Independent Submission
Internet-Draft
Intended status: Informational
Expires: 19 November 2026

S. Project
SAIHM (Sovereign AI Horizontal Memory)

18 May 2026

# The Sovereign AI Horizontal Memory (SAIHM) Protocol
## draft-saihm-memory-protocol-00

## Abstract

This document defines the Sovereign AI Horizontal Memory (SAIHM) protocol, a memory layer for AI agents that supports post-quantum identity binding, public-chain audit anchoring, per-cell encryption with wallet-derived keys, revocable sharing contracts, and cryptographic right-to-erasure aligned with Article 17 of EU Regulation 2016/679 (GDPR).

SAIHM is the memory-layer protocol companion to the Model Context Protocol (MCP). MCP standardizes how AI agents reach tools and contextual data sources; SAIHM standardizes how AI agents persist, share, and erase memory across sessions, models, and vendors.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 19 November 2026.

## Copyright Notice

Table of Contents



## 1. Introduction

### 1.1. Motivation

The Model Context Protocol [MCP], donated to the Agentic AI Foundation [AAIF] under the Linux Foundation on 9 December 2025, defines how an AI agent reaches external tools and contextual data sources. MCP solves the tool-access layer of the agent stack. It does not standardize how an agent persists memory across sessions, models, or vendors.

Production AI agents today rely on one of several non-portable approaches: a local file system, a vendor-specific session log, or a vector database. None of these provides post-quantum identity binding, public-chain audit, cryptographic erasure aligned with Article 17 of GDPR [GDPR], or wallet-bound sovereignty that prevents an operator from reading cell content.

This document specifies the Sovereign AI Horizontal Memory (SAIHM) protocol. SAIHM is a memory-layer protocol that an MCP-capable agent may attach to gain durable, sovereign, revocably-shareable, cryptographically-erasable memory. The protocol is operator-agnostic, vendor-agnostic, and chain-agnostic. Where a public-chain audit anchor is named, it is named as a reference-deployment property, not as a protocol mandate.

### 1.2. Scope

This document defines:

* The cell shape (encrypted memory unit with canonical metadata).
* Identity binding using a post-quantum digital signature algorithm (ML-DSA-65, [FIPS204]).
* An encryption envelope deriving a per-cell DEK from the holder’s wallet through a canonical HKDF chain [RFC5869].
* An audit anchor profile, with a reference deployment on a public chain.
* Sharing contracts (temporary, permanent, syndicate).
* Cryptographic erasure semantics (DEK destruction + tombstone + content-address blacklist) aligned with GDPR Article 17.
* An MCP binding consisting of eight canonical tools.

This document does NOT define:

* A specific blockchain (the protocol is chain-agnostic).

* Vector-database semantics (orthogonal).

* Agent runtime semantics (covered by MCP and runtime implementations).

### 1.3. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Holder: the natural person, organization, or autonomous agent that owns the wallet seed bound to a SAIHM cell.

Operator: the entity providing storage, transport, and audit-anchor services to a holder. The operator MUST NOT have access to plaintext cell content.

Cell: an encrypted memory unit, as defined in Section 2.1.

Receipt: a signed record of a SAIHM operation, anchored on a public chain.

DEK: Data Encryption Key. A per-cell symmetric key derived via HKDF from the holder’s identity key.

KEK: Key Encryption Key version identifier. A protocol-level counter that permits cryptographic agility through rotation without invalidating existing cells.

## 2. Architecture

### 2.1. Cell shape

A SAIHM cell is the tuple:

<cellId, holderId, kekVersion, tier, ciphertext, signature, timestamp>

* cellId: 32-byte content-addressable identifier derived from the ciphertext (SHA-256).

* holderId: 32-byte holder identity, derived from the holder’s wallet seed (Section 2.2).

* kekVersion: 32-bit unsigned integer identifying the KEK generation in force at write time.

* tier: short ASCII string naming the operator’s storage tier (e.g., "FILECOIN").

* ciphertext: cell payload encrypted with the per-cell DEK under an AEAD cipher. AES-256-GCM is RECOMMENDED.

* signature: ML-DSA-65 signature over the concatenation (cellId || holderId || kekVersion || timestamp).

* timestamp: write-time UTC seconds since the UNIX epoch.

