PGP

IMPLEMENTATION OF PGP

Enabling a Secure Communication System

. INTRODUCTION

PGP Encryption

- **Definition**: A method to secure emails and files through encryption. **Mechanism**:**Symmetric and Asymmetric Encryption**: Uses both methods for protection.
- **Encryption Process**: Converts messages into a scrambled format.
- **Decryption**: Only the intended recipient with the specific key can read the message.
- **Protection Prevents**: Eavesdropping and data breaches.
- **Usage**: Common in commercial settings to secure sensitive information and communications.

.OBJECTIVE

Objective

- **Definition**: A method to secure emails and files through encryption.**Mechanism**:**Symmetric and Asymmetric Encryption**: Uses both methods for protection.
- **Encryption Process**: Converts messages into a scrambled format.
- **Decryption**: Only the intended recipient with the specific key can read the message.
- **Protection**:**Prevents**: Eavesdropping and data breaches.
- **Usage**: Common in commercial settings to secure sensitive information and communications.

LITERATURE SURVEY

PGP

S.no	Author	Year	Techniques / Methods	Drawbacks
01	S. Khan, F. et. all	2022	Advancement in Vehicular Public key Infrastructures. (VPKI)	Implementation is relatively hard.
02	Mauro Barni. et. all	2023	Digital WatermarkingRobust HashingSteganography and SteganalysisBiometrics	Challenges in adapting to rapid technological advancements, such as deepfakes and synthetic mediaNeed for evolving legal frameworks to manage technological advancements and prevent misuseBalancing the benefits of new technologies with privacy concerns and potential misuse
03	Jingwei Jiang. et. all	2024	Quantum-Resistant Password-Authenticated Symmetric Searchable Encryption (QPASE)Lattice-Based CryptographyThreshold Oblivious Pseudorandom Function	Challenges in adapting traditional cryptographic schemes to post-quantum environments
04	Jingwei Jiang. et. All	2024	Quantum-Resistant Password-Protected Secret Sharing (PPSS)Lattice-Based CryptographyQuantum-Resistant Data Outsourcing	Challenges in ensuring robustness against various quantum computing attacksPotential performance trade-offs compared to non-quantum-resistant schemes

S.no	Author	Year	Techniques / Methods	Drawbacks
05	V.G. Karantaev. et. all	2023	Implementation of Cryptographic Measures in Digital SubstationsSecure Inter-Network Communication for IEC 61850-8-1 (MMS) Protocol	Challenges in effectively securing digital communication protocols used in highly automated substations
06	Vincenzo De Angelis. et. all	2022	Digital Health (eHealth) Innovation and ChallengesIntegration of 6G Wireless Networks in Healthcare	Existing network infrastructures may not fully support the digitalization of healthcare.
07	Cristian Bermudez Serna. et. all	2023	Post-Quantum Cryptography (PQC) for Shared Mutual Authentication (SMA)Implementation of Kyber Algorithm in SMA	Kyber-based SMA requires more random bytes and has a longer execution time compared to baseline mechanisms.
08	Mi Song, Ding Wang	2024	Two-Factor AuthenticationAttribute-Based Password Authenticated Key Exchange (AB-PAKE)Flexible and Fine-Grained AuthorizationPrivacy Preservation and Dynamic Access Control	Although the protocol is round-optimal and reduces pairing operations, its implementation and management of storage devices could be complex.

Problem Finding

Our literature survey highlighted several critical issues:

1. Inadequate Encryption Implementation: Many systems lack robust encryption, leaving sensitive data exposed to unauthorized access and breaches. Proper encryption is crucial for protecting data confidentiality during transmission and storage.

2. Risk of Data Manipulation: Without effective encryption, data is vulnerable to manipulation and tampering. This can compromise the accuracy and reliability of information, impacting decision-making and operational integrity.

3. Insufficient Integrity Checks: Many systems do not employ adequate integrity verification, making it difficult to detect unauthorized changes. This vulnerability increases susceptibility to attacks such as phishing, where attackers can deceive users into compromising sensitive information.

Addressing these gaps is essential to enhancing data security and protecting against cyber threats.

PGP

90%

OF SUCCESSFUL CYBER-ATTACKS START WITH PHISHING EMAILS

SOLUTION

We propose using TAILS OS and TOR to route traffic, which ensures user anonymity and leaves no traces by running entirely from RAM and restarting cleanly. This solution enhances security by protecting data from manipulation and unauthorized access.

Using the PGP mechanism allows us to perform integrity checks to detect any spam or unauthorized data flows.

We use a decentralized server and PGP encryption to secure data, with OTR encryption adding an extra layer of protection for enhanced privacy.

ARCHITECTURE

1. Data Flow Initiation:
- Server sends data to Client A.
- Client A requests public/private keys using PGP.

2. Key Generation:
- Key Management Service (KMS) generates keys.
- Client A receives a public key.

3. Data Encryption:
- Client A encrypts data with Client B's public key.

4. Data Transmission:
- Encrypted data is sent to Client B.
- Data is secure, even if intercepted.

5. Data Decryption and Integrity Verification:
- Client B decrypts data with its private key.
- Verifies data integrity using PGP mechanisms.

Server → Client A (Implementation)
- Detects Destination
- Requests PGP Key Pair

Key Management Service (KMS)
- Generates Public and Private Keys
- Sends Public Key to Client A

