**Project Report: Secure P2P Chat Application**

**1. Introduction The goal of this project is to build a secure peer-to-peer (P2P) chat application that enables two clients to communicate securely over a network. The application ensures confidentiality by using hybrid encryption: asymmetric encryption is used initially to securely exchange a symmetric key, which is then used for efficient message encryption during the chat session.**

**2. Objectives**

* **Enable two users to chat securely over the network.**
* **Use asymmetric encryption to share a symmetric key.**
* **Use symmetric encryption for fast, secure message exchange.**
* **Ensure the chat runs over a direct peer-to-peer connection.**

**3. Tools and Technologies**

* **Python: Programming language**
* **socket: For creating network communication**
* **cryptography.fernet: For symmetric encryption**
* **Threading: To handle sending and receiving messages concurrently**
* **OOP (Object-Oriented Programming): For organized and clean code structure**

**4. System Components**

* **Signaling Server: Helps clients find each other and exchange connection information. It doesn’t handle message forwarding.**
* **Client (Host): Acts as a server that waits for a connection from a peer.**
* **Client (Peer): Connects to the host to begin communication.**
* **Encryption Module: Handles all encryption and decryption operations using Fernet.**

**5. How It Works**

1. **The signaling server is started and waits for clients.**
2. **The first client (Host) connects and registers itself.**
3. **The second client (Peer) connects and requests to chat with the Host.**
4. **Once matched, the Peer connects directly to the Host’s IP and port.**
5. **The Host sends a symmetric key using Fernet.**
6. **Both clients begin sending encrypted messages using the shared key.**

**6. Security Mechanism**

* **The symmetric key is generated by the host.**
* **The key is securely sent to the peer at the start of the chat.**
* **All messages are encrypted using Fernet encryption before being sent and decrypted upon receipt.**

**7. Features**

* **Fully encrypted peer-to-peer messaging**
* **Simple and readable terminal interface**
* **Object-oriented codebase for maintainability**

**8. Limitations**

* **Currently supports only two users in one session.**
* **Manual key exchange (can be automated using public-key cryptography).**
* **No graphical interface (CLI only).**

**9. Future Enhancements**

* **Add graphical user interface using Tkinter or PyQt**
* **Enable group chatting and broadcasting**
* **Automate secure key exchange with RSA**
* **Improve signaling to work with NAT/firewalled networks**

**10. Conclusion This project demonstrates the implementation of a secure, efficient, and lightweight peer-to-peer chat application using Python and encryption. It provides a practical approach to understanding secure communications and encryption mechanisms in real-world scenarios.**