Project Report

Secure Chat Application with AES and RSA Encryption

Developed by
Phaneendra Peravarapu

Table of Contents

Secure Chat Application with AES and RSA Encryption

---

1. Introduction

1.1 Project Overview

This project is a command-line based secure chat application developed in Python that facilitates encrypted messaging between multiple clients through a server. It leverages Advanced Encryption Standard (AES) for message encryption and Rivest-Shamir-Adleman (RSA) for secure key exchange, ensuring confidentiality and integrity of the data being transmitted.

1.2 Objective

The objective is to build a secure and scalable communication system using symmetric and asymmetric encryption methods to prevent unauthorized access and interception of messages.

---

2. Technology Stack

2.1 Programming Language

- Python 3.12

2.2 Libraries and Tools Used

- socket for network communication
- threading for handling multiple clients
- PyCryptodome for AES and RSA encryption
- os for file and key management

---

3. System Architecture

3.1 Components

- Server
- Multiple Clients
- AES and RSA Encryption Modules

3.2 Data Flow

1. Server waits for incoming connections.
2. Client connects and shares its encrypted AES key using RSA.
3. Server decrypts the AES key and uses it to decrypt future messages.

3.3 Encryption Workflow

- RSA is used once to securely share the AES key.
- AES is then used for all ongoing encrypted communication.

---

4. Modules Description

4.1 Server Module

- Accepts multiple client connections
- Handles RSA-based key exchange and AES decryption
- Displays decrypted client messages

4.2 Client Module

- Connects to the server
- Encrypts and sends messages using AES
- Shares AES key using RSA upon connection

4.3 GUI Client (Optional)

- Not implemented in this version

4.4 AES Encryption Module

- Handles symmetric encryption of chat messages

4.5 RSA Encryption Module

- Encrypts the AES key using server's public key
- Decrypts AES key at the server using private key

---

5. Working Mechanism

5.1 Key Exchange

- Client encrypts a random AES key using server's RSA public key.

5.2 Encrypted Communication

- Chat messages are encrypted using AES and sent to the server.

5.3 Multi-client Handling

- Server spawns a new thread for every incoming client.

---

6. Installation and Setup

6.1 Prerequisites

- Python 3.x
- pip install pycryptodome

6.2 Running the Server

```
python server.py
```

6.3 Running the Client

```
python client.py
```

---

7. Testing and Demonstration

7.1 Single Client Chat

- Messages encrypted using AES are properly decrypted at server.

7.2 Multiple Client Interaction

- Server handles multiple connections with individual threads.

---

8. Security Features

8.1 AES Encryption

- Ensures confidentiality of chat messages.

8.2 RSA Key Protection

- Prevents man-in-the-middle attacks during AES key exchange.

8.3 Limitations and Future Enhancements

- Add authentication
- Add message integrity checks
- Build GUI for better usability

---

9. Conclusion

This project successfully demonstrates secure, encrypted communication using hybrid encryption techniques. It showcases a foundational understanding of networking, encryption, and multi-threading in Python.

---

10. References

- PyCryptodome Documentation
- Python Socket Programming Guide