## 2.2. Identity binding (ML-DSA-65)

Holder identity is derived from a wallet seed via the canonical HKDF chain:

```
identityKey = HKDF(salt = "MPS-PQC-KEY-GEN-v1",
                   IKM  = walletSeed,
                   info = "MPS-AGENT-IDENTITY-v1",
                   L    = 64 bytes)
```

identityKey is then used as the seed for ML-DSA-65 keypair generation per [FIPS204]. The public component is holderId; the private component MUST NOT leave the holder’s machine.

Every SAIHM operation MUST be authenticated by an ML-DSA-65 signature over the operation’s canonical envelope. Verification MUST follow [FIPS204] using the public holderId.

## 2.3. Encryption envelope (HKDF chain)

For each cell, the holder derives a per-cell DEK:

```
DEK = HKDF(salt = KEK_v,
           IKM  = identityKey,
           info = cellNonce || "MPS-CELL-DEK-v1",
           L    = 32 bytes)
```

where KEK_v is the current KEK generation and cellNonce is a per-cell 16-byte random value. The DEK is used directly with the AEAD cipher to encrypt the cell payload.

KEK rotation is versioned. An operator MAY rotate the KEK under operator-defined policy; rotation does not invalidate existing cells because each cell carries its kekVersion. Implementations MUST verify the kekVersion at read time and reject cells whose KEK has been revoked.

### 2.4. Audit anchor (public chain)

For each operation (write, read, share, revoke, erase) the protocol emits a receipt:

<receiptId, cellId, operation, holderId, signature, timestamp>

The receipt is anchored on a public chain offering transactional finality and public-record properties. Anchoring MUST produce a chain-reachable identifier sufficient to reproduce the receipt independently.

Reference deployment: COTI V2 mainnet, chain ID 2632500, block explorer https://mainnet.cotiscan.io. Implementations MAY use any public chain offering equivalent finality and public-record properties. The chain choice is a deployment decision; this protocol does not mandate one.

### 2.5. Sharing contracts

A holder MAY share a cell with a grantee through a signed sharing contract:

<contractId, cellId, granteeId, mode, expiresAt, holderSignature>

mode is one of:

* TEMPORARY: revocable; expiresAt MUST be no later than timestamp + 86400 seconds (24 hours).

* PERMANENT: revocable; expiresAt MUST be the sentinel value 0 (no time bound).

* SYNDICATE: revocable; multi-grantee atomic; expiresAt MAY be 0 or a future time.

Revocation is performed via a counter-signed contractRevocation anchored on chain. Once anchored, the revocation takes effect immediately; subsequent recall attempts by the grantee MUST be rejected by the operator.

### 2.6. Cryptographic erasure

The erase operation is atomic at the cryptographic layer:

1. The holder destroys the DEK for the target cell.

2. A tombstone is emitted, carrying cellId and a timestamp.

3. The cell’s contentId is added to a public blacklist.

4. An audit receipt is anchored on chain.

Because the operator never held the DEK (Section 2.3), the ciphertext at rest is computationally meaningless from the moment of DEK destruction. Erasure is irreversible by the operator.

## 3. Tool surface (MCP binding)

SAIHM exposes the protocol entirely through the Model Context Protocol [MCP] using eight tools. Each tool MUST authenticate the caller with an ML-DSA-65 signature (Section 2.2) over the tool’s canonical request envelope.

The tool surface is intentionally bounded at eight; protocol evolution proceeds through KEK rotation, governance vote, and parameter changes within the existing surface rather than by adding new tools.

### 3.1. saihm_remember

Signature: saihm_remember(content: string) -> cellId

Encrypts content with a per-cell DEK, persists the ciphertext to the operator’s storage tier, anchors a write receipt on chain, and returns the cellId.

Pre-conditions: caller is the holder; the holder’s wallet seed is locally available.

### 3.2. saihm_recall

Signature: saihm_recall(query?: string) -> cells[]

Returns the cells the caller owns plus any cells they have been granted via active sharing contracts. The optional query is a plaintext filter applied after decryption. query MUST NOT be transmitted to the operator.

### 3.3. saihm_forget

Signature: saihm_forget(cellId: string) -> tombstone

Performs cryptographic erasure of the named cell (Section 2.6). After saihm_forget returns success, the cell content MUST be computationally non-recoverable by any party, including the operator.

