SQEP-Zero: A Cryptographic Standard for Verifiable Proof of Digital Actions

Version 1.0 — Frozen Standard Genesis Hash: `858a81309f473dacc70e4a94b21a09a6d56241ae810ee6b9308e6a49d00038b7` **Date:** January 31, 2026 **Author:** ElevitaX

Executive Summary

Every day, billions of digital actions occur—files are accessed, data is modified, decisions are made by AI systems, and sensitive information changes hands. Yet most of these actions leave no verifiable trace. When disputes arise, organizations cannot prove what happened, when it happened, or who was responsible.

SQEP-Zero (Secure Quantum Encryption Protocol — Zero) is an open cryptographic standard that solves this problem. It creates **mathematical proof** that a specific action occurred on data, without exposing the data itself.

The core innovation is simple but powerful: **every action on your data deserves a receipt.**

Unlike blockchain systems that require consensus networks and cryptocurrency, SQEP-Zero operates independently. Unlike traditional logging that can be altered, SQEP-Zero creates an immutable chain of cryptographic proofs that anyone can verify using only mathematics—no trust in the issuer required.

The Problem

1. Actions Are Invisible

When you access a file, modify a database, or run an AI model on sensitive data, what proof exists? Log files can be deleted. Timestamps can be forged. Audit trails can be altered by anyone with system access.

2. Trust Is Required

Current systems require you to trust: - The software vendor - The system administrator - The cloud provider - The auditor

If any of these parties is compromised, malicious, or simply makes a mistake, your "proof" becomes worthless.

3. Privacy vs. Proof Trade-off

Traditional audit systems face a dilemma: to prove an action occurred, they often must store sensitive data. This creates GDPR liability, security risks, and privacy violations.

4. Verification Is Expensive

Proving what happened typically requires hiring auditors, lawyers, or forensic specialists. Small businesses and individuals cannot afford this verification.

The SQEP-Zero Standard

SQEP-Zero is defined by **10 immutable principles**:

#	Principle
1	**SQEP-Zero** is an open cryptographic standard for producing verifiable proof of digital actions.
2	A **Receipt** is a cryptographic record proving that a specific action occurred on data.
3	Each Receipt contains: action type, timestamp, actor reference, object reference, and a SHA-256 hash.
4	The **HashChain** links each Receipt to the previous one via `previous_hash`, forming an immutable chain.
5	The **Genesis Receipt** is the origin of any HashChain—it has no predecessor (`previous_hash = null`).
6	Chain integrity is verified by recalculating all hashes sequentially from Genesis to head.
7	SQEP-Zero hashes **actions**, never raw data—ensuring privacy, universality, and GDPR compliance.
8	Verification is public and requires no trust in the issuer—only the mathematical chain.
9	The standard is **use-case agnostic**: it works for files, AI decisions, compliance, legal proof, or any domain.
10	**Every action on your data deserves a receipt.**

Core Principle

> **"Security is a promise. Proof is a fact."**

How It Works

The Receipt

A Receipt is the atomic unit of proof. Each Receipt captures:

```
┌──────────────────────────────────────────────────────────┐
│                       SQEP RECEIPT                       │
├──────────────────────────────────────────────────────────┤
│  Timestamp:    1706745600 (Unix seconds)                 │
│  Action:       FILE_ENCRYPTED                            │
│  Message:      "file_id=doc-2024-001 algo=AES-256-GCM"   │
│  Prev Hash:    64f9943a2b...                             │
│  This Hash:    7c8d1e2f3a...                             │
└──────────────────────────────────────────────────────────┘
```

Key insight: The Receipt contains a description of the action, not the data itself. You can prove "this file was encrypted at this time" without revealing the file contents.

The HashChain

Receipts are linked cryptographically:

```
┌──────────┐     ┌──────────┐     ┌──────────┐     ┌──────────┐
│ GENESIS  │────►│ Receipt  │────►│ Receipt  │────►│ Receipt  │
│          │     │    #2    │     │    #3    │     │    #4    │
│ hash: A  │     │ prev: A  │     │ prev: B  │     │ prev: C  │
│          │     │ hash: B  │     │ hash: C  │     │ hash: D  │
└──────────┘     └──────────┘     └──────────┘     └──────────┘
```

Each Receipt's hash is computed from:

```
hash = SHA-256(timestamp | action | message | previous_hash)
```

This creates a **tamper-evident chain**: modifying any Receipt breaks all subsequent hashes.

