Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 21, 2026

L.J. Reilly
Independent Researcher

March 21, 2026

Reilly Resilience Protocol (RRP): Tamper-Evident Proof of System Resilience

draft-reilly-resilience-protocol-01

Abstract

The Reilly Resilience Protocol (RRP) standardizes a verifiable method to prove that IT systems, cloud infrastructures, and AI pipelines are continuously exercised and resilient. RRP transforms resilience claims into cryptographically signed, tamper-evident evidence persisted in immutable storage and batched into a daily Merkle root that is publicly time-anchored on public blockchains. The protocol outputs an executive Resilience Scorecard backed by independently verifiable cryptographic receipts. RRP composes with the Reilly EternaMark (REM) protocol to ensure dual-layer digital permanence using both DOI archival and blockchain timestamping.

This document supersedes draft-reilly-resilience-protocol-00 and extends the protocol with a formal evidence schema, compliance mapping, a scoring algorithm specification, expanded threat model, implementation guidance, and key management requirements. The foundational whitepaper underpinning this work (Reilly Resilience Protocol Whitepaper v2) is permanently archived at:

DOI: 10.5281/zenodo.17100703
https://zenodo.org/records/17100703

That document was blockchain timestamped and DOI archived on September 11, 2025, at the time of its upload to Zenodo, providing cryptographic attestation of its existence and integrity as of that date. The initial IETF submission (draft-reilly-resilience-protocol-00) was recorded in the IETF Datatracker on September 27, 2025.

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 September 21, 2026.

Copyright Notice

Table of Contents

1. Introduction

Organizations routinely assert claims of "high availability," "recoverability," and "regulatory compliance," yet significant service outages, failed backup restores, configuration drift, and silent AI model degradation continue to occur with alarming frequency. Traditional monitoring logs and dashboard indicators are vulnerable to post-hoc modification, incomplete capture, or absence of objective time-validity proofs. A log that shows 99.9% uptime provides no cryptographic guarantee that the log itself was not modified after the fact.

RRP closes this trust gap by producing daily, tamper-evident proof that critical resilience controls executed successfully. The protocol mandates that all evidence be cryptographically signed, persisted in Write-Once-Read-Many (WORM) or equivalent immutable storage, and anchored into a public blockchain Merkle root for

independent time attestation.

The protocol is designed for three primary audiences:

1. Operators: DevOps, SRE, and security teams who run and automate the actual resilience checks, sign evidence, and maintain the evidence store.

2. Executives: CISOs, CIOs, and board-level stakeholders who consume the Resilience Scorecard and need assurance without burdening themselves with cryptographic details.

3. Auditors and Regulators: Independent third parties who need to re-derive and verify every resilience claim without trusting any single vendor or organizational assertion.

RRP is intentionally designed to be composable with existing monitoring systems, CI/CD pipelines, and cloud-native tooling. Implementers are expected to wrap existing check outputs to produce canonical RRP evidence records rather than replace their monitoring stacks wholesale.

This revision (-01) supersedes draft-reilly-resilience-protocol-00, which was submitted to the IETF on September 27, 2025, and is available at:

https://datatracker.ietf.org/doc/draft-reilly-resilience-protocol/00/

The whitepaper from which this protocol is derived, "The Reilly Resilience Protocol (RRP): A Tamper-Evident, Blockchain-Anchored Framework for Proving IT, Cloud, and AI System Resilience" (Reilly Resilience Protocol Whitepaper v2), is permanently archived under:

DOI: 10.5281/zenodo.17100703
https://zenodo.org/records/17100703

The whitepaper was blockchain timestamped and DOI archived on September 11, 2025, the date of its upload to Zenodo. This blockchain attestation cryptographically proves the document’s existence and integrity as of that date and constitutes timestamped prior art for all protocol concepts described herein.

2. Conventions Used in This Document

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.