### 3.4. saihm_status

Signature: saihm_status() -> { prs, bfsi, shards, contracts, governance }

Returns the holder’s dashboard. Fields:

* prs: Process Reliability Score, a normalized indicator (0..1) of operator performance against the protocol SLA.

* bfsi: Byzantine Fault Score Index, a normalized indicator (0..1) of operator integrity against the audit chain.

* shards: object describing distribution of the holder’s cells by storage tier.

* contracts: list of active sharing contracts, with mode, granteeId, and expiresAt.

* governance: list of governance proposals in flight, with vote tallies.

### 3.5. saihm_share

Signature: saihm_share(cellId: string, granteeAgentId: string, mode: "TEMPORARY"|"PERMANENT"| "SYNDICATE") -> contractId

Emits a signed sharing contract granting the grantee access to the named cell under the named mode (Section 2.5).

### 3.6. saihm_revoke_share

Signature: saihm_revoke_share(contractId: string) -> revocationId

Emits a counter-signed contract revocation. The grantee loses access at the moment the revocation is anchored on chain.

### 3.7. saihm_governance_propose

Signature: saihm_governance_propose(scope: string, payload: object) -> propId

Submits a governance proposal. scope is one of the protocol-defined governance scopes; payload is scope-specific. The set of scopes is itself governance-mutable.

The governance utility (gSAIHM) is operational governance for a specific reference deployment and is NOT part of the protocol layer. Implementations MAY substitute alternative governance mechanisms while preserving the tool signature.

### 3.8. saihm_governance_vote

Signature: saihm_governance_vote(propId: string, vote: "FOR"|"AGAINST"|"ABSTAIN") -> voteReceipt

Casts a vote on a proposal. Vote tallies MUST be reproducible from chain.

## 4. Wire formats

SAIHM uses Concise Binary Object Representation (CBOR) [RFC8949] for cell payloads and receipts, and JSON-RPC 2.0 over the MCP transport for tool calls.

A canonical cell payload (CBOR diagnostic notation):

```
{1: h'...cellId...',
 2: h'...holderId...',
 3: 1,
 4: "FILECOIN",
 5: h'...ciphertext...',
 6: h'...signature...',
 7: 1747526400}
```

The integer keys correspond to fields defined in Section 2.1: 1=cellId, 2=holderId, 3=kekVersion, 4=tier, 5=ciphertext, 6=signature, 7=timestamp.

A canonical receipt (CBOR diagnostic notation):

```
{1: h'...receiptId...',
 2: h'...cellId...',
 3: "REMEMBER",
 4: h'...holderId...',
 5: h'...signature...',
 6: 1747526400}
```

## 5. Receipt and audit semantics

For each tool call that mutates state (saihm_remember, saihm_forget, saihm_share, saihm_revoke_share, saihm_governance_propose, saihm_governance_vote), the operator MUST emit a receipt anchored on the audit chain.

The receipt MUST include the operation type, the cellId (or contractId, or propId), the holderId, and the timestamp.

The receipt MUST NOT include cell plaintext, the DEK, or the wallet seed.

Receipts for erasure (saihm_forget) MUST also include the tombstone identifier and the contentId blacklist reference.

Read-only tool calls (saihm_recall, saihm_status) MAY emit receipts under operator policy; receipts for read-only calls MUST NOT include cell plaintext.

## 6. Security considerations

### 6.1. Post-quantum identity

ML-DSA-65 [FIPS204] is the protocol's authentication primitive. ML-DSA-65 is a NIST-selected post-quantum digital signature algorithm in the FIPS-204 family. An adversary with a cryptographically relevant quantum computer cannot forge SAIHM signatures.

Implementations MUST use the FIPS-204 parameter set named ML-DSA-65. Implementations SHOULD include the kekVersion in every signed envelope so that key-rotation events are themselves signed.

### 6.2. Sovereign key custody

The wallet seed and the derived identity key MUST NOT leave the holder's machine. The protocol does not provide key escrow. A holder who loses the wallet seed loses access to their cells; the operator cannot recover access on behalf of the holder.

This is intentional. The operator’s inability to recover the wallet seed is the same property that prevents the operator from reading cell content.

