RFC-003: Cryptographic Audit Trail

Status: Implemented **Date:** January 2026 **Author:** Derrell Piper ddp@eludom.net **Implementation:** audit.scm

---

Abstract

This RFC specifies the cryptographic audit trail system for the Library of Cyberspace, providing tamper-evident, hash-chained logging with SPKI principal attribution and Ed25519 signatures.

Motivation

Distributed systems require accountability. Who did what, when, and under whose authority?

Traditional logging fails on all counts: - **Tamperable:** Text files can be edited - **Anonymous:** No cryptographic identity - **Disconnected:** No provable ordering - **Unverifiable:** No mathematical proof of integrity

Cyberspace audit trails provide: 1. **Content-addressed entries** - Tamper-evident by hash 2. **Hash-chained structure** - Append-only ordering 3. **SPKI attribution** - Cryptographic actor identity 4. **Ed25519 seals** - Mathematical proof of authenticity 5. **Dual context** - Human-readable motivation + machine-parseable environment

---

Specification

Entry Structure

```
(audit-entry
  (id "sha512:b14471cd57ea557f...")
  (timestamp "Mon Jan 5 23:38:20 2026")
  (sequence 1)
  (parent-id "sha512:previous...")
  (actor
    (principal #${public-key-blob})
    (authorization-chain))
  (action
    (verb seal-publish)
    (object "1.0.0")
    (parameters "/path/to/remote"))
  (context
    (motivation "Published to filesystem")
    (language "en"))
  (environment
    (platform "darwin")
    (timestamp 1767685100))
  (seal
    (algorithm "ed25519-sha512")
    (content-hash "...")
    (signature "...")))
```

Core Fields

Field	Type	Description
id	string	Content-addressed hash (SHA-512, first 32 hex chars)
timestamp	string	Human-readable time
sequence	integer	Monotonic counter within audit trail
parent-id	string/nil	ID of previous entry (hash chain)
actor	record	SPKI principal who performed action
action	record	What was done (verb, object, parameters)
context	record	Human-readable motivation
environment	alist	Machine environment snapshot
seal	record	Cryptographic signature

Actor Record

```
(define-record-type <audit-actor>
  (make-audit-actor principal authorization-chain)
  audit-actor?
  (principal actor-principal)              ; Public key blob
  (authorization-chain actor-authorization-chain))  ; SPKI cert chain
```

The actor is identified by: - **Principal:** Ed25519 public key (32 bytes) - **Authorization chain:** Optional SPKI certificate chain proving delegation

Action Record

```
(define-record-type <audit-action>
  (make-audit-action verb object parameters)
  audit-action?
  (verb action-verb)        ; Symbol: seal-commit, seal-publish, etc.
  (object action-object)    ; Primary target
  (parameters action-parameters))  ; Additional arguments
```

Standard verbs: - `seal-commit` - Version control commit - `seal-publish` - Release publication - `seal-subscribe` - Subscription to remote - `seal-synchronize` - Bidirectional sync - `seal-release` - Version tagging

Context Record

```
(define-record-type <audit-context>
  (make-audit-context motivation relates-to language)
  audit-context?
  (motivation context-motivation)    ; Human explanation
  (relates-to context-relates-to)    ; Related entries
  (language context-language))       ; ISO 639-1 code
```

Context provides: - **Motivation:** Why the action was taken (human-readable) - **Relates-to:** Cross-references to related audit entries - **Language:** For internationalization

Seal Record

```
(define-record-type <audit-seal>
  (make-audit-seal algorithm content-hash signature)
  audit-seal?
  (algorithm seal-algorithm)        ; "ed25519-sha512"
  (content-hash seal-content-hash)  ; SHA-512 of unsealed entry
  (signature seal-signature))       ; Ed25519 signature
```

Operations

audit-init

Initialize audit trail for a vault.

```
(audit-init signing-key: key audit-dir: ".vault/audit")
```

audit-append

Create and sign a new audit entry.

```
(audit-append
  actor: public-key-blob
  action: '(seal-commit "hash123")
  motivation: "Added new feature"
  signing-key: private-key-blob)
```

