

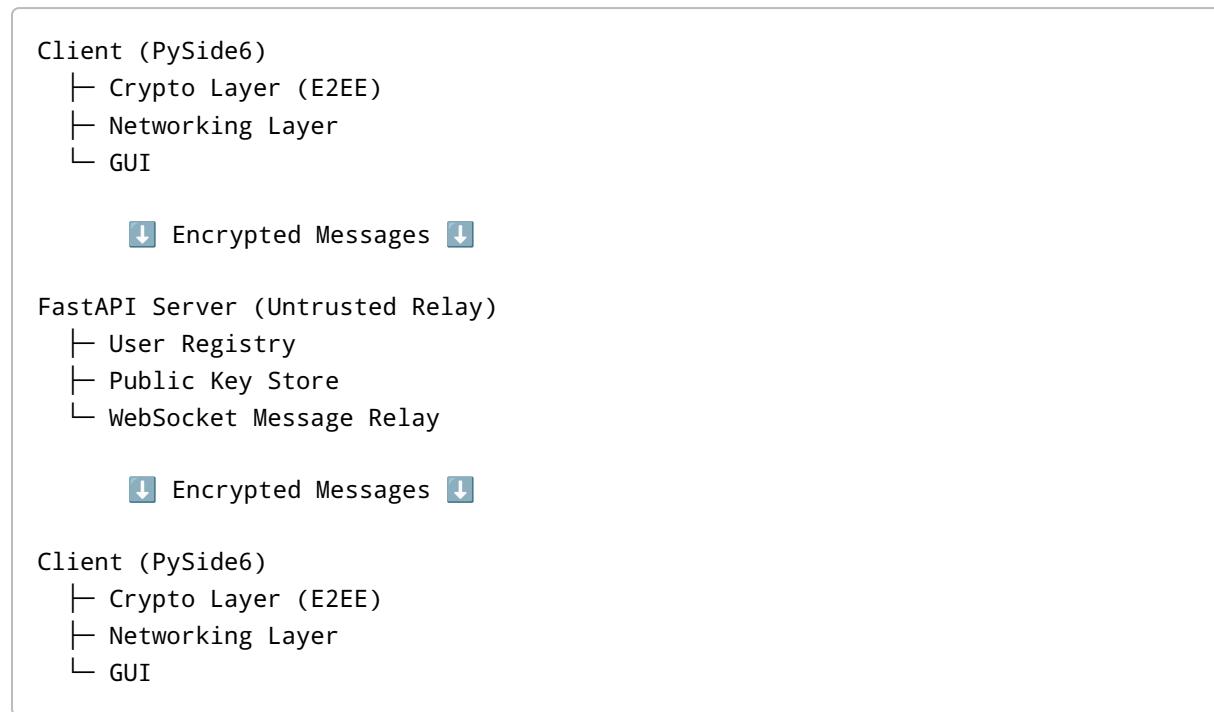
Secure Messaging Project — Team Work Division & Architecture Plan

Overview

This document defines how work is divided across a 3-person team building a **real-time, end-to-end encrypted (E2EE) messaging system** using Python and FastAPI. The goal is to ensure clear ownership, minimal overlap, and a structure that mirrors industry best practices for secure systems.

The system follows a strict separation of concerns: - Cryptography & protocol logic - Backend networking & message relay - Client application & user experience

High-Level Architecture



Security rule: The server never has access to plaintext messages, private keys, or derived symmetric keys.

Team Roles & Responsibilities

Person 1 — Crypto & Security Engineer

Primary responsibility: Correctness and safety of cryptographic design.

Responsibilities

- Design and implement cryptographic primitives
- Define key lifecycles and message formats
- Own the security model and threat analysis

Tasks

- Implement:
 - Ed25519 identity key generation & signing
 - X25519 Diffie-Hellman key exchange
 - HKDF key derivation
 - AEAD encryption (AES-GCM or XChaCha20-Poly1305)
- Define message envelope format
- Ensure correct nonce usage and key separation
- Write threat model and security assumptions

Deliverables

- `crypto/` module
- Minimal crypto API (e.g. `encrypt_message()`, `decrypt_message()`)
- Unit tests validating encryption, decryption, and signature verification

Key rule: Only one person touches cryptographic internals to avoid inconsistencies.

Person 2 — Backend & Networking Engineer

Primary responsibility: Reliable message transport and user coordination.

Responsibilities

- FastAPI server development
- WebSocket message relay
- User registry and public key distribution

Tasks

- Build REST endpoints:
 - `/register` (user + public keys)
 - `/keys/{username}` (public key lookup)
- Implement WebSocket messaging
- Manage:
 - Authentication / sessions
 - Online/offline user tracking
 - Rate limiting and basic abuse prevention
 - Log metadata events (no message content)

Deliverables

- `server/` directory
- FastAPI application with WebSocket relay
- Documentation explaining the server trust model

Key rule: The backend treats encrypted payloads as opaque blobs and never performs cryptographic operations.

Person 3 — Client & UX Engineer

Primary responsibility: Usability, persistence, and client integration.

Responsibilities

- PySide6 GUI development
- Client-side state management
- Local key storage

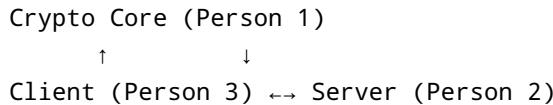
Tasks

- Build UI components:
- Login / registration
- Chat interface
- Contact list
- Handle:
- Secure local key storage
- Message history
- File selection and upload
- Implement trust UX:
- Identity fingerprint display
- Contact verification flow
- Error handling

Deliverables

- `client/` directory
 - Fully working GUI client
 - UX documentation describing trust decisions
-

How the Components Integrate



- Crypto logic exposes a small, well-defined API
- Client calls crypto functions but never re-implements them
- Server simply relays encrypted data

This structure minimizes security risk and merge conflicts.

Shared Responsibilities

All Team Members

- Protocol review and validation
- End-to-end testing
- Final documentation and presentation

Recommended Sync

- Weekly 20–30 minute check-in
- Review:
 - Message format changes
 - API contracts
 - Threat model updates

Suggested Repository Structure

```
project/
├── crypto/
│   ├── identity.py
│   ├── dh.py
│   ├── hkdf.py
│   ├── aead.py
│   └── protocol.py
├── server/
│   ├── main.py
│   ├── websocket.py
│   └── registry.py
└── client/
```

```
|   └── gui/
|   └── storage/
|   └── network/
|       └── app.py
└── docs/
    ├── threat_model.md
    └── protocol_flow.md
```

Summary

This division of labor ensures:

- Clear ownership of security-critical code
- Independent development without overlap
- An architecture aligned with real-world secure messaging systems

This plan can be used directly as a development roadmap or shared with collaborators and reviewers.