VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

B.Tech. Winter Semester 2024-25
School Of Computer Science and Engineering (SCOPE)

Digital Assignment - I
Information Security

Apurva Mishra: 22BCE2791
Date: 23 February 2024

Contents

1 P2P Chat App 2
1.1 Stack 2
1.2 Data Flow Diagram 2
1.3 Module Contributions 2
1.4 Algorithm Description 2
1.5 Output 4
1.6 Information Security 4

1 P2P Chat App

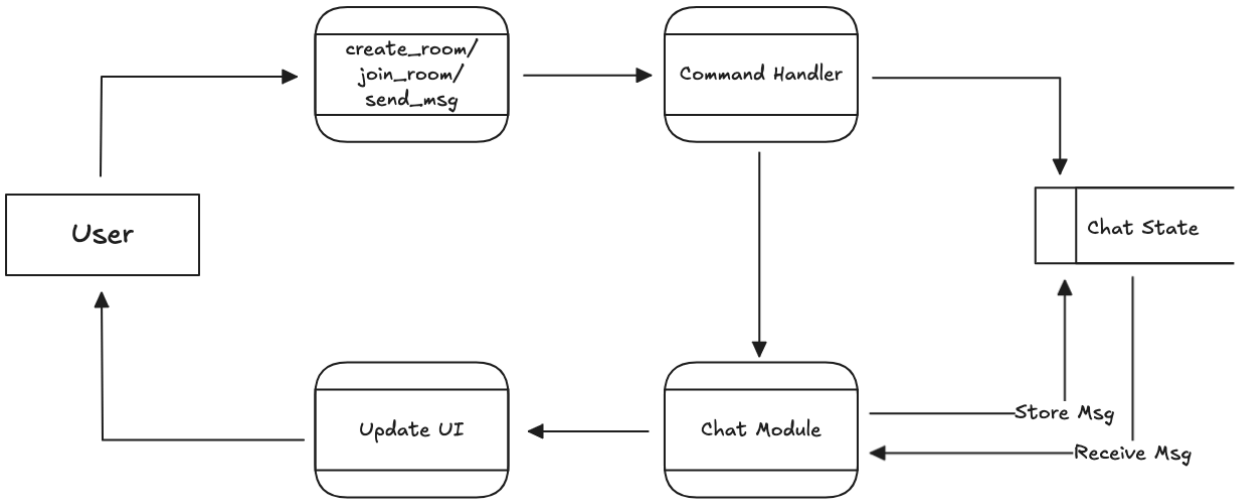
A peer to peer chat application which is end to end encrypted and does not require login to ensure complete user privacy

Code: https://github.com/mav3ri3k/p2p-chat

1.1 Stack

- **Tauri**: Framework for creating cross-platform applications with ui in javascript and logic in rust.
- **Iroh**: Library which helps orchestracte p2p communication.

1.2 Data Flow Diagram



1.3 Module Contributions

Implementation of core backend logic and communication.

Link to Github Commits

1.4 Algorithm Description

- Uses Iroh/QUIC for encrypted transport.
- Asynchronous tasks handle connections and messages.
- Event Driven Architecture

There are 4 core events:

1. Create Chat Room:
    1. Get/Generate Secret Key.
    2. Create Iroh Endpoint (with key, ALPN).
    3. Bind Endpoint.
    4. Get Node Address (from relay).
    5. Create NodeTicket.
    6. **Async Task:** Accept connection, bi-directional stream. Read/ Verify Handshake.