3. Terminology

The following terms are defined for the purposes of this document:

Anchor Receipt:
: A cryptographic proof returned by a blockchain anchoring operation, which ties a specific Merkle root to a blockchain transaction identifier and block height. An anchor receipt enables independent verification that the Merkle root was published at or before a given block.

Blockchain Timestamping:
: The process of publishing a cryptographic digest or Merkle root

to a public blockchain, thereby establishing that the committed data existed no later than the block in which the commitment appears. This is distinct from a claim of authorship; it is a claim of existence-by-time.

Collector:
A software agent or automated process responsible for executing a specific resilience control check and emitting a normalized evidence record conforming to the RRP evidence schema defined in Section 6.

Control:
A defined, measurable resilience check with associated SLOs. Examples include database backup restore verification, TLS certificate expiry checks, AI model drift scans, and network partition recovery tests.

DOI Archival:
The process of registering a document or dataset with a Digital Object Identifier (DOI) through a recognized DOI registration agency (such as Zenodo/DataCite), providing a persistent, resolvable identifier that anchors the artifact in a citeable, academically recognized registry.

Dual-Layer Digital Permanence:
A term originating in the work of L.J. Reilly (see REM protocol references) describing the combination of DOI archival and blockchain timestamping as complementary, independent permanence mechanisms. Neither layer alone is considered sufficient; both are required for full permanence assurance.

Evidence Record:
A structured JSON or CBOR document conforming to the RRP evidence schema that captures the result of a single control execution, including metadata, SLO outcome, cryptographic digest, and collector signature.

Evidence Store:
WORM or functionally equivalent immutable object storage in which evidence records and associated artifacts are preserved with configurable retention policies.

Merkle Root:
The root digest of a binary Merkle hash tree constructed over all evidence record digests collected within a given aggregation period (typically one calendar day in UTC).

REM Protocol:
The Reilly EternaMark Protocol. A complementary protocol that specifies dual-layer digital permanence for arbitrary documents and data artifacts using DOI registration and blockchain timestamping. See [DRAFT-REM-00].

Resilience Scorecard:
A structured report, consumable in PDF or dashboard form, that presents weighted pillar scores derived from the day’s evidence records, along with the Merkle root, anchor receipt reference, and independently verifiable evidence URIs.

SLO (Service Level Objective):
A quantitative target associated with a Control. An SLO defines the threshold below which a control result is classified as PASS, WARN, or FAIL.

WORM Storage:
Write-Once-Read-Many storage. Storage in which written objects

cannot be modified or deleted for a configured retention period, providing tamper-evidence for stored evidence records.

4. Protocol Principles

RRP is founded on four non-negotiable design principles:

4.1. Test Restores, Not Backups

Backup existence is not proof of recoverability. Recovery MUST be demonstrated by executing an actual restore operation and validating that the restored artifact is consistent with the original. Merely confirming that a backup job completed is explicitly insufficient evidence under this protocol.

4.2. Evidence Over Narrative

Every resilience claim MUST map to a signed, immutable evidence record stored at a resolvable URI. Narrative assertions such as "our system is highly available" MUST be supported by verifiable evidence records or they carry no weight under RRP.

4.3. Minimum On-Chain

RRP recognizes that publishing full evidence records to public blockchains raises cost, latency, throughput, and privacy concerns. Only the daily Merkle root MUST be anchored on-chain. Full evidence records, artifacts, and supporting logs MUST remain off-chain in the Evidence Store. The on-chain commitment is sufficient to provide time attestation for all off-chain evidence linked through the Merkle tree.

4.4. Composability

RRP is not a replacement for existing monitoring, observability, or testing infrastructure. Existing systems SHOULD be wrapped to produce standardized RRP evidence records. A Collector MAY be a thin adapter that invokes an existing check and transforms its output into the RRP evidence schema.

