Project Report: Secure Chat Application with End-to-End Encryption (E2EE)

---

1. Introduction

In today's digital era, online communication is essential but also vulnerable to privacy breaches, unauthorized access, and data leaks. Traditional chat systems often store messages in plain text on servers, leaving them exposed to potential interception.
To address these concerns, this project — **Secure Chat Application with End-to-End Encryption (E2EE)** — was developed for **Elevate Labs**, ensuring that only the sender and recipient can read the exchanged messages.

This system uses **public-key cryptography (RSA)** and **symmetric encryption (AES)** to achieve a secure, real-time chat platform where even the server cannot access the message content.

---

2. Objectives

- Implement **real-time communication** with complete message confidentiality.
- Use **End-to-End Encryption (E2EE)** to ensure only intended recipients can decrypt messages.
- Protect sensitive user data from unauthorized access or interception.
- Enable **secure key exchange** using RSA encryption.
- Provide an optional feature to **store encrypted chat logs** on the server.

---

3. Technology Stack

- **Programming Language:** Python, JavaScript
- **Backend Framework:** Flask with Flask-SocketIO
- **Frontend:** HTML, CSS, JavaScript
- **Cryptography:** RSA and AES (from Python's cryptography and CryptoJS libraries)
- **Communication Protocol:** WebSockets (via Flask-SocketIO)
- **Database (optional):** SQLite / MySQL for encrypted chat logs
- **Development Tool:** Visual Studio Code

4. System Architecture

1. **Key Generation:**
   Each user generates an RSA key pair — a **public key** (shared) and a **private key** (kept secret).
2. **Key Exchange:**
   Public keys are exchanged between users securely.
3. **Message Encryption:**
   Each message is encrypted using **AES**, and the AES key is then encrypted using the recipient's **RSA public key**.
4. **Real-Time Communication:**
   Flask-SocketIO manages the exchange of encrypted messages between users instantly.
5. **Decryption:**
   On the client side, the recipient's browser decrypts the AES key using their RSA **private key**, then decrypts the message content.
6. **Encrypted Storage (optional):**
   Messages can be stored on the server in encrypted form for future retrieval.

---

5. Features

- **End-to-End Encryption (E2EE):** Messages are encrypted before leaving the sender and decrypted only by the recipient.
- **RSA & AES Hybrid Encryption:** Combines the strength of RSA (for key exchange) and AES (for fast symmetric encryption).
- **Real-Time Messaging:** Built using Flask-SocketIO for instant communication.
- **Private Key Management:** Users load their private keys locally — never shared with the server.
- **Secure Chat Logs:** Optional encrypted storage for auditing and record-keeping.
- **User-Friendly Interface:** Simple and intuitive chat interface with live updates.

---

6. Challenges & Solutions

Challenge	Solution
Implementing secure real-time communication	Integrated Flask-SocketIO with proper event handling for message exchange.
Ensuring secure key exchange between users	Used RSA encryption for transmitting AES keys safely.
Preventing message exposure on the server	Decryption happens entirely on the client-side using the user’s private key.
Managing user keys securely	Implemented local key loading to avoid uploading private keys to the server.
Synchronizing users in real time	Employed WebSocket-based communication for bidirectional real-time updates.

7. Results & Benefits

- **High Security:** Achieved complete end-to-end encryption; server cannot read message content.
- **Improved Privacy:** Each user controls their private key, ensuring personal data remains safe.
- **Real-Time Efficiency:** Instant encrypted communication using Flask-SocketIO.
- **Scalable Architecture:** Can be expanded to group chats and enterprise-level messaging.
- **User Trust:** Demonstrates strong data protection and cryptographic best practices.

8. Future Enhancements

- **Group Chat Encryption:** Extend E2EE to multi-user group chats.
- **Two-Factor Authentication:** Add user verification using OTP or biometric login.
- **Encrypted File Sharing:** Enable secure transmission of files alongside messages.
- **Mobile App Integration:** Develop an Android/iOS version using the same encryption model.
- **Blockchain Logging:** Implement blockchain-based message integrity verification.

9. Conclusion

The **Secure Chat Application with End-to-End Encryption** ensures privacy, security, and trust in digital communication. By combining RSA and AES encryption with real-time messaging, this system eliminates the risks associated with traditional chat systems.
This project demonstrates the practical application of cryptographic principles in enhancing communication security and can serve as a foundation for future secure communication platforms.

Prepared By: Bhagyashree Satpathy
Date: 23rd October 2025
Organization: Elevate Labs
