Network Working Group
Internet-Draft
Intended status: Informational
Expires: 27 September 2026

L. J. Reilly
Independent

March 2026

Reilly Government Integrity Protocol (RGIP): Multi-Layer, Quantum-Resilient Framework for Permanent and Tamper-Evident Public Records

draft-reilly-government-integrity-01

Abstract

The Reilly Government Integrity Protocol (RGIP) defines a standards-aligned method for producing permanent, independently verifiable public records by combining multi-algorithm content hashing, public blockchain anchoring via OpenTimestamps, DOI-based archival, decentralized storage, and web archiving into a single automated pipeline. This revision (01) expands the protocol to incorporate a Triple-Hash Chain architecture running SHA-256 [RFC6234], SHA3-512 [FIPS202], and BLAKE3 [BLAKE3] in parallel, bound together by a cryptographic Cross-Chain Hash at every block. RGIP introduces the REMID (Reilly EternaMark Identifier), a self-sovereign persistent identifier derived directly from artifact content, replacing reliance on externally assigned identifiers as the sole chain anchor. The protocol specifies the Evidence Receipt (ER), an expanded canonicalized metadata object binding the artifact's triple-hash fingerprint, a REMID, a DOI, an OpenTimestamps Bitcoin proof, an IPFS CID, and an Internet Archive URI. Processing steps, chain integrity procedures, autonomous verification, and post-quantum resilience properties are defined.

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 27 September 2026.

Copyright Notice

Table of Contents

1. Introduction

Public institutions produce records whose integrity is foundational to democratic governance, legal accountability, and public trust. Legislative acts, regulatory filings, judicial decisions, agency rulemakings, procurement records, budget authorizations, and intelligence assessments all depend on an unbroken chain of custody from the moment of creation. When that chain is interrupted - whether by accident, negligence, or deliberate suppression, the record loses its evidentiary value and public accountability fails.

Existing mechanisms for establishing record permanence are structurally insufficient. Institutional custody concentrates control in the entity whose conduct the record is meant to document. Paper archives are subject to physical destruction. Certified copies depend on the integrity of the certifying authority. Digital records stored in government-controlled systems are alterable by the same administrators who produce them. None of these mechanisms provides independently verifiable proof that a record existed at a specific moment and has not been modified since.

The threat is not hypothetical. Public records have been altered, selectively deleted, retroactively reclassified, and withheld in contexts ranging from financial regulation to military operations to electoral administration. The absence of a tamper-evident anchoring layer means that any alteration, if discovered at all, is discovered too late.

RGIP addresses this structural gap by anchoring government records to infrastructure that no single institution controls. The Bitcoin blockchain provides a timestamp that cannot be altered without recomputing proof-of-work for the entire subsequent chain, a computation that is infeasible for any actor including nation-states. IPFS provides content-addressed storage independent of any hosting provider. DOI archival at Zenodo provides a citable, persistent reference backed by CERN infrastructure. The Internet Archive provides an additional independent crawl. Together these four layers ensure that no single point of failure, and no single institutional actor, can suppress or alter a record that has been processed through RGIP.

RGIP is relevant to a wide range of government contexts:

* Legislative records: bills, amendments, votes, and committee reports anchored at the moment of publication.

* Regulatory records: rulemaking notices, agency guidance, and public comments with tamper-evident timestamps proving the sequence of publication and response.

* Judicial records: court filings, orders, and opinions with cryptographic proof of content at time of issuance.

* Procurement and contracting: solicitations, awards, and modifications with an immutable audit trail.

* Budget and financial records: appropriations, expenditures, and audit findings anchored before any revision cycle.

* Intelligence and national security: where chain of custody for assessments and supporting evidence is operationally critical.

* Election administration: voter rolls, ballot definitions, and canvass reports with independently verifiable creation timestamps.

RGIP does not mandate a specific regulatory framework or legal standing for anchored records. It provides the cryptographic infrastructure on which legal and regulatory frameworks can rely.

RGIP addresses this by cryptographically anchoring records to infrastructure that no single actor controls.

Version -01 significantly expands the -00 specification in four dimensions:

1. The content hashing layer is upgraded from single-algorithm SHA-256 to a Triple-Hash Chain running SHA-256 [RFC6234], SHA3-512 [FIPS202], and BLAKE3 [BLAKE3] simultaneously. All three chains are bound at every block by a Cross-Chain Hash, meaning an adversary must defeat all three algorithms simultaneously, not independently, to forge a valid record.

2. A self-sovereign identifier, the REMID, is introduced. The REMID is derived deterministically from artifact content and submission timestamp, making it independent of any registrar or external authority. The REMID namespace is defined and normatively specified.

3. The permanence layer is expanded from DOI-plus-blockchain to a four-layer stack: Bitcoin OpenTimestamps anchoring, IPFS content-addressed pinning, DOI archival via Zenodo, and Wayback Machine web archival. Each layer is independently verifiable; failure of any single layer does not invalidate the others.

