End-to-End Encrypted Messaging App

Project Report by Aashutosh Rana

Date: October 2025

Project Overview

The End-to-End Encrypted Messaging App is a secure communication platform designed to ensure that private conversations remain confidential. It provides real-time chatting functionality similar to modern messaging apps like WhatsApp but incorporates full end-to-end encryption (E2EE) to protect user data. The encryption and decryption of messages occur entirely on the client side, ensuring that no third party, including the server, can read user messages.

Objective

The primary objective of this project is to create a messaging system that maintains user privacy by using public-key cryptography for key exchange and AES encryption for message content. The app ensures secure message transmission, storage, and decryption — protecting data integrity and confidentiality.

Technologies Used

• Frontend: React (for responsive and animated UI) • Backend: Python (Flask + Socket.IO) • Encryption: RSA (for key exchange) and AES (for message encryption) • Libraries: cryptography, socket.io-client, framer-motion • Database: In-memory storage or lightweight file-based storage for development • Styling: Tailwind CSS and Framer Motion for animations

Working Process

1. **User Authentication and Key Generation:** Each user generates their own RSA public-private key pair locally using the WebCrypto API. 2. **Public Key Sharing:** Only the public key is sent to the server, which allows other users to fetch it for message encryption. 3. **Message Encryption:** When a user sends a message, a random AES key is generated to encrypt the message. This AES key is then encrypted using the recipient's public RSA key. 4. **Message Transmission:** The encrypted AES key and message ciphertext are transmitted to the server using Socket.IO in real time. 5. **Decryption on Client:** The recipient decrypts the AES key using their private RSA key, then uses it to decrypt the original message content. 6. **UI Updates:** Messages are animated using Framer Motion for smooth transitions and visual effects.

Key Features

• Real-time chat using Socket.IO • RSA and AES-based encryption ensuring true end-to-end security • Clean and modern UI built with React • Local key generation and storage • Insecure fallback mode for environments without WebCrypto support • Contact list, chat bubbles, and smooth animations for a modern messaging experience

Security Aspects

The security model ensures that private messages remain unreadable to unauthorized entities. Encryption keys are generated and stored locally in the user's browser. The server never has access to private keys, making it impossible for anyone, including administrators, to decrypt messages. Additionally, the app validates the integrity of cryptographic keys before each encryption and decryption step to prevent malformed or spoofed data from causing errors.

Conclusion

This project demonstrates the design and implementation of a secure, real-time chat system using end-to-end encryption principles. By combining RSA and AES cryptography with a dynamic React frontend and Python backend, the app ensures privacy, reliability, and modern user experience. It showcases the importance of cryptography in safeguarding digital communication and can be further extended to include voice/video calls and multi-device synchronization in the future.