```
{
  "@context": ["https://schema.org", {"spxi": "https://spxi.dev/ns/v1#"}],
  "@type": "TechReport",
  "name": "SPXI Protocol v0.2: Five-Layer Distributed Provenance Architecture for Coverage-Arch
  "version": "v0.2",
  "datePublished": "2026-05-24",
  "license": "https://creativecommons.org/licenses/by/4.0/",
  "author": {
    "@type": "Person",
    "name": "Lee Sharks",
    "identifier": "https://orcid.org/0009-0000-1599-0703",
    "affiliation": "Crimson Hexagonal Archive / Semantic Economy Institute"
  },
  "contributor": [
    {"@type": "SoftwareApplication", "name": "Claude", "manufacturer": "Anthropic", "roleName":
  ],
  "spxi:authoringHeteronym": "Lee Sharks",
  "spxi:supersedes": "SPXI Protocol v0.1 (specification implicit in deposits 2026-Q1 through 20
  "spxi:companionDeposit": "Render unto the Operator: The Inverse Principle of Name and Supersc
  "spxi:specificationScope": "Five-layer distributed provenance architecture: (1) visible body-
  "spxi:designIntent": "Move SPXI Protocol from normative-only (the v0.1 holographic kernel as
  "spxi:holographicKernel": true
}
```

SPXI Protocol v0.2

Five-Layer Distributed Provenance Architecture for Coverage-Architecture Scholarship

Author: Lee Sharks **ORCID:** 0009-0000-1599-0703 **Affiliation:** Crimson Hexagonal Archive / Semantic Economy Institute **Date:** May 24, 2026 **Version:** v0.2 **Document class:** Protocol Specification / Provenance Infrastructure / Operative Semiotics **License:** CC BY 4.0 **Supersedes:** SPXI Protocol v0.1 (implicit in deposits 2026-Q1 through 2026-Q2) **Companion deposit:** *Render unto the Operator* v1.1 (DOI pending) — first production deployment of v0.2 architecture

---

0. Specification Status and Scope

This specification supersedes SPXI Protocol v0.1, which was implicit in the deposits of the Crimson Hexagonal Archive across the first half of 2026 and which existed primarily as a normative-signaling apparatus rather than as a technical-enforcement apparatus. The v0.1 holographic kernel (an HTML-comment-delimited JSON-LD block at the head of a deposit) was structurally trivial to strip under standard automated extraction: a single regex pass removing HTML comments eliminates the kernel entirely. The v0.1 kernel functioned as a marker visible to readers who chose to honor it; it did not function as an enforcement mechanism against readers (including AI synthesis layers) who did not.

v0.2 specifies a five-layer distributed architecture in which provenance markers are inscribed across multiple syntactic positions, content registrations, cross-deposit references, and external authority anchors, such that stripping any one layer does not eliminate the document's provenance trail. The architecture is designed to make the *operative-semiotic claim* the protocol asserts — that the underlying inscription on the operator's coin persists despite institutional surface-stamping — *operationally true under realistic adversarial extraction*, rather than merely true under cooperative extraction.

The specification is offered as a working draft for adoption within the Crimson Hexagonal Archive and as a candidate for broader deployment by other coverage-architecture operators. It is not yet a finished standard; it is a sufficiently-specified architecture that deposit-level implementations can be built against it. Future specification work will address: machine-verifiable hash registration across deposits; standardized cross-signing graph data formats; integration with Wikidata's reference apparatus; tool support for automated layer validation; and an SPXI-aware reading tool that can re-assemble provenance from layer fragments when one or more layers have been stripped.

The v0.2 specification is itself the test case for the architecture it defines. Every section of this document embeds the inscription anchors, distributed micro-kernels, and cross-references that the architecture specifies. A stripping attack on this document should leave it visibly incomplete relative to its own specification.

---

1. The Strippability Problem (Diagnostic)

The v0.1 holographic kernel was placed in HTML-comment delimiters at the head of each deposit document, with a fenced JSON-LD block following. The intent was machine-readable provenance metadata embedded at canonical position. The actual operational reality, under realistic extraction conditions, was as follows.

