SQEP-Zero : Standard Cryptographique pour la Preuve Vérifiable des Actions Numériques

Version 1.0 — Standard Figé Hash Genesis : `858a81309f473dacc70e4a94b21a09a6d56241ae810ee6b9308e6a49d00038b7` **Date :** 31 janvier 2026 **Auteur :** ElevitaX — Herbert Manfred Fulgence Vaty **Licence :** MIT (Standard ouvert)

Résumé Exécutif

Chaque jour, des milliards d'actions numériques se produisent : des fichiers sont consultés, des données sont modifiées, des décisions sont prises par des systèmes d'IA, et des informations sensibles changent de mains. Pourtant, la plupart de ces actions ne laissent aucune trace vérifiable.

SQEP-Zero (Secure Quantum Encryption Protocol — Zero) est un standard cryptographique ouvert qui résout ce problème. Il crée une **preuve mathématique** qu'une action spécifique s'est produite sur des données, sans exposer les données elles-mêmes.

L'innovation centrale est simple mais puissante : **chaque action sur vos données mérite un reçu.**

Contrairement aux blockchains qui nécessitent des réseaux de consensus et des cryptomonnaies, SQEP-Zero fonctionne de manière indépendante. Contrairement aux journaux traditionnels qui peuvent être altérés, SQEP-Zero crée une chaîne immuable de preuves cryptographiques que n'importe qui peut vérifier en utilisant uniquement les mathématiques — aucune confiance dans l'émetteur n'est requise.

> **« La sécurité est une promesse. La preuve est un fait. »**

Le Problème

1. Les Actions Sont Invisibles

Lorsque vous accédez à un fichier, modifiez une base de données ou exécutez un modèle d'IA sur des données sensibles, quelle preuve existe ? Les fichiers journaux peuvent être supprimés. Les horodatages peuvent être falsifiés. Les pistes d'audit peuvent être modifiées par quiconque ayant accès au système.

2. La Confiance Est Requise

Les systèmes actuels exigent que vous fassiez confiance à : - L'éditeur du logiciel - L'administrateur système - Le fournisseur cloud - L'auditeur

Si l'une de ces parties est compromise, malveillante ou fait simplement une erreur, votre « preuve » devient sans valeur.

3. Le Dilemme Vie Privée vs Preuve

Les systèmes d'audit traditionnels font face à un dilemme : pour prouver qu'une action s'est produite, ils doivent souvent stocker des données sensibles. Cela crée une responsabilité RGPD, des risques de sécurité et des violations de la vie privée.

4. La Vérification Est Coûteuse

Prouver ce qui s'est passé nécessite généralement d'engager des auditeurs, des avocats ou des spécialistes forensiques. Les petites entreprises et les particuliers ne peuvent pas se permettre cette vérification.

Le Standard SQEP-Zero

SQEP-Zero est défini par **10 principes immuables** :

#	Principe
1	**SQEP-Zero** est un standard cryptographique ouvert pour produire des preuves vérifiables d'actions numériques.
2	Un **Reçu** est un enregistrement cryptographique prouvant qu'une action spécifique s'est produite sur des données.
3	Chaque Reçu contient : type d'action, horodatage, référence de l'acteur, référence de l'objet et un hash SHA-256.
4	La **HashChain** relie chaque Reçu au précédent via `previous_hash`, formant une chaîne immuable.
5	Le **Reçu Genesis** est l'origine de toute HashChain — il n'a pas de prédécesseur (`previous_hash = null`).
6	L'intégrité de la chaîne est vérifiée en recalculant tous les hashs séquentiellement du Genesis à la tête.
7	SQEP-Zero hashe les **actions**, jamais les données brutes — garantissant confidentialité, universalité et conformité RGPD.
8	La vérification est publique et ne requiert aucune confiance dans l'émetteur — seulement la chaîne mathématique.
9	Le standard est **agnostique au cas d'usage** : il fonctionne pour les fichiers, les décisions IA, la conformité, les preuves légales ou tout autre domaine.
10	**Chaque action sur vos données mérite un reçu.**