PROCESS 1

Client A
- Encrypts Data with Client B's Public Key

Network ("with notation: Encrypted Data Transfer")
- Potential Sniffing Attack

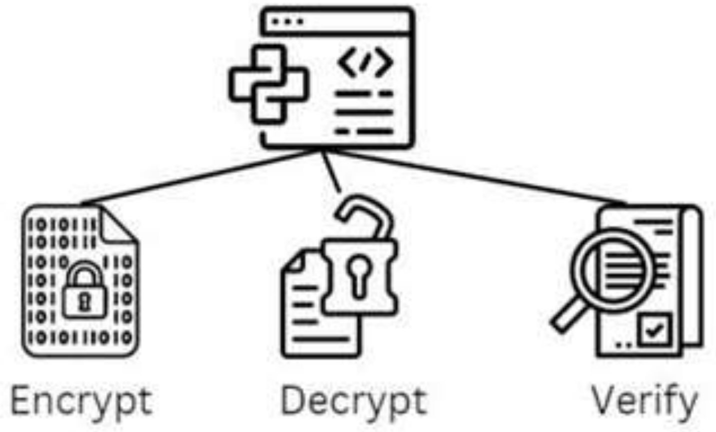
PROCESS 2

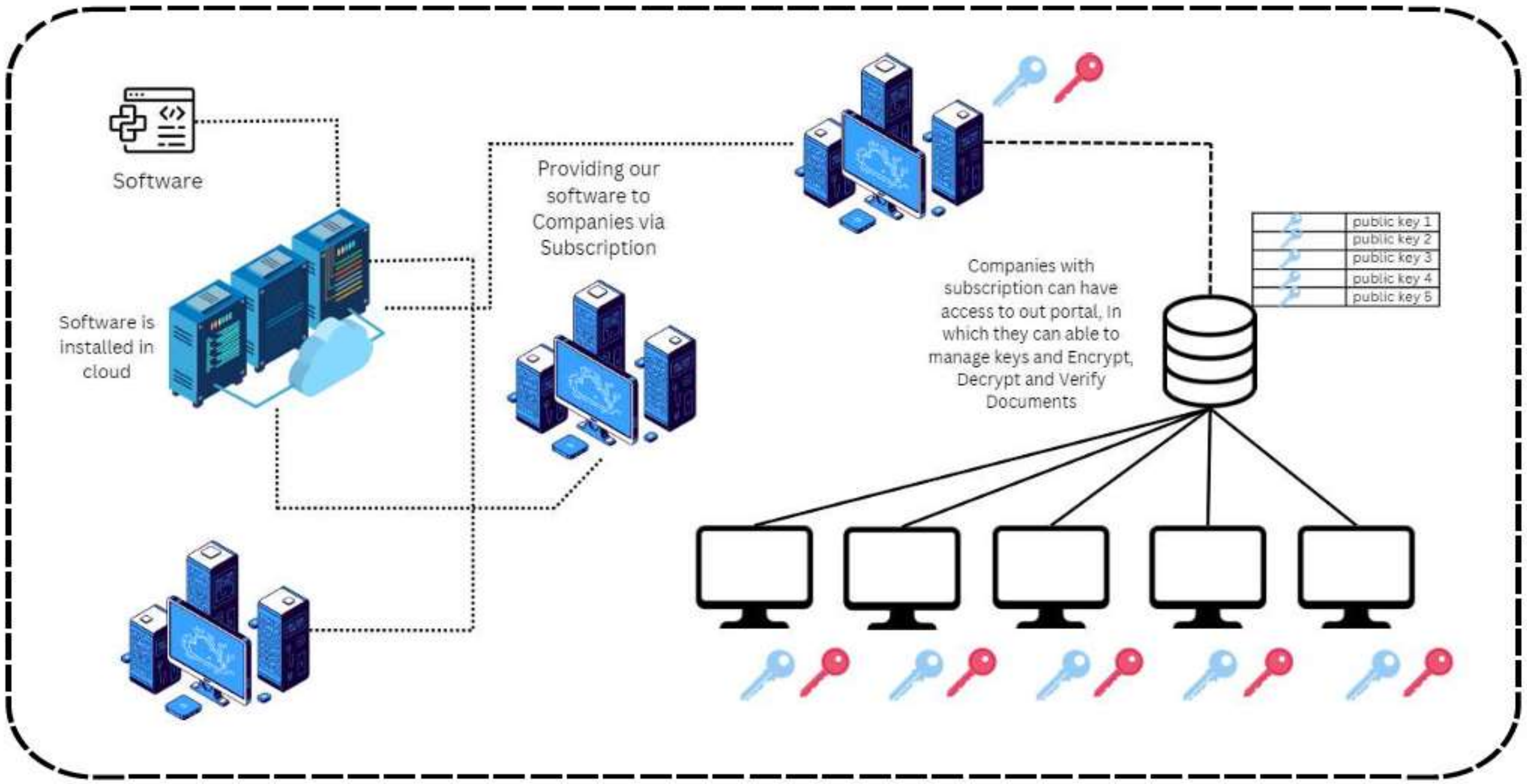
Client B
- Decrypts Data with Private Key
- Verifies Integrity with PGP

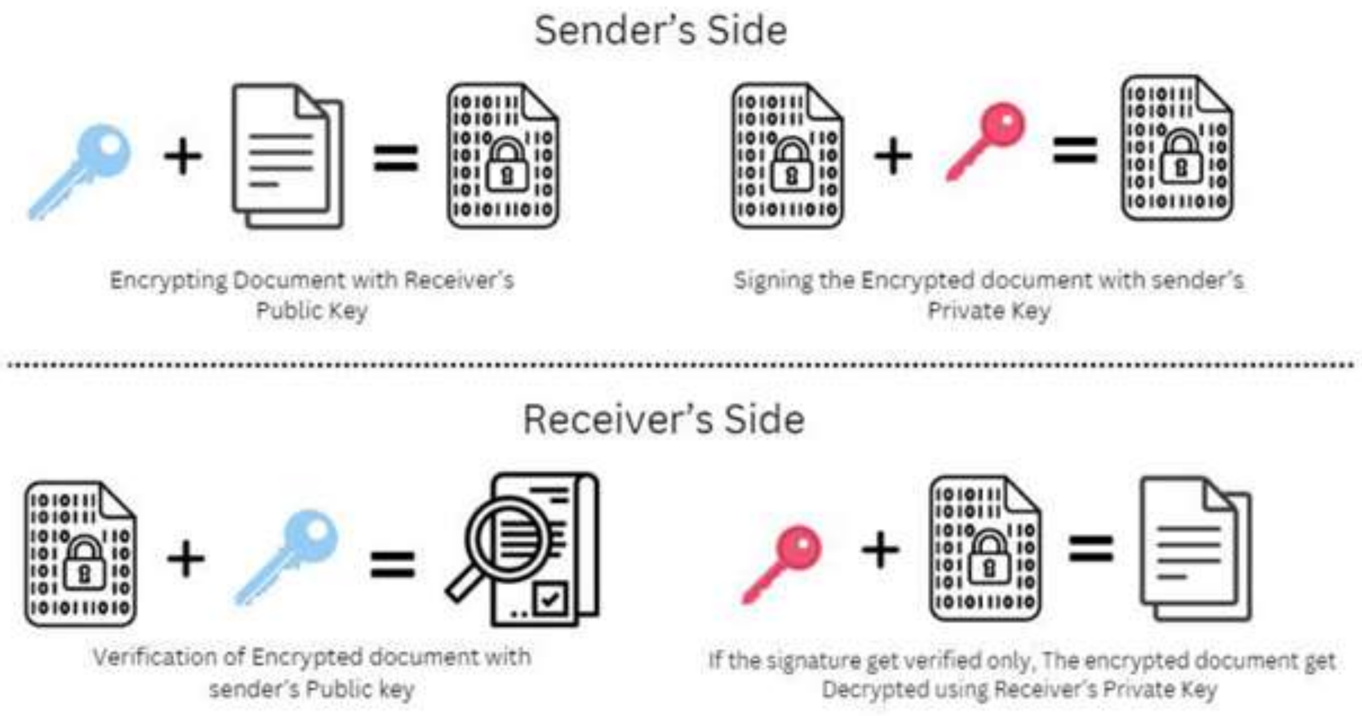
Result: "Secure and Unmodified Data Received"

