Reviewer: Samin Yeasar Seaum
Student ID: a1976022
Course: Secure Programming (SOCP Phase 3 — Testing and Peer Review)

Peer Review Report — Group 70

1. Executive Summary

Group 70 delivers a well-structured and comprehensive secure chat prototype. This report highlights four educational backdoors, overall security posture, compliance with SOCP v1.3, and key recommendations. The feedback is constructive and focused on enhancing the codebase towards professional secure programming standards.

2. Security Findings (Including Backdoors)

Backdoor 1 — Hardcoded JWT / Secret Keys
Location: src/auth/auth.py, src/transport/ws_server.py
Description: Static secret keys used for signing tokens. Highlights the risks of embedded secrets and rotation issues.
Remediation: Store secrets in environment variables or a secure secrets manager. Enforce rotation and remove literals.

Backdoor 2 — Placeholder / Missing Message Signatures
Location: src/msgFormat.py (sig field placeholder)
Description: Signature field static; not verified. Demonstrates signature validation necessity for authenticity.
Remediation: Implement RSASSA-PSS signing and verification; reject unsigned or altered messages.

Backdoor 3 — Weak RSA Key Size Default (2048 bits)
Location: src/protocols/HELLO.py, src/mykeys.py
Description: Key generation defaults to 2048 bits instead of 4096. Demonstrates crypto downgrade risks.
Remediation: Default to 4096-bit keys; validate modulus length during import and registration.

Backdoor 4 — Service Bound to All Interfaces (0.0.0.0)
Location: src/auth/main.py (uvicorn.run)
Description: Binding to all interfaces exposes unnecessary attack surface; demonstrates network scoping issues.

Remediation: Default to 127.0.0.1 and require explicit configuration for external exposure.

Additional Security Observations:

• Broad try/except/pass blocks in src/client/chat_client.py hide exceptions; use targeted exception handling.
• HTTP requests lack timeouts; implement sensible timeouts and retry/backoff logic.
• Missing docstrings and long functions reduce maintainability; refactor into smaller, single-purpose handlers.

3. Strengths

• Modular design separating authentication, transport, and cryptographic logic.
• Good understanding of key management and signing mechanisms.
• Protocol modules (HELLO, MSG, PEERLIST, HEARTBEAT) structured for scalability.
• Use of FastAPI and uvicorn for asynchronous communication, improving efficiency.
• Consistent naming and file structure enhance maintainability and readability.

4. SOCP v1.3 Compliance Mapping

The implementation was compared against major SOCP v1.3 requirements. Below is a summary of compliance and actions required for full alignment.

Cryptography – RSA-4096 Requirement
Status: Partial. RSA implemented but default 2048-bit generation. Action: Update to 4096-bit keys and enforce key-size checks.
Transport Signatures ('sig' Verification)
Status: Partial. Signature field exists but unverified in some flows. Action: Add RSASSA-PSS verification before trust-impacting state changes.
Bootstrap & Introducer Pinning
Status: Mostly Compliant. Ensure pinned introducer keys verify SERVER_WELCOME and assigned_id integrity.
Presence Gossip & Registration
Status: Partial. Deduplication not fully enforced. Action: Add seen-IDs cache (LRU/TTL) and malformed-input checks.
Loop Suppression & Replay Protection
Status: Unclear. Action: Add nonce/timestamp replay protection consistent with spec.
Mandatory Features & Heartbeats
Status: Mostly Compliant. Add end-to-end tests verifying /list, /tell, /all, /file and heartbeat timeouts.

5. Recommendations

1. Externalize all hardcoded secrets; integrate environment-based configuration.
2. Implement signature verification across all protocol handlers.
3. Enforce 4096-bit RSA key generation and validation.
4. Limit default binding to localhost; document external exposure policy.
5. Refactor complex handlers and improve inline documentation.
6. Implement message deduplication and replay protection.

6. Testing and CI Integration

• Add pytest-based unit and integration tests for weak-key rejection, unsigned frame rejection, and replay detection.
• Integrate Pylint and Bandit in CI pipeline; enforce quality thresholds.
• Add pre-commit hooks for linting, formatting, and dependency checks.
• Include dependency installation guide (requirements.txt).

Appendix — Tool Evidence Summary

Top Bandit Findings:

- **B104 (MEDIUM) — src/auth/main.py:20:** Binding to all interfaces (0.0.0.0)
- **B105 (LOW) — src/auth/auth.py:6:** Hardcoded secret key
- **B110 (LOW) — src/client/chat_client.py:87:** Broad try/except/pass
- **B113 (MEDIUM) — src/client/chat_client.py:110:** HTTP request without timeout
- **B303 (LOW) — src/crypto/mykeys.py:22:** Use of weak hash algorithm

Top Pylint Findings:

- **C0114 (Convention) — src/protocol.py:** Missing module docstring
- **R0915 (Refactor) — src/transport/ws_server.py:134:** Function too complex; refactor into smaller units
- **E0401 (Error) — src/main.py:** Missing dependency import (fastapi)
- **C0301 (Convention) — src/client/chat_client.py:55:** Line too long (>100 chars)
- **W0702 (Warning) — src/client/chat_client.py:87:** Bare except; specify exception type