Security Design Doc

Project: Semester 7 - Individual (E2EE Messaging Platform)

Version: 0.1 (2025-11-02)

---

1. Purpose & Scope

This document consolidates all security-related decisions taken to date and describes the current E2EE protocol design for one-to-one and group messaging, including file sharing. It also lists assumptions, threat model, trust boundaries, and open issues for future iterations.

In scope: Backend services (Gateway, Auth, Messaging), client–server contracts for E2EE, CI/CD security controls.

Out of scope: Full cryptographic proofs, mobile/desktop client UI/UX.

---

2. High-Level Security Goals

- **Confidentiality**: Server never has plaintext of messages/files; only ciphertext + minimal metadata.
- **Integrity & Authenticity**: Recipients detect tampering via AEAD auth tags; messages are bound to sender identity.
- **Forward Secrecy (FS)**: Compromise of long-term keys should not reveal past messages (see current status below).
- **Post-Compromise Safety (PCS)**: Limit blast radius of a compromised device via key rotation/re-establishment.
- **Availability**: Rate-limiting and basic DoS hardening at Gateway; at-least-once delivery with client de-dupe.
- **Privacy by Design**: Data minimization, purpose limitation, retention, and subject-rights flows.

---

3. Assets, Adversaries, Assumptions

3.1 Assets

- **Message content** (1-to-1 & group).
- **Attachment content** (planned).
- **Long-term identity keys** (client-side).
- **Session/ephemeral keys** (client-side).
- **Account credentials** (email/username + password).
- **Server secrets** (JWT signing key, DB creds).

3.2 Adversaries

- **Network attacker**: eavesdrop/modify traffic; attempts downgrade/miTM.
- **Honest-but-curious server**: can read DB/object store but should learn no plaintext.
- **External attacker**: attempts credential stuffing, token theft, replay.
- **Malicious participant**: valid group member sharing content/keys. (Non-goals: preventing authorized recipients from re-sharing plaintext outside the system.)

3.3 Assumptions

- Clients implement crypto correctly and protect private keys at rest.
- TLS termination is secure; certificates are valid and pinned at the client.
- Users control their devices; device compromise is detectable/recoverable (session revoke).

---

4. Trust Boundaries & Data Flow

- **Client ↔ Gateway**: Will be TLS-protected HTTP/WS, once deployed on the cloud.
- **Gateway ↔ Services (Auth/Messaging/File)**: internal network (Docker), also TLS-terminable if externalized.
- **Services ↔ Postgres**: private network; credentials via env; least privilege.

---

5. Implemented Security Controls (Server-Side)

- **Passwords**: `Argon2id` hash(target ~64 MB memory, p=2, t≈1).
- **Authentication**: Short-lived **JWT access** (≈15 min) + **refresh tokens** (≈30 days) bound to a **server-side session** (revocation on logout/compromise).
- **Sessions**: Store IP as Postgres `inet` (host only); rotate refresh on use; single-device binding per token.
- **Input Validation**: Strict JSON validation; server **rejects plaintext** fields where ciphertext is required.
- **Rate Limiting**: Auth ≤ 5 req/sec/IP; Messaging ≤ 60 req/min/user (configurable).
- **Secrets Management**: Environment variables via CI/CD; no secrets in VCS; GH Actions **Secrets** for sensitive values; **Variables** for non-sensitive.
- **Logging**: Structured logs; **never** log secrets or plaintext content; correlation IDs planned.
- **Health & Rollback**: Container healthchecks; immutable image tags `:sha-<commit>` for rollback.

