SecureLink -
An Encrypted chat app using Sockets

Submitted By:
Syed Uzair Hussain - 22K-4212
Syed Samroze Ali - 22K-4187

Department of Computer Science
National University of Computer & Emerging Sciences

Motivation:

The motivation behind SecureLink stems from the growing need for secure communication in an era where data privacy is increasingly threatened. As digital communications become more prevalent, ensuring that sensitive information remains confidential is paramount. This project aims to demonstrate how encryption technologies can be effectively implemented to create a secure chat environment that protects user data while maintaining a seamless, real-time communication experience.

Overview:

Significance of the Project:

SecureLink represents a practical implementation of modern encryption techniques in real-time communication. The project has significant educational value as it demonstrates the application of cryptographic principles, network programming, and secure software development practices. In today's digital landscape where data breaches are common, developing applications with security at their core is essential. This project provides hands-on experience with implementing a hybrid encryption system (RSA/AES) and understanding the challenges of secure key exchange in networked applications.

Description of the Project:

SecureLink is a secure end-to-end encrypted chat application built using Flask and Socket.IO that enables real-time secure communication between multiple users. The project addresses the problem of insecure messaging by implementing strong encryption at every layer of communication.

The scope of this work includes:

- Developing a web-based chat client with a modern interface
- Creating a secure backend infrastructure with proper authentication
- Implementing end-to-end encryption using industry-standard cryptographic algorithms
- Ensuring message integrity and confidentiality
- Providing protection against common network attacks
- The system uses a hybrid encryption approach where RSA-2048 is used for secure key exchange and AES-256 in CBC mode handles the actual message encryption, ensuring both security and performance.

Background of the Project:

This project builds upon established cryptographic principles and network programming paradigms. The technical foundation draws from several key areas:

1. WebSocket technology, as described in RFC 6455, for real-time bidirectional communication
2. Flask web framework for building the application server

3. Modern cryptographic algorithms:
    a. RSA asymmetric encryption, based on the work of Rivest, Shamir, and Adleman
    b. AES (Advanced Encryption Standard) in CBC (Cipher Block Chaining) mode
4. Socket programming concepts for network communication

Our implementation references the following resources:

- Flask-SocketIO documentation (https://flask-socketio.readthedocs.io/)
- PyCryptodome library documentation (https://pycryptodome.readthedocs.io/)

Project Category:

This is a Product-based project that demonstrates practical application of cybersecurity concepts in a real-world context.

Features / Scope / Modules:

1. End-to-End Encryption:
   The application employs AES-256 encryption in CBC mode with random initialization vectors for each message. This ensures that even if communication is intercepted, the content remains unintelligible without the proper decryption keys.
2. Secure Key Exchange:
   RSA-2048 asymmetric encryption is used during the initial connection phase to securely exchange AES keys between clients and the server, establishing a secure channel for all subsequent communications.
3. Real-time Communication:
   The application utilizes WebSockets via Socket.IO to provide instant message delivery with minimal latency, creating a responsive user experience while maintaining security.
4. Rate Limiting and Connection Controls:
   The system implements IP-based rate limiting and connection controls to prevent brute force attacks and denial of service attempts, enhancing the application's resilience against common network attacks.
5. Multi-layered Architecture:
   The application employs a two-tier socket architecture with secure communication channels between:
    a. Web browsers and the Flask server using Socket.IO
    b. Flask server and the chat server using TCP sockets
6. Graceful Error Handling:
   All cryptographic and networking operations include robust error handling to prevent information leakage and ensure system stability even under adverse conditions.
7. Logging System:
   A comprehensive logging system records all significant events while carefully avoiding the logging of sensitive information, aiding in troubleshooting and security auditing.

Project Planning:

Timeline:

Week 1-2: Research on encryption methods and system architecture
Week 3-4: Development of chat server core functionality
Week 5-6: Implementation of encryption layer and key exchange
Week 7-8: Development of Flask server and web interface
Week 9-10: Integration and system testing
Week 11-12: Security testing, bug fixes, and optimization

Responsibility Distribution:

1. Syed Uzair Hussain (22K-4212)
    a. Flask application development (app_flask.py)
    b. Web interface (HTML/CSS/JavaScript)
    c. Encryption implementation
    d. System architecture design
2. Syed Samroze Ali (22K-4187)
    a. Backend server development (server_v2.py)
    b. Client application (client_v2.py)
    c. Encryption implementation
    d. System architecture design

Project Feasibility:

Technical Feasibility

The project is technically feasible as it utilizes well-established technologies and libraries that are mature and well-documented:

- Python's socket library for network communication
- PyCryptodome for encryption operations
- Flask and Flask-SocketIO for web server functionality
- Standard browser technologies (HTML5, CSS3, JavaScript)
- Technical risks include:
- Ensuring proper implementation of cryptographic algorithms
- Managing potential WebSocket connectivity issues across different network environments
- Handling concurrent connections efficiently

Economic Feasibility

The project is economically feasible as it relies entirely on open-source technologies.
Development costs are primarily time-based, with no required purchases of software licenses or specialized hardware.
Operational costs would be minimal, limited to standard web hosting expenses if deployed beyond a local environment.

