E2EE Explained

Project: Semester 7 - Individual (E2EE Messaging Platform)

Scope: Cryptographic model for one-to-one, groups, and attachments - concepts, keys, message flows, and security properties.

---

1) Why E2EE and What It Guarantees

- **Goal:** Only the intended recipients can decrypt messages/files; the server stores **ciphertext + minimal metadata**.
- **Properties targeted:** Confidentiality, integrity, authenticity, **forward secrecy (FS)**, **post-compromise security (PCS)**, and deniability (optional, see signatures notes).
- **Non-goals:** Preventing recipients from re-sharing plaintext outside the app; hiding traffic timing/size completely.

---

2) Cryptographic Building Blocks (Concepts)

- **Asymmetric DH (X25519):** Derives shared secrets between two parties from a pair of private/public keys.
- **Symmetric AEAD (AES-GCM or ChaCha20-Poly1305):** Encrypts and authenticates message content; rejects tampering.
- **KDF (HKDF):** Expands a shared secret into multiple independent keys with domain separation (context strings).
- **Signatures (Ed25519):** Authenticates public keys or protocol data (e.g., signed prekeys). May be omitted for deniability in some designs.
- **Nonces:** Unique per message for AEAD; generated randomly or via counters/ratchets. Reuse under the same key is forbidden.
- **AAD (Associated Data):** Unencrypted metadata bound into the AEAD tag (e.g., conversation_id, message_id) to prevent malleability.

---

3) Key Types and Their Roles

3.1 Identity Keys (Long-Term)

- **What:** Long-lived asymmetric key pair per device or per user (preferred: per device).
- **Purpose:** Identify the party, sign prekeys, and authenticate session establishment.
- **Server sees: Public** key only. Private key remains on client.

3.2 Prekeys (Asynchronous Setup)

- **What:** Medium-term signed prekeys + a pool of one-time prekeys published to the server.
- **Purpose:** Allow a sender to start a secure session while the recipient is offline (X3DH-style).
- **Rotation:** Regularly (e.g., weeks) or on compromise; one-time prekeys are consumed once.

3.3 Session Keys (Short-Lived, Per Conversation/Per Message)

- **What:** Symmetric keys derived during session setup.
- **Purpose:** Encrypt messages. With **Double Ratchet**, each message uses a fresh key for FS/PCS.
- **Storage:** Ephemeral; retained only as needed to decrypt history locally (via key chains).

3.4 Group Sender Keys (Shared or Per-Device)

- **What:** Symmetric keys used to encrypt group messages efficiently.
- **Variants:** (a) **Single shared sender key** per group (simpler); (b) **Per-device sender keys** (better PCS and auditability).
- **Rotation:** On membership changes (join/leave/device add-remove) and periodically.

3.5 Attachment Keys (Per-File)

- **What:** Random symmetric keys generated client-side per file.
- **Purpose:** Encrypt attachments before upload; keys are wrapped for recipients inside E2EE messages.

---

4) One-to-One Protocol (From MVP to Full)

4.1 MVP (Static DH + AEAD)

1. Sender fetches recipient identity public key.
2. Compute DH shared secret `K = X25519(a_priv, B_pub)`.
3. Derive `k_enc = HKDF(K, context="1to1:enc")`.
4. For each message: generate **nonce**, encrypt with AEAD; bind AAD (ids, timestamps) to the tag.

4.2 Full (X3DH + Double Ratchet)

- **X3DH setup:** Identity key + signed prekey (+ one-time prekey) → initial shared secret(s).
- **Double Ratchet:** Each message advances sender/receiver chains, deriving a fresh key; compromise doesn't reveal past or future messages after the next ratchet step.
- **Key verification:** Short authentication strings (SAS)/QR compare to defeat MiTM.

---

5) Metadata & Privacy

- **Minimize server-visible data:** `message_id`, `conversation_id/group_id`, `sender_id`, timestamps, sizes.
- **Bind essential metadata with AAD** to prevent mix-and-match attacks.
- **GDPR implications:** Subject-rights cover metadata and blobs; no content keys are stored server-side.

---

6) Compromise & Recovery

- **Lost device:** Revoke sessions (server-side), rotate prekeys, and re-establish sessions (new DR chains).
- **Password breach:** Argon2id + rate limiting.
- **Backup/restore:** Client key backups must be encrypted end-to-end; recovery should not reveal keys to the server.

---

7) Correctness Rules & Developer Notes

- **Never reuse a nonce with the same key.**
- **Derive separate keys** for encrypt vs. MAC (AEAD already integrates MAC) and for protocol stages via HKDF labels.
- **Authenticate identities** before trusting public keys (verification UX).
- **Reject plaintext** at the API boundary; treat ciphertext and envelope fields as the only acceptable payload.
- **Clock skew tolerance** (~60s) for token/session timestamps; don't bind cryptographic validity to wall-clock time except for token expiry.
- **Store only what you must:** ciphertext, nonces, tags, minimal metadata.

---

8) Message & File Envelope Sketches

```
// Message (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"
}
```

9) Messaging chart

```
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- recipient signed_prekey_pub\n- signed_prekey_signature\n- recipient one_time_prekey_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\nHandshake fields sent:\n- sender ephemeral_pub\n- ids of recipient spk and opk 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
```

9) Quick Glossary

- **FS (Forward Secrecy):** Past messages stay safe even if long-term keys are compromised later.

- **PCS (Post-Compromise Security):** The protocol self-heals after compromise, limiting future damage.
- **Prekey:** A published key that enables session startup while the peer is offline.
- **AAD:** Data that isn't encrypted but is authenticated by AEAD.