5. Architecture Overview

5.1. Roles and Responsibilities

RRP defines the following distinct roles within an implementation:

Collector:
: MUST execute one or more assigned Controls on a configured schedule. MUST emit a canonical evidence record per execution. MUST compute the SHA-256 digest of the evidence record prior to signing. SHOULD be isolated such that compromise of one Collector does not compromise others.

Signer:
: MUST hold a signing key (Ed25519 [RFC8032] or ECDSA P-256) in a hardware security module (HSM) or Key Management Service (KMS). MUST sign the SHA-256 digest of each evidence record using the associated private key. MUST NOT expose the private key outside the KMS/HSM boundary. A Collector and Signer MAY be co-located in a single process if the signing key is protected by a KMS API call rather than in-process memory.

Evidence Store:
: MUST accept signed evidence records and associated artifacts

(e.g., restore validation logs, screenshots, structured test outputs). MUST enforce WORM semantics or functionally equivalent immutability for the configured retention period (RECOMMENDED minimum: 7 years for compliance-relevant implementations). MUST return a stable, resolvable URI for each stored record.

Aggregator:
MUST retrieve all evidence records produced within the aggregation window (default: one UTC calendar day). MUST validate each collector signature before including the record in the Merkle tree. Records failing signature validation MUST be excluded and flagged. MUST construct a Merkle tree over the validated set and compute the root digest per the algorithm in Section 7.2. MUST output a signed Aggregate Manifest referencing all included record URIs, their digests, and the Merkle root.

Anchor:
MUST publish the daily Merkle root to at least one public blockchain within the Anchor Window. The Anchor Window opens at 00:00:00 UTC on the day following the aggregation window and MUST close no later than 23:59:59 UTC of that same day. MUST retain the anchor receipt returned by the anchoring service. SHOULD publish to a second independent chain to mitigate single-chain risk.

Scorecard Generator:
SHOULD compute weighted pillar scores per Section 10 and produce a human-readable Resilience Scorecard in PDF or dashboard format. The Scorecard MUST include the Merkle root, an anchor receipt reference, and the aggregation date. The Scorecard SHOULD include evidence record URIs for auditor self-service verification.

Verifier:
Any party wishing to independently verify a resilience claim. A Verifier MUST be able to re-derive evidence digests from stored records, validate Collector signatures against published public keys, verify Merkle inclusion proofs linking individual records to the published Merkle root, and confirm the Merkle root matches the on-chain anchor without trusting any organizational assertion.

5.2. Data Flow

The following describes the end-to-end data flow for a single aggregation cycle:

(1) Collectors execute Control checks on schedule.

(2) Each Collector produces an evidence record (Section 6) and computes its SHA-256 digest.

(3) Each Signer signs the digest; the signature is embedded in the evidence record’s "sig" field.

(4) Signed evidence records are submitted to the Evidence Store, which returns a stable URI for each record.

(5) At end of aggregation window (UTC day boundary), the Aggregator retrieves all records, validates signatures, and constructs the Merkle tree.

(6) The Anchor publishes the Merkle root to the blockchain(s) and retains the anchor receipt.

(7) The Scorecard Generator computes pillar scores from the day's evidence and produces the Resilience Scorecard.

(8) Auditors use the Scorecard's evidence URIs, Merkle root, and anchor receipt to independently verify any claim.

6. Evidence Schema

6.1. Evidence Record Structure

Each evidence record MUST be a valid JSON document [RFC8259] or CBOR document [RFC8949] conforming to the following schema. JSON is RECOMMENDED for interoperability; CBOR is OPTIONAL for constrained environments.

The canonical field names and their semantics are as follows:

"rrp_version" (string, REQUIRED)
The RRP version string. For records conforming to this specification: "1.1".

"record_id" (string, REQUIRED)
A globally unique identifier for this evidence record. MUST be a UUID v4 [RFC4122] formatted as a lowercase hyphenated string.

"control_id" (string, REQUIRED)
A stable, human-readable identifier for the Control that was executed. RECOMMENDED format: reverse-DNS prefix followed by a descriptive slug (e.g., "com.example.db.restore-verify").

