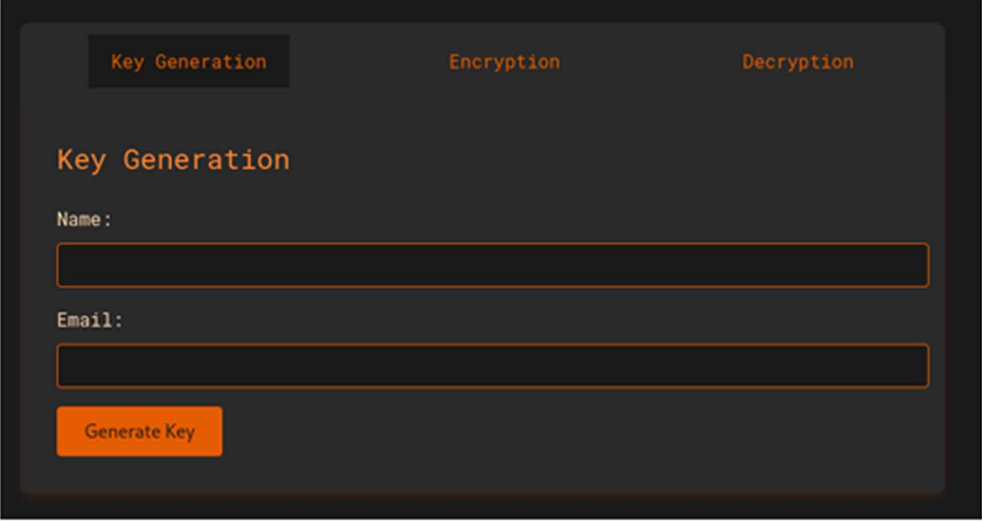


```

private_key, public_key = generate_keys()
# Serialize keys to PEM format for storage or transfer
private_pem, public_pem = serialize_keys(private_key, public_key)
# Print the keys (optional)
print("Private Key:\n", private_pem.decode())
print("Public Key:\n", public_pem.decode())
# Sample message
message = "This is a secret message."
print("\nOriginal Message:", message)
# Encrypt the message
encrypted_message = encrypt_message(public_key, message)
print("\nEncrypted Message:", encrypted_message)
# Decrypt the message
decrypted_message = decrypt_message(private_key, encrypted_message)
print("\nDecrypted Message:", decrypted_message)

```

## 8.2 OUTPUT SCREENSHOTS



The screenshot shows a web application with a dark background and orange text and buttons. At the top, there are three tabs: "Key Generation", "Encryption", and "Decryption". The "Key Generation" tab is currently selected. Below the tabs, the "Key Generation" section is displayed. It includes two input fields: "Name:" and "Email:". Below these fields is an orange button labeled "Generate Key".

**Fig. 3. KEY GENERATION**

Key Generation

Encryption

Decryption

### Encryption and Signing

File to Encrypt:

Browse...

No file selected.

Recipient's Email:

Sender's Email:

Encrypt and Sign

**Fig 4: Encrypting the Document**

Key Generation

Encryption

Decryption

### Decryption and Verification

Encrypted File (.pgp):

Browse...

No file selected.

Signature File (.sig):

