

# DEFENSIVE PUBLICATION

## Sovereign: A Local Architecture for Creator IP Protection

Author: Wayne Gordon

Date of Publication: May 2026

Version 4.2 — Adds AIP-7 alignment, updates brand to Alterna Axiom Labs

Alterna Axiom Labs · [contact@alternaaxiomlabs.com](mailto:contact@alternaaxiomlabs.com)

### Abstract

Sovereign is a local-first architecture designed to protect digital intellectual property by establishing cryptographic identity, artifact lineage, and verifiable authorship proof without reliance on centralized services or cloud infrastructure.

Version 4.2 adds alignment documentation with the AIP-7 agentgovernance/v1 open standard for append-only hash-chained audit logs, establishing that Sovereign's provenance chain architecture predates and independently implements the same core primitives. Brand updated to Alterna Axiom Labs.

All prior elements remain: SHA3-256 quantum-resilient fingerprinting, single deterministic provenance engine, frequency-domain watermarking, vault capacity enforcement, offline HMAC-SHA256 licensing, C2PA alignment, and portable proof bundle export.

### 1. Problem Statement

Digital creators currently lack reliable tools to demonstrate authorship and creation timelines. Existing solutions rely on centralized cloud services, copyright registries, email timestamps, blockchain systems, and platform-specific logs. These approaches depend on third-party infrastructure and rarely capture the complete lifecycle of an artifact.

Metadata-based watermarking (EXIF, XMP, ID3) can be stripped by standard image processing operations including resizing, format conversion, and social media upload pipelines. A more durable approach embeds authorship evidence directly in the frequency-domain coefficients of the image data, surviving transformations that remove or ignore metadata fields.

Current cryptographic fingerprinting in IP protection tools relies on SHA-256, which faces theoretical vulnerability to quantum computing attacks. A quantum-resilient alternative based on the SHA-3 (Keccak) family provides forward-looking security for authorship claims that may need to be verified years or decades after creation.

### 2. Proposed Solution

Sovereign introduces a creator-side evidence architecture recording artifact identity, lineage, and verification evidence locally. Key characteristics include: local processing, deterministic artifact identity via SHA3-256 (quantum-resilient), append-only provenance history with hash-chained entries, optional immutable artifact archiving, deterministic provenance verification, exportable verification bundles, frequency-domain image watermarking, vault capacity enforcement, and offline HMAC-signed license verification.

### 3. System Architecture

The architecture processes artifacts through a sequential six-stage pipeline:

Stage	Component	Function
1. Intake	Creator Artifact	Digital file enters via filesystem watcher or manual import
2. Integrity	Provenance Engine	File validation, format detection, integrity baseline
3. Identity	SHA3-256 Hash	Quantum-resilient cryptographic fingerprint generation
4. Watermark	Dual Watermark	Metadata embed plus frequency-domain watermark
5. Logging	Provenance Log	Append-only event recording in local SQLite database
6. Export	Proof Bundle	Portable verification package generation

Table 1: Pipeline stage descriptions

## 4. Provenance Engine Architecture

Version 4.0 consolidated the prior dual-agent architecture into a single provenance engine performing all functions: artifact intake, integrity validation, fingerprint generation, watermark embedding, provenance log creation, and provenance verification.

### 4.1 Provenance Engine

Responsible for the complete artifact protection pipeline: intake, SHA3-256 fingerprint generation, metadata and frequency-domain watermark embedding, append-only provenance logging, and deterministic verification assessment. All operations are deterministic.

### 4.2 Deterministic Verification

The provenance verification step generates a structured assessment confirming: artifact identity (SHA3-256 fingerprint), watermark status, vault artifact count, and provenance chain integrity. No AI inference is involved.

### 4.3 Vault

Provides the provenance database via local SQLite. Enforces append-only semantics on the provenance log. All timestamps use ISO 8601 format with explicit timezone offsets.

### 4.4 Provenance Integrity — Hash Chain

Each provenance entry includes a SHA3-256 hash of the previous entry, forming a hash chain. Any modification or deletion within the log invalidates all subsequent hashes, making tampering detectable.

## 5. Artifact Identity — Quantum-Resilient Fingerprinting

SHA3-256 (Keccak, FIPS 202) serves as the primary cryptographic fingerprint algorithm. The Keccak sponge construction is structurally distinct from SHA-256's Merkle-Damgard construction, providing defense against potential quantum attacks. The fingerprint is computed using the sha3 crate (Rust). The verify.py script uses Python's hashlib.sha3\_256().

