Internship Project Report

Secure Chat Application with End-to-End Encryption

Abstract

This project implements a secure chat application in Python that supports **end-to-end encryption (E2EE)**. Messages are encrypted before leaving the sender and decrypted only by the intended receiver. The server stores ciphertext only and cannot read messages. We use a hybrid encryption scheme (**RSA + AES-GCM**) for confidentiality and **digital signatures (RSA-PSS)** for authenticity. Access to the server API is restricted by a secret token, ensuring private use. The final app enables two users to exchange encrypted, signed messages interactively on a command-line chat client.

Introduction

Secure communication is a fundamental requirement in today’s connected world. Traditional chat systems often leave messages exposed on the server. To address this, we built a **Flask-based chat server** and a **Python client** that enforces E2EE. The objective of this project was to apply cryptography in practice, build secure APIs, and demonstrate an interactive secure chat system suitable for internship learning and real-world scenarios.

Tools Used

- Python 3
- Flask (HTTP server)
- Requests (HTTP client)
- Cryptography library (RSA, AES-GCM, signatures)
- curl (API testing), ngrok (remote demo)
- Environment: Ubuntu VM with Python virtualenv

Methodology

1. Environment Setup: Created project, configured virtualenv, and installed dependencies.
2. Plaintext Chat Baseline: Implemented /send and /messages endpoints; tested with curl.
3. Key Generation: RSA-3072 keypairs generated for users; SHA-256 fingerprints displayed.
4. Public Key Distribution: Users register keys with /register_key and fetch with /get_key/.
5. Hybrid E2EE: AES-256-GCM encrypts messages; AES key encrypted with RSA-OAEP.
6. Digital Signatures: Messages signed with RSA-PSS and verified by receivers.
7. Private Server Security: Server bound to localhost and protected with API token.
8. Interactive CLI Chat: Built chat_cli.py enabling two-way encrypted chat between Alice and Bob.

Results

The secure chat app successfully runs on a local VM with two users. The server never stores plaintext messages, only ciphertext blobs. Clients decrypt and verify messages locally, ensuring confidentiality and authenticity. Unauthorized requests are blocked via the API token. The project demonstrates two-way end-to-end encrypted communication with message authenticity verification.

Conclusion

This project demonstrates how to design and implement a secure end-to-end encrypted chat system. It covers **confidentiality**, **integrity**, **authenticity**, and **access control**. It goes beyond theory to a working interactive app. Future scope includes persistent storage, forward secrecy using ephemeral Diffie-Hellman, message expiry, and a GUI-based client.

Recommended Screenshots

- Flask server running (localhost)
- curl baseline test (/send, /messages)
- RSA key generation with fingerprints
- Public key registration and retrieval
- Secure signed message decrypted
- Alice and Bob chatting interactively