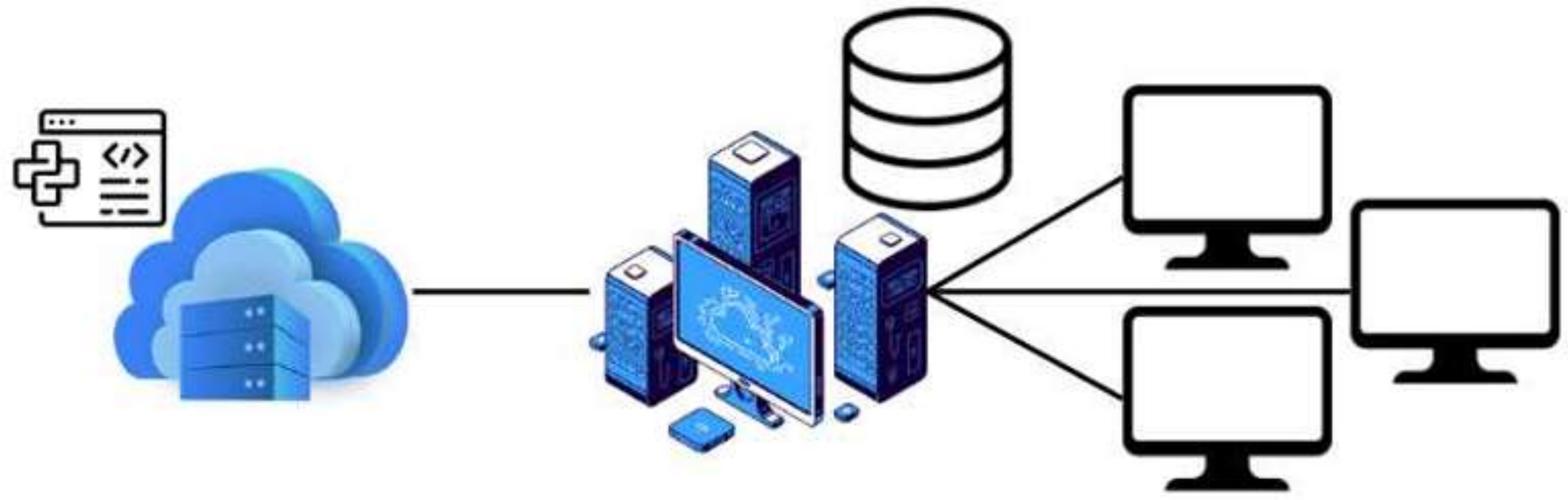
PROCESS 3

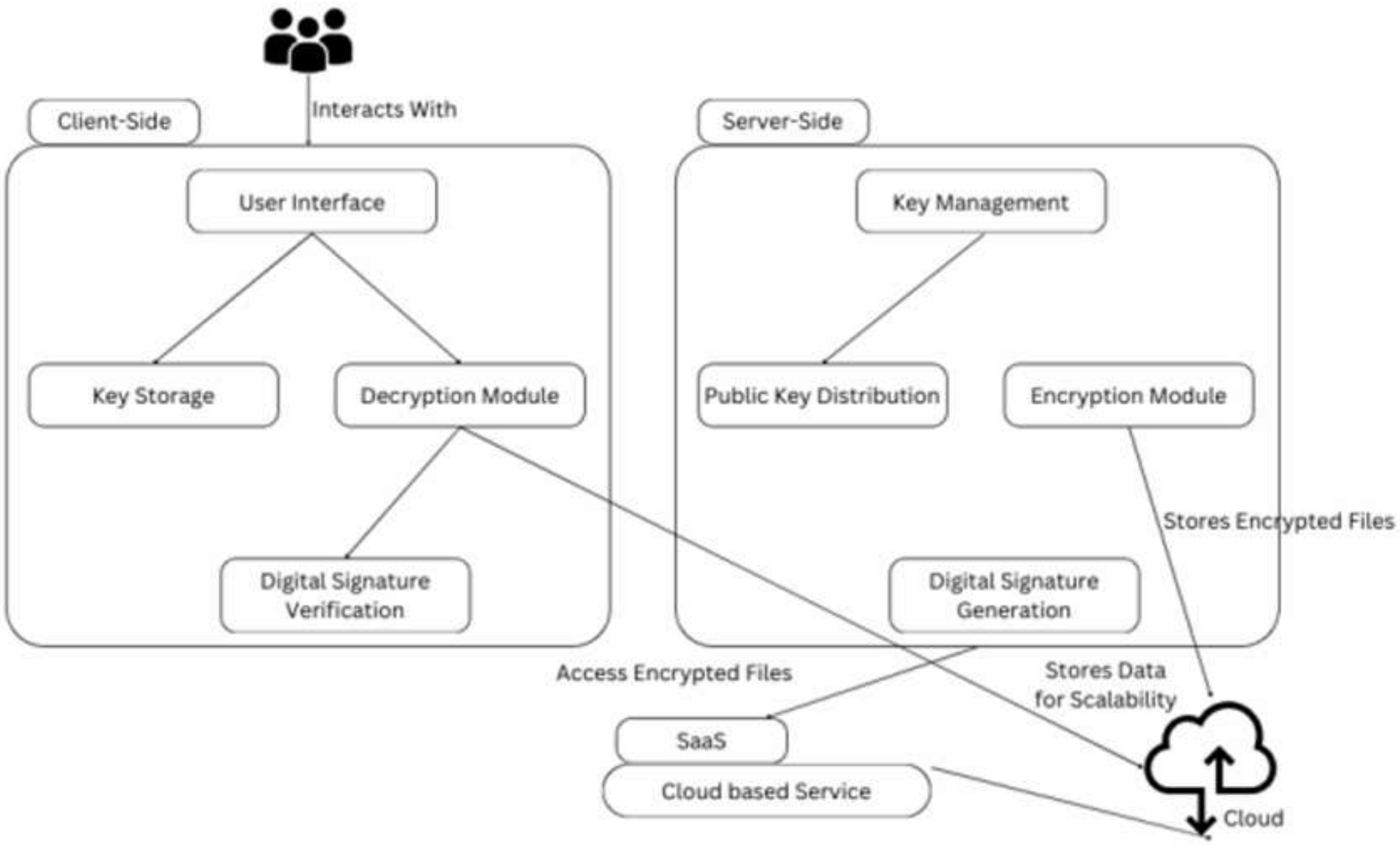

Technologies
Used
python
HTML
CSS
BASH
JS


Encrypt
Decrypt
Verify


Software
Software is
installed in
cloud
Providing our
software to
Companies via
Subscription
public key 1
public key 2
public key 3
public key 4
public key 5
Companies with
subscription can have
access to out portal, In
which they can able to
manage keys and Encrypt,
Decrypt and Verify
Documents


Sender's Side
Encrypting Document with Receiver's
Public Key
Signing the Encrypted document with sender's
Private Key
Receiver's Side
Verification of Encrypted document with
sender's Public key
If the signature get verified only, The encrypted document get
Decrypted using Receiver's Private Key





Architecture Diagram

Client-Side: Users interact with modules for key storage, decryption, and digital signature verification.

Server-Side: Manages key distribution, encryption, and digital signature generation.

Cloud: Stores encrypted files for scalability and accessibility via a SaaS model.