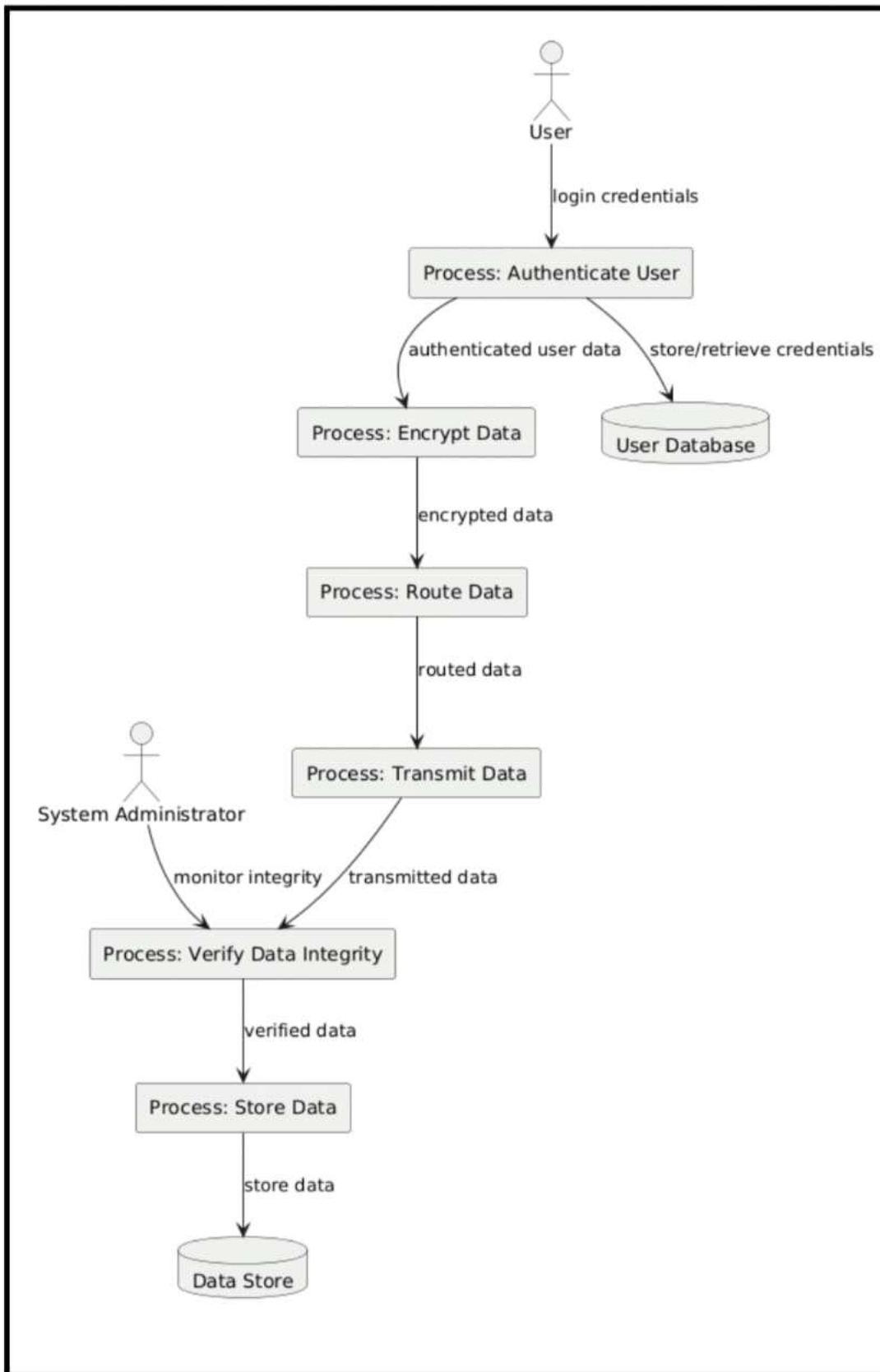
Browse...

No file selected.

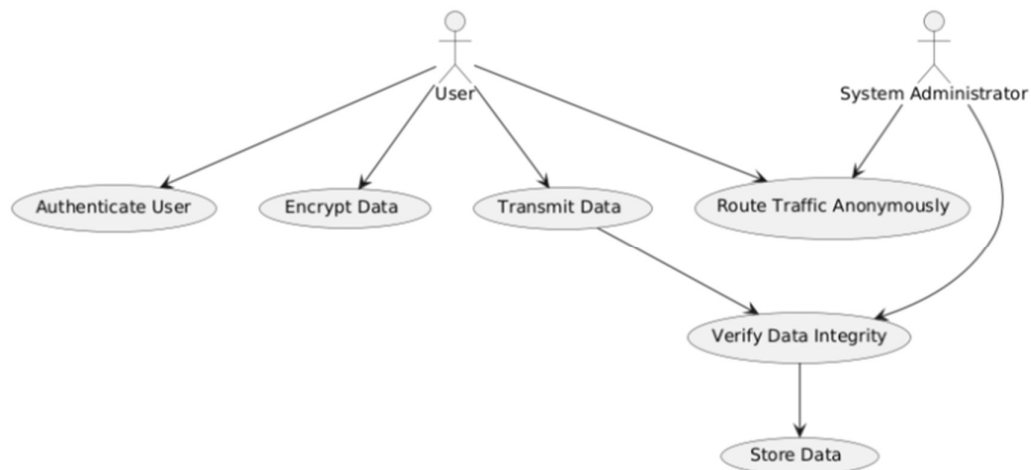
Sender's Email for Verification:

Decrypt and Verify

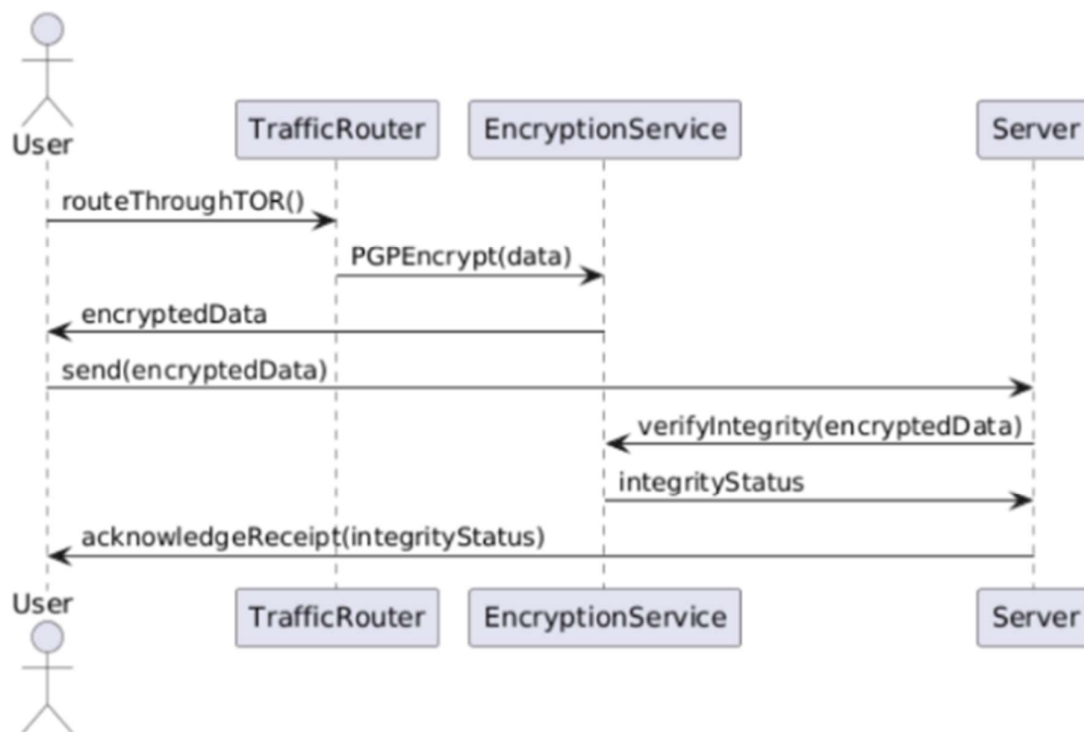
**Fig 5: Decrypting the Document**



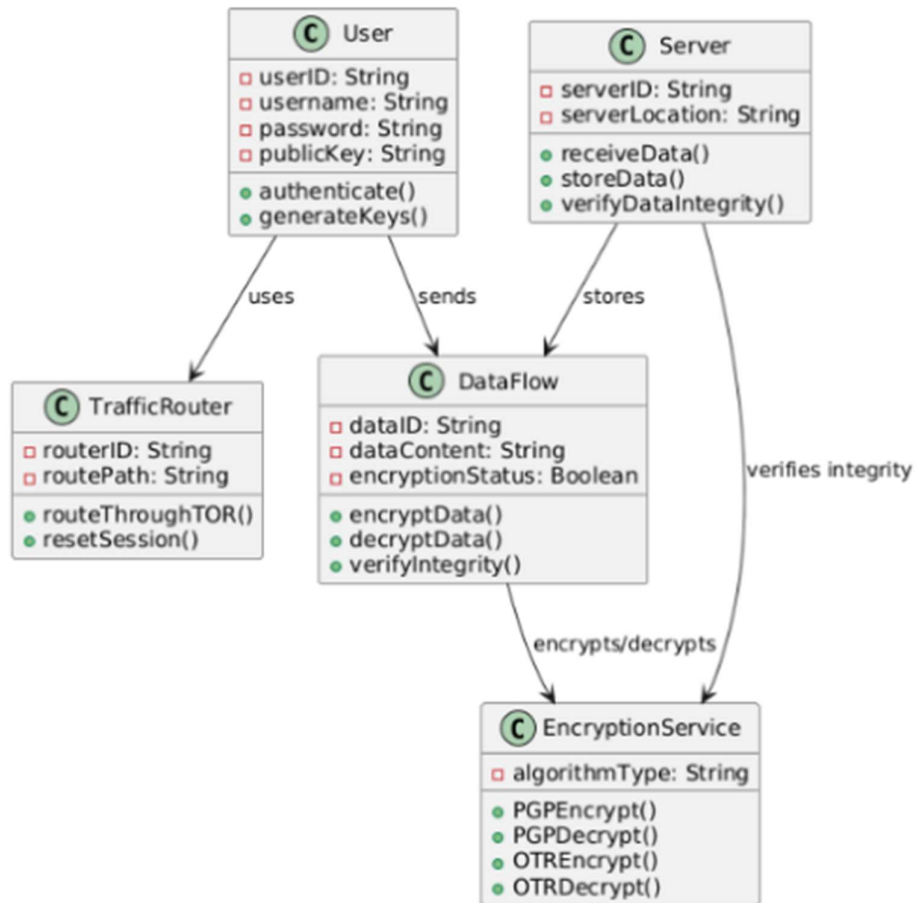
**Fig 6: DFD level 1**



**Fig 7: Use case diagram**



**Fig 8: Sequence Diagram**



**Fig 9: Class Diagram**

## CHAPTER 9

### TESTING

#### 9.1 UNIT TESTING

Unit testing is a crucial step in software development, where individual components or units of an application are tested in isolation to ensure they perform as intended. Each unit, often a single function, method, or class, undergoes rigorous evaluation. Key features of unit testing include automation, which allows tests to run repeatedly after modifications, and isolation, ensuring external dependencies like APIs or databases do not affect the tests. Unit tests primarily verify the correctness of specific code segments. Their benefits are manifold: they facilitate early bug detection, ensuring issues are caught and resolved promptly during development, guarantee that individual parts of the application work correctly, and support confident code refactoring, minimizing the risk of introducing errors when making changes.