"control_version" (string, REQUIRED)
The version string of the Control definition in use, allowing traceability to a specific control specification revision.

"collector_id" (string, REQUIRED)
The stable identifier for the Collector that produced this record, corresponding to a published public key.

"ts_start" (string, REQUIRED)
ISO 8601 timestamp with UTC timezone (ending in "Z") indicating when the Control execution began.

"ts_end" (string, REQUIRED)
ISO 8601 timestamp with UTC timezone indicating when the Control execution completed.

"result" (string, REQUIRED)
One of: "PASS", "WARN", "FAIL", "ERROR". See Section 6.2 for semantics.

"slo_target" (object, REQUIRED)
An object describing the SLO threshold. MUST contain:
"metric": the metric name (string),
"operator": one of "lte", "gte", "lt", "gt", "eq",
"value": the threshold value (number or string),
"unit": the unit of measurement (string).

"slo_actual" (object, REQUIRED)
The same structure as "slo_target" with the "value" field set to the actual measured value.

```
   "artifacts"  (array, OPTIONAL)
      An array of artifact descriptor objects, each containing:
      "uri": resolvable URI to the artifact in the Evidence
             Store (string, REQUIRED),
      "sha256": hex-encoded SHA-256 digest of the artifact
                (string, REQUIRED),
      "type": MIME type or descriptor (string, OPTIONAL).

   "metadata"  (object, OPTIONAL)
      Arbitrary key-value pairs for implementation-specific
      metadata.  Values MUST be strings.

   "digest"  (string, REQUIRED)
      The hex-encoded SHA-256 digest of the canonical
      serialization of this record with the "digest" and
      "sig" fields set to empty strings.  See Section 6.3.

   "sig"  (object, REQUIRED)
      An object containing:
      "alg": the signing algorithm ("Ed25519" or "ES256"),
      "kid": key identifier referencing the Signer's
             published public key,
      "value": base64url-encoded signature over "digest".
```

An example evidence record (JSON, line-wrapped for readability):

```
   {
     "rrp_version": "1.1",
     "record_id": "550e8400-e29b-41d4-a716-446655440000",
     "control_id": "com.example.db.restore-verify",
     "control_version": "2.0.1",
     "collector_id": "collector-db-prod-us-east-1",
     "ts_start": "2026-03-21T00:05:00Z",
     "ts_end":   "2026-03-21T00:23:41Z",
     "result": "PASS",
     "slo_target": {
       "metric": "restore_duration_minutes",
       "operator": "lte",
       "value": 30,
       "unit": "minutes"
     },
     "slo_actual": {
       "metric": "restore_duration_minutes",
       "operator": "lte",
       "value": 18,
       "unit": "minutes"
     },
     "artifacts": [
       {
         "uri": "s3://evidence-store/2026-03-21/restore-log.txt",
         "sha256": "a3f1...c9d2",
         "type": "text/plain"
       }
     ],
     "metadata": { "env": "production", "region": "us-east-1" },
     "digest": "4e2a...b7f0",
     "sig": {
       "alg": "Ed25519",
       "kid": "collector-db-prod-us-east-1-key-v3",
       "value": "base64url-encoded-signature"
     }
   }
```

6.2. Control Result Codes

PASS:

The control executed successfully and the measured SLO actual value satisfies the SLO target threshold. No action required.

WARN:
The control executed successfully but the measured SLO actual value approaches (within a configurable margin of) the SLO target threshold, or a non-critical advisory condition was detected. Implementers SHOULD define warning margins per control.

FAIL:
The control executed but the measured SLO actual value did not satisfy the SLO target threshold. FAIL results MUST trigger alerting and MUST cap the affected pillar score at a maximum of 60 per Section 10.2.

ERROR:
The control execution itself failed to complete (e.g., the Collector encountered an infrastructure error, timeout, or unexpected exception before producing a valid measurement). ERROR results MUST be treated as FAIL for scoring purposes. The distinction is retained for operational root-cause analysis.