## 6. Embedded Watermarking

Sovereign embeds authorship metadata into artifact files using format-appropriate methods.

Format	Method	Status Label
.wav, .mp3	ID3 tag metadata	ID3 Watermarked (WAV)
.flac, .aiff, .ogg	Vorbis comments / ID3 via lofty	ID3 Watermarked (FLAC)
.jpg, .tiff, .webp	EXIF metadata + frequency domain	Frequency Watermarked (JPG)
.pdf	Document info dictionary via lopdf	XMP Watermarked (PDF)
.docx	Custom XML property injection	Custom XML Watermarked (DOCX)
Other formats	Sidecar .sovereign file	Sidecar Watermarked

Table 2: Watermarking methods by format

## 7. Frequency-Domain Watermarking

Embeds authorship evidence in DCT coefficient domain of image data. The watermark is derived from the artifact's SHA3-256 fingerprint and embedded in mid-frequency DCT coefficients. A content-hash deduplication guard with

RAII Drop semantics prevents redundant application.

## 8. Vault Capacity Model

Hard vault capacity limit of 20 protected files per vault, enforced at the Rust processing pipeline level prior to any computational work. Revisions to already-protected files bypass the capacity check.

## 9. Offline Per-Vault Licensing

Offline licensing architecture using HMAC-SHA256 signed license files. License resolution checks executable directory first (enabling USB deployment), then application data directory. Cumulative vault count model with 999+ treated as unlimited.

Field	Type	Description
product	string	Product identifier — prevents cross-product signature reuse
version	integer	License format version
customer	string	Normalized customer email (lowercase, trimmed)
vaults	integer	Number of vaults allowed (999+ = unlimited)
issued	string	ISO 8601 date of issuance
hmac	string	Base64-encoded HMAC-SHA256 signature

Table 3: License file fields

## 10. C2PA Alignment

Sovereign's architecture overlaps significantly with C2PA concepts while operating entirely locally.

C2PA Concept	Sovereign Implementation
Content Credential	Proof bundle — portable, self-contained, independently verifiable
Claim	Provenance log entry — typed event with timestamp and hash chain link
Claim Generator	Provenance engine — generates intake, watermark, and verification events
Hard Binding	Content hash (SHA3-256) binding provenance record to artifact
Soft Binding	Frequency-domain watermark surviving metadata stripping
Manifest Store	SQLite provenance database — append-only, hash-chained
Trust Signal	Proof bundle with verify.py — zero external dependencies

Table 4: C2PA alignment

## 11. AIP-7 (agentgovernance/v1) Alignment

The AIP-7 specification ([agentproto.sh/docs/aip-7](https://agentproto.sh/docs/aip-7)) defines an open standard for append-only hash-chained audit logs, approval signatures, and declarative governance policies for AI agent systems. Sovereign's provenance architecture independently implements several core primitives defined in AIP-7, predating the standard's formalization.

### 11.1 Conceptual Alignment

AIP-7 Primitive	Sovereign Implementation
audit-event (append-only hash-chained log)	Provenance log entry — each includes SHA3-256 hash of previous entry
Hash chain verification protocol	verify.py — recomputes chain hashes with zero dependencies
Append-only invariant	SQLite provenance log enforces insert-only semantics
Third-party verifiability	Proof bundle export — portable ZIP with self-verification
Filesystem-first artifact format	Proof bundle as ZIP of plain files

Table 5: AIP-7 agentgovernance/v1 alignment

### 11.2 Distinctions

AIP-7 additionally defines signature primitives (cryptographic approval events), declarative policy engines, and workspace composition for multi-agent governance. Sovereign does not implement these elements, as its architecture serves single-user creator protection rather than multi-agent coordination. However, Sovereign's provenance chain could be exported in AIP-7 audit-event JSONL format with a straightforward mapping.

### 11.3 Prior Art Significance

Sovereign's append-only hash-chained provenance log was implemented and documented in Defensive Publication v1.0 (DOI 10.5281/zenodo.19056811) prior to the publication of AIP-7. This establishes independent prior art for the core audit-chain primitive that AIP-7 subsequently formalized as a standard.

## 12. Artifact Lineage and Version History

Artifacts maintain complete version history. Events generating version entries include: initial intake, provenance verification, file rename detection, content modification, and Do Not Train flag changes. File renames are detected by hash comparison.