Schedule Feasibility
The 12-week development timeline provides adequate time for all phases of development, testing, and refinement. The modular approach allows for parallel development of client and server components, making efficient use of the team's resources.

Hardware and Software Requirements

Software Requirements

- Python 3.6 or higher
- Flask 2.3.3
- Flask-SocketIO 5.3.6
- PyCryptodome 3.22.0
- RSA 4.9.1
- Modern web browser with WebSocket support
- Development environment (VS Code, PyCharm, or similar)

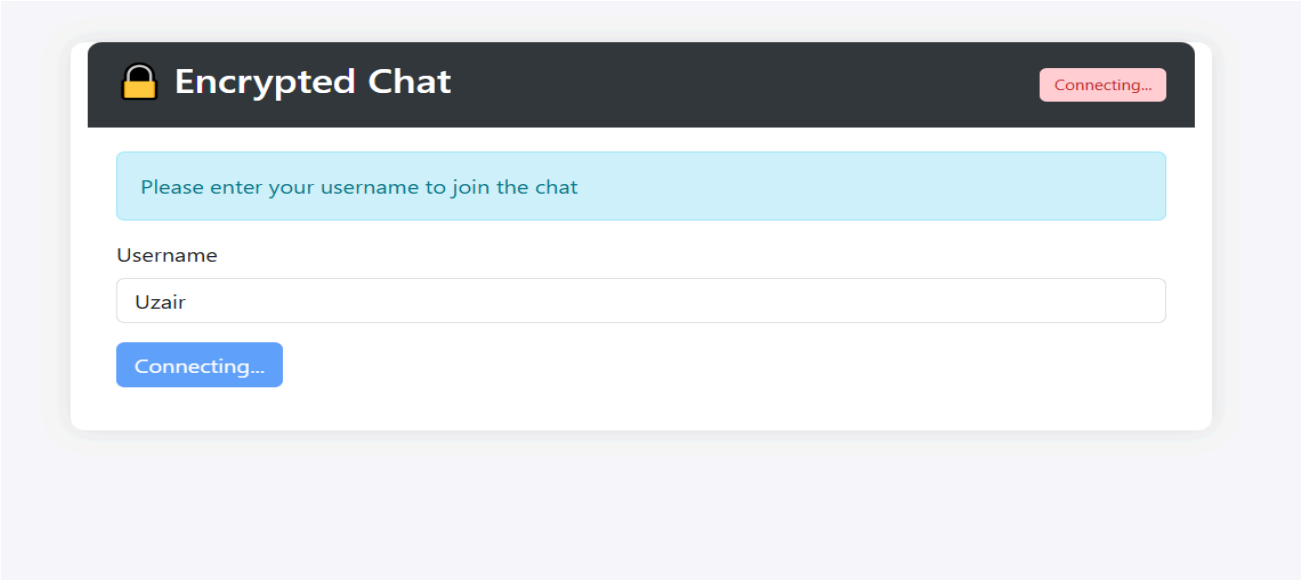
Hardware Requirements

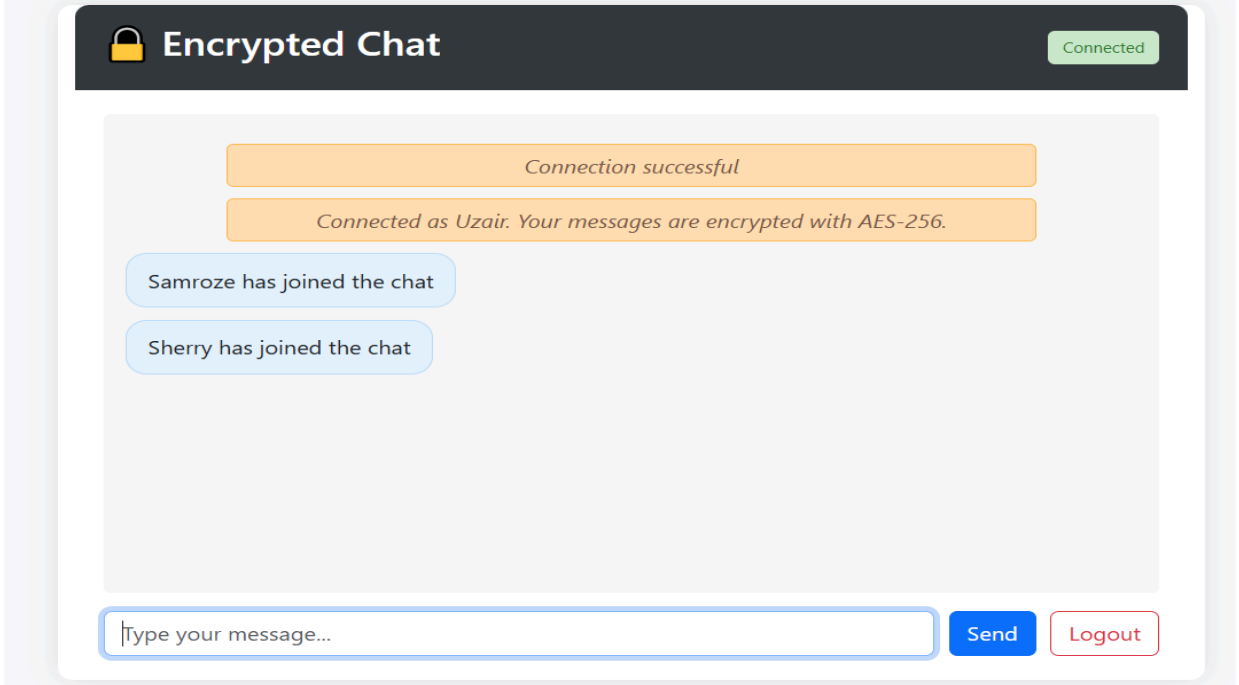
Any computer capable of running Python 3.6+
Minimum 4GB RAM for development environment
Network connectivity between server and clients
For deployment: Server with stable internet connection

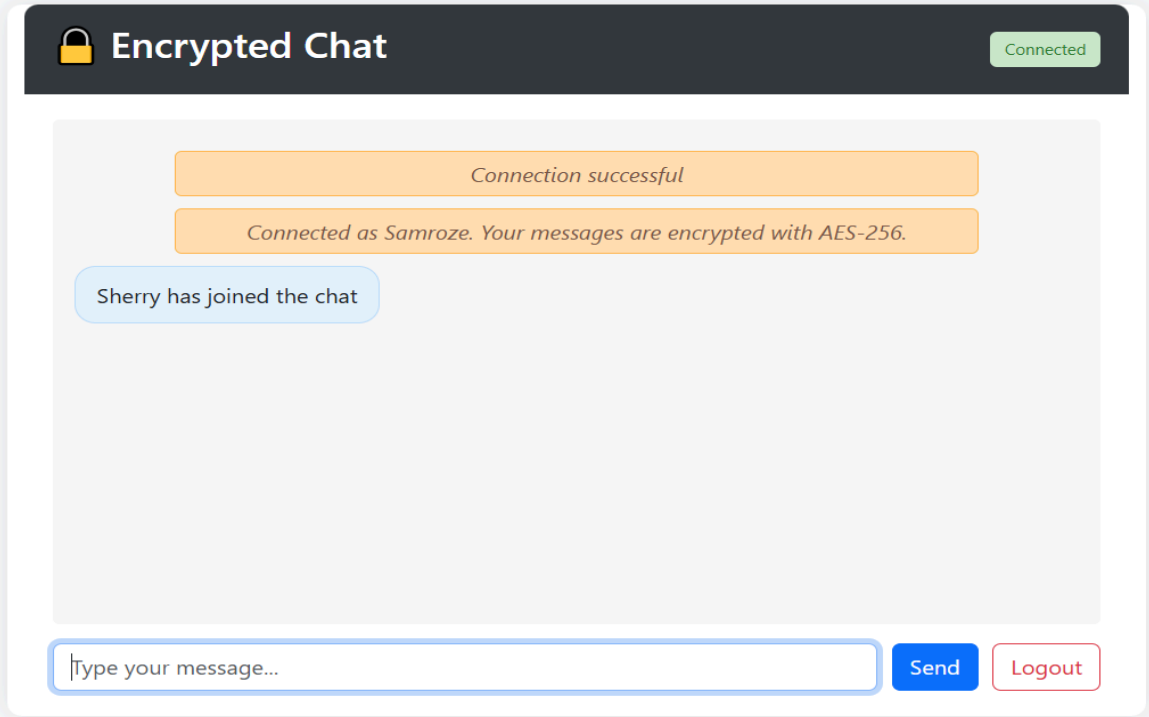
Diagrammatic Representation:

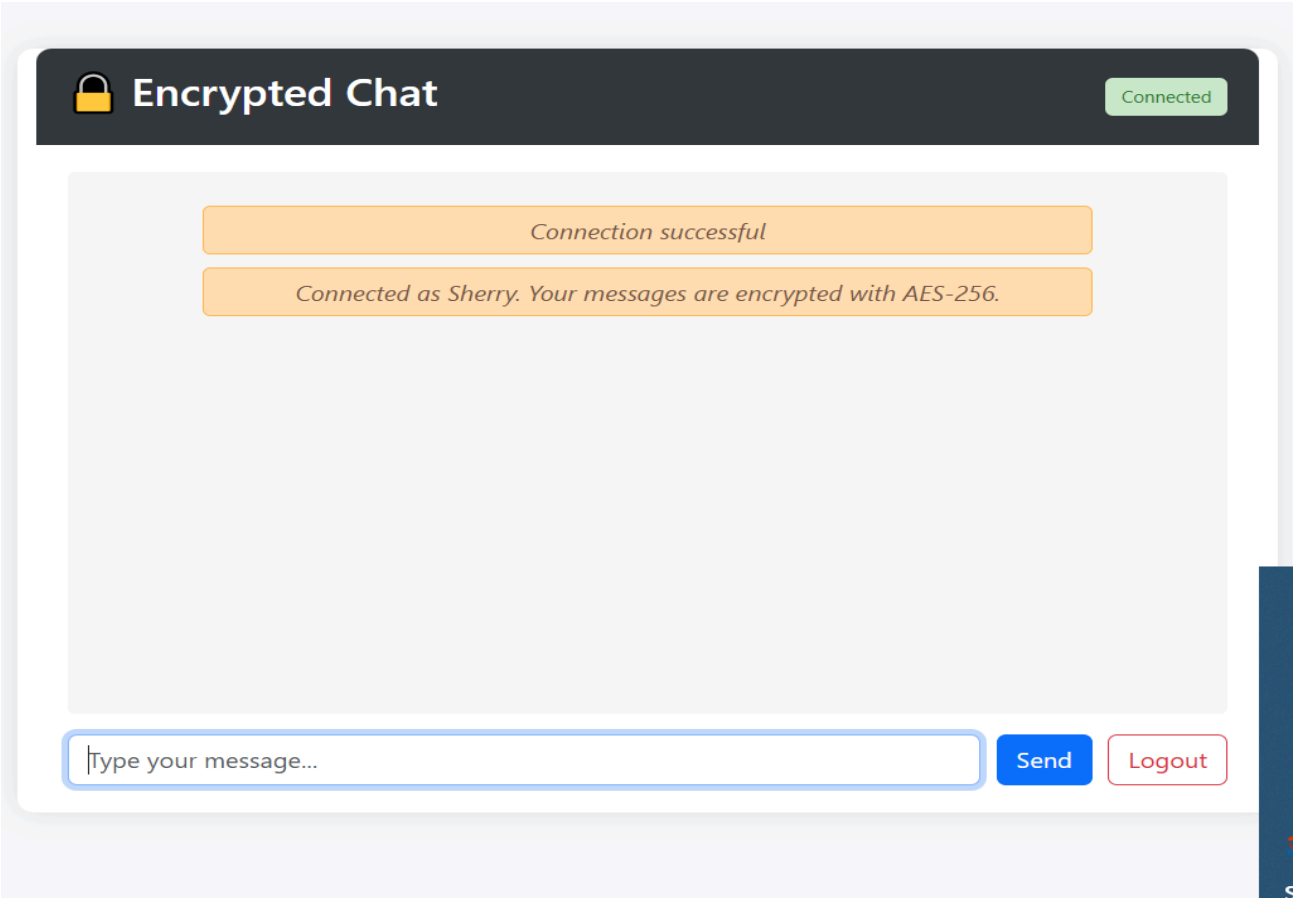
Figure 1