6.3. Canonical Serialization

To compute the "digest" field, implementers MUST:

(1) Set the "digest" field to the empty string "".
(2) Set the "sig" field to {"alg":"","kid":"","value":""}.
(3) Serialize the record to UTF-8 JSON with keys in lexicographic (alphabetical) order, no trailing whitespace, and no trailing newline.
(4) Compute SHA-256 over the resulting UTF-8 byte sequence.
(5) Hex-encode the resulting 32-byte digest (lowercase).
(6) Set the "digest" field to this hex-encoded value.
(7) Sign the "digest" value per Section 7.1.

7. Cryptographic Operations

7.1. Evidence Signing

Collector signing keys MUST be one of the following algorithms:

Ed25519 [RFC8032]:
RECOMMENDED. Produces 64-byte signatures. Key pairs are 64 bytes (private) and 32 bytes (public). Deterministic; does not require a random number generator at signing time.

ECDSA with P-256 and SHA-256 (ES256):
ACCEPTABLE where Ed25519 is not supported by the KMS. Implementers SHOULD use deterministic ECDSA [RFC6979] to mitigate nonce-reuse vulnerabilities.

RSA-PSS with SHA-256 (PS256):
MAY be used in legacy environments where neither Ed25519 nor ECDSA is available. Minimum key size: 3072 bits. NOT RECOMMENDED for new implementations.

All Signer public keys MUST be published in JWK format [RFC7517] at a stable, resolvable URI and SHOULD be published in a JWK Set document alongside the collector identity metadata.

Signed evidence records SHOULD additionally be wrapped in a COSE Sign1 structure [RFC9052] when interoperability with

COSE-aware verification toolchains is required.

7.2. Merkle Tree Construction

The Aggregator MUST construct a binary Merkle tree using the following algorithm:

(1) Collect the set D = {d_1, d_2, ..., d_n} of SHA-256 digests from all validated evidence records in the aggregation window, sorted in ascending lexicographic order.

(2) If n = 0, no anchor SHALL be produced for this window. The Aggregator MUST log the empty window condition.

(3) If n is odd, duplicate the last element: append d_n to D so that |D| is even. This is consistent with the construction in [RFC9162].

(4) Compute parent nodes by hashing pairs:
parent(i) = SHA-256(0x01 || child_left || child_right)
Leaf nodes are computed as:
leaf(d_i) = SHA-256(0x00 || d_i)
The 0x00/0x01 domain separation prefixes prevent second-preimage attacks on the tree [RFC9162] Section 2.1.

(5) Recurse until a single root digest remains.

(6) The resulting root is the Merkle Root for this window.

Inclusion proofs for individual evidence records MUST be generated and stored alongside the Aggregate Manifest, enabling any Verifier to prove that a specific evidence record is included in the published Merkle root without requiring access to all other records.

7.3. Blockchain Anchoring

The Anchor MUST publish the Merkle root using at least one of the following mechanisms:

Bitcoin via OpenTimestamps:
The Merkle root is committed using OpenTimestamps' OP_RETURN embedding or equivalent calendar server mechanism. The returned .ots file constitutes the anchor receipt and MUST be retained alongside the Aggregate Manifest. RECOMMENDED for implementations prioritizing immutability and chain longevity.

Ethereum or EVM-Compatible L2:
The Merkle root MAY be submitted as calldata in a standard Ethereum transaction, or via a purpose-built smart contract. The transaction hash and block number constitute the anchor receipt. Implementers SHOULD prefer a Layer 2 network with Ethereum settlement for cost efficiency.

The Anchor Window is defined as the 24-hour UTC period immediately following the close of the aggregation window. Anchoring MUST complete within the Anchor Window.

SHOULD use two independent chains (e.g., Bitcoin and an EVM chain) to mitigate the risk of a single chain reorganization or availability failure invalidating the anchor.