### 13. Layered Evidence Model

Layer	Mechanism	Purpose
1	Embedded Watermark (metadata)	Authorship metadata inside the artifact file
2	Frequency-Domain Watermark	Authorship evidence in DCT coefficients
3	Cryptographic Fingerprint	Quantum-resilient identity via SHA3-256
4	Append-Only Provenance Log	Immutable timeline with ISO 8601 timestamps and hash chain
5	Portable Proof Bundle	Self-contained, independently verifiable package

Table 6: Five-layer evidence model

### 14. Proof Bundle

File	Contents
artifact_file	Original digital artifact
fingerprint.txt	SHA3-256 hash
provenance_export.json	Full provenance chain with typed events and ISO 8601 timestamps
chain_hash.txt	Tamper-evidence hash of the provenance log
verify.py	Self-verification script — Python 3, zero external dependencies
verify.bat	Windows convenience wrapper
provenance_report.pdf	Human-readable provenance report

Table 7: Proof bundle contents

### 15. Do Not Train (DNT) Flag

Each artifact supports a Do Not Train boolean flag. DNT state changes are logged as immutable provenance events with timestamps. The DNT flag and its complete history are included in proof bundle exports.

### 16. Portable Cold Storage Operation

The Sovereign architecture supports deployment as a fully portable application on removable storage media. Each device operates as an isolated, independent evidence vault.

## 17. Demonstration Environment

Component	Specification
Processor	Intel Core i9
Memory	32 GB RAM
GPU	None required
Framework	Tauri v2 (Rust backend, React/TypeScript frontend)
Database	SQLite (local, single-file, append-only provenance log)
Hashing	SHA3-256 (Keccak, FIPS 202) via sha3 crate
Watermarking	lofty crate (audio/image), lopdf (PDF), XML injection (DOCX), DCT (images)
Licensing	HMAC-SHA256 offline license file, no server required
Development time	Approximately four months

Table 8: Prototype implementation specifications

## 18. Defensive Disclosure

This document constitutes a public defensive disclosure of the Sovereign architecture. Its publication establishes prior art for systems combining the following elements:

- Local-only artifact identity generation using quantum-resilient cryptographic hashing (SHA3-256, Keccak/FIPS 202)
- Append-only provenance logging with ISO 8601 timestamps and timezone offsets
- Hash-chained provenance entries using SHA3-256 where each entry includes a cryptographic hash of the previous entry, making tampering detectable
- Single deterministic provenance engine performing integrity verification, fingerprinting, watermarking, and verification without AI inference
- Deterministic provenance verification generating structured assessments from cryptographic facts without language model inference
- Optional immutable artifact archiving with version lineage preservation and rename tracking
- Metadata-embedded watermarking across multiple format families (audio, image, video, document) with sidecar fallback
- Frequency-domain watermarking using DCT coefficient embedding for image formats
- Content-hash deduplication guard with RAI Drop semantics preventing redundant pipeline execution
- Vault capacity enforcement at the Rust processing pipeline level — hard limit prior to any computational work
- Offline per-vault licensing using HMAC-SHA256 signed license files with no server, network, or activation service
- License file resolution checking executable directory before application data directory, enabling portable deployment
- Product-prefixed HMAC signed strings preventing cross-product signature reuse
- Cumulative vault count model — single license file encodes total authorized vaults, replacing prior license on upgrade
- Portable proof bundle export with self-verification script and tamper-evident chain hash

- Do Not Train (DNT) flag with immutable provenance audit trail
- Portable cold storage operation on removable media with isolated evidence vaults
- Local-first C2PA-aligned provenance architecture operating without centralized signing infrastructure
- Complete local operation without reliance on cloud services, blockchain, AI models, or centralized infrastructure
- Migration from SHA-256 to SHA3-256 as a quantum-resilience measure for long-term artifact identity preservation
- Independent prior art for the append-only hash-chained audit-event primitive subsequently formalized in the AIP-7 agentgovernance/v1 open standard

Publication of this architecture is intended to prevent future patent claims covering the same or substantially similar system design. Any patent application filed after the date of this publication that claims the above combination of elements may be challenged on the basis of this prior art disclosure.

### **Sovereign: A Local Architecture for Creator IP Protection**

Defensive Publication v4.2 · Wayne Gordon · May 2026

Alterna Axiom Labs · [contact@alternaaxiomlabs.com](mailto:contact@alternaaxiomlabs.com)

*This document is released for the purpose of establishing prior art.*

Original publication (v1.0): DOI 10.5281/zenodo.19056811