4. An autonomous agent architecture is defined for continuous chain integrity verification, self-healing of degraded permanence layers, tamper detection, and AI-governed protocol state analysis. This architecture enables RGIP deployments to operate without continuous human oversight.

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.

2. Terminology

Artifact: The public record content being made permanent (e.g., PDF, text, JSON, image, or document bundle).

Triple-Hash Fingerprint: The ordered triple (SHA-256, SHA3-512, BLAKE3) of an Artifact's canonical byte stream, computed simultaneously and stored as the artifact's content identifier.

Cross-Chain Hash: The SHA3-512 digest of the concatenation of the current block's SHA-256 chain hash, SHA3-512 chain hash, and BLAKE3 chain hash. Binds all three chains at every block.

REMID: Reilly EternaMark Identifier. A self-sovereign persistent identifier for an Artifact derived from the artifact's SHA-256 hash prefix and submission date, in the namespace REMID:YYYY.MMDD/<hash8>.

CID: Content Identifier per the IPFS content addressing specification [IPFS-SPEC].

DOI: Digital Object Identifier resolving to an archival landing page for the Artifact.

OTS Proof: An OpenTimestamps [OTS] proof file linking an artifact hash to a Bitcoin block header via a Merkle path.

ER: Evidence Receipt defined by this document.

Block: A single entry in the RGIP chain, corresponding to one Artifact submission.

ni URI: A content-hash URI per [RFC6920].

Anchor: The inclusion of a content hash or commitment in a public blockchain transaction.

Genesis Block: The initial block of an RGIP chain, with all previous chain hash fields set to the all-zeros string of the appropriate length.

3. Protocol Goals and Non-Goals

Goals

* Cryptographically binding each Artifact to a triple-hash fingerprint that is resistant to both classical and quantum adversaries.

* Chaining all Artifacts in a verifiable sequence where any retrospective alteration is detectable.

* Anchoring the chain to a public blockchain providing an immutable, independently observable timestamp.

* Distributing Artifacts and their metadata across multiple independent permanence layers with no single point of failure.

* Enabling autonomous, continuous verification without requiring human oversight.

* Supporting post-quantum migration by including SHA3-512 and BLAKE3 as primary chain algorithms from the outset.

Non-Goals

* Mandating a specific blockchain beyond specifying Bitcoin OpenTimestamps as the RECOMMENDED anchor.

* Defining new cryptographic primitives.

* Governing the legal or regulatory standing of records.

* Prescribing access control policies for Artifact content.

4. High-Level Overview

An RGIP producer processes an Artifact through the following pipeline:

(1) Compute the Triple-Hash Fingerprint (SHA-256, SHA3-512, BLAKE3) of the Artifact’s canonical byte stream.

(2) Derive a REMID from the SHA-256 prefix and submission timestamp.

(3) Compute the new block’s three chain hashes by chaining the previous block’s chain hashes with the current artifact hashes, REMID, and timestamp.

(4) Compute the Cross-Chain Hash binding all three chain hashes.

(5) Anchor the SHA-256 hash to Bitcoin via OpenTimestamps [OTS], receiving a timestamped proof.

(6) Pin the Artifact and its Evidence Receipt to IPFS, obtaining a CID.

(7) Archive the Artifact in a DOI-issuing repository (RECOMMENDED: Zenodo), obtaining a DOI.

(8) Submit the REMID resolver URL to the Internet Archive Wayback Machine for web archival.

(9) Sign and publish the Evidence Receipt (ER) incorporating all of the above.

Verifiers retrieve the ER, recompute all hashes, validate the chain, check each permanence layer independently, and verify the signature.

5. Government Use Cases

This section is informative. It illustrates the application of RGIP to specific government record categories. In each case, the RGIP pipeline produces an Evidence Receipt that is independently verifiable by any party with access to the public permanence layers.

5.1. Legislative and Regulatory Records

A legislative body or regulatory agency publishes a document (bill text, final rule, guidance, or public notice). The agency processes the document through RGIP at the moment of publication, producing an Evidence Receipt with a Bitcoin OTS timestamp, IPFS CID, DOI, and Wayback Machine record.

Any subsequent claim that the document was published at a different time, or that its content differs from the published version, is refutable by recomputing the artifact hashes and validating the Bitcoin timestamp. The timestamp is anchored in a block mined by the global Bitcoin network and cannot be altered retroactively by any party.

5.2. Procurement and Contracting

Government procurement generates records that are frequently disputed: solicitation language, proposal submissions, evaluation scores, and award decisions. Each document in the procurement lifecycle is processed through RGIP at the moment of creation or receipt.

The chain structure ensures that the sequence of records is provable. A chain where a proposal submission precedes a solicitation amendment cannot be forged without invalidating the cross-chain hash for every subsequent block. Audit agencies can independently verify the full procurement record chain without relying on the contracting agency’s internal systems.

5.3. Financial and Budget Records

Appropriations, expenditure authorizations, and audit findings processed through RGIP receive Bitcoin timestamps at the moment of issuance. Any retroactive amendment to a financial record requires creating a new block, preserving the original record in the chain. The original record remains independently retrievable via its REMID, DOI, and IPFS CID regardless of subsequent amendments.