The on-chain commitment MUST include only the Merkle root digest and a minimal protocol identifier. Full evidence records, PII, and sensitive operational data MUST NOT be published on-chain.

8. Key Management

Proper key management is essential to the integrity of all RRP evidence. The following requirements apply:

(1) All Signer private keys MUST be stored in a hardware security module (HSM) or a cloud KMS with HSM-backed key storage (e.g., AWS CloudHSM, Google Cloud HSM, Azure Dedicated HSM).

(2) Private keys MUST NOT be stored in filesystem files, source control, container images, or environment variables.

(3) Each Collector MUST have a distinct signing key pair. Shared keys across Collectors are NOT RECOMMENDED as they prevent individual Collector attribution in the event of compromise.

(4) Collector public keys MUST be published to a well-known endpoint prior to the Collector emitting any evidence records. The published JWK Set MUST be updated within 24 hours of any key rotation.

(5) Signing keys MUST be rotated at least annually. Key rotation MUST NOT invalidate previously issued signatures. The previous public key MUST remain published and resolvable for the duration of the evidence retention period.

(6) Key compromise MUST be declared by publishing a revocation notice at the public key endpoint within 24 hours of discovery. All evidence records signed by the compromised key after the declared compromise time MUST be treated as invalid.

(7) Implementers SHOULD maintain an out-of-band key backup in a geographically separate HSM to enable recovery without evidence continuity interruption.

9. Operational Procedures

9.1. Control Definition

Implementers MUST define a minimum of 5 Controls and SHOULD define 8 to 12 for a production deployment. Each Control MUST have:

(a) A stable "control_id" (reverse-DNS format RECOMMENDED).

(b) A human-readable description and rationale.

(c) A measurable SLO target with a specific metric, operator, threshold value, and unit.

(d) An assigned Collector or set of Collectors.

(e) An execution schedule (RECOMMENDED: at minimum once per 24-hour period).

(f) Mapping to at least one compliance framework control (see Section 11) where applicable.

The following are RECOMMENDED baseline Controls:

Backup Restore Verification:
   Execute a full restore of a production backup to an
   isolated environment. Validate restored artifact
   consistency. SLO example: restore completes in <= 30 min
   with 100% data integrity checksum match.

TLS Certificate Expiry:
   Verify all externally facing TLS certificates have
   remaining validity > N days (RECOMMENDED: N >= 30).

Database Replication Lag:
   Confirm replication lag across all read replicas is
   below a defined threshold.

AI Model Drift Scan:
   Compare current inference distribution against a
   validated baseline distribution. Trigger WARN if
   drift exceeds a configurable threshold; FAIL if it
   exceeds a critical threshold.

Network Partition Recovery:
   Inject a controlled network partition and verify
   that the system recovers within the defined RTO.

Incident Response Drill:
   Log evidence of a completed tabletop or live incident
   response exercise, including participant count, scenario,
   and findings.

9.2. Evidence Collection

Each Collector MUST:

(1) Execute the assigned Control check at the scheduled time.

(2) Record ts_start immediately before execution begins.

(3) Capture raw output, logs, and any relevant artifacts.

(4) Map the result to one of: PASS, WARN, FAIL, ERROR.

(5) Populate all REQUIRED fields of the evidence schema
    (Section 6.1).

(6) Store raw artifacts in the Evidence Store and populate
    the "artifacts" array with their URIs and digests.

(7) Compute the canonical serialization (Section 6.3) and
    set the "digest" field.

(8) Submit the pre-signature record to the Signer.

Collectors MUST be designed to be idempotent with respect to
re-execution. A re-run of the same Control within the same
aggregation window MUST produce a new record with a unique
"record_id"; it MUST NOT overwrite the prior record.

9.3. Signing and Storage

The Signer MUST:

(1) Receive the pre-signature evidence record from a Collector
    over an authenticated, encrypted channel (TLS 1.3 REQUIRED
    per [RFC8446]).

