Ledger – C Core Design Document (Phase 1)

The C core provides the cryptographic foundation of the Ledger system. It computes and verifies SHA-256 hashes for every event record stored in SQLite, ensuring data integrity and tamper detection.

Architecture Overview

File	Purpose
ledger.c	Implements hashing and verification logic.
ledger.h	Declares function prototypes and constants.

Compiled as a shared library (ledger.so), accessed from Python using ctypes.

Functional Design

compute_hash(const char* input)
- Input: concatenated string (timestamp|actor|action|details|prev_hash)
- Output: 64-character SHA-256 hex string
- Steps: receive input → compute digest → convert to hex → return hash

verify_hash(const char* input, const char* target)
- Recomputes hash and compares with target
- Returns 1 if identical, 0 if mismatch

Integration Flow

1. Flask collects event data
2. Flask sends concatenated data to C core
3. C core computes and returns hash
4. Flask stores hash in database
5. During verification, Flask recomputes hashes via C core and compares results

Security Notes

- Algorithm: SHA-256 (Phase 1 standard)
- Input validation: check for null or empty strings
- Memory safety: manage buffers securely
- Future: add post-quantum algorithms (Dilithium, Kyber)

Compilation

```
gcc -shared -o ledger.so -fPIC ledger.c
```

Then load in Python:

```
from ctypes import CDLL
lib = CDLL('./ledger.so')
```

Testing Plan

Test	Objective
Unit Test	Ensure compute_hash() is consistent.
Cross-Check	Compare with Python hashlib SHA-256.
Tamper Test	Modify data and ensure detection.
Integration Test	Verify Flask–C–SQLite interaction.

End of Document