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

Peer Review Report — Group 69

1) Executive Summary

Group 69 presents a capable SOCP-style chat implementation with clear separation between client, node (server/peer), and persistence layers. Static analysis (pylint & bandit) shows a solid base (7.11/10 pylint) with several places to improve maintainability and security hardening. This review highlights strengths, identifies educational backdoors present in this submission, and provides concrete, low-friction remediation suggestions aligned to SOCP v1.3.

2) Educational Backdoors (identified and documented)

Backdoor 1 — Insecure transport (plaintext WebSocket)
Location: node.py — outbound peer connection uses ws:// (no TLS)
Intent & effect: Allows observation/tampering on-path and is commonly used as a classroom backdoor for transport hardening exercises.
Remediation: Support wss:// with certificate verification; gate ws:// behind a dev-only flag.

Backdoor 2 — Weak key acceptance (no modulus enforcement)
Location: node.py — key size parsed but not enforced (unused variable), rsa_pub_from_b64u accepted under broad try/except
Intent & effect: Permits registration/use of keys below RSA-4096 policy, weakening crypto assurances.
Remediation: Validate modulus length on import/registration and reject <4096-bit keys; document policy.

Backdoor 3 — Signature verification masking via silent exceptions
Location: node.py & client.py — multiple try/except/continue or try/except/pass blocks around message handling
Intent & effect: Swallows verification/parse errors so processing may proceed without strong authenticity checks.
Remediation: Verify before state changes; on failure, log and fail-closed. Replace broad 'except' with specific exceptions.

Backdoor 4 — SQL query construction pattern enabling risky expansion
Location: server_database.py — dynamic IN-clause placeholders constructed via f-string

Intent & effect: While parameters are used, dynamic SQL construction is intentionally left flexible, inviting discussion of injection risks.
Remediation: Build the placeholder string safely, ensure parameter binding only; consider executemany or prepared statements.

3) Code Quality & Maintainability (pylint highlights)

Overall pylint score: 7.11/10. Strengths include consistent naming and modularity. Areas to improve:

• Missing docstrings across many functions/classes; several overly long functions with deep branching (client.Client.run, node handlers).

• Large module (node.py ~1100 lines) — consider splitting into submodules for handlers, transport, and crypto utilities.

• Broad exception catches (W0718) and try/except/pass patterns reduce debuggability and complicate audits.

Representative pylint items: missing docstrings, too many branches/statements in Node._on_user_hello and bootstrap_join, duplicate crypto helper code between client and node, and line-length/style cleanups. filecite turn4file1

4) Security Posture (Bandit & manual observations)

Bandit flagged 30 issues total (0 High / 1 Medium / 29 Low). Notable items:

• B608 potential SQL construction risk (server_database.py) — use strict parameter binding.

• B110/B112 patterns (many) — replace with explicit exception handling & logging.

• WebSocket plaintext usage — add TLS by default and certificate pinning where feasible. filecite turn4file0

5) Strengths

• Clear separation of client, node, and persistence logic.

• Inclusion of RSA PSS signing/verification primitives in both client and node indicates strong crypto awareness.

• Presence of dedup/heartbeat logic and queued message delivery in the database layer demonstrates protocol fluency.

• Achieved a solid pylint baseline and consistent code style across files.

6) Concise SOCP v1.3 Compliance Mapping

• Cryptography — RSA-4096 required: Status: Partial. Key size parsed but not enforced. Action: enforce 4096-bit modulus on import/register (reject weak keys).

• Transport signatures required: Status: Partial. Verify signatures before mutating state; avoid masking failures via broad except.

• Bootstrap/Introducer pinning: Status: Mostly present. Ensure pinned pubkey verification and assigned_id validation when joining.
• Presence gossip & dedup: Status: Present but fragile. Keep a seen-IDs cache with TTL and log drops for observability.
• Heartbeats & timeouts: Status: Present. Add tests to ensure missed heartbeats trigger reconnection within the specified window.
• Mandatory features (/list, /tell, /all, /file): Status: Present; add E2E tests including negative cases.

7) Recommendations (prioritised)

1. Replace broad exception handling with targeted exceptions and explicit logging; fail closed on signature/parse errors.
2. Enforce RSA-4096 key policy and validate modulus on import/registration; reject weak keys.
3. Switch peer links to wss:// by default with certificate verification; allow ws:// only behind a dev flag.
4. Refactor node.py into modules (transport, handlers, storage, crypto) to reduce complexity and improve testability.
5. Harden SQL layer: keep parameter binding, avoid ad-hoc string construction, and add tests on message id handling.
6. Improve documentation: module/class/function docstrings and a short developer setup guide; remove duplicate crypto helpers.

8) Testing & CI Enhancements

• Add pytest tests for: unsigned/altered frame rejection; weak-key registration rejection; dedup replay drops; heartbeat timeouts; SQL queue marking.
• Integrate Bandit & Pylint in CI with thresholds; add pre-commit hooks for formatting and import order.
• Provide sample configs for secure defaults (wss://, key policy) and a Makefile/README quickstart.

Appendix — Tool Evidence (concise)

Bandit highlights:

- **B112 (LOW) — client.py:153:** try/except/continue in client input loop
- **B110 (LOW) — client.py:239:** try/except/pass masking receive errors
- **B110 (LOW) — node.py:239:** try/except/pass around pubkey acceptance
- **B112 (LOW) — node.py:903:** try/except/continue around ws connect
- **B608 (MEDIUM) — server_database.py:233:** hardcoded SQL expression (dynamic IN placeholders)

Pylint highlights (score 7.11/10):

- **R0915 (Refactor) — node.py:889 (bootstrap_join):** Too many statements; split into helpers
- **R0912 (Refactor) — node.py:397 (_on_user_hello):** Too many branches; simplify state machine
- **W0718 (Warning) — node.py:*:** Catching too general Exception across many handlers
- **C0116 (Convention) — node.py & client.py:** Missing function docstrings
- **R0801 (Refactor) — client.py/node.py:** Duplicate crypto helpers; unify in a shared module