Verification

Anyone can verify the entire chain:

1. Start at Genesis (known, published hash)
2. For each Receipt:
3. Recompute the hash from its fields
4. Confirm it matches the stored hash
5. Confirm `previous_hash` matches the prior Receipt
6. If all checks pass, the chain is intact

No trust required. No special software. No permission from the issuer. Just mathematics.

Why Not Blockchain?

Aspect	Blockchain	SQEP-Zero
Consensus required	Yes (PoW/PoS)	No
Cryptocurrency needed	Usually	No
Energy consumption	High	Minimal
Latency	Minutes to hours	Milliseconds
Complexity	High	Low
Offline operation	No	Yes
Single-organization use	Impractical	Ideal

SQEP-Zero is designed for **organizational sovereignty**: you control your own chain, you issue your own receipts, and anyone can verify them independently.

Use Cases

1. GDPR Compliance

Problem: Under GDPR, you must prove you handled personal data correctly.

Solution: Every data access, modification, or deletion generates a Receipt. When audited, you provide the HashChain as mathematical proof of compliance.

2. AI Decision Auditing

Problem: AI systems make decisions affecting people's lives (loans, hiring, medical diagnoses). Regulators demand explainability.

Solution: Every AI inference generates a Receipt linking the input hash, model version, and output. The decision trail is immutable and verifiable.

3. Legal Document Integrity

Problem: Contracts, evidence, and legal documents can be altered after signing.

Solution: Hash the document at creation and record a Receipt. Any subsequent version can be verified against the original hash.

4. Supply Chain Provenance

Problem: Proving a product's origin and handling history.

Solution: Each handoff generates a Receipt. The complete chain proves provenance from source to destination.

5. Financial Audit Trail

Problem: Transaction logs can be manipulated.

Solution: Every transaction generates a Receipt. Auditors verify the chain mathematically, not through trust.

Technical Specification

Receipt Schema (JSON)

```
{
  "ts_unix": 1769832901,
  "tag": "FILE_ENCRYPTED",
  "message": "file_id=doc-2024-001 algo=AES-256-GCM actor=user@example.com",
  "prev_hash": "64f9943a2b8c1d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a9b0c1d2e3f4a5b6c7d8e",
  "hash": "7c8d1e2f3a4b5c6d7e8f9a0b1c2d3e4f5a6b7c8d9e0f1a2b3c4d5e6f7a8b9c0d"
}
```

Hash Computation

```
content = "{ts_unix}|{tag}|{message}|{prev_hash}"
hash = hex(SHA-256(UTF-8(content)))
```

Example:

```
content = "1769832901|FILE_ENCRYPTED|file_id=doc-2024-001|64f9943a2b..."
hash = SHA-256(content) → "7c8d1e2f3a..."
```

Storage Format

JSONL (JSON Lines): one Receipt per line, append-only.

```
{"ts_unix":
1769832901,"tag":"CHAIN_GENESIS","message":"...","prev_hash":"","hash":"858a813..."}
{"ts_unix":
1769832952,"tag":"FILE_ENCRYPTED","message":"...","prev_hash":"858a813...","hash":"64
f9943..."}
```

Verification Algorithm (Pseudocode)

```
function verify_chain(receipts):
    prev_hash = ""
    for receipt in receipts:
        expected = sha256(receipt.ts_unix | receipt.tag | receipt.message | receipt.p
rev_hash)
        if expected != receipt.hash:
            return INVALID("hash mismatch at receipt")
        if receipt.prev_hash != prev_hash:
            return INVALID("chain break at receipt")
        prev_hash = receipt.hash
    return VALID
```

API Endpoints

Endpoint	Method	Description
`/v1/journal/append`	POST	Add a new Receipt
`/v1/journal/verify`	GET	Verify entire chain
`/v1/journal/head`	GET	Get latest Receipt hash
`/v1/journal/tail?n=10`	GET	Get last N Receipts

Supported Action Types

SQEP-Zero is action-agnostic, but common types include:

- `CHAIN_GENESIS` — Chain origin
- `FILE_CREATE`, `FILE_OPEN`, `FILE_SAVE`, `FILE_DELETE`
- `FILE_ENCRYPTED`, `FILE_DECRYPTED`
- `POLICY_APPLIED`, `POLICY_REVOKED`
- `USER_LOGIN`, `USER_LOGOUT`
- `AI_INFERENCE`, `AI_DECISION`
- `DATA_EXPORT`, `DATA_IMPORT`
- `CONSENT_GRANTED`, `CONSENT_REVOKED`

Security Properties

Tamper Evidence

Any modification to a Receipt invalidates its hash and all subsequent hashes. Tampering is mathematically detectable.