---

6. E2EE Protocol - Current Design (MVP)

```
sequenceDiagram
  autonumber
  participant A as Sender (Client A)
  participant G as Gateway
  participant M as Messaging Service
  participant B as Recipient (Client B)
  A->>G: POST /messages/send { envelope(ciphertext, nonce, AAD) }
  G->>M: forward
  M-->>B: push over WebSocket
  B->>B: Derive k_enc (X25519 + HKDF) → AEAD decrypt
```

```
sequenceDiagram
    autonumber
    participant UI as Web UI (React)
    participant WASM as WASM Bridge
    participant MC as Msg Client
    participant CC as Crypto Core
    participant KS as Keys Service
    participant MS as Messages Service
    participant DB as Messages DB

    Note over UI: User presses Send

    UI->>WASM: prepareSend with state and plaintext
    WASM->>MC: PrepareSend

    alt No existing session for this conversation
        MC->>KS: GET /keys/bundle for recipient device
        KS-->>MC: PreKeyBundle

        Note over KS,MC: PreKeyBundle fields\n- recipient identity_pub\n- recipi
ent signed_prekey_pub\n- signed_prekey_signature\n- recipient one_time_pre
key_pub optional\n- key identifiers for spk and opk

        MC->>CC: InitSession using bundle
        CC-->>MC: SessionState and HandshakeMessage

        Note over MC,MS: First message will include Handshake in the header\n
Handshake fields sent:\n- sender ephemeral_pub\n- ids of recipient spk and o
pk used\n- authentication tag or transcript hash
    end

    MC->>CC: Encrypt with SessionState
    CC-->>MC: ciphertext and MessageHeader

    Note over MC,MS: Every message header contains ratchet data\nRatchet h
```

```
eader fields:\n- sender ratchet_dh_pub\n- pn previous chain count\n- n curren
t chain index\n- nonce for AEAD

    MC-->>WASM: request with header and ciphertext
    WASM-->>UI: request ready to send

    UI->>MS: POST /messages/send with header and ciphertext
    MS->>DB: store envelope as ciphertext only
    MS-->>UI: 200 OK

    rect rgb(1,1,1)
    Note over MS,DB: Delivery pipeline keeps ciphertext intact
    MS-->>UI: recipient WS push envelope unchanged
    end

    UI->>WASM: handleEnvelope with header and ciphertext
    WASM->>MC: HandleEnvelope
    opt First envelope for this conversation
        MC->>CC: AcceptSession using Handshake from header
        CC-->>MC: SessionState ready for decrypt
    end
    MC->>CC: Decrypt with header and ciphertext
    CC-->>MC: plaintext
    MC-->>WASM: plaintext to UI
```

6.1 Key Material (Client-Side)

- **Identity Key Pair**: Long-term keypair (Curve25519 for DH; Ed25519 for signatures — optional for MVP). Public identity key is uploaded to server; **private key never leaves client**.
- **Per-Conversation Session Key** (MVP): Established via **X25519** DH between sender and recipient identity keys (or prekeys, see planned); derives a symmetric key via HKDF.
- **Symmetric Cipher**: **AES-GCM** or **ChaCha20-Poly1305** (AEAD). Ciphertext includes **nonce** and **auth tag**.

6.2 One-to-One Messaging (MVP)

Establish session:

1. Sender fetches recipient **public identity key**.
2. Sender computes DH: `K = X25519(sender_priv, recipient_pub)`.
3. Derive `k_enc = HKDF(K, context="1to1:enc", … )`.

Encrypt & send:

1. Generate random **nonce** (unique per message).
2. **AAD** binds metadata (e.g., `conversation_id`, `sender_id`, `message_id`, `created_at`).
3. Encrypt payload with AEAD to produce `ciphertext` + `auth_tag`.
4. Send **envelope** to server: `{message_id, conv_id, sender_id, created_at, ciphertext, nonce, auth_tag, content_type, parent_id?}`.

Receive & decrypt:

1. Recipient derives same `k_enc` via DH.
2. Verify AEAD tag over **ciphertext + AAD**; on success, return plaintext.

Delivery semantics: at-least-once; clients **de-duplicate** via `message_id`.

---

7. Message Envelope (Server-Visible)

```
{
  "message_id": "uuid-v4",
  "conversation_id": "uuid-v4",   // or group_id
  "sender_id": "uuid-v4",
  "created_at": "RFC3339",
  "ciphertext": "base64",
  "nonce": "base64",
  "auth_tag": "base64",
  "content_type": "text|file-pointer|...",
  "parent_id": "uuid-v4|null"      // optional reply threading
}
```

---

8. Secure Development Lifecycle (SDLC) Controls

- **Linters/Static Analysis**: `golangci-lint` (gofumpt, gocyclo, revive, unparam, misspell).
- **CI**: `go vet`, `go test`, build per service; fail-fast.
- **Supply Chain**: Images built in CI; pushed to GHCR; deploys pin `:sha-<commit>`.
- **Environments**: `.env` templating via Actions; secrets injected only at deploy; no secrets in logs.
- **Observability (MVP)**: Health endpoints + container healthchecks; planned centralized logs/metrics.