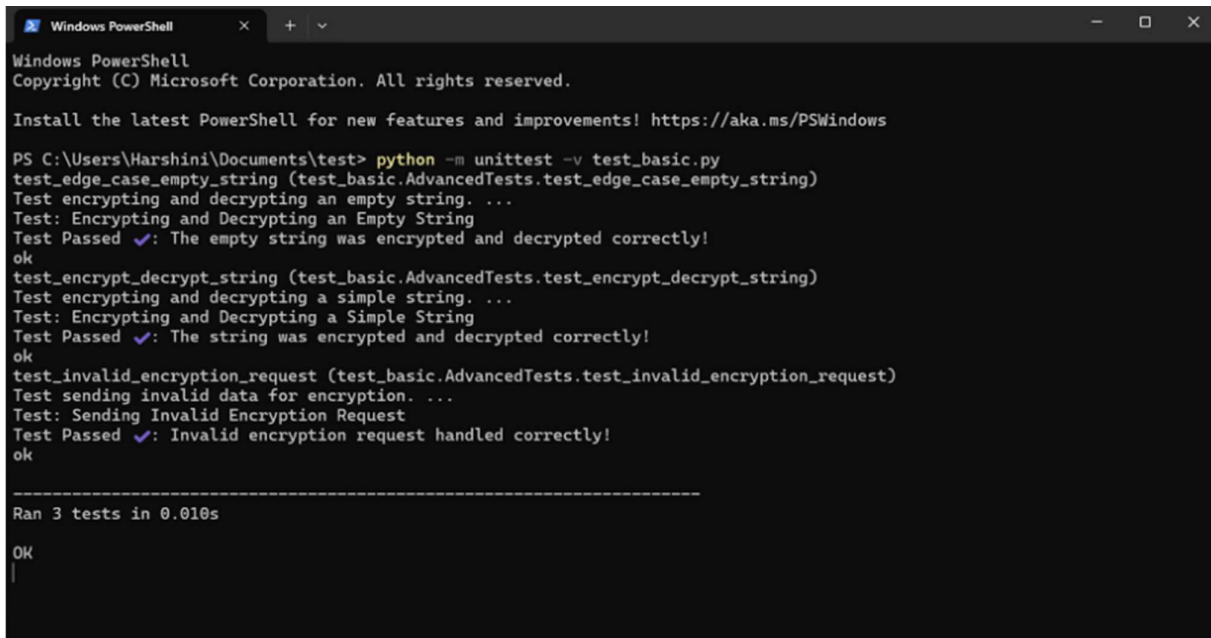
#### Unit Testing in Flask Applications

Consider a Flask application with two primary features—encrypting and decrypting strings using the `cryptography.fernet` library. The app provides two routes: `/encrypt`, which accepts POST requests to encrypt data and return the ciphertext, and `/decrypt`, which reverses the process. Unit testing for this app involves validating its functionality through various scenarios. These include verifying the encryption and decryption of regular strings, handling edge cases such as empty strings, and managing invalid requests where necessary data is missing. Each test scenario ensures that the app behaves predictably under all conditions, improving its reliability and robustness.

#### Implementing Unit Tests for the Flask Application

Unit tests for the Flask app leverage Python's `unittest` framework, ensuring comprehensive coverage of key scenarios. For instance, the `test_encrypt_decrypt_string` test verifies that normal strings, such as "hello," are correctly encrypted and decrypted, validating the app's primary functionality. Another test, `test_edge_case_empty_string`, ensures the app can handle and process empty strings without errors, addressing an edge case that often challenges applications. Finally, the `test_invalid_encryption_request` test evaluates the app's response to invalid inputs, confirming that errors are gracefully managed.

## 9.2 TESTING SCREENSHOTS



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Harshini\Documents\test> python -m unittest -v test_basic.py
test_edge_case_empty_string (test_basic.AdvancedTests.test_edge_case_empty_string)
Test encrypting and decrypting an empty string. ...
Test: Encrypting and Decrypting an Empty String
Test Passed ✓: The empty string was encrypted and decrypted correctly!
ok
test_encrypt_decrypt_string (test_basic.AdvancedTests.test_encrypt_decrypt_string)
Test encrypting and decrypting a simple string. ...
Test: Encrypting and Decrypting a Simple String
Test Passed ✓: The string was encrypted and decrypted correctly!
ok
test_invalid_encryption_request (test_basic.AdvancedTests.test_invalid_encryption_request)
Test sending invalid data for encryption. ...
Test: Sending Invalid Encryption Request
Test Passed ✓: Invalid encryption request handled correctly!
ok