Forward Secrecy

Old Receipts cannot be forged without knowing the chain state at that point. The chain grows forward only.

Privacy by Design

Receipts contain action descriptions, not raw data. You prove "file X was accessed" without revealing file X's contents.

Offline Operation

No network required. The chain exists locally. Verification requires only the chain data and a SHA-256 implementation.

Post-Quantum Considerations

SHA-256 remains secure against known quantum attacks (Grover's algorithm provides only quadratic speedup). Future versions may adopt SHA-3 or quantum-resistant signatures for additional assurance.

Implementation

Reference Implementation

The reference implementation is written in Rust, chosen for:

- Memory safety without garbage collection
- Zero-cost abstractions
- No runtime dependencies
- Cross-platform support (Linux, Windows, macOS, ARM, Android)

Metrics: - Binary size: ~15 MB - Memory footprint: < 50 MB typical - Test coverage: 1477 passing tests - External dependencies: Minimal (no LLM, no GPU required)

Open Source

SQEP-Zero is an open standard. The specification is freely available. Reference implementations will be published under permissive licenses.

The Genesis Receipt

The SQEP-Zero standard was frozen on January 31, 2026, with the following Genesis Receipt:

```
{
  "ts_unix": 1769832901,
  "tag": "CHAIN_GENESIS",
  "message": "SQEP-Zero HashChain v1.0 Genesis | authority=ElevitaX | actor=SYSTEM |
spec_version=1.0 | standard=SQEP-Zero | chain_id=SQEP-ELEVITAX-PROD-V1",
  "prev_hash": "",
  "hash": "858a81309f473dacc70e4a94b21a09a6d56241ae810ee6b9308e6a49d00038b7"
}
```

This hash serves as the **anchor** for the ElevitaX production chain. Anyone can verify that a chain traces back to this Genesis.

AI Runtime: Cryptographic Intelligence

The Vision

SQEP-Zero is not just a receipt system—it is a **Cryptographic AI Runtime**. Every AI decision, every inference, every reasoning step is captured in the HashChain, creating AI systems that can mathematically prove their behavior.

> **Conceptual Analogy** Bitcoin's core insight was that you don't need to trust an institution to verify a transaction—you need a mathematical proof that it occurred. SQEP-Zero applies that same insight to digital actions: cryptographic receipts that can be independently verified by anyone, without trusting the issuer.
>
> *This is not a decentralized currency. It is a cryptographically verifiable receipt protocol, grounded in well-studied primitives (SHA-256, HMAC, AEAD).*

Processing Architecture

The AI Runtime implements a deterministic processing engine:

```
┌─────────────────────────────────────────────────────────────────┐
│                    SQEP-Zero AI Runtime                          │
├─────────────────────────────────────────────────────────────────┤
│                                                                  │
│   ┌──────────────┐    ┌──────────────┐    ┌──────────────┐      │
│   │   Quality    │    │    Speed     │    │   Offline    │      │
│   │    Tier      │    │    Tier      │    │    Tier      │      │
│   │              │    │              │    │              │      │
│   │   Ollama     │    │ Candle+CUDA  │    │ Candle+CPU   │      │
│   │  (Remote)    │    │   (Local)    │    │   (Local)    │      │
│   │              │    │              │    │              │      │
│   │   ~2-5s      │    │   ~3-8s      │    │   ~60-120s   │      │
│   └──────────────┘    └──────────────┘    └──────────────┘      │
│          │                   │                   │               │
│          └───────────────────┼───────────────────┘               │
│                              │                                    │
│                    ┌─────────▼─────────┐                         │
│                    │  Smart Router     │                         │
│                    │  (Automatic       │                         │
│                    │   Selection)      │                         │
│                    └─────────┬─────────┘                         │
│                              │                                    │
│                    ┌─────────▼─────────┐                         │
│                    │    HashChain      │                         │
│                    │    (Every         │                         │
│                    │    inference      │                         │
│                    │    logged)        │                         │
│                    └───────────────────┘                         │
│                                                                   │
└─────────────────────────────────────────────────────────────────┘
```

Three-Tier LLM Inference

Tier	Engine	Latency	Use Case
Quality	Ollama (Gemma 2B, Llama)	~2-5s	Complex reasoning, long context
Speed	Candle + CUDA GPU	~3-8s	Real-time interaction, local privacy
Offline	Candle + CPU	~60-120s	Air-gapped systems, no network

The Smart Router automatically selects the optimal tier based on: - GPU availability (CUDA detection) - Network connectivity (Ollama health check) - Task complexity (token count, context length) - Privacy requirements (local-only mode)

GPU Acceleration

The Speed tier leverages CUDA for significant performance gains:

- **28x speedup** on RTX 4060 vs CPU-only inference
- Quantized models (GGUF format) for memory efficiency

- Automatic fallback to CPU if GPU unavailable

AI Receipt Generation

Every AI operation generates a cryptographic receipt:

```
{
  "ts_unix": 1769833100,
  "tag": "AI_INFERENCE",
  "message": "model=gemma-2b tier=speed tokens_in=128 tokens_out=64 latency_ms=3200",
  "prev_hash": "7c8d1e2f3a...",
  "hash": "a1b2c3d4e5..."
}
```

This enables: - **Auditable AI**: Prove which model made which decision - **Reproducibility**: Track exact versions and parameters - **Compliance**: Demonstrate AI governance to regulators - **Cost tracking**: Monitor resource usage per inference

Supported Models

Model	Parameters	Format	Recommended Tier
Gemma 2B	2B	GGUF Q4	Speed (GPU)
TinyLlama	1.1B	GGUF Q4	Offline (CPU)
Llama 3.2	1B-3B	Native	Quality (Ollama)

Offline-First Design

The AI Runtime operates without internet connectivity: - All models can run locally - No API calls to external services required - HashChain verification is entirely local - Ideal for air-gapped, secure, or remote environments

---

Conclusion

SQEP-Zero transforms digital accountability from a matter of trust to a matter of mathematics.

- **For organizations:** Prove compliance, demonstrate integrity, reduce audit costs.
- **For individuals:** Know that actions on your data are recorded and verifiable.
- **For regulators:** Verify claims independently, without trusting the regulated party.

- **For developers:** A simple, well-defined standard that integrates into any system.

The digital world generates billions of actions daily. Most leave no verifiable trace. SQEP-Zero changes this by providing a universal, open, and mathematically sound method for proving that actions occurred.

Every action on your data deserves a receipt.

Contact & Resources

- **Organization:** ElevitaX
- **Standard Version:** 1.0 (Frozen)
- **Genesis Hash:** `858a81309f473dacc70e4a94b21a09a6d56241ae810ee6b9308e6a49d00038b7`

Appendix A: Glossary

Term	Definition
Receipt	A cryptographic record proving an action occurred
HashChain	A sequence of Receipts linked by cryptographic hashes
Genesis	The first Receipt in a HashChain (no predecessor)
SHA-256	The hash algorithm used (256-bit output)
JSONL	JSON Lines format (one JSON object per line)
Tamper-evident	Any modification is mathematically detectable

Appendix B: Verification Example

Given this chain:

```
Receipt 1 (Genesis):
  ts_unix: 1769832901
  tag: CHAIN_GENESIS
  message: "SQEP-Zero HashChain v1.0 Genesis..."
  prev_hash: ""
  hash: 858a81309f473dacc70e4a94b21a09a6d56241ae810ee6b9308e6a49d00038b7

Receipt 2:
  ts_unix: 1769832952
  tag: CHAIN_INITIALIZED
  message: "HashChain v1 initialized"
  prev_hash: 858a81309f473dacc70e4a94b21a09a6d56241ae810ee6b9308e6a49d00038b7
  hash: [computed]
```

Verification steps:

1. Compute: `SHA-256("1769832901|CHAIN_GENESIS|SQEP-Zero HashChain...|")`
2. Verify result equals `858a813...`
3. Compute: `SHA-256("1769832952|CHAIN_INITIALIZED|HashChain v1 initialized|858a813...")`
4. Verify result equals Receipt 2's hash
5. Verify Receipt 2's `prev_hash` equals Receipt 1's `hash`

If all checks pass, the chain is mathematically verified.