5.4. Judicial Records

Court filings, orders, and opinions processed through RGIP at the moment of issuance carry an immutable creation timestamp. The Triple-Hash Fingerprint ensures that any modification of the document after issuance is detectable by recomputing the artifact hashes. This is particularly relevant for records subject to expungement or retroactive sealing orders, where the existence of an original record may itself be material.

5.5. Intelligence and National Security Records

Assessments, briefings, and supporting evidence processed through RGIP carry a tamper-evident record of their content at the time of production. This is operationally relevant for chain of custody in legal proceedings, for inspector general oversight, and for historical declassification review where the original content must be distinguishable from subsequent revisions.

RGIP’s post-quantum resilience (Section 15) is directly relevant to national security contexts where adversaries with access to quantum computing resources may attempt to undermine the cryptographic integrity of anchored records.

5.6. Election Administration

Voter registration files, ballot definitions, cast vote records, and canvass reports processed through RGIP at defined intervals create an independently verifiable audit trail. The Bitcoin timestamp anchors the state of each file to a specific block, enabling post-election audit by any party with access to the public permanence layers, without reliance on any election administrator's internal systems.

6. Triple-Hash Chain Architecture

RGIP's chain architecture runs three independent hash chains in parallel. Each block N in the chain is defined by three chain hashes and one cross-chain hash.

6.1. SHA-256 Chain

The SHA-256 chain hash at block N is defined as:

```
sha256_chain[N] = SHA-256(
    prev_sha256_chain[N-1] ||
    artifact_sha256 ||
    record_id ||
    remid ||
    timestamp
)
```

where || denotes concatenation of UTF-8 encoded strings and SHA-256 is defined in [RFC6234]. For the Genesis Block, prev_sha256_chain[0] is the all-zeros 64-character hex string.

SHA-256 provides Bitcoin-native compatibility. The SHA-256 hash of the Artifact is the value submitted to OpenTimestamps [OTS], ensuring that Bitcoin timestamping and chain integrity share a common cryptographic root.

6.2. SHA3-512 Chain

The SHA3-512 chain hash at block N is defined as:

```
sha3_chain[N] = SHA3-512(
    prev_sha3_chain[N-1] ||
    artifact_sha3_512 ||
    record_id ||
    remid ||
    timestamp
)
```

where SHA3-512 is defined in [FIPS202]. For the Genesis Block, prev_sha3_chain[0] is the all-zeros 128-character hex string.

SHA3-512 provides 256-bit post-quantum security, meeting the threshold established by NIST for post-quantum cryptographic strength [NIST-PQC]. SHA3-512 is based on the Keccak sponge construction [KECCAK], which is structurally independent of SHA-2 and not susceptible to length-extension attacks. Its inclusion ensures that a quantum adversary capable of attacking SHA-256 via Grover's algorithm [GROVER] cannot simultaneously invalidate the SHA3-512 chain.

6.3. BLAKE3 Chain

The BLAKE3 chain hash at block N is defined as:

```
blake3_chain[N] = BLAKE3(
    prev_blake3_chain[N-1] ||
    artifact_blake3 ||
```

```
        record_id ||
        remid ||
        timestamp
    )
```

where BLAKE3 is defined in [BLAKE3]. For the Genesis Block, prev_blake3_chain[0] is the all-zeros 64-character hex string.

BLAKE3 provides a modern, high-performance hash function designed for parallelism and future-proof security. Its structural independence from both SHA-2 and SHA-3 provides a third, orthogonal attack surface that a chain adversary must overcome simultaneously.

6.4. Cross-Chain Binding Hash

The Cross-Chain Hash at block N is defined as:

```
cross_chain[N] = SHA3-512(
    sha256_chain[N] ||
    sha3_chain[N] ||
    blake3_chain[N]
)
```

The Cross-Chain Hash commits to the state of all three chains simultaneously at every block. Its properties are:

* Any modification to any single chain hash at block N changes cross_chain[N], which cascades forward to invalidate all subsequent cross-chain hashes.

* An adversary attempting to forge a single block must produce valid outputs for SHA-256, SHA3-512, and BLAKE3 simultaneously, then satisfy the SHA3-512 Cross-Chain Hash binding. This converts three independent attack problems into one simultaneous multi-algorithm problem.

* SHA3-512 is used as the outer algorithm for the Cross-Chain Hash because it provides post-quantum resilience [FIPS202], ensuring the binding itself is protected against quantum attacks.

* A verifier can confirm an entire block’s chain state integrity by checking only the cross_chain value, which serves as a single-hash summary of all three chains at that height.

6.5. Chain Integrity Verification

Verifiers MUST validate all four chain hash values at every block:

1. Recompute sha256_chain[N] from the inputs specified in Section 6.1 and compare to the stored value.

2. Recompute sha3_chain[N] from the inputs specified in Section 6.2 and compare to the stored value.

3. Recompute blake3_chain[N] from the inputs specified in Section 6.3 and compare to the stored value.

4. Recompute cross_chain[N] as SHA3-512 of the concatenation of the three stored chain hashes and compare to the stored value.