(2) Re-compute the canonical digest and verify it matches

the submitted "digest" field before signing. A mismatch MUST result in rejection of the record.

(3) Sign the digest using the Collector’s assigned private key.

(4) Return the signed evidence record to the Collector.

The Evidence Store MUST:

(1) Accept only signed evidence records (records missing a valid "sig" block MUST be rejected).

(2) Verify the Collector signature against the published public key before storing.

(3) Return a stable, resolvable URI for the stored record.

(4) Enforce the configured WORM retention period.

9.4. Aggregation and Anchoring

At the close of each aggregation window, the Aggregator MUST:

(1) Query the Evidence Store for all records with "ts_end" falling within the closed window.

(2) For each record, validate the Collector signature against the published JWK Set. Exclude records failing validation.

(3) Log all excluded records with the reason for exclusion.

(4) Construct the Merkle tree per Section 7.2.

(5) Produce and sign an Aggregate Manifest containing:
- Aggregation window (ts_window_start, ts_window_end),
- List of included record URIs and their digests,
- List of excluded record identifiers and reasons,
- Merkle root (hex-encoded SHA-256),
- Aggregator’s own signature over the manifest.

(6) Store the Aggregate Manifest in the Evidence Store.

(7) Submit the Merkle root to the Anchor.

(8) Receive and store the anchor receipt alongside the Aggregate Manifest.

9.5. Scorecard Publication

The Scorecard Generator SHOULD publish the Resilience Scorecard within 4 hours of the Anchor completing its on-chain commitment.

The Scorecard MUST include:

(a) The aggregation date (UTC).
(b) The overall RRP composite score (0-100).
(c) Per-pillar scores with weights and contributing controls.
(d) Result summary (counts of PASS, WARN, FAIL, ERROR).
(e) The Merkle root for the period.
(f) A reference to the anchor receipt (chain and txid or proof identifier).
(g) Evidence Store URIs for all included records.

The Scorecard SHOULD be produced in a machine-parseable format (JSON) in addition to a human-readable format (PDF).

9.6. Independent Verification

Any Verifier MUST be able to perform the following steps without access to internal systems or organizational trust:

(1) Retrieve a specific evidence record from its published URI.

(2) Re-compute the canonical serialization and SHA-256 digest, confirming it matches the "digest" field in the record.

(3) Retrieve the Collector’s public key from the published JWK Set using the "kid" in the "sig" field.

(4) Validate the signature in the "sig" field over the "digest" value using the retrieved public key.

(5) Retrieve the Aggregate Manifest for the relevant window.

(6) Confirm the record’s digest appears in the manifest’s included record list.

(7) Recompute the Merkle root from the manifest’s record list and the provided inclusion proof.

(8) Confirm the recomputed Merkle root matches the manifest’s stated Merkle root.

(9) Confirm the Merkle root matches the on-chain commitment by independently querying the referenced blockchain.

A Verifier completing all nine steps has independently confirmed the evidence record’s authenticity without trusting any organizational assertion.

10. Resilience Scorecard Specification

10.1. Pillars and Weights

RRP organizes resilience across five standard Pillars. Implementers MAY redefine pillar weights for their specific risk profile, provided weights sum to 1.0 (100%). The default weights are:

Pillar	Symbol	Weight
Recovery Capability	RC	0.25
Availability & Continuity	AC	0.25
Security Integrity	SI	0.20
AI Pipeline Resilience	AI	0.15
Operational Compliance	OC	0.15
Total		1.00

Each Control defined in Section 9.1 MUST be assigned to exactly one Pillar. Implementers SHOULD ensure at least one Control is assigned to each active Pillar.