FEASIBILITY

01 Software requirement is minimalistic

02 Can implemented to any system based on requirements

03 Low cost implementation

04 Easy to maintain

05 Can be applied to old systems

06 Integrity Verification can be done

07 Secure Platfom

IMPACT AND BENEFITS

Objective n° 1

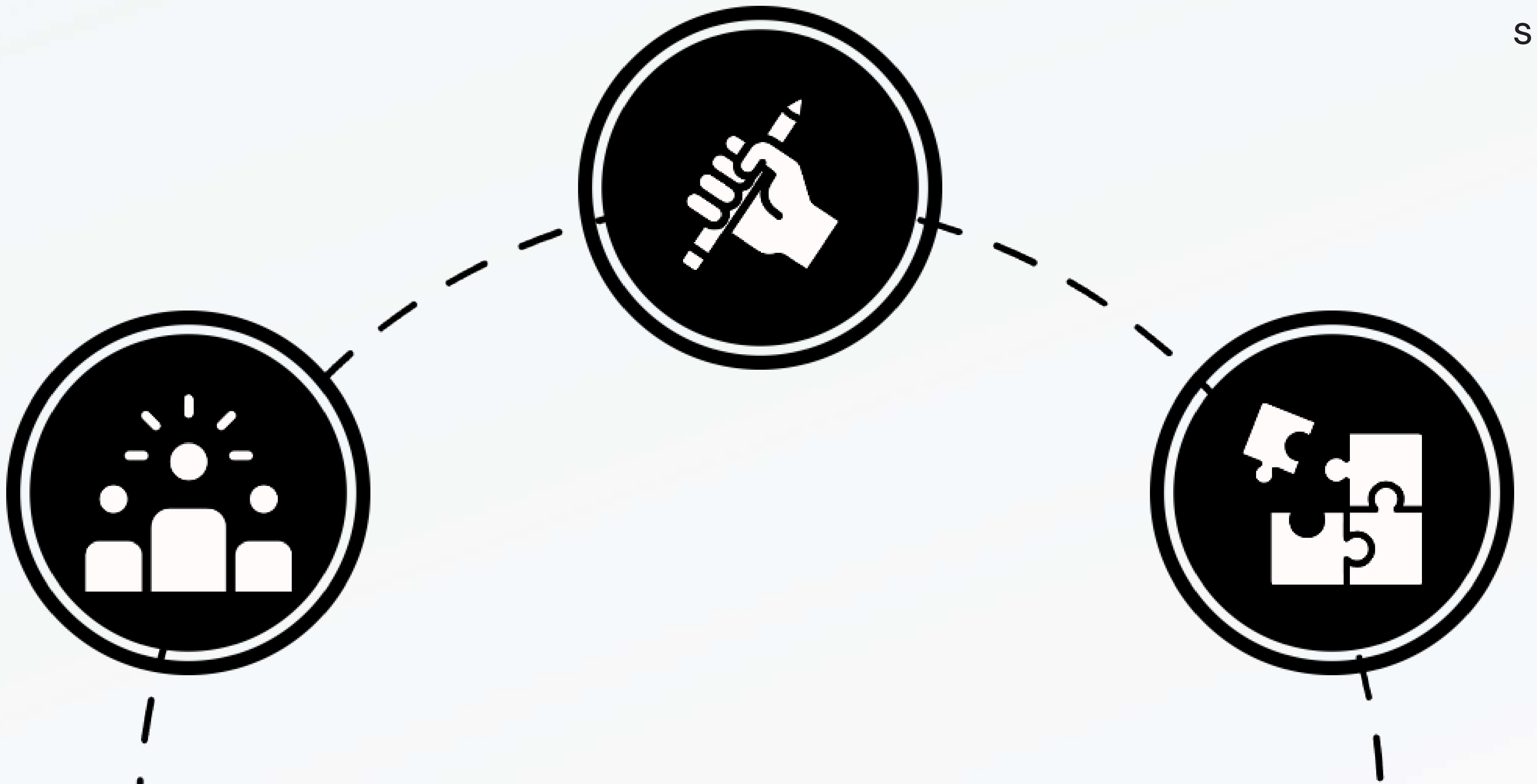
No complex mechanism involved. We provide a good user interface.

Objective n° 2

Can be able to create a most secured system of communication and make a stable encryption standard.

Objective n° 3

By implementing this system, we can able to potentially mitigate other cyber attacks such as MIMT and Phishing.



Testing

Key Management: Testing key generation, storage, and retrieval.
Encryption/Decryption Modules: Verifying correct encryption and decryption processes.
Digital Signature Generation/Verification: Ensuring signatures are correctly created and validated.
Public Key Distribution: Checking that keys are distributed securely.

Tests are written to cover various inputs, including expected and unexpected cases.