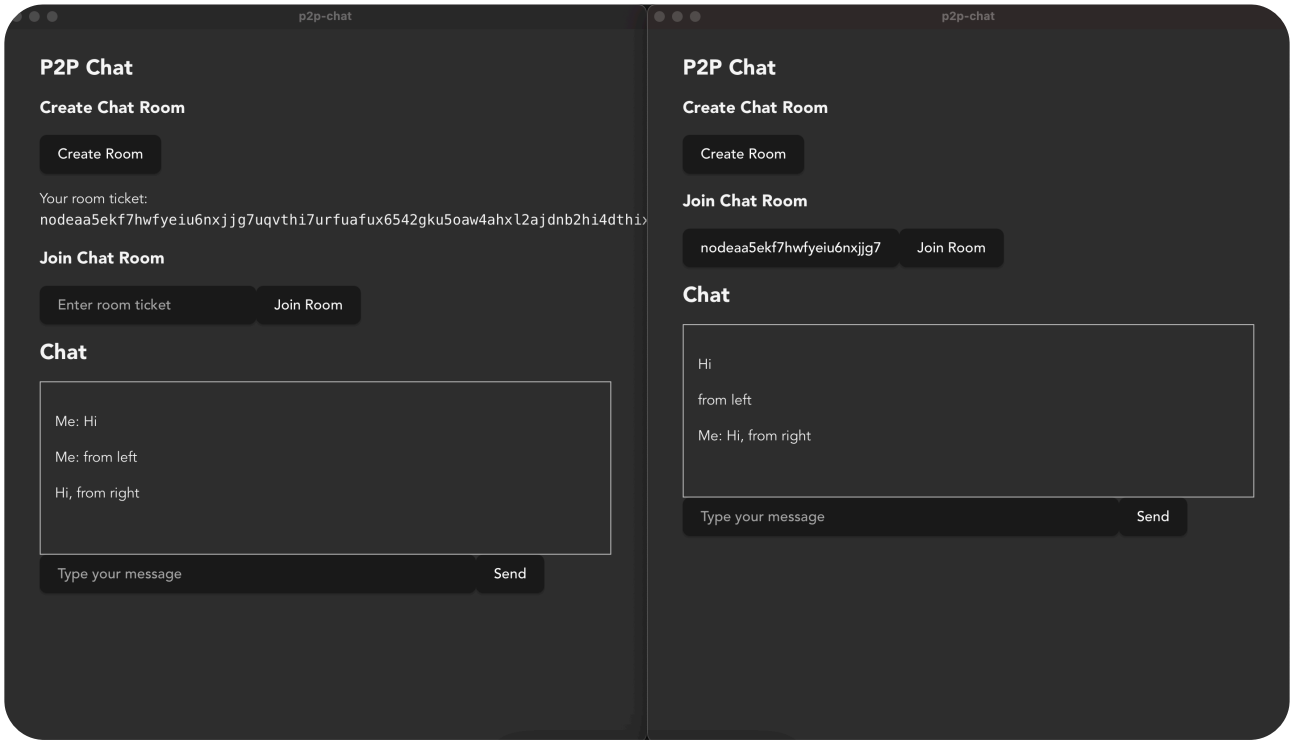
7. **Async Task:** Read messages, emit “new-message” event. Store SendStream in ChatState.

2. Join Chat Room:
   1. Parse NodeTicket.
   2. Get/Generate Secret Key.
   3. Create Iroh Endpoint
   4. Bind Endpoint.
   5. Connect to peer (using NodeID).
   6. Open bi-directional stream. Send Handshake.
   7. **Async Task:** Read messages, emit “new-message” event. Store SendStream in ChatState.

3. Send Message:
   1. Lock ChatState.
   2. Lock SendStream (from ChatSession).
   3. Write message to SendStream.
   4. Release Locks.

4. Receiving a Message:
   1. Read from RecvStream: Asynchronously read data from the `RecvStream` of the established bi-directional QUIC stream.
   2. Decode received bytes to to string.
   3. Emit Tauri Event for emit a “new-message” event.
   4. The frontend is subscribed to the “new-message” event. Upon receiving the event, the frontend updates the UI.
   5. Loop: Repeat steps 1-4 to continuously listen for new messages on the stream.

1.5 Output



1.6 Information Security

The implementation is fundamentally based on information security and the CIA Triad:

- **Confidentiality:** Use of TLS encryption, protecting the confidentiality of data in transit. The use of `SecretKey` and `PublicKey` pairs ensures that only the intended recipient can decrypt the data.
- **Integrity:** QUIC provides built-in integrity protection through its authenticated encryption.
- **Availability:** The combination of P2P architecture, QUIC's resilience to network changes, and the use of relay servers as a fallback mechanism enhances availability.
- **Authenticity:** QUIC provides strong **peer** authentication. The connecting node verifies the identity of the other node using its `PublicKey` (NodeId).
- **Accountability:** Requires user authentication as a prerequisite.
- **Privacy:** Encryption protects the content.