

Secure Real-Time Encrypted Chat System

GROUP MEMBERS:

1.Fahad (054-72148)

2.Faizan (054-72198)

3.M.anas (054-72259)

# Overview

This project is a secure, real-time messaging system that allows users to send and receive encrypted messages using RSA, AES, or Fernet encryption. It features both a WebSocket-based live chat frontend and a REST API backend. All user registrations, messages, and keys are stored persistently to support long-term secure communication.

# Objectives

* • Build a secure communication system for real-time chat.
* • Support multiple encryption algorithms: RSA (asymmetric), AES, and Fernet (symmetric).
* • Provide an intuitive web interface for user interaction.
* • Enable REST API access for integration with external systems.
* • Persist user data and chat history across sessions.
* • Enforce that only registered users can communicate.

# System Architecture

## Components

* Backend Server: Flask + Flask-SocketIO
* Web GUI: HTML + JavaScript + Socket.IO
* REST API: Flask
* Encryption: Python cryptography & pycryptodome
* Data Storage: JSON files (messages.json, registered\_users.json)

# Features

## User Management

• Users must register before sending/receiving messages.  
• User identities are stored in registered\_users.json.

## End-to-End Encryption

• Supported algorithms: RSA, AES, Fernet.  
• Messages are encrypted before sending and decrypted only by recipients.

## Real-Time Communication

• Live chat with Flask-SocketIO.  
• System broadcasts real-time messages like “User has joined.”

## Inbox & History

• Each user's messages are stored in messages.json.  
• Users can fetch inbox securely.

## Persistent Storage

• All data remains available even after server restarts.

# User Interface

A web GUI (chat.html) is provided to:  
• Register and join  
• Send encrypted messages  
• Choose algorithm  
• Fetch inbox  
• View real-time chat  
• Display connection status

# REST API Endpoints

|  |  |  |
| --- | --- | --- |
| Endpoint | Method | Description |
| /register | POST | Register a new user |
| /encrypt | POST | Encrypt a message for a recipient |
| /decrypt | POST | Decrypt an encrypted message |
| /inbox/<username> | GET | Fetch decrypted messages for user |

# Testing Tools

• Manual testing via GUI  
• API testing with Postman  
• Debugging via browser console and logs

# File Structure

project/  
├── app.py # WebSocket chat server  
├── kms\_api.py # REST API backend  
├── kms\_core.py # Key management and crypto logic  
├── chat.html # Web GUI  
├── registered\_users.json # Persistent user data  
├── messages.json # Encrypted messages

# Future Improvements

* • Password-based login and authentication
* • JWT or token authentication for APIs
* • Admin dashboard for moderation
* • Message deletion and expiration
* • Search/filter by sender or date

# Conclusion

This project demonstrates a complete, secure, real-time messaging platform with strong cryptographic standards and extensible architecture. It ensures privacy, integrity, and authentication while being accessible through both API and web interface.