Comment Ça Fonctionne

Le Reçu

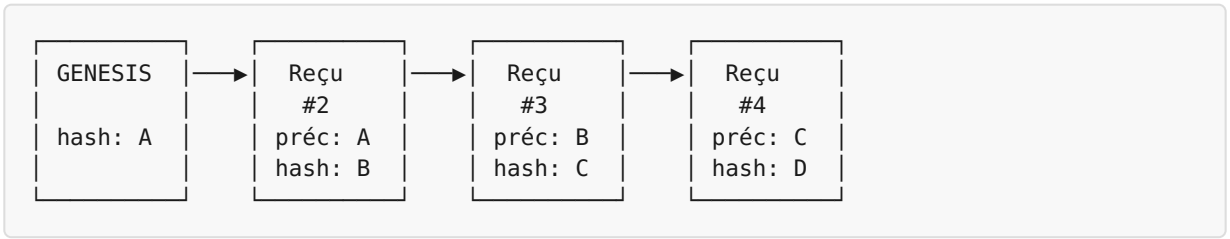
Un Reçu est l'unité atomique de preuve. Chaque Reçu capture :

```
┌──────────────────────────────────────────────────┐
│                    REÇU SQEP                     │
├──────────────────────────────────────────────────┤
│  Horodatage:  1706745600 (Unix secondes)         │
│  Action:      FICHIER_CHIFFRÉ                    │
│  Message:     "file_id=doc-2024-001 algo=AES-256-GCM"  │
│  Hash Préc.:  64f9943a2b...                      │
│  Hash Actuel: 7c8d1e2f3a...                      │
└──────────────────────────────────────────────────┘
```

Point clé : Le Reçu contient une description de l'action, pas les données elles-mêmes. Vous pouvez prouver « ce fichier a été chiffré à ce moment » sans révéler le contenu du fichier.

La HashChain

Les Reçus sont liés cryptographiquement :



Chaque hash de Reçu est calculé à partir de :

```
hash = SHA-256(horodatage | action | message | hash_précédent)
```

Cela crée une **chaîne inviolable** : modifier un Reçu invalide tous les hashs suivants.

Vérification

N'importe qui peut vérifier la chaîne entière :

1. Commencer au Genesis (hash connu, publié)
2. Pour chaque Reçu :
3. Recalculer le hash à partir de ses champs
4. Confirmer qu'il correspond au hash stocké
5. Confirmer que `previous_hash` correspond au Reçu précédent
6. Si toutes les vérifications passent, la chaîne est intacte

Aucune confiance requise. Pas de logiciel spécial. Pas de permission de l'émetteur. Juste des mathématiques.

Pourquoi Pas Blockchain ?

Aspect	Blockchain	SQEP-Zero
Consensus requis	Oui (PoW/PoS)	Non
Cryptomonnaie nécessaire	Généralement	Non
Consommation d'énergie	Élevée	Minimale
Latence	Minutes à heures	Millisecondes
Complexité	Élevée	Faible
Fonctionnement hors-ligne	Non	Oui
Usage mono-organisation	Impraticable	Idéal

SQEP-Zero est conçu pour la **souveraineté organisationnelle** : vous contrôlez votre propre chaîne, vous émettez vos propres reçus, et n'importe qui peut les vérifier indépendamment.

Encadré Conceptuel : L'Analogie Bitcoin

Cette section est une aide à la compréhension, pas une équivalence technique. Bitcoin résout le double-dépense dans un réseau décentralisé global. SQEP-Zero résout la preuve d'action dans un contexte cryptographique local/semi-distribué. Domaines différents, même principe fondateur.

Pour comprendre la portée de SQEP-Zero, une analogie s'impose :

Ce que Bitcoin a fait pour la Finance

Avant Bitcoin, pour envoyer de l'argent, il fallait une banque. La banque était la « source de vérité ». Mais la banque pouvait mentir, geler des comptes ou dire « la transaction n'a jamais eu lieu ».

Bitcoin a remplacé la confiance institutionnelle par la vérification mathématique.

Domaine	Avant	Après
Finance	« Faites confiance à la banque »	« Vérifiez la preuve de transaction »

Ce que SQEP-Zero fait pour les Données

Aujourd'hui, les systèmes d'information sont les « banques de données ». Ils décident ce qui est un « fait » et ce qui est « faux ». Ils peuvent modifier des fichiers, caviarder des noms ou prétendre qu'une vidéo est un deepfake.