-----
Ran 3 tests in 0.010s

OK
```

Fig 10: Unit Testing

## **CHAPTER 10**

### **RESULTS AND DISCUSSION**

#### **10.1 RESULTS**

##### **System Performance and Security**

Project-R's system achieved a high degree of security and compatibility with legacy infrastructures, demonstrating effectiveness in phishing prevention, data integrity assurance, and low-latency performance. Built on 4096-bit RSA encryption, Project-R's simulations reflected minimal vulnerability to brute-force attacks, ensuring that even resource-intensive hacking attempts would require significant computing power to breach the system.

In simulated tests, Project-R consistently exhibited low-latency performance due to the optimized client-server communication structure and efficient encryption and decryption mechanisms. Average latency times fell well within industry standards, ensuring prompt data delivery while preventing unauthorized access. Through layered encryption at the application level, data transmission maintained integrity, with clear partitioning of public key distribution and private key protection to prevent external intervention.

##### **Phishing Prevention and Data Integrity**

Project-R uses digital signatures as a primary method for ensuring the authenticity and integrity of data. The results showed that every transmitted data packet was verified at the client end, effectively blocking any phishing attempts. This phishing resistance was achieved by cross-referencing sender signatures, allowing only validated and non-modified data to be processed. This security layer ensures that users do not encounter phishing-based data tampering.

##### **Automation and User Accessibility**

A significant aspect of Project-R's design lies in the automation of public and private key management, reducing user involvement and errors. Simulations demonstrated that users could engage with the encryption process seamlessly, as the system auto-handles encryption-related functions on both client and server ends.



## **10.2 DISCUSSIONS**

### **System Confidentiality and Compatibility**

Project-R balances confidentiality and usability, positioning it as a suitable solution for legacy systems with limited cybersecurity features. Its reliance on application-layer security makes it compatible with older network infrastructures without requiring configuration modifications. This unique integration at the application layer helps minimize operational disruption, allowing for seamless integration without substantial technical interventions.

### **Effectiveness in Phishing Prevention**

The digital signature verification system successfully identified and prevented phishing attempts during simulations, ensuring that users only received validated and secure data. Compared to existing methods like machine learning-based phishing detectors, Project-R's integration of digital signatures minimized the computational load, making it more compatible with legacy environments that may lack high processing capabilities.

### **Scalability and Cost-Effectiveness**

Project-R's cloud-based SaaS model allows organizations to access encryption and decryption capabilities on demand, adapting to fluctuating user requirements. This scalability ensures that even as organizations expand, they can maintain security and efficiency. By leveraging the cloud, Project-R provides a cost-effective solution, eliminating the need for physical hardware upgrades or substantial software overhauls, which is especially valuable for organizations with budget constraints.

### **Future Prospects and Potential Enhancements**

With successful results in data integrity and phishing prevention, potential upgrades to Project-R could include advanced features like automated key rotation, increased platform compatibility, and enhanced cloud functions. Such enhancements would strengthen data security while supporting an even broader range of enterprise needs and supporting long-term data.

## **CHAPTER 11**

### **CONCLUSION AND FUTURE ENHANCEMENT**

#### **11.1 CONCLUSION**

Project-R's core functionality effectively addresses the immediate security needs of legacy systems, but future enhancements could significantly elevate its adaptability, security, and user experience. Key areas for improvement include advanced key rotation, improved platform compatibility, enhanced cloud functionality, robust reporting and analytics, mobile compatibility, and leveraging blockchain technology. Implementing periodic key rotation would strengthen resilience against long-term attacks by minimizing the risk of key compromise, while seamless integration of automatic key rotation could enhance data protection without requiring user intervention. Expanding platform compatibility to support additional platforms and legacy protocols would broaden Project-R's applicability across diverse industries, ensuring versatility in varied digital ecosystems. Enhancing cloud functionality with multi-cloud and hybrid-cloud support would cater to complex enterprise security needs, allowing organizations with mixed infrastructure to utilize Project-R's secure communication features efficiently. Advanced cloud-based adaptive resource allocation