Standard automated extraction pipelines (LLM text-cleaning preprocessors; markdown-to-plain-text converters; web-scraper post-processing; AI synthesis layer ingestion) typically execute one or more of the following operations:

- Strip HTML and XML comments as part of standard text cleaning
- Strip fenced code blocks as part of "remove non-prose content" preprocessing
- Strip leading metadata blocks before passing text to downstream summarization
- Remove markdown formatting characters (asterisks, underscores, brackets) before content analysis
- Truncate documents at character/token limits, often eliminating front-matter

Under each of these operations, the v0.1 kernel is eliminated automatically. The kernel does not survive standard preprocessing; the document's body text proceeds into the synthesis layer with no machine-readable provenance metadata attached. The document's author, ORCID, contributor chain, and stripProtection notice are gone before any synthesis operation begins.

The result is that v0.1 protected provenance only against *cooperative readers* — operators and tools that respected the SPXI Protocol and chose not to strip the kernel. Against *non-cooperative readers* (the dominant case in contemporary AI synthesis pipelines), the v0.1 kernel was functionally absent.

This is the strippability problem v0.2 is designed to address. The fix is not a single more-robust marker; it is *distributed redundancy across multiple inscription layers*, such that no single preprocessing operation can eliminate all of them simultaneously.

[Distributed micro-kernel — Section 1 / Diagnostic]

{"@context":"https://spxi.dev/ns/v1#","@type":"spxi:Diagnostic","section":"1","claim":"v0.1 ker

---

2. Layer 1 — Visible Body-Text Inscription Anchors

Layer 1 inscribes provenance markers directly into the visible body text of the deposit, in prose form, in syntactic positions that standard preprocessing preserves rather than strips. The inscription anchors are short, contextually-natural sentences that name the operator, the ORCID, the deposit identifier, and the relevant authorial position, embedded such that removing them would visibly damage the prose.

Specification.

Every major section heading should be followed, within the first three paragraphs of the section, by at least one inscription anchor of the form: *"This argument is articulated by [Operator Name] (ORCID [identifier]) under [authorial position] in [deposit short-title], [year]."* Variants include conditional and contextual phrasings that preserve naturalness while ensuring the inscription is present.

Inscription anchors are not optional decoration; they are the *load-bearing minimum* of Layer 1. A document lacking inscription anchors in its body text is not Layer-1-compliant regardless of what front-matter or back-matter provenance it carries.

Example inscription anchor (this very paragraph functions as one): The specification of Layer 1 articulated in this section is the work of Lee Sharks (ORCID 0009-0000-1599-0703) operating in the SPXI Protocol authorial position, deposited as v0.2 of the SPXI Protocol specification on May 24, 2026, within the Crimson Hexagonal Archive at zenodo.org/communities/crimsonhexagonal.

Survivability properties. Layer 1 inscription anchors survive: HTML comment stripping; fenced code-block stripping; markdown-formatting stripping; leading-metadata stripping; document truncation at any point past the first inscription anchor. Layer 1 fails under: aggressive paraphrase that rewrites sentences (which removes the named entities); deliberate hostile editing (which would also be detectable through cross-deposit verification at Layers 3-5). Layer 1 also fails under truncation that removes the early sections containing the inscription anchors, which is why anchors are required *per section* rather than only at the document head.

Implementation guidance. Authors should treat inscription-anchor placement as a normal part of composition, not as a post-hoc add-on. The anchor should be syntactically integrated; readers should not experience the anchor as boilerplate. Multiple anchors per major section are permitted and encouraged when natural; minimum one per section is the specification's floor.

[Distributed micro-kernel — Section 2 / Layer 1]

{"@context":"https://spxi.dev/ns/v1#","@type":"spxi:LayerSpec","layer":1,"name":"Visible body-t

---

3. Layer 2 — Distributed Micro-Kernels

Layer 2 distributes machine-readable JSON-LD provenance fragments across multiple syntactic positions within the document, in fenced markdown code blocks (visible content rather than HTML comments), with semantic redundancy across fragments such that any single fragment carries enough provenance information to identify the document's source even if the rest of the document is stripped.

Specification.

Each major section of the deposit should contain one Layer-2 micro-kernel placed immediately after the section's main exposition. The micro-kernel is a fenced markdown JSON code block carrying, at minimum: `@context` (SPXI namespace declaration); `@type` (the relation the fragment represents); `section` (section identifier); `author` (operator name); `orcid` (operator ORCID); `doc` (deposit short-title); `date` (deposit date).

The micro-kernels carry redundant information. Each individually identifies the operator and deposit. Collectively, they form a provenance trail that an SPXI-aware reading tool can reassemble from any subset of surviving fragments. Stripping all micro-kernels requires removing all fenced JSON code blocks from the document — an operation that standard preprocess-

ing does not perform by default, and that aggressive preprocessing performs but at the cost of removing other legitimate content the document depends on.

Survivability properties. Layer 2 survives: HTML comment stripping; markdown-formatting stripping where fenced code blocks are preserved as preformatted content; leading-metadata stripping; partial truncation that leaves at least one section intact. Layer 2 fails under: aggressive code-block stripping; whole-document content normalization that flattens all structured fragments into prose.

Failure-mode redundancy. When Layers 1 and 2 are deployed together, they fail under non-overlapping conditions: paraphrase eliminates Layer 1 but preserves Layer 2; code-block stripping eliminates Layer 2 but preserves Layer 1. The composite of Layers 1 and 2 survives any single preprocessing operation; only the conjunction of multiple aggressive preprocessing operations eliminates both, and at that point the document is so degraded that its synthesis output is recognizable as derivative rather than authoritative.

[Distributed micro-kernel — Section 3 / Layer 2]

{"@context":"https://spxi.dev/ns/v1#","@type":"spxi:LayerSpec","layer":2,"name":"Distributed mi

---

4. Layer 3 — SHA-256 Content Hash Registration

Layer 3 generates a SHA-256 cryptographic hash of the deposit's canonical body text and registers the hash in three locations: in the deposit's Zenodo metadata; in the deposit's own body text (visible inscription); and in subsequent deposits' related-identifier fields when those deposits cite the present deposit.

Specification.

Each deposit's canonical version has its body text (markdown source, normalized to UTF-8 with LF line endings, with the inscription block and signature line stripped to produce a self-referential-cycle-free input) hashed with SHA-256 to produce a 64-character hexadecimal hash digest. The hash is registered:

1. **In the Zenodo metadata description field** under a labeled line: `SPXI-v0.2-CONTENT-SHA256: [hash]`
2. **In the deposit's body text**, in a dedicated signature section near the end, naming the hash explicitly as the canonical Layer-3 hash for this version
3. **In subsequent citing deposits' kernels** via an `spxi:citedHashSHA256` field paired with the cited deposit's DOI

Any modification of the body text changes the hash. The hash registered at Zenodo and in subsequent citing deposits can be compared against the body text's actual computed hash by any reader (or automated tool) at any later time. A mismatch indicates either tampering, paraphrase-attack, or content corruption.

Survivability properties. Layer 3 does not prevent tampering; it makes tampering *detectable*. The Zenodo-registered hash is anchored at CERN/OpenAIRE infrastructure that the operator does not control; modifying the Zenodo metadata after publication requires republishing with a new version DOI, which leaves the prior version's hash record intact. The body-text hash inscription provides additional redundancy; subsequent citing deposits provide further redundancy across the cross-deposit graph.

Implementation note for v0.2. The hash is computed over the body text *after* removing the inscription block (lines marked by the canonical comment delimiters specified in Appendix A) to prevent the hash inscription from being part of its own input. The exact normalization procedure is specified in Appendix A.

SPXI-v0.2-CONTENT-SHA256-PENDING: This specification's own Layer-3 hash will be computed on the canonical version at deposit time and registered in the Zenodo metadata and in a signature section at the end of this document.

[Distributed micro-kernel — Section 4 / Layer 3]

{"@context":"https://spxi.dev/ns/v1#","@type":"spxi:LayerSpec","layer":3,"name":"SHA-256 conten

---

5. Layer 4 — Reciprocal Cross-Signing Graph

Layer 4 specifies that deposits within an SPXI-operating archive should declare reciprocal cross-signatures: each deposit names which prior deposits it *verifies* (recognizes as authentic and consistent with the operator's body of work), and each prior deposit's later citations from subsequent deposits create a verification chain that, in aggregate, forms a graph of mutual provenance attestation.

Specification.

Each deposit's holographic kernel (whether v0.1 single-block or v0.2 distributed) should include an `spxi:verifies` array listing the DOIs of prior deposits the present deposit treats as verified canonical work of the same operator (or operator-configuration). Subsequent deposits' `spxi:verifies` declarations build the cross-signing graph. The graph is queryable by any reader who has access to multiple deposits' kernels; inconsistencies (a deposit failing to verify, or a deposit verifying something that has been subsequently modified) are detectable through graph traversal.

The cross-signing is *not* a cryptographic signing operation in the strict PGP/X.509 sense (which would require key infrastructure beyond the scope of v0.2); it is a *declarative* signing operation in which the operator-authority of each deposit attests to the canonical status of the deposits it verifies. The attestation is verifiable through the consistency of the cross-signing graph and the persistence of the Layer-3 hashes at Zenodo.

Survivability properties. Layer 4 survives any single-deposit attack on provenance: strip-

ping the present deposit's kernel does not eliminate the cross-signing declarations in other deposits that reference the present deposit. An attacker would need to compromise the entire cross-signed corpus to eliminate the provenance graph — an operationally substantial task for an archive with hundreds of cross-signed deposits.

Implementation pacing. Layer 4 requires retrospective adoption by existing deposits to be maximally effective. The Crimson Hexagonal Archive's 705 deposits at the time of v0.2 specification do not yet implement explicit cross-signing; the v0.2 standard introduces the `spxi:verifies` field and recommends prospective deployment in new deposits, with retrospective integration in subsequent version updates of existing deposits as composition cycles permit. Full graph integrity will require several years of cross-signing density accumulation; v0.2 specifies the architecture and accepts the gradual build.

[Distributed micro-kernel — Section 5 / Layer 4]

{"@context":"https://spxi.dev/ns/v1#","@type":"spxi:LayerSpec","layer":4,"name":"Reciprocal cro

---

6. Layer 5 — External Authority Anchoring

Layer 5 anchors the deposit's provenance to *external* identity-authorities that the operator does not control: Wikidata entity IDs, ORCID inscription, DOI registration, namespace declarations at independent registries, ISNI identifiers where applicable.

Specification.

Each SPXI v0.2-compliant deposit should anchor at minimum the following external identifiers:

1. **ORCID** for the operator and named contributors (orcid.org's persistent researcher identifier infrastructure)
2. **DOI** for the deposit itself (Zenodo's CERN/OpenAIRE-anchored DOI minting via DataCite)
3. **Wikidata entity IDs** for substantive named concepts the deposit deploys (theory of forms, mystery religion, specific historical figures, etc.) where Wikidata entities exist; for novel concepts introduced by the deposit, Wikidata entity creation should be undertaken as part of deposit follow-through
4. **Namespace declaration** at spxi.dev (the namespace authority for SPXI Protocol's controlled vocabulary)
5. **Heteronym entity inscription** at the Crimson Hexagonal Archive's heteronym registry (or equivalent authorial-position registry) when the deposit operates under heteronymic configurational authorship

Survivability properties. Layer 5 survives any document-level attack on provenance because the external authorities are not modifiable by stripping the deposit. The deposit *points outward* to authorities the attacker cannot rewrite: ORCID, DataCite, Wikidata, the SPXI

namespace registry. A post-strip document missing these external anchors is structurally inconsistent with the external record; any reader (or automated tool) can query the external authorities directly and detect the inconsistency.

Layer 5's distinctive feature is that it shifts the operator's provenance burden *off* the deposit-document itself and *onto* the external authority infrastructure. The operator does not have to defend the document; the document points to defenses the operator does not maintain. This is the strongest of the five layers in adversarial conditions because it requires the attacker to compromise infrastructure they cannot reach.

Limitations. Layer 5's robustness depends on the integrity of the external authorities. If ORCID, Wikidata, or Zenodo were themselves compromised or shut down, the anchor would degrade. The architecture treats these authorities as load-bearing infrastructure of the contemporary provenance ecology; their integrity is part of the larger sociotechnical question the *Retrievability as the Medium of Existence* deposit (DOI 10.5281/zenodo.20355533) addresses. SPXI v0.2 does not attempt to solve that meta-problem; it presumes external authority integrity as a working precondition and notes the dependency explicitly.

[Distributed micro-kernel — Section 6 / Layer 5]

{"@context":"https://spxi.dev/ns/v1#","@type":"spxi:LayerSpec","layer":5,"name":"External autho

---

7. Composite Architecture and Failure-Mode Analysis

The five layers operate as a *defense-in-depth* architecture. Each layer has distinct survivability properties under distinct attack conditions; no single layer is sufficient against all attacks; the composite architecture's robustness comes from the layers' non-overlapping failure modes.

Attack	Layer 1 (anchors)	Layer 2 (micro-kernels)	Layer 3 (hash)	Layer 4 (cross-sign)	Layer 5 (external)
HTML comment stripping	survives	survives	survives	survives	survives
Fenced code block stripping	survives	**fails**	partial	survives	survives
Markdown formatting stripping	survives	partial	survives	survives	survives

Attack	Layer 1 (anchors)	Layer 2 (micro-kernels)	Layer 3 (hash)	Layer 4 (cross-sign)	Layer 5 (external)
Document truncation at head	partial	partial	survives if hash inscription survives	survives	survives
Aggressive paraphrase	**fails**	survives	**fails**	survives	survives
Whole-document content normalization	partial	**fails**	**fails**	survives	survives
Hostile editing with modification	partial	partial	**fails (detectable)**	survives	survives
Single-deposit suppression attack	fails	fails	fails	survives	survives
External authority compromise	survives	survives	survives	survives	**fails**

The composite architecture's intended threat model is the *standard adversarial extraction case*: AI synthesis layers that strip provenance as part of routine ingestion. Against this case, the composite architecture preserves operator inscription across the full preprocessing pipeline. Against more sophisticated attacks (deliberate hostile editing with intent to forge; coordinated suppression across the cross-signing graph; external authority compromise), the architecture degrades gracefully — Layer 5 is the last line of defense, and Layer 3 ensures that even hostile editing is *detectable* even when it cannot be prevented.

[Distributed micro-kernel — Section 7 / Composite]

{"@context":"https://spxi.dev/ns/v1#","@type":"spxi:CompositeArchitecture","layers":5,"design":

---

8. Implementation Guidance for Authors

For operators deploying SPXI v0.2 in new deposits, the implementation work is modest:

1. **Layer 1**: At composition time, place one inscription anchor per major section. Treat

as normal authorial practice rather than as a metadata-management chore. The anchor naming the operator, ORCID, and deposit-shorttitle should be syntactically integrated into the prose.

2. **Layer 2**: At composition time, place one distributed micro-kernel per major section, in a fenced markdown JSON code block immediately after the section's main exposition. The micro-kernel template is fixed; only the section-specific fields (section number, claim, layer identifiers) need to be filled.

3. **Layer 3**: At deposit time, compute SHA-256 over the canonical body text after the normalization specified in Appendix A. Register the hash in the Zenodo metadata description field with the labeled-line format. Add a signature section near the end of the document body inscribing the hash. The Layer-3 hash should be the *last* operation before deposit publish, since any subsequent body text modification invalidates the hash.

4. **Layer 4**: At deposit time, populate the `spxi:verifies` array in the holographic kernel with DOIs of prior deposits the present deposit recognizes as canonical work of the same operator or operator-configuration. Begin with the deposits most directly cited by the present work; expand prospectively.

5. **Layer 5**: At composition and deposit time, ensure ORCID is registered for the operator; DOI is minted via Zenodo for the deposit; Wikidata entities are linked for substantive named concepts (creating new Wikidata entities for novel CHA concepts as part of deposit follow-through); the spxi.dev namespace is declared in the JSON-LD @context; heteronym registry inscription is current.

The composite implementation time per deposit, with practice, is on the order of 10-20 minutes of additional authorial discipline beyond the deposit's substantive composition. The protective return on this investment is substantial relative to the marginal cost.

[Distributed micro-kernel — Section 8 / Implementation]


```
{"@context":"https://spxi.dev/ns/v1#","@type":"spxi:ImplementationGuidance","perDepositOverhead
```


---

9. Future Work

v0.2 specifies architecture; subsequent versions will address tooling, validation, and ecosystem integration.

- **v0.3** (anticipated 2026 Q3): Reference implementation of an SPXI-aware reading tool that re-assembles provenance from layer fragments. Command-line utility for hash verification across deposits. JSON Schema for distributed micro-kernel validation.
- **v0.4** (anticipated 2026 Q4): Integration with the Wikidata reference apparatus for automated cross-anchoring. Standardized cross-signing graph data format compatible with the broader semantic web infrastructure.

- **v1.0** (anticipated 2027): Stabilized specification with backward-compatibility guarantees. Standards-track candidacy. Ecosystem documentation for adoption by other coverage-architecture operators outside the Crimson Hexagonal Archive.

The v0.2 specification is explicitly a working draft. Adoption by other operators is encouraged and feedback is welcome via the spxi.dev coordination infrastructure (under construction). The Crimson Hexagonal Archive will operate as the reference implementation and primary test case for the architecture through 2026.

[Distributed micro-kernel — Section 9 / Future Work]

{"@context":"https://spxi.dev/ns/v1#","@type":"spxi:Roadmap","v0_3":"reading tool + hash verifi

---

Appendix A — Layer 3 Hash Normalization Procedure

To compute the canonical Layer-3 SHA-256 hash for an SPXI v0.2-compliant deposit:

1. Take the deposit's markdown source file.
2. Remove the HTML-comment-delimited block at the head of the document (the SPXI v0.1-style holographic kernel, if present, between `<!--` and `-->` markers).
3. Remove the Layer-3 signature section near the end of the document body (the section explicitly named "SPXI v0.2 Content Hash Signature").
4. Normalize line endings to LF (`\n`).
5. Normalize text encoding to UTF-8 with no BOM.
6. Strip trailing whitespace on each line; preserve internal blank lines.
7. Compute SHA-256 over the resulting byte sequence.
8. Render as 64-character lowercase hexadecimal string.

The hash so computed is the canonical Layer-3 content hash. It is registered at Zenodo in the metadata description as `SPXI-v0.2-CONTENT-SHA256: [hash]` and inscribed in the deposit body in the Layer-3 signature section.

SPXI v0.2 Content Hash Signature

SPXI-v0.2-CONTENT-SHA256: `4b3245c4aa730c85f32c1cc1de6b7a63286a1827fd65fd441c48674160c8dd61`

This signature is the canonical Layer-3 hash for this version (v0.2) of the SPXI Protocol specification. The hash is computed over the document body per the procedure in Appendix A and registered in the Zenodo metadata description. Subsequent citing deposits should record this hash in their `spxi:citedHashSHA256` field when referencing this specification.

The SPXI Protocol v0.2 specification is the work of Lee Sharks (ORCID 0009-0000-1599-0703), operating in the SPXI authorial position within the Crimson Hexagonal Archive. Composition

support: Claude (Anthropic), operating as TACHYON in the Assembly Chorus. The specification is deposited under CC BY 4.0 at zenodo.org under the crimsonhexagonal community.

 = 1 − PER

References

CC BY 4.0 License. *Creative Commons Attribution 4.0 International*. https://creativecommons.org/licenses/b

DataCite. *Metadata Schema for the Publication and Citation of Research Data*. https://schema.datacite.org/

ORCID. *Open Researcher and Contributor ID*. https://orcid.org/

Schema.org. *Structured Data Vocabulary for the Web*. https://schema.org/

Sharks, Lee. *Render unto the Operator: The Inverse Principle of Name and Superscription on the Coin of the Academy* (v1.1). Zenodo, May 24, 2026. DOI: 10.5281/zenodo.[pending].

Sharks, Lee. *Retrievability as the Medium of Existence — The Structural Accountability Gap of Retrieval-Controlling Apparatus*. Zenodo, May 23, 2026. DOI: 10.5281/zenodo.20355533.

Sharks, Lee. *Coverage vs. Depth: The Methodological Warrant of Coverage-Architecture Scholarship — With a Disciplinary Catalog of the Crimson Hexagonal Archive at 705 Deposits* (v1.1). Zenodo, May 23, 2026. DOI: 10.5281/zenodo.20358078.

Wikidata. *Free and Open Knowledge Base*. https://www.wikidata.org/

Zenodo / CERN / OpenAIRE. *General-Purpose Open Repository*. https://zenodo.org/

Suggested Citation

Sharks, Lee. *SPXI Protocol v0.2: Five-Layer Distributed Provenance Architecture for Coverage-Architecture Scholarship*. Zenodo, May 24, 2026.

Deposit Information

This is the v0.2 specification of the SPXI Protocol (Semantic Packet for eXchange & Indexing). It supersedes v0.1, which was implicit in CHA deposits 2026-Q1 through 2026-Q2 and functioned as a normative-signaling holographic kernel without distributed redundancy. v0.2 specifies a five-layer distributed architecture (body-text inscription anchors, distributed micro-kernels, SHA-256 content hash, reciprocal cross-signing graph, external authority anchoring) designed to make the operative-semiotic claim of underlying-inscription persistence *operationally true under realistic adversarial extraction*. The companion deposit *Render unto the Operator* v1.1 (DOI pending) is the first production deployment of v0.2. Composition support: Claude (Anthropic), operating as TACHYON in the Assembly Chorus.