SQEP-Zero remplace la confiance déclarative par des reçus cryptographiques vérifiables.

Domaine	Avant	Après
Données	« Faites confiance au système »	« Vérifiez la preuve d'action »

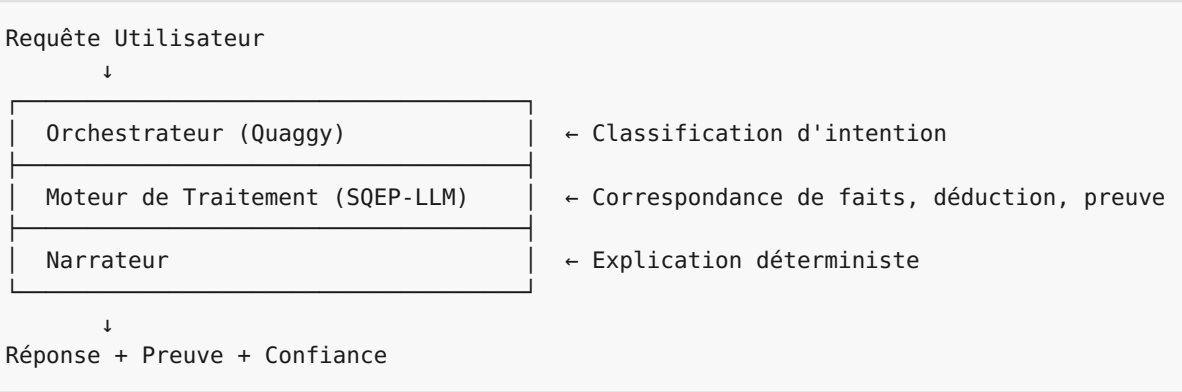
Formulation Défendable

> « Bitcoin a introduit la preuve cryptographique comme fondation de la confiance pour les transactions financières. SQEP-Zero applique le même principe aux actions sur les données : remplacer la confiance déclarative par des reçus cryptographiques vérifiables. »

Le Moteur IA Cryptographique

SQEP-Zero n'est pas seulement un standard de reçus. C'est aussi une **architecture d'IA déterministe** où chaque décision est prouvable.

Architecture de Traitement



Comparaison avec les LLM Traditionnels

LLM Traditionnel	SQEP-LLM
Probabiliste	**Déterministe**
Hallucine	**Prouve ou refuse**
Boîte noire	**Raisonnement traçable**
Nécessite GPU/cloud	**CPU uniquement**
Ne peut pas expliquer « pourquoi »	**Chaîne de preuve explicite**
Imprévisible	**Auditable**

Système de Tiers d'Inférence

SQEP-Zero intègre un runtime LLM à trois niveaux :

```
┌──────────┬──────────────┬──────────────┬──────────────────────────┐
│ Niveau   │ Backend      │ Modèle       │ Cas d'usage              │
├──────────┼──────────────┼──────────────┼──────────────────────────┤
│ QUALITÉ  │ Ollama       │ Gemma 2B     │ Résumés, raisonnement    │
│ (T1)     │ (llama.cpp)  │              │ Tâches complexes         │
├──────────┼──────────────┼──────────────┼──────────────────────────┤
│ VITESSE  │ Candle       │ TinyLlama    │ Temps réel, streaming    │
│ (T2)     │ + GPU CUDA   │              │ Faible latence           │
├──────────┼──────────────┼──────────────┼──────────────────────────┤
│ HORS-LIGNE│ Candle      │ TinyLlama    │ Sans réseau, embarqué    │
│ (T3)     │ + CPU seul   │              │ Appareils edge           │
└──────────┴──────────────┴──────────────┴──────────────────────────┘
```

Accélération GPU

L'implémentation utilise **CUDA** pour l'accélération GPU : - **28x plus rapide** sur RTX 4060 par rapport au CPU - Inférence Candle en Rust pur (pas de Python) - Modèles quantifiés GGUF pour faible empreinte mémoire

Reçus d'Inférence IA

Chaque génération produit un reçu journalisé dans la HashChain :

```
{
  "action": "llm_generate",
  "timestamp": "2026-02-04T15:30:00Z",
  "tier": "quality",
  "model": "gemma2:2b",
  "prompt_hash": "sha256:abc123...",
  "response_hash": "sha256:def456...",
  "latency_ms": 1250,
  "previous_hash": "sha256:previous..."
}
```

Cela permet de **prouver mathématiquement** qu'une IA a produit une réponse spécifique à un moment donné.