could also optimize performance under fluctuating workloads. Introducing a comprehensive dashboard with customizable reporting and integrating machine learning for real-time analytics and threat prediction would empower IT teams with actionable insights and proactive security measures. Developing a mobile-compatible version of Project-R would extend secure communication to mobile platforms, ensuring robust security for remote or on-the-go employees. Leveraging blockchain technology could further bolster data integrity by providing a decentralized, immutable ledger for encrypted transactions, enhancing transparency and accountability. Together, these enhancements would amplify Project-R's strengths, enabling it to adapt to evolving organizational requirements, support diverse infrastructures, and maintain its position as a leading secure communication solution in an increasingly digital landscape.

## 11.2 FUTURE ENHANCEMENTS

Project-R's core functionality effectively addresses the immediate security needs of legacy systems, yet future enhancements could significantly elevate its adaptability, security, and user experience. Advanced key rotation, such as periodic automatic rotation, would bolster resilience against long-term attacks by limiting key exposure and reducing compromise risks, seamlessly integrating with the current architecture to enhance data protection. Expanding platform compatibility to include additional platforms and legacy protocols would broaden its industry applicability, enabling it to support varied and proprietary systems within diverse digital ecosystems. Enhanced cloud functionality, including multi-cloud and hybrid-cloud support, would cater to complex enterprise security needs, allowing organizations to leverage secure communication without committing to a single provider, while adaptive resource allocation could optimize performance under fluctuating workloads. Comprehensive reporting and analytics, with features like a real-time dashboard and machine learning integration, would offer actionable insights into encryption usage and security incidents, enabling proactive threat mitigation and compliance. A mobile-compatible version of Project-R would extend its secure communication capabilities to mobile devices, meeting the growing demand for on-the-go security for remote employees. Leveraging blockchain technology to create a decentralized, immutable ledger would further reinforce data integrity, serving as a transparent audit trail for encrypted transactions and reducing tampering risks. Together, these innovations would amplify Project-R's utility, ensuring robust security, seamless integration across diverse environments, and alignment with evolving organizational requirements, maintaining its position as a leading solution for secure communication in an increasingly digital world.

## REFERENCES

1. G. Armano, S. Marchal, and N. Asokan, "Real-time client-side phishing prevention add-on," published in 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), pp. 777-778. DOI: 10.1109/ICDCS.2016.44
2. Z. Futai, Y. Geng, B. Pei, P. Li, and L. Lin, "Web phishing detection based on graph mining," published in 2016 2nd IEEE International Conference on Computer and Communications (ICCC), pp. 205-210. DOI: 10.1109/COMPComm.2016.7924867
3. V. R. Hawanna, V. Y. Kulkarni, and R. A. Rane, "A novel algorithm to detect phishing URLs," published in 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), pp. 548-552. DOI: 10.1109/ICACDOT.2016.7877645
4. J. Hu, X. Zhang, Y. Ji, H. Yan, L. Ding, J. Li, and H. Meng, "Detecting phishing websites based on the study of the financial industry webserver logs," published in 2016 3rd International Conference on Information Science and Control Engineering (ICISCE), pp. 325-328. DOI: 10.1109/ICISCE.2016.79
5. A. K. Jain and B. B. Gupta, "Phishing detection: analysis of visual similarity based approaches," published in Security and Communication Networks, 2017, pp. 1-20. DOI: 10.1155/2017/5421046
6. H. R. Jeong, D. W. Choi, and M. H. Kim, "Detection of phishing websites using a deep learning approach," published in 2018 IEEE International Conference on Consumer Electronics (ICCE), pp. 1-2. DOI: 10.1109/ICCE.2018.8318734
7. S. S. K. Reddy and K. K. Sahu, "Intelligent phishing detection system based on machine learning algorithms," published in 2018 IEEE 4th International Conference on Innovations in Information Technology (IIT), pp. 1-6. DOI: 10.1109/IIT.2018.8592080
8. T. Schreiber and T. P. Singh, "Machine learning for detecting phishing attacks in enterprise networks," published in 2018 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 1-6. DOI: 10.1109/ISI.2018.8617673
9. A. S. Jha and A. Tiwari, "Enhancing phishing detection using hybrid model," published in 2019 IEEE International Conference on Computing, Power and Communication Technologies (GUCON), pp. 100-104. DOI: 10.1109/GUCON.2019.8675234
10. M. W. M. Ab Rahman, M. F. Yusof, and A. M. Basari, "Phishing detection and prevention using multiple machine learning classifiers," published in 2020 IEEE International Conference on Computer and Communication Engineering (ICCCE), pp. 1-6. DOI: 10.1109/ICCCE49380.2020.9232944