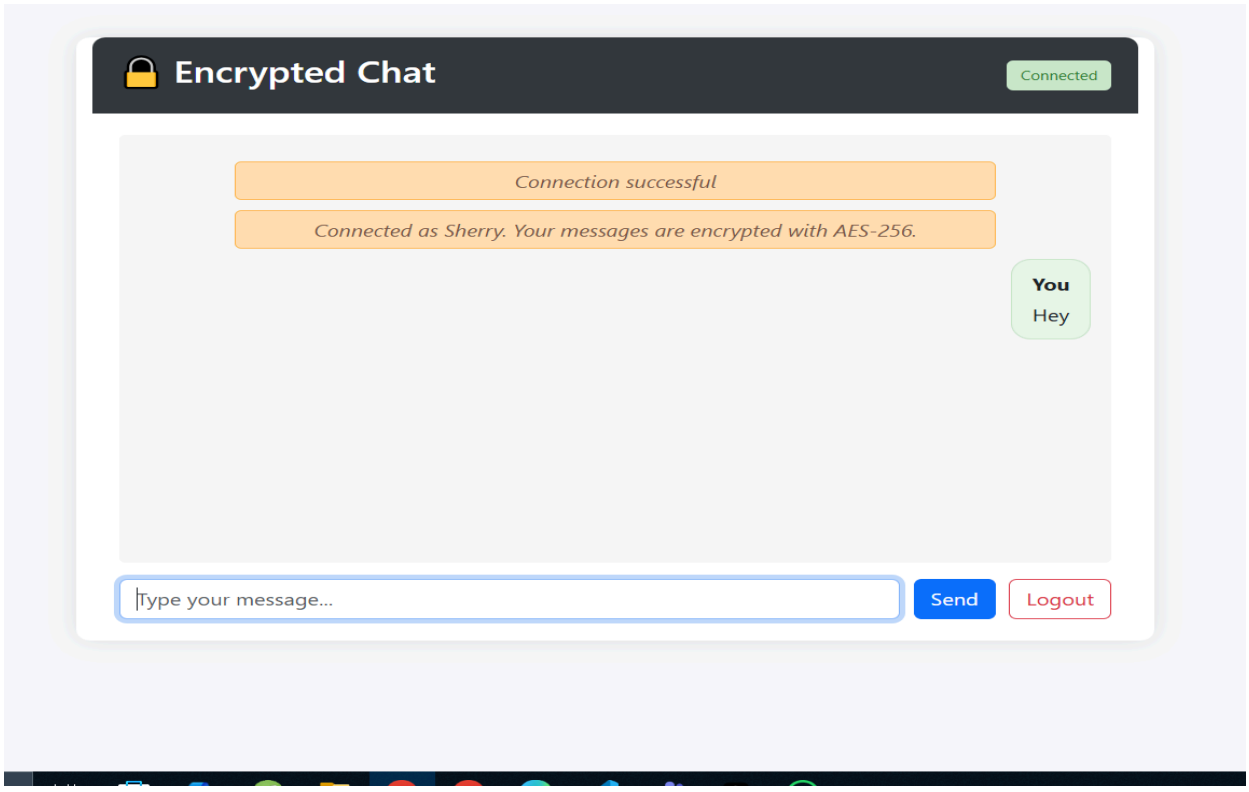
Working Demo:

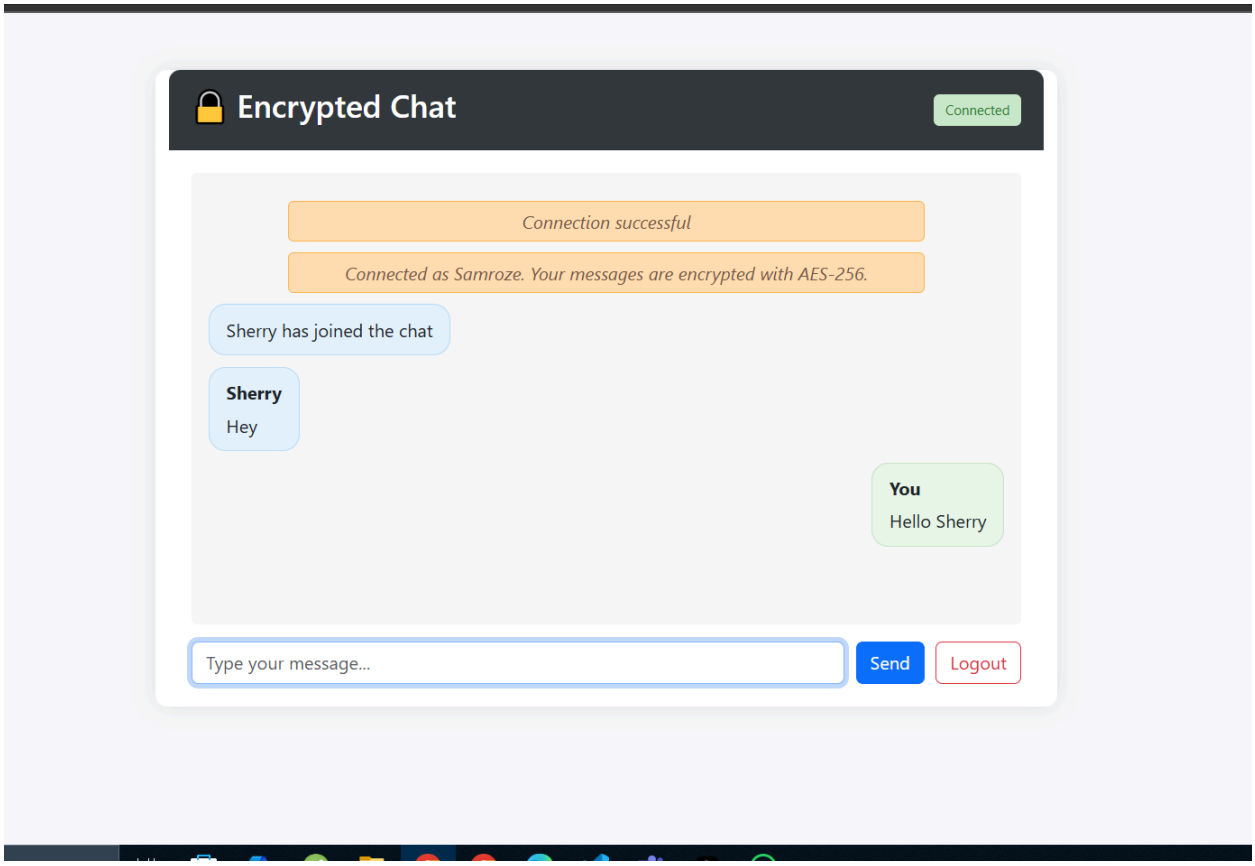


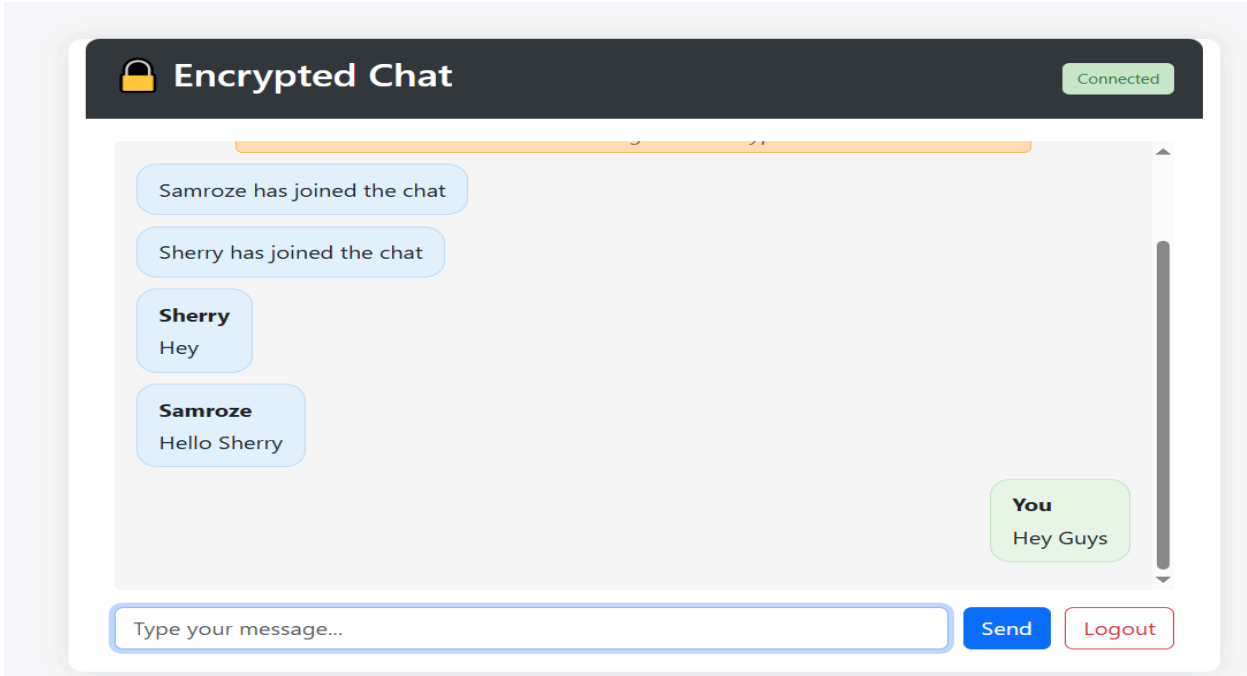

Encrypted Chat
Connected
Connection successful
Connected as Uzair. Your messages are encrypted with AES-256.
Samroze has joined the chat
Sherry has joined the chat
Type your message...
Send
Logout