5. Confirm that prev_sha256_chain[N] equals sha256_chain[N-1], prev_sha3_chain[N] equals sha3_chain[N-1], and prev_blake3_chain[N] equals blake3_chain[N-1].

6. Confirm that the Genesis Block’s prev_* fields are all-zeros

strings of the appropriate length.

Verification MUST fail if any of the above comparisons do not match. Partial verification (checking only a subset of the four hashes) is NOT RECOMMENDED and MUST be disclosed if used.

7. REMID: Self-Sovereign Artifact Identifier

The REMID (Reilly EternaMark Identifier) is a persistent identifier derived from artifact content and submission time. It does not depend on any registrar, certificate authority, or external naming authority.

REMID Format

REMID:YYYY.MMDD/<sha256_prefix8>

where:

* YYYY is the four-digit UTC year of submission.
* MM is the two-digit UTC month of submission.
* DD is the two-digit UTC day of submission.
* sha256_prefix8 is the first eight hexadecimal characters (four bytes) of the Artifact’s SHA-256 digest.

Example:

REMID:2026.0327/3e48edc8

Properties

* Deterministic: The REMID for a given Artifact submitted on a given UTC date is always the same, regardless of the system computing it.

* Content-addressed: The REMID embeds content identity via the SHA-256 prefix, enabling rapid matching against the full artifact hash.

* Self-sovereign: No authority assigns or revokes REMIDs. A REMID is valid if and only if its embedded SHA-256 prefix matches the corresponding Artifact.

* Collision-resistant: The probability of two distinct Artifacts sharing the same REMID on the same date is bounded by the collision resistance of SHA-256 [RFC6234] truncated to 32 bits, approximately 1 in 2^32. Implementations encountering a collision SHOULD append additional hash characters to disambiguate.

REMID Resolver

Implementations SHOULD publish a REMID resolver endpoint at a stable URL that maps a REMID to the corresponding Evidence Receipt and Artifact metadata. The resolver SHOULD return JSON conforming to the ER data model in Section 7.

8. Evidence Receipt (ER) Data Model

The ER is a UTF-8 JSON object [RFC8259], canonicalized per JCS [RFC8785] for signing and hashing. The following members are REQUIRED unless marked OPTIONAL.

```
{
  "er_version": "2",
  "subject": {
```

```
    "title":         string,
    "author":        string,
    "remid":         string,          // REMID per Section 6
    "doi":           string,          // DOI URI OPTIONAL
    "artifact_uri":  string OPTIONAL  // direct URL if available
  },
  "content": {
    "media_type":    string,          // e.g., application/pdf
    "byte_length":   integer,
    "sha256":        string,          // hex lowercase [RFC6234]
    "sha3_512":      string,          // hex lowercase [FIPS202]
    "blake3":        string,          // hex lowercase [BLAKE3]
    "ni":            string           // ni URI per [RFC6920]
  },
  "chain": {
    "block_index":           integer,
    "sha256_chain_hash":     string,  // hex lowercase
    "sha3_chain_hash":       string,  // hex lowercase
    "blake3_chain_hash":     string,  // hex lowercase
    "cross_chain_hash":      string,  // hex lowercase SHA3-512
    "prev_sha256_chain":     string,
    "prev_sha3_chain":       string,
    "prev_blake3_chain":     string,
    "quantum_resilient":     boolean
  },
  "event": {
    "event_type":    string,          // e.g., "publish" | "amend"
    "created":       string           // RFC 3339 timestamp [RFC3339]
  },
  "permanence": {
    "bitcoin": {
      "status":            string,    // "pending" | "confirmed"
      "submitted_at":      string,    // RFC 3339 [RFC3339]
      "block_height":      integer OPTIONAL,
      "block_date":        string OPTIONAL,
      "calendars":         array      // OTS calendar URLs used
    },
    "ipfs": {
      "status":            string,
      "cid":               string OPTIONAL,
      "gateway":           string OPTIONAL
    },
    "zenodo": {
      "status":            string,
      "doi":               string OPTIONAL,
      "record_url":        string OPTIONAL
    },
    "archive": {
      "status":            string,
      "wayback_search":    string OPTIONAL
    }
  },
  "security": {
    "binding_hash":   string,   // SHA-256(record_id||remid||created)
    "binding_fields": string
  },
  "sign": {
    "alg":        string,       // COSE algorithm name or number
    "cose_sign1": string        // base64url COSE_Sign1 [RFC9052]
  }
}
```

Field Notes

* URIs MUST conform to [RFC3986].

* The "ni" value MUST be an ni URI [RFC6920] encoding the SHA-256 content hash.

* Timestamps MUST be RFC 3339 [RFC3339] with UTC offset.

* The "cross_chain_hash" MUST be computed per Section 6.4.

* The "binding_hash" is computed as SHA-256(record_id || remid || created_at) and enables tamper detection independent of the chain hashes.