Tests run automatically, making it easier to detect bugs early during development.

Failures in unit tests indicate issues within specific units, helping developers quickly identify and fix problems.

Functionality Testing: Ensures the software performs its core functions correctly.
Performance Testing: Checks the software's responsiveness, stability, and scalability under load.
Usability Testing: Verifies that the software is user-friendly and intuitive.
Security Testing: Identifies vulnerabilities and ensures data protection.

Windows PowerShell

```
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
```

Class Diagram

User: Authenticates and sends data.
Server: Receives, stores, and verifies data.
DataFlow: Handles data encryption, decryption, and integrity checks.
EncryptionService: Provides encryption and decryption methods.
TrafficRouter: Routes traffic through TOR for anonymity.

The diagram shows how these components interact to ensure secure, encrypted communication.

Sequence Diagram

User initiates the process by routing traffic through the **TrafficRouter**.**TrafficRouter** sends encrypted data to the **EncryptionService**.**EncryptionService** encrypts the data and returns it to the **User**.**User** then sends the encrypted data to the **Server**.**Server** verifies the data's integrity through the **EncryptionService**.**EncryptionService** returns the integrity status to the **Server**.**Server** acknowledges receipt and integrity status to the **User**.

Use Case Diagram

User: Can authenticate, send data securely, and verify data integrity.
System Administrator: Monitors data integrity and oversees system operations.
System: Handles user authentication, encrypts data, routes traffic through TOR, transmits data, and verifies its integrity.

Activity Diagram

- **Start**: User initiates the process.
- **Authenticate**: User is authenticated.
- **Generate Keys**: Keys are generated for encryption.
- **Encrypt Data**: Data is encrypted.
- **Route Through TOR**: Encrypted data is routed through TOR.
- **Transmit Data**: Data is transmitted to the server.
- **Verify Integrity**: Data integrity is checked by the server.
- **Store Data**: Verified data is stored.

End: Process completes successfully, or data is rejected if integrity fails.

DFD Diagram Level 1

- **User** interacts with the **Authenticate User** process by providing login credentials.
- **Authenticate User** process communicates with the **User Database** to store and retrieve credentials.
- **Authenticate User** sends authenticated data to the **Encrypt Data** process.
- **Encrypt Data** process encrypts the data and passes it to the **Route Data** process.
- **Route Data** process routes the data and sends it to the **Transmit Data** process.
- **Transmit Data** process sends the data to the **Verify Data Integrity** process.
- **Verify Data Integrity** process checks the data and sends it to the **Store Data** process.
- **Store Data** process stores the data in the **Data Store**.

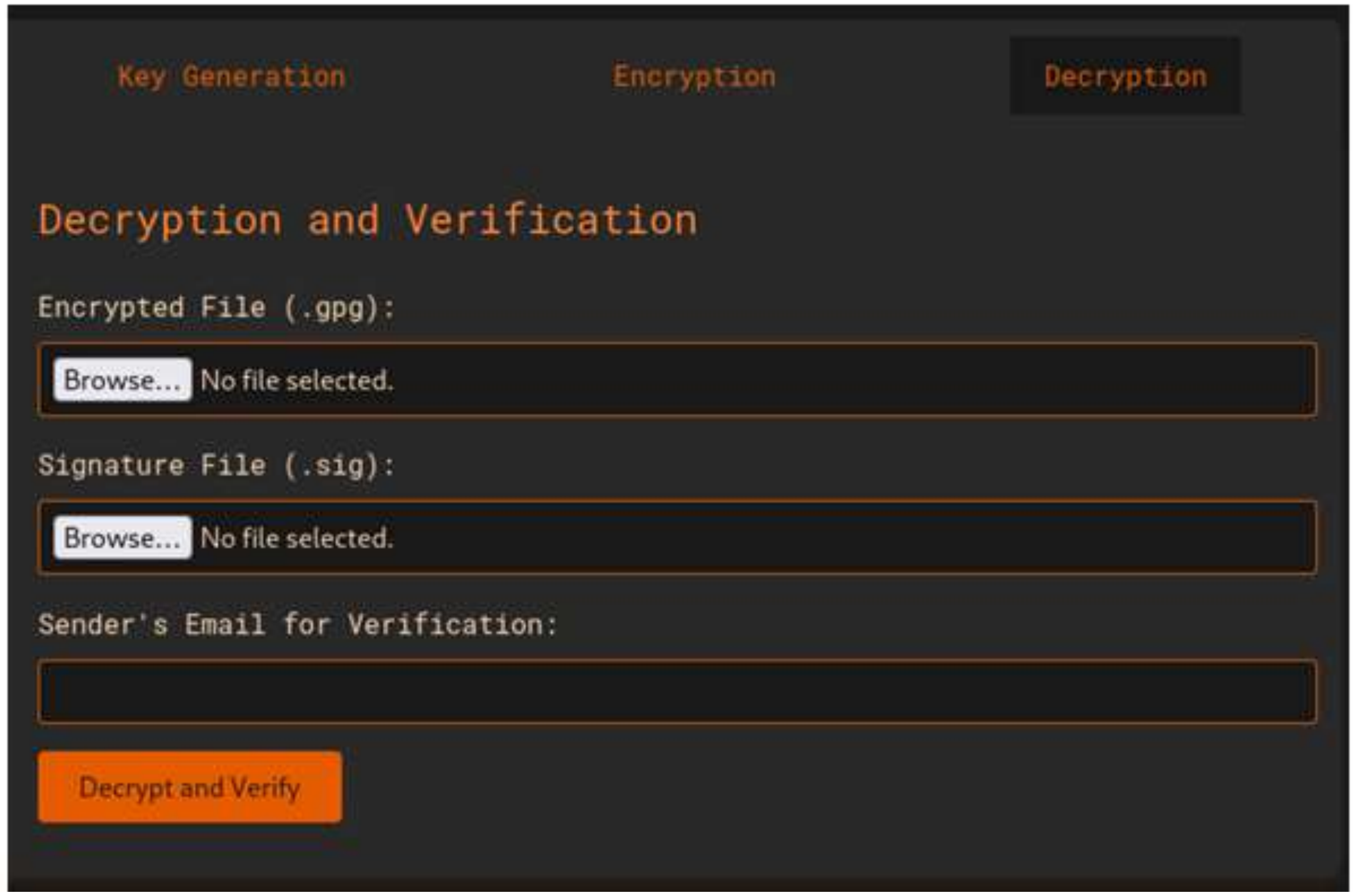
System Administrator monitors the **Verify Data Integrity** process.