Process: 1. Increment sequence counter 2. Get parent entry ID (hash chain link) 3. Build unsealed entry structure 4. Compute SHA-512 hash of canonical S-expression 5. Sign hash with Ed25519 6. Create seal record 7. Save entry to disk

audit-verify

Verify cryptographic seal on an entry.

```
(audit-verify entry public-key: key)
```

Verification steps: 1. Reconstruct unsealed entry 2. Compute SHA-512 hash 3. Compare with stored content-hash 4. Verify Ed25519 signature

audit-chain

Verify entire audit chain.

```
(audit-chain verify-key: public-key)
```

Verifies: - Each entry's signature is valid - Parent-id references form valid chain - Sequence numbers are monotonic

audit-read

Read specific audit entry.

```
(audit-read sequence: 42)
(audit-read id: "sha512:...")
```

Storage Format

Entries stored as individual S-expression files:

```
.vault/audit/
  1.sexp
  2.sexp
  3.sexp
  ...
```

File naming by sequence number enables efficient: - Sequential reads - Range queries - Latest entry lookup

---

Security Considerations

Threat Model

Trusted: - Local filesystem (during operation) - Ed25519 implementation (libsodium) - Private keys

Untrusted: - Storage medium (after creation) - Network transport - Other actors

Attack Mitigations

Attack	Mitigation
Entry modification	SHA-512 hash detects tampering
Entry deletion	Chain breaks are detectable
Entry insertion	Hash chain prevents backdating
Actor impersonation	Ed25519 signatures verify identity
Replay attacks	Sequence numbers detect duplicates

Non-Repudiation

Once an entry is signed and published: - Actor cannot deny performing the action - Timestamp cannot be backdated - Content cannot be altered - Signature mathematically proves authorship

---

Integration Points

Vault Operations

All vault operations record audit entries:

```
(seal-commit "message")     → (action (verb seal-commit) ...)
(seal-publish "1.0.0" ...)  → (action (verb seal-publish) ...)
(seal-subscribe remote ...) → (action (verb seal-subscribe) ...)
```

SPKI Authorization

Audit entries can include authorization chains:

```
(actor
  (principal #${bob-public-key})
  (authorization-chain
    (signed-cert ...)    ; Alice delegated to Bob
    (signed-cert ...))) ; Root delegated to Alice
```

This proves not just who acted, but under whose authority.

Export Formats

S-expression Export

```
(audit-export-sexp output: "audit-export.sexp")
```

Produces:

```
(audit-trail
  (audit-entry ...)
  (audit-entry ...)
  ...)
```

Human-readable Export

```
(audit-export-human output: "audit-export.txt")
```

Produces:

```
AUDIT TRAIL - Library of Cyberspace
===================================

Entry #1
  ID: sha512:b14471cd57ea557f...
  Time: Mon Jan 5 23:38:20 2026
  Action: seal-publish
  Why: Published release to filesystem

Entry #2
  ...
```

Implementation Notes

Dependencies

- `crypto-ffi` - Ed25519 signatures, SHA-512 hashing
- `srfi-1` - List utilities
- `srfi-4` - u8vectors for binary data
- `srfi-13` - String utilities

Performance Considerations

- Content-addressed IDs enable O(1) lookup by hash
- Sequential file naming enables efficient range queries
- Lazy verification: verify on read, not on load

References

1. Haber, S., & Stornetta, W. S. (1991). How to time-stamp a digital document.
2. Merkle, R. C. (1987). A digital signature based on a conventional encryption function.
3. Bernstein, D. J. (2006). Curve25519: new Diffie-Hellman speed records.
4. SPKI/SDSI - RFC 2693, RFC 2692

Changelog

- **2026-01-06** - Initial specification

Implementation Status: Complete **Test Status:** Passing (test-audit.scm) **Integration:** Vault operations fully audited