* Signatures MUST use COSE_Sign1 [RFC9052] with a RECOMMENDED key algorithm of Ed25519 [RFC8032]. Implementations MAY support additional COSE algorithms per [RFC9053].

9. Permanence Layer Stack

9.1. Bitcoin Anchoring via OpenTimestamps

The SHA-256 hash of the Artifact MUST be submitted to one or more OpenTimestamps [OTS] calendar servers. The protocol RECOMMENDS submission to at least three independent OTS calendar servers simultaneously to ensure receipt survival.

The OTS calendar servers aggregate submitted hashes into a Merkle tree [RFC6962] and commit the Merkle root to a Bitcoin block via a Bitcoin OP_RETURN transaction. Once a Bitcoin block confirms the commitment, the OTS proof is "upgraded" to include the full Merkle path from the submitted hash to the Bitcoin block header, providing a compact and independently verifiable timestamp.

Implementations MUST store the raw OTS receipt bytes for each submission. When an upgraded proof becomes available, implementations SHOULD parse the Bitcoin block height SHOULD parse the Bitcoin block height and confirmation date from the proof and update the permanence.bitcoin fields of the ER accordingly.

The Bitcoin timestamp provides an immutable, independently observable timestamp. No institution, including the protocol operator, can alter or remove a committed Bitcoin block header.

9.2. Decentralized Storage via IPFS

The Artifact and its Evidence Receipt JSON SHOULD be pinned to the InterPlanetary File System [IPFS-SPEC] using a pinning service. Pinning produces a Content Identifier (CID) that is a content-addressed, self-describing multihash [MULTIHASH] uniquely identifying the pinned content.

The CID MUST be stored in the permanence.ipfs.cid field of the ER. An IPFS gateway URL resolving the CID to the Artifact SHOULD be stored in permanence.ipfs.gateway.

IPFS provides decentralized, content-addressed storage that remains accessible independently of any single hosting provider. If the original host becomes unavailable, any IPFS node holding a copy of the content can serve it under the same CID.

9.3. DOI Archival via Zenodo

Artifacts SHOULD be archived in a DOI-issuing repository. Zenodo (https://zenodo.org), operated by CERN, is RECOMMENDED due to its open access policy, long-term preservation commitment, and institutional independence.

Upon archival, a DOI is assigned that provides a persistent,

human-readable resolver URL independent of any hosting provider. The DOI and the Zenodo record URL SHOULD be stored in the permanence.zenodo fields of the ER.

DOI archival provides a citable, stable reference that integrates with academic and institutional citation infrastructure.

9.4. Web Archival via Internet Archive

The REMID resolver URL for each Artifact SHOULD be submitted to the Internet Archive Wayback Machine for web archival. This creates a permanent crawl of the public-facing record page, providing an additional, institutionally independent copy of the Artifact metadata.

The Wayback Machine search URL for the REMID resolver SHOULD be stored in permanence.archive.wayback_search of the ER.

10. Step-by-Step Implementation (Normative)

This section is normative.

Step 1: Prepare the Artifact

1. Determine the Artifact format. For JSON, producers MUST canonically serialize using JCS [RFC8785]. For other formats, producers MUST use the exact byte stream intended for publication.

2. Compute the Triple-Hash Fingerprint simultaneously:
   a. SHA-256 [RFC6234] of the canonical byte stream.
   b. SHA3-512 [FIPS202] of the canonical byte stream.
   c. BLAKE3 [BLAKE3] of the canonical byte stream.

3. Construct an ni URI of the form "ni:///sha-256;<digest>" [RFC6920] using the SHA-256 digest.

Step 2: Assign a REMID

4. Determine the UTC submission date.

5. Derive the REMID as REMID:YYYY.MMDD/<sha256_prefix8> per Section 6.

Step 3: Compute Chain Hashes

6. Retrieve the previous block’s sha256_chain_hash, sha3_chain_hash, and blake3_chain_hash. For the Genesis Block, use all-zeros strings.

7. Generate a unique record_id (RECOMMENDED: UUID v4 [RFC9562]).

8. Record the current UTC timestamp in RFC 3339 format [RFC3339].

9. Compute sha256_chain_hash, sha3_chain_hash, and blake3_chain_hash per Sections 5.1, 5.2, and 5.3 respectively.

10. Compute cross_chain_hash per Section 6.4.

11. Compute the binding_hash as SHA-256(record_id || remid || created_at).

Step 4: Anchor to Bitcoin

12. Submit the SHA-256 artifact hash to at least three OTS calendar servers simultaneously. Record submitted_at, calendars used,

and initial receipt bytes.