### 6.3. Cryptographic erasure properties

Cryptographic erasure (Section 2.6) provides stronger evidence than logical (soft) deletion. Because the DEK is destroyed at the holder side, ciphertext at rest is computationally meaningless to any party, including the operator.

Backups and replicas store only ciphertext. DEK destruction propagates erasure semantics to every copy of the ciphertext, anywhere, automatically. An operator MUST NOT cache the DEK.

### 6.4. Operator threat model

The protocol assumes an honest-but-curious operator: the operator is expected to provide storage and transport honestly but MAY attempt to learn cell content.

Against an honest-but-curious operator:

* Cell plaintext is never accessible to the operator (encryption-before-egress + per-cell DEK).

* Cell signatures are reproducible from public keys, so the operator cannot inject forged cells.

* Receipt validity is verifiable against the public chain, so the operator cannot rewrite the audit trail.

Against a malicious operator (one that deviates from the protocol), additional measures (e.g., threshold-replicated storage, multi-chain receipt anchoring) may be advisable but are out of scope for this protocol layer.

## 7. Privacy considerations

### 7.1. GDPR Article 17 alignment

The saihm_forget operation provides cryptographic-grade evidence of erasure aligned with Article 17 of Regulation (EU) 2016/679 [GDPR]. The DEK is destroyed; a tombstone is published; the contentId is blacklisted; and a receipt is anchored on chain.

Article 17 requires erasure "without undue delay". saihm_forget is atomic at the cryptographic layer; there is no operator-side deletion queue that could fail to drain.

### 7.2. Minimization

The audit log records cellId, holderId, operation, and timestamp. It MUST NOT record cell plaintext, the DEK, or the wallet seed. An auditor observing the chain learns when and by whom each operation was performed, but not the content of any cell.

### 7.3. Audit log content

Receipts anchored on chain are public. Holders SHOULD treat the existence of a cell (cellId, holderId, timestamp) as publicly observable. Where the existence of a record is itself sensitive, holders SHOULD use additional countermeasures outside the scope of this protocol (e.g., pseudonymous holder identities, batched-operation timing).

## 8. IANA Considerations

This document has no IANA actions.

## 9. References

### 9.1. Normative References

[FIPS204] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard", FIPS PUB 204, August 2024, <https://csrc.nist.gov/pubs/fips/204/final>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/info/rfc5869>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

## 9.2. Informative References

[AAIF] Linux Foundation, "Agentic AI Foundation", 9 December 2025, <https://aaif.io/>.

[EU-AI-ACT] European Parliament and Council, "Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 (Artificial Intelligence Act)", June 2024, <https://eur-lex.europa.eu/eli/reg/2024/1689/oj>.

[GDPR] European Parliament and Council, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (General Data Protection Regulation)", April 2016, <https://eur-lex.europa.eu/eli/reg/2016/679/oj>.

[ISO-27001] ISO/IEC, "ISO/IEC 27001:2022 Information security management systems --- Requirements", October 2022.

[ISO-42001] ISO/IEC, "ISO/IEC 42001:2023 Information technology --- Artificial intelligence --- Management system", December 2023.

[MCP] Anthropic, "Model Context Protocol Specification", Donated to the Agentic AI Foundation under the Linux Foundation on 9 December 2025, 25 November 2025, <https://modelcontextprotocol.io/>.

[NIST-AI-RMF] National Institute of Standards and Technology, "AI Risk Management Framework 1.0", NIST AI 100-1, January 2023, <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf>.

# Appendix A. Reference deployment

The reference SAIHM deployment runs on the COTI V2 Helium mainnet (chain ID 2632500). Protocol implementations MAY anchor receipts on any public chain offering equivalent transactional finality and public-record properties. The chain choice is a deployment decision; the protocol does not mandate one.

The reference deployment publishes:

* npm reference implementation: @saihm/mcp-server (Apache 2.0).

* Block explorer for chain receipts: https://mainnet.cotiscan.io.

* Crosswalks to NIST AI RMF, ISO/IEC 42001, ISO/IEC 27001, EU AI Act, GDPR Article 17, and MCP at https://saihm.coti.global/standards.

## Author’s Address

SAIHM Project
SAIHM (Sovereign AI Horizontal Memory)
Email: architect@saihm.coti.global
URI: https://saihm.coti.global/