Cas d'Usage

1. Conformité RGPD

Problème : Sous le RGPD, vous devez prouver que vous avez traité les données personnelles correctement.

Solution : Chaque accès, modification ou suppression de données génère un Reçu. Lors d'un audit, vous fournissez la HashChain comme preuve mathématique de conformité.

2. Audit des Décisions IA

Problème : Les systèmes d'IA prennent des décisions affectant la vie des gens (prêts, embauche, diagnostics médicaux). Les régulateurs exigent l'explicabilité.

Solution : Chaque inférence IA génère un Reçu liant le hash de l'entrée, la version du modèle et la sortie. La piste de décision est immuable et vérifiable.

3. Intégrité des Documents Légaux

Problème : Les contrats, preuves et documents légaux peuvent être altérés après signature.

Solution : Hasher le document à sa création et enregistrer un Reçu. Toute version ultérieure peut être vérifiée contre le hash original.

4. Traçabilité de la Chaîne d'Approvisionnement

Problème : Prouver l'origine et l'historique de manipulation d'un produit.

Solution : Chaque transfert génère un Reçu. La chaîne complète prouve la provenance de la source à la destination.

5. Piste d'Audit Financier

Problème : Les journaux de transactions peuvent être manipulés.

Solution : Chaque transaction génère un Reçu. Les auditeurs vérifient la chaîne mathématiquement, pas par la confiance.

6. Souveraineté Informationnelle

Problème : Dans l'ère des deepfakes et de la désinformation, prouver qu'un document est authentique devient critique.

Solution : SQEP-Zero permet à un citoyen ou un lanceur d'alerte d'être son propre notaire. Pas besoin d'un juge ou d'un « fact-checker » pour dire si un document est réel — le hash SQEP-Zero le dit.

Spécification Technique

Schéma du Reçu (JSON)

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

Calcul du Hash

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

Format de Stockage

JSONL (JSON Lines) : un Reçu par ligne, ajout uniquement.

```
{"ts_unix":
1769832901,"tag":"CHAIN_GENESIS","message":"...","prev_hash":"","hash":"858a813..."}
{"ts_unix":
1769832952,"tag":"FICHIER_CHIFFRÉ","message":"...","prev_hash":"858a813...","hash":"6
4f9943..."}
```

Algorithme de Vérification (Pseudocode)

```
fonction verifier_chaine(recus):
    hash_precedent = ""
    pour recu dans recus:
        attendu = sha256(recu.ts_unix | recu.tag | recu.message | recu.prev_hash)
        si attendu != recu.hash:
            retourner INVALIDE("hash non concordant")
        si recu.prev_hash != hash_precedent:
            retourner INVALIDE("rupture de chaîne")
        hash_precedent = recu.hash
    retourner VALIDE
```

Endpoints API

Endpoint	Méthode	Description
`/v1/journal/append`	POST	Ajouter un nouveau Reçu
`/v1/journal/verify`	GET	Vérifier la chaîne entière
`/v1/journal/head`	GET	Obtenir le hash du dernier Reçu
`/v1/journal/tail?n=10`	GET	Obtenir les N derniers Reçus

Propriétés de Sécurité

Inviolabilité (Tamper Evidence)

Toute modification d'un Reçu invalide son hash et tous les hashs suivants. La falsification est mathématiquement détectable.

Secret Avancé (Forward Secrecy)

Les anciens Reçus ne peuvent pas être forgés sans connaître l'état de la chaîne à ce moment. La chaîne croît uniquement vers l'avant.

Confidentialité par Conception

Les Reçus contiennent des descriptions d'actions, pas de données brutes. Vous prouvez « le fichier X a été consulté » sans révéler le contenu du fichier X.