Step 5: Pin to IPFS

13. Pin the Artifact bytes to IPFS. Record the resulting CID and gateway URL.

Step 6: Archive to DOI Repository

14. Upload the Artifact to Zenodo or another DOI-issuing repository. Record the DOI and record URL.

Step 7: Archive to Wayback Machine

15. Submit the REMID resolver URL to the Internet Archive Wayback Machine. Record the resulting wayback_search URL.

Step 8: Build and Sign the ER

16. Populate all ER fields per Section 7.

17. Canonicalize the ER using JCS [RFC8785].

18. Create a COSE_Sign1 object [RFC9052] over the JCS-canonicalized ER using Ed25519 [RFC8032] (RECOMMENDED).

19. Embed the base64url-encoded COSE_Sign1 in sign.cose_sign1.

Step 9: Publish

20. Publish the Artifact and ER. The DOI landing page and REMID resolver SHOULD link both.

Step 10: OTS Upgrade

21. Monitor OTS calendar servers for proof upgrades. When an upgraded proof is available, parse the Bitcoin block height and date, update permanence.bitcoin.status to "confirmed", and update the ER.

11. Verification Procedure (Normative)

Verifiers MUST perform the following steps:

1. Retrieve the ER for the Artifact to be verified.

2. Compute the Triple-Hash Fingerprint of the Artifact byte stream and confirm that all three digests match the stored values in content.sha256, content.sha3_512, and content.blake3.

3. Recompute all four chain hash values per Section 6.5 and confirm they match the stored values.

4. Verify the cross_chain_hash by computing SHA3-512 of the concatenation of the three stored chain hashes and comparing to the stored cross_chain_hash.

5. Re-derive the REMID from the Artifact’s SHA-256 prefix and submission date and confirm it matches subject.remid.

6. Recompute the binding_hash as SHA-256(record_id || remid || created_at) and confirm it matches security.binding_hash.

7. Validate the OTS proof by:
   a. Parsing the OTS receipt bytes to extract the Merkle path.
   b. Confirming the path connects the SHA-256 artifact hash to

a Bitcoin block header.
c. Confirming the Bitcoin block header appears in the Bitcoin blockchain at the stated block_height.

8. Resolve the CID via any IPFS gateway and confirm the returned content matches the Artifact's SHA-256 digest.

9. Resolve the DOI and confirm the landing page links to the correct Artifact.

10. Re-canonicalize the ER with JCS [RFC8785] and verify the COSE signature [RFC9052] using the publisher's public key.

11. Confirm all timestamps are valid RFC 3339 [RFC3339] and consistent with anchor observation and DOI publication time.

Verifiers SHOULD independently confirm at least three permanence layers (Bitcoin, IPFS, DOI). A record MAY be considered unverified if fewer than two layers return consistent results.

12. Autonomous Agent Architecture

RGIP deployments SHOULD implement autonomous verification agents to provide continuous integrity assurance without human oversight. This section defines the recommended agent roles.

Agent 1 (Hasher):
Computes the Triple-Hash Fingerprint for each incoming Artifact simultaneously in memory. MUST produce all three digests before any subsequent agent runs.

Agent 2 (Bitcoin Anchor):
Submits the SHA-256 hash to OTS calendar servers. SHOULD submit to at least three calendars simultaneously to ensure receipt redundancy.

Agent 3 (REMID):
Derives and mints the REMID for each Artifact per Section 6.

Agent 4 (IPFS):
Pins the Artifact and ER to IPFS using a pinning provider. Stores the resulting CID in the ER.

Agent 5 (Archive):
Submits the REMID resolver URL to the Wayback Machine API.

Agent 5.5 (DOI):
Creates a deposition in a DOI-issuing repository and publishes it to obtain a citable DOI.

Agent 6 (Database):
Persists each ER to a local datastore. SHOULD implement integrity-checked storage with hash verification on read.

Agent 7 (OTS Watcher):
Polls OTS calendar servers for upgraded proofs on pending records. SHOULD poll at intervals not exceeding 15 minutes. When an upgraded proof is found, MUST parse the block height, update permanence.bitcoin.status to "confirmed", and persist the updated ER.

Agent 11 (Self-Heal):
Periodically scans all stored ERs for degraded permanence layers (e.g., IPFS pins that have lapsed, missing Bitcoin submissions, unarchived records) and re-executes the relevant agent for each degraded layer. SHOULD run at intervals not

exceeding one hour.

Agent 13 (Sentinel):
Periodically recomputes all four chain hash values for all stored blocks and compares against stored values per Section 6.5. MUST alert on any discrepancy. SHOULD also monitor OTS calendar server availability and enforce rate limits on artifact submission. SHOULD run at intervals not exceeding 15 minutes.

Agent 14 (Intelligence):
Applies AI-assisted analysis to the protocol state snapshot, producing a risk assessment, findings, and remediation directives dispatched to Agent 11. SHOULD run at intervals not exceeding 30 minutes.

Agents MUST be implemented as independent processes or coroutines with crash isolation. Failure of any single agent MUST NOT prevent other agents from continuing operation. Agents MUST handle CancelledError and similar asynchronous termination signals gracefully.

13. Operational Considerations

Key Management:
Publishers SHOULD use Ed25519 [RFC8032] keys for ER signing. Keys SHOULD be rotated periodically. Public keys SHOULD be published via the DOI landing page or a verifiable channel.

Chain Continuity:
Implementations MUST maintain an unbroken chain of blocks from Genesis. If the datastore is lost, chain state MAY be reconstructed from persisted ERs by sorting by block_index and recomputing chain hashes in order. The reconstruction procedure is equivalent to the backfill operation described in Section 12 (Agent 11).