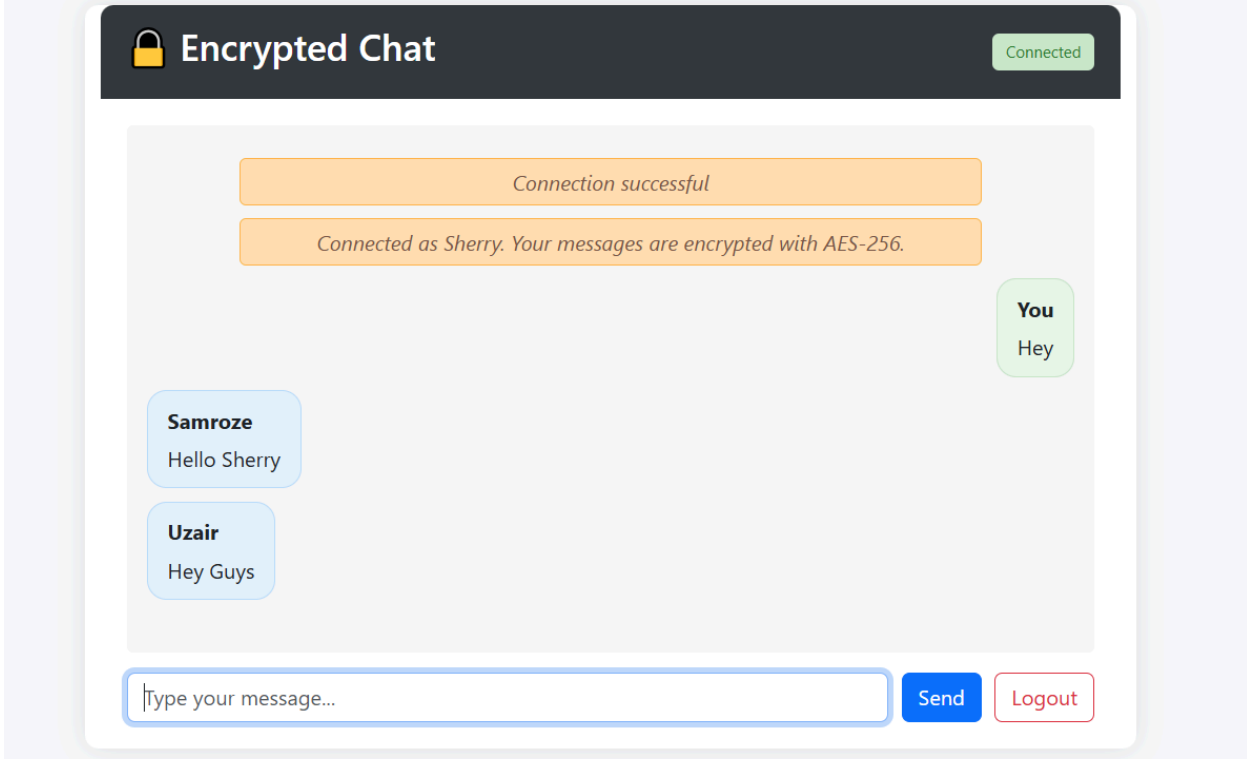

Encrypted Chat
Connected
Connection successful
Connected as Samroze. Your messages are encrypted with AES-256.
Sherry has joined the chat
Type your message...
Send
Logout

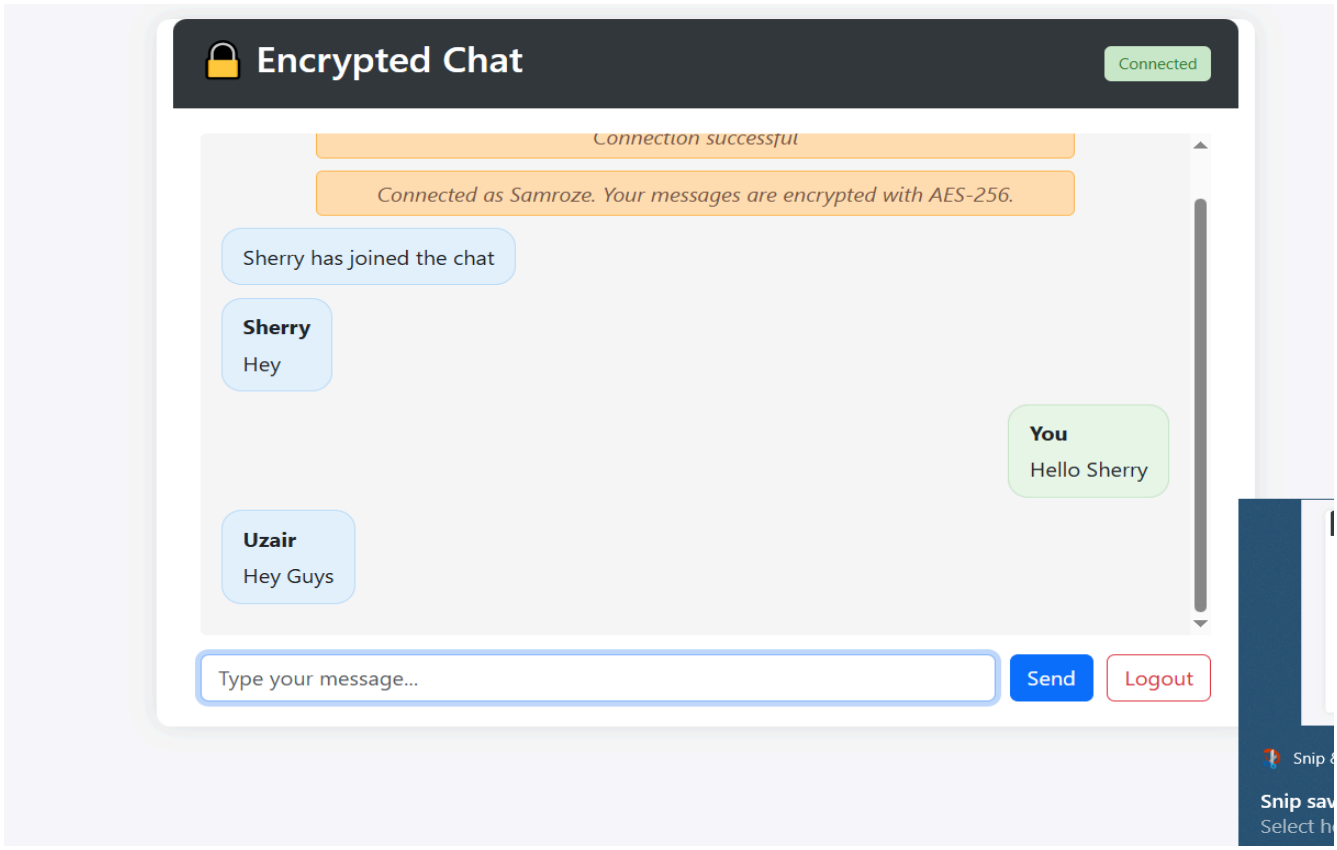

Encrypted Chat
Connected
Connection successful
Connected as Sherry. Your messages are encrypted with AES-256.
Type your message...
Send
Logout

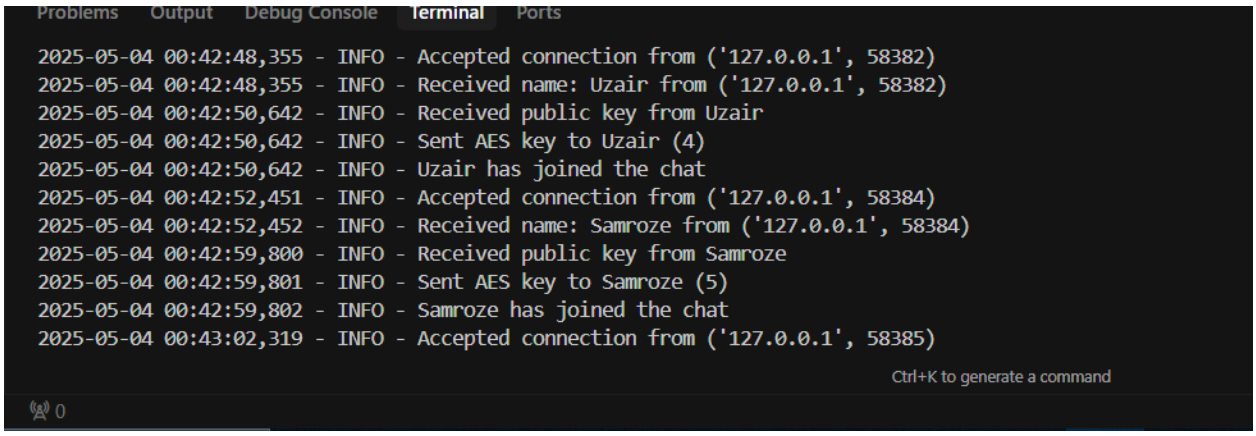

Encrypted Chat
Connected
Connection successful
Connected as Sherry. Your messages are encrypted with AES-256.
You
Hey
Type your message...
Send
Logout


Encrypted Chat
Connected
Connection successful
Connected as Samroze. Your messages are encrypted with AES-256.
Sherry has joined the chat
Sherry
Hey
You
Hello Sherry
Type your message...
Send
Logout


Encrypted Chat
Connected
Samroze has joined the chat
Sherry has joined the chat
Sherry
Hey
Samroze
Hello Sherry
You
Hey Guys
Type your message...
Send
Logout


Encrypted Chat
Connected
Connection successful
Connected as Sherry. Your messages are encrypted with AES-256.
You
Hey
Samroze
Hello Sherry
Uzair
Hey Guys
Type your message...
Send
Logout





References

[1] Flask-SocketIO, "Documentation", https://flask-socketio.readthedocs.io/
[2] PyCryptodome, "API Documentation", https://pycryptodome.readthedocs.io/

==========