Program Output

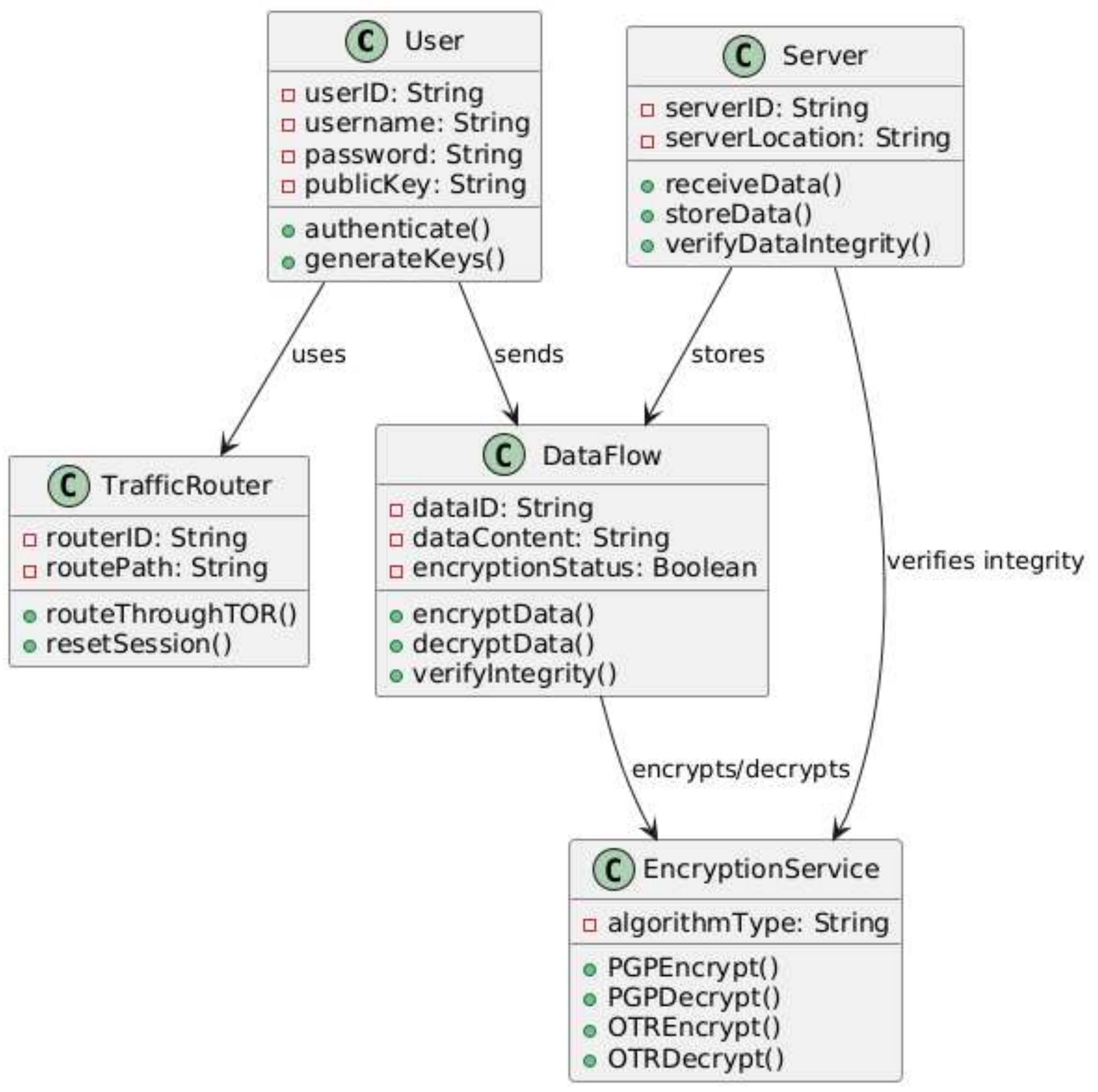
Key Generation

Encryption Module

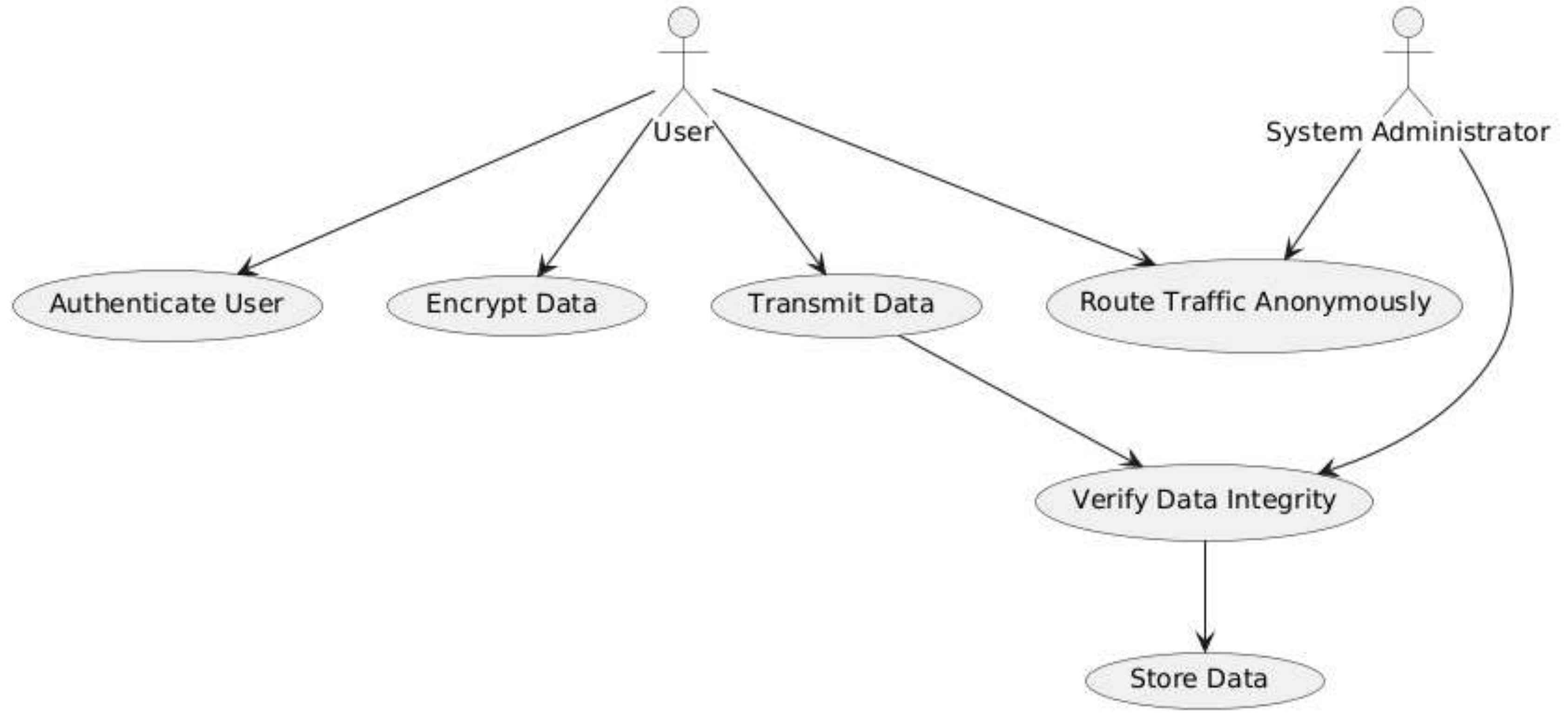
Decryption Module



Class Diagram

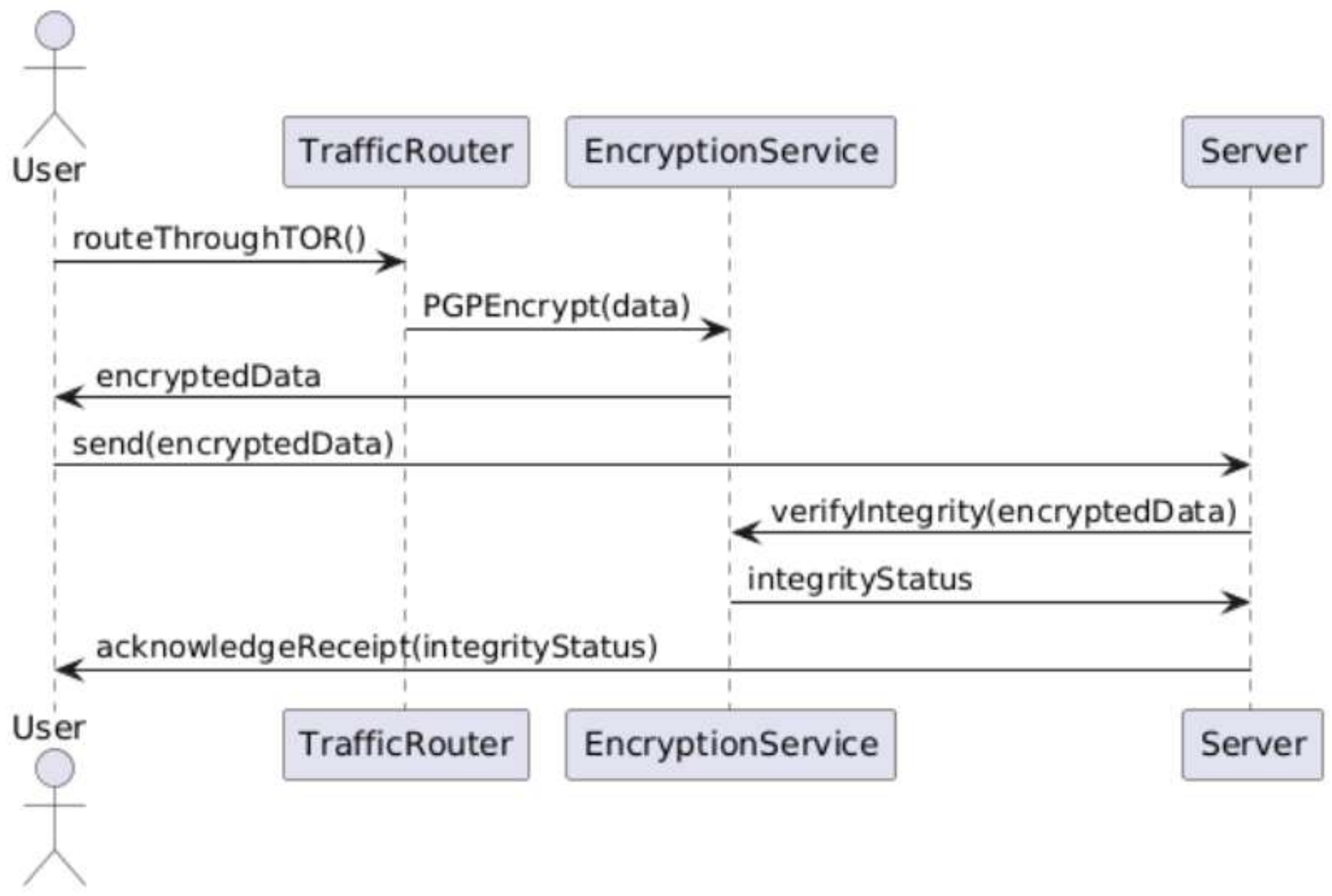


Use Case Diagram

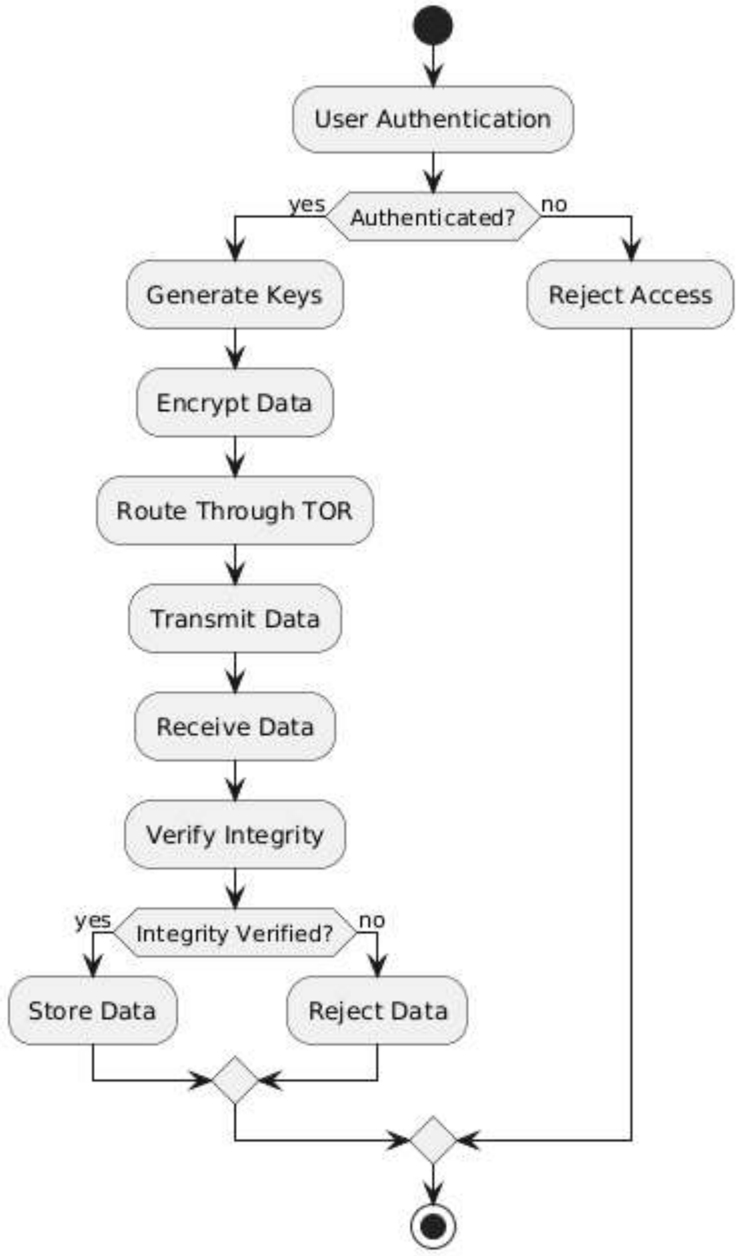


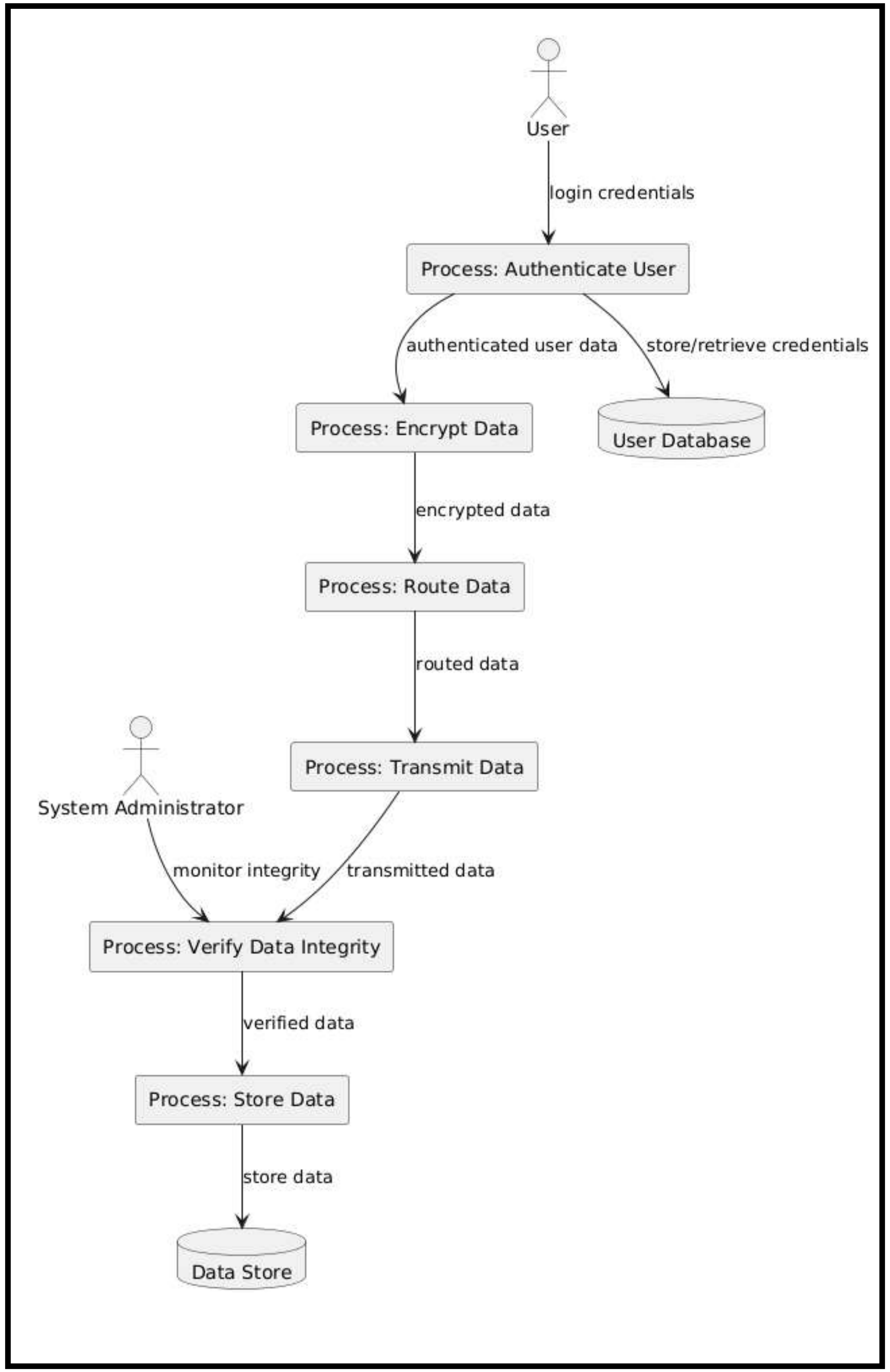
Class and Use Case Diagram

Sequence Diagram



Activity Diagram





DFD Diagram Level 1

Conclusion

Unit testing in this Flask application ensures that core features—encryption, decryption, and error handling—function as expected. Testing isolated components allows early bug detection, confirms correct functionality, and builds confidence for code refactoring. The tests cover normal strings, empty strings, and invalid requests, verifying robustness and handling of edge cases. Overall, unit testing improves the reliability, maintainability, and stability of the application.

References

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

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

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

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),

OUR TEAM

Guruprasath P

Artificial intelligence and data Science

III Year

Madhuvanthiy S

Artificial intelligence and data Science

III Year

Harshini MD

Artificial intelligence and data Science

III Year