📝 Flash Storage & Security Design

1. Key Hierarchy & Root of Trust

- **DMS (Device Master Secret)**: one-time provisioned, never leaves device. Burned at provisioning.
- **KEK (Key Encryption Key)**: derived from DMS + boot counter + device UID. Used to wrap DEKs.
- **DEK (Data Encryption Key)**: per-record. Derived via HKDF. Used to encrypt record payloads.
- **KEK_ODD**: session-limited KEK derived after PIN/UV success. Used for records that require user presence/verification.
- **Session tokens**: short-lived, kept in RAM only, never persisted.
- **PIN stretching**: Argon2id (software) → stretched key → input to KEK_ODD derivation.
- **All derivations**: HKDF-SHA256 (CC310-backed).

2. Record Format

Each record = append-only unit in data segment.

- **Header**
  - Record UUID (random 128b)
  - Sequential number (monotonic, prevents replay/reinsertion)
  - Length (payload size)
  - Flags (alive/dead, type, etc.)
- **Body**
  - Encrypted payload (AEAD: AES-256-GCM, 96b nonce, 128b tag)
- **Trailer**
  - CRC32 (fast fail detection, optional)
  - AEAD authentication tag (mandatory integrity)

```
struct RecordHeader {
  u32 magic = 0x5458524B  // "TXRK"
  u8  version = 1
  u8  type    // 0=cred, 1=large-blob, 2=meta-note, ...
  u16 header_len
  u128 record_uuid
  u256 rpIdHash       // SHA-256(RP ID), for lookup
  u32  seqno          // monotonically increasing per record replacement
  u32  created_ts
  u32  last_used_ts
  u16  aead_algo      // 1=AES-GCM-256, 2=CH20P1305
  u16  payload_len    // plaintext length before AEAD
  u32  crc_header     // CRC-32 over header (excl. this field)
}
struct RecordBody {
  u96  nonce          // AEAD nonce; ensure per-(DEK,seqno) uniqueness
  u8   payload[payload_len]   // e.g., FIDO2 credential source incl. pubkey
  u128 tag            // AEAD tag
}
struct RecordTrailer {
  u32  crc_physical   // CRC-32 over full physical record for corruption triage
  u32  commit_mark    // 0x8BADF00D when fully written (last word)
}
```

3. Log-Structured Layout (Flash Partitioning)

Fixed layout for **8 MB serial NOR flash (**Append-only, no in-place updates):

- **Superblock (metadata)** → 2 × 128 KB = 256 KB
- **Index region** → 256 KB
- **Data segment** → 6 MB (~5000 records @ 1.2 KB avg size)
- **Large blob area** → 1.5 MB (for CTAP2 largeBlob storage)

4. Indexing & Lookup

- Index maps UUID → physical location in data segment.
- Stored in fixed index region.
- Supports fast lookup & GC scan.
- Metadata confidentiality: index stores only hashes of user handles/PII.

5. Anti-Rollback & Replay Protection

- **Boot counter (monotonic)**
  - Increment on each successful boot.
  - One-way increment → stored in OTP or protected flash.
  - Used in KEK derivation → rollback makes KEK mismatch → old data undecryptable.
- **Superblock hash chain**
  - Hash = SHA256(prev superblock || boot counter || layout metadata).
  - Ensures continuity across reboots → tamper-evident.
- **Record sequential number**
  - Each UUID increments seq# on update.
  - Prevents replay of old record with same UUID.
  - Checked during lookup.

6. PIN / User Verification Binding & Rate Limiting

- After PIN/UV success: derive **KEK_ODD**.
- Store only a short-lived **ODD session token in RAM**.
- Encrypt UV-bound records with DEKs derived from KEK_ODD → useless without fresh verification.
- **Rate limit counters & back-off values**
  - Stored as small meta-records in flash.
  - Increment on failed PIN attempts.
  - Enforce exponential backoff (delay grows per failure).
  - Written with same commit protocol → atomic.

7. Garbage Collection

- Append-only writes → dead/stale records accumulate.
- GC process:
  1. Select segment for reclaim.
  2. Copy live records to new space.
  3. Erase segment.
- Trigger when free segments drop below threshold.
- Fragmentation handled automatically (copy compacts records).

8. Crypto Modes, Parameters, Randomness

- **AEAD**: AES-256-GCM (primary).
- **Nonces**: 96-bit, from TRNG (CC310).
- **KDFs**: HKDF-SHA256 (hardware), Argon2id (software, PIN only).
- **Integrity**: AEAD tags mandatory; CRC32 optional fast-fail.

9. Power Loss & Crash Consistency

- All mutating ops (writes, updates, deletes) = **two-phase commit**:
  - **Prepare**: write new record(s) as append.
  - **Publish**: atomically mark new record as live + old as dead.
- If crash before publish → old record still valid.
- If crash after publish → new record valid, old dead.
- Ensures no half-written corruption.

10. Boot & Provisioning Checklist

Step-by-step burn-in flow:

1. **Entropy check**: verify CC310 RNG working.
2. **Generate DMS**: 256b random, store in OTP/protected flash.
3. **Derive KEK** = HKDF(DMS || UID || boot counter).
4. **Provision root keys**: wrap and store securely.
5. **Initialize superblocks**: write SB0 + SB1 with clean layout + hash chain base.
6. **Initialize index region**: empty.
7. **Reserve GC space**: mark one free segment.
8. **Lock debug interface**.
9. **PIN provisioning**: store Argon2id-stretched key record.
10. **Initialize rate-limit counters** (0 attempts).
11. **First-boot test**: ensure KEK re-derivation, rollback check, GC run.

11. Performance & Wear

- Must always keep ≥1 free segment → GC breathing room.
- If data segment fills (6 MB used):
  - GC runs to reclaim space.
  - If no dead space left → out-of-storage error → reject new record.
- ~5000 records @ 1.2 KB average. Likely sufficient for FIDO2 credentials (tens to hundreds per user, not thousands).
- **QSPI-XIP**:
  - Quad-SPI Execute-in-Place → allows direct mapped read from flash → speeds lookups & index reads.
  - No need to copy into RAM for reads.

12. Security Hardening Extras

- **Metadata confidentiality**: never store raw user handles/PII. Only hashed IDs.
- **Side-channel hygiene**:
  - Zeroize secrets and buffers.
  - Use constant-time memcmp, memcpy.
  - Avoid secret-dependent branching.
  - Keep sensitive ops inside CC310 hardware.
- **Firmware-level hardening**: disable debug, lock fuses, enforce secure boot.

13. Practical FIDO2 Fields Stored

- Resident keys (RKs).
- User handles (hashed only).
- LargeBlob extension data.
- PIN-stretched key material.
- Rate-limit counters.
- Superblock + index metadata.