10.2. Scoring Algorithm

For each Pillar P, let:

C_P = set of Controls assigned to Pillar P
r(c) = result of Control c: PASS=100, WARN=75, FAIL=0, ERROR=0

The raw pillar score is:

```
raw_P = (sum of r(c) for c in C_P) / |C_P|
```

The capped pillar score accounts for critical failures:

```
If any c in C_P has result FAIL or ERROR:
   pillar_P = min(raw_P, 60)
Else:
   pillar_P = raw_P
```

The composite RRP score is:

```
RRP_score = sum over P of ( weight_P * pillar_P )
```

Where weight_P is the pillar weight from Section 10.1.

The composite score is a value in [0, 100]. For reporting purposes, RECOMMENDED bands are:

```
[90, 100]: Excellent - All controls passing with strong SLO
           margin.
[75, 90):  Good - Minor warnings present; review recommended.
[60, 75):  Moderate - Multiple warnings or a capped pillar;
           remediation required.
[0, 60):   At Risk - One or more critical failures; immediate
           remediation required.
```

10.3. Score Decay

Evidence currency is critical to the integrity of the Scorecard. To prevent stale evidence from inflating scores:

(1) If a Control has not produced any evidence record within the 48-hour period ending at scorecard generation time, its result MUST be treated as ERROR for scoring purposes.

(2) If a Control's most recent record has "result": "ERROR" due to staleness, the Scorecard Generator MUST annotate the Scorecard with a "stale evidence" warning for that Control.

(3) Implementers MAY define shorter decay windows for higher-frequency controls (e.g., controls scheduled hourly may apply a 4-hour decay window).

11. Compliance Mapping

RRP evidence records directly support audit evidence requirements across the following compliance frameworks:

SOC 2 (Trust Services Criteria):
   CC7.1 (System Monitoring), CC7.2 (Anomaly Identification),
   A1.2 (Recovery Plan Testing), A1.3 (Recovery Objectives).
   RRP Pillars: Recovery Capability, Availability & Continuity.

ISO 27001:2022:
   Annex A 8.6 (Capacity Management),
   Annex A 5.29 (Information Security During Disruption),
   Annex A 5.30 (ICT Readiness for Business Continuity).
   RRP Pillars: Recovery Capability, Operational Compliance.

NIST SP 800-53 Rev 5:
   CP-9 (System Backup), CP-10 (System Recovery and

Reconstitution), SI-12 (Information Management and Retention), SI-7 (Software, Firmware, and Information Integrity). RRP Pillars: All pillars.

PCI DSS v4.0:
Requirement 10 (Log and Monitor All Access), Requirement 12.10 (Incident Response Plan).
RRP Pillars: Security Integrity, Operational Compliance.

HIPAA Security Rule (45 CFR Part 164):
164.308(a)(7) (Contingency Plan), 164.312(c)(1) (Integrity).
RRP Pillars: Recovery Capability, Security Integrity.

DORA (EU Digital Operational Resilience Act):
Article 11 (Backup Policies), Article 12 (Recovery Plans and Procedures), Article 25 (Advanced TLPT).
RRP Pillars: Recovery Capability, Availability & Continuity, Operational Compliance.

Implementers SHOULD include compliance framework mapping metadata in the Control definition and in the "metadata" field of evidence records to facilitate automated compliance reporting.

12. Threat Model

12.1. Adversary Classes

RRP considers the following adversary classes:

Internal Attacker (IA):
An employee or contractor with access to Collector infrastructure who may attempt to falsify evidence records or suppress evidence of failures.

External Attacker (EA):
An adversary without initial access who may attempt to tamper with stored evidence records, compromise Collector signing keys, or perform a blockchain reorganization.

Collusion (CO):
Two or more internal parties (e.g., Collector operator and Evidence Store administrator) colluding to produce and store fraudulent evidence records.

Availability Adversary (AA):
An adversary whose goal is to prevent the Anchor from publishing the Merkle root within the Anchor Window, thereby creating gaps in the evidence chain.

12.2. Attack Scenarios and Mitigations

Scenario 1: Evidence Record Tampering