Updates and Amendments:
Amendments MUST create a new Artifact with a new REMID, a new ER, and a new block. The amended ER SHOULD reference the original REMID in the subject.artifact_uri field.

Multiple Formats:
If the same content is distributed in multiple formats, each format SHOULD receive its own ER to avoid ambiguity.

Interoperability:
Profiles MAY restrict permanence layer implementations, mandate specific COSE algorithms, or define PKI discovery methods to support procurement or sectoral policies.

Calendar Redundancy:
At least three independent OTS calendar servers SHOULD be used for each submission. If a calendar server becomes permanently unavailable, the OTS proof from remaining servers remains independently valid.

14. Security Considerations

Chain Integrity:
RGIP’s chain integrity relies on the collision resistance of all three hash functions. An adversary who can find a SHA-256 collision does not thereby break the SHA3-512 or BLAKE3 chains, and cannot satisfy the SHA3-512 Cross-Chain Hash binding without simultaneously producing valid outputs for all three algorithms.

Tamper Detection:

The binding_hash field (SHA-256 of record_id, REMID, and created_at) provides tamper detection independent of the chain hashes. Implementations SHOULD verify the binding_hash separately from chain verification.

Bitcoin Anchor Security:
Anchoring on the Bitcoin blockchain provides an append-only timestamp. An adversary altering a committed Bitcoin block would require recomputing proof-of-work for that block and all subsequent blocks, which is computationally infeasible for the current Bitcoin network.

Key Compromise:
Compromise of an ER signing key allows an adversary to sign fraudulent ERs but does not allow alteration of the chain hashes or Bitcoin anchors. Publishers SHOULD maintain key revocation metadata.

Replay and Substitution:
Verifiers SHOULD acquire Artifacts from the DOI host over authenticated TLS [RFC8446] and compare size and media type metadata. The REMID derivation ties the identifier to content, limiting substitution attacks.

Rate Limiting:
Implementations SHOULD enforce rate limits on Artifact submission per source address to prevent chain flooding.

15. Post-Quantum Resilience

RGIP is designed to remain cryptographically valid in the presence of large-scale quantum computers.

SHA3-512 Security:
SHA3-512 [FIPS202] provides 256-bit post-quantum security under the model of Grover's algorithm [GROVER]. This meets the NIST minimum threshold for post-quantum hash security [NIST-PQC]. SHA3-512 is based on the Keccak sponge construction [KECCAK], which is structurally orthogonal to SHA-2 and is not weakened by quantum speedups that exploit SHA-2's Merkle-Damgard structure.

BLAKE3 Security:
BLAKE3 [BLAKE3] provides a modern construction with 128-bit classical security (256-bit output truncated by Grover's algorithm to approximately 128-bit quantum resistance). Its structural independence from SHA-2 and SHA-3 provides a third orthogonal chain.

Cross-Chain Binding:
The Cross-Chain Hash uses SHA3-512 as the outer algorithm, ensuring the binding commitment retains post-quantum security that the binding commitment retains post-quantum security even if SHA-256 is weakened by quantum hardware.

Future Migration:
Implementations MAY extend the Triple-Hash Chain to include additional post-quantum hash functions as NIST standards mature (e.g., those based on [NIST-PQC] finalists). The ER data model is designed to accommodate additional content hash fields without breaking backward compatibility.

Signature Migration:
As post-quantum signature standards mature, implementations SHOULD plan migration from Ed25519 [RFC8032] to a NIST-approved post-quantum signature algorithm. COSE [RFC9052] supports algorithm agility, enabling this migration without restructuring

the ER format.

16. Privacy Considerations

ERs SHOULD avoid embedding personal data beyond what is necessary for Artifact identification and attribution. When personal data is unavoidable, publishers SHOULD minimize fields and consider pseudonymization.

Bitcoin anchors are public and permanent. Publishers MUST NOT include sensitive personal data in anchor commitments. The SHA-256 hash of an Artifact is a one-way commitment and does not by itself reveal Artifact content.

IPFS CIDs are public. If Artifact content is sensitive, publishers SHOULD apply appropriate access controls at the pinning layer rather than relying on CID obscurity.

DOI landing pages SHOULD offer appropriate access controls if required by law or policy. Zenodo supports restricted access depositions.

17. IANA Considerations

This document has no IANA actions.

18. References

18.1. Normative References

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

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, July 2002. <https://www.rfc-editor.org/info/rfc3339>

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005. <https://www.rfc-editor.org/info/rfc3986>

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 6234, May 2011. <https://www.rfc-editor.org/info/rfc6234>

[RFC6920] Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B., Keranen, A., and P. Hallam-Baker, "Naming Things with Hashes", RFC 6920, April 2013. <https://www.rfc-editor.org/info/rfc6920>

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, January 2017. <https://www.rfc-editor.org/info/rfc8032>

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

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, December 2017. <https://www.rfc-editor.org/info/rfc8259>

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018.