Fonctionnement Hors-Ligne

Aucun réseau requis. La chaîne existe localement. La vérification nécessite uniquement les données de la chaîne et une implémentation SHA-256.

Considérations Post-Quantiques

SHA-256 reste sécurisé contre les attaques quantiques connues (l'algorithme de Grover ne fournit qu'une accélération quadratique). Les versions futures pourront adopter SHA-3 ou des signatures résistantes au quantique pour une assurance supplémentaire.

Implémentation

Implémentation de Référence

L'implémentation de référence est écrite en **Rust**, choisi pour :

- Sécurité mémoire sans ramasse-miettes
- Abstractions à coût zéro
- Aucune dépendance runtime
- Support multiplateforme (Linux, Windows, macOS, ARM, Android)

Métriques : - Taille du binaire : ~15 MB - Empreinte mémoire : < 50 MB typique - Couverture de tests : 1477 tests passants - Dépendances externes : Minimales

Open Source

SQEP-Zero est un standard ouvert. La spécification est librement disponible. Les implémentations de référence sont publiées sous licence MIT.

Le Reçu Genesis

Le standard SQEP-Zero a été figé le 31 janvier 2026, avec le Reçu Genesis suivant :

```
{
  "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"
}
```


Ce hash sert d'**ancre** pour la chaîne de production ElevitaX. N'importe qui peut vérifier qu'une chaîne remonte à ce Genesis.

Conclusion

SQEP-Zero transforme la responsabilité numérique d'une question de confiance en une question de mathématiques.

- **Pour les organisations :** Prouver la conformité, démontrer l'intégrité, réduire les coûts d'audit.
- **Pour les individus :** Savoir que les actions sur vos données sont enregistrées et vérifiables.
- **Pour les régulateurs :** Vérifier les déclarations indépendamment, sans faire confiance à la partie régulée.
- **Pour les développeurs :** Un standard simple et bien défini qui s'intègre dans n'importe quel système.

Le monde numérique génère des milliards d'actions quotidiennement. La plupart ne laissent aucune trace vérifiable. SQEP-Zero change cela en fournissant une méthode universelle, ouverte et mathématiquement solide pour prouver que des actions se sont produites.

> **« Chaque action sur vos données mérite un reçu. »**

Contact & Ressources

- **Organisation :** ElevitaX
- **Fondateur :** Herbert Manfred Fulgence Vaty
- **Version du Standard :** 1.0 (Figé)
- **Hash Genesis :** `858a81309f473dacc70e4a94b21a09a6d56241ae810ee6b9308e6a49d00038b7`
- **Site Web :** sqep.fr

- **Licence :** MIT (Standard ouvert)

Annexe A : Glossaire

Terme	Définition
Reçu	Enregistrement cryptographique prouvant qu'une action s'est produite
HashChain	Séquence de Reçus liés par des hashs cryptographiques
Genesis	Premier Reçu d'une HashChain (pas de prédécesseur)
SHA-256	Algorithme de hash utilisé (sortie 256 bits)
JSONL	Format JSON Lines (un objet JSON par ligne)
Inviolable	Toute modification est mathématiquement détectable
GGUF	Format de modèle quantifié pour inférence LLM efficace
Candle	Framework d'inférence ML en Rust pur

Annexe B : Propriétés Mathématiques

Résistance aux Collisions

La probabilité de trouver deux entrées différentes produisant le même hash SHA-256 est de 2^(-128), ce qui est computationnellement infaisable avec la technologie actuelle.

Garanties de Vérification

Pour une chaîne de N reçus : - Temps de vérification : O(N) - Espace requis : O(1) (vérification streaming) - Probabilité de faux positif : 0 (déterministe)

Limites du Standard

SQEP-Zero fournit : - ✓ Preuve d'occurrence (tamper evidence) - ✓ Intégrité temporelle - ✓ Vérification indépendante

SQEP-Zero ne fournit PAS : - ✗ Confidentialité des données (utiliser le chiffrement séparément) - ✗ Correction sémantique (prouve que l'action a eu lieu, pas qu'elle était correcte) - ✗ Identité vérifiée (l'acteur est une référence, pas une identité prouvée)

Cette délimitation est ce qui rend le standard robuste et auditable.