<https://www.rfc-editor.org/info/rfc8446>

[RFC8785] Johansson, A., "The JSON Canonicalization Scheme (JCS)", RFC 8785, June 2020. <https://www.rfc-editor.org/info/rfc8785>

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", RFC 9052, August 2022. <https://www.rfc-editor.org/info/rfc9052>

[RFC9053] Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", RFC 9053, August 2022. <https://www.rfc-editor.org/info/rfc9053>

[RFC9562] Davis, K., Peabody, B., and P. Leach, "Universally Unique IDentifiers (UUIDs)", RFC 9562, May 2024. <https://www.rfc-editor.org/info/rfc9562>

[FIPS202] National Institute of Standards and Technology, "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", FIPS PUB 202, August 2015. <https://doi.org/10.6028/NIST.FIPS.202>

[BLAKE3] O'Connor, J., Ausmasson, J.-P., Neves, S., and Z. Wilcox-O'Hearn, "BLAKE3: One Function, Fast Everywhere", 2020. <https://github.com/BLAKE3-team/BLAKE3-specs>

18.2. Informative References

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, August 2001. <https://www.rfc-editor.org/info/rfc3161>

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, June 2013. <https://www.rfc-editor.org/info/rfc6962>

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

[OTS] Todd, P., "OpenTimestamps: Scalable, Trust-Minimized, Distributed Timestamping with Bitcoin", 2016. <https://opentimestamps.org>

[IPFS-SPEC] Protocol Labs, "IPFS: Content Addressed, Versioned, P2P File System", 2014. <https://ipfs.tech>

[MULTIHASH] Protocol Labs, "Multihash: Self-Describing Hash Functions", 2015. <https://multiformats.io/multihash/> 2015. <https://multiformats.io/multihash/>

[NIST-PQC] National Institute of Standards and Technology, "Post-Quantum Cryptography Standardization", 2024. <https://csrc.nist.gov/projects/post-quantum-cryptography>

[GROVER] Grover, L.K., "A fast quantum mechanical algorithm for database search", STOC '96, pp. 212-219, 1996.

[KECCAK] Bertoni, G., Daemen, J., Peeters, M., and G. Van Assche, "Keccak reference 3.0", 2011. <https://keccak.team>

[REM] Reilly, L. J., "Reilly EternaMark Protocol (REM): Multi-Layer Permanence for Digital Artifacts", draft-reilly-rem-protocol-01 (work in progress), 2026.

[RGIP00] Reilly, L. J., "Reilly Government Integrity Protocol (RGIP)", DOI:10.5281/zenodo.17114518, September 2025.

Changes from draft-reilly-government-integrity-00

The following substantive changes were made in this revision:

* Section 5 (Triple-Hash Chain Architecture) is new. The single SHA-256 content hash from -00 is replaced by a Triple-Hash Chain running SHA-256 [RFC6234], SHA3-512 [FIPS202], and BLAKE3 [BLAKE3] simultaneously. All three chains are bound at every block by a SHA3-512 Cross-Chain Hash (Section 6.4).

* Section 6.5 (Chain Integrity Verification) is new, providing a normative four-step verification procedure covering all four hash values per block.

* Section 6 (REMID) is new. The REMID replaces reliance on externally assigned DOIs as the sole persistent identifier. The REMID namespace and derivation algorithm are normatively defined.

* Section 7 (Evidence Receipt Data Model) is substantially expanded. The ER version is updated from "1" to "2". New required fields include: content.sha3_512, content.blake3, all chain.* fields including cross_chain_hash, the full permanence.* object covering all four layers, and security.binding_hash.

* Section 8 (Permanence Layer Stack) is new. The -00 Bitcoin anchoring section is expanded and joined by normative subsections for IPFS (Section 9.2), DOI archival via Zenodo (Section 8.3), and Internet Archive web archival (Section 8.4).

* Section 9 (Implementation) is expanded from 12 steps to 21 steps to cover REMID derivation, triple-hash computation, cross-chain hash computation, binding hash computation, IPFS pinning, and Wayback Machine submission.

* Section 10 (Verification) is expanded from 5 steps to 11 steps to cover all three artifact hashes, all four chain hashes, REMID re-derivation, binding hash verification, OTS proof parsing, and IPFS CID resolution.

* Section 12 (Autonomous Agent Architecture) is new, defining normative agent roles for continuous integrity assurance, self-healing, sentinel verification, and AI-assisted protocol state analysis.

* Section 14 (Post-Quantum Resilience) is new, providing detailed analysis of SHA3-512 and BLAKE3 quantum resistance properties, Cross-Chain binding post-quantum properties, and a migration path for signatures.

* Normative references added: RFC 8259, RFC 8446, RFC 9053, RFC 9562, FIPS202, BLAKE3.
FIPS202, BLAKE3.

* Informative references added: NIST-PQC, GROVER, KECCAK, OTS, IPFS-SPEC, MULTIHASH, REM.

Acknowledgments

The author thanks the IETF community for review and discussion.

Author’s Address

Lawrence J. Reilly
Independent

Email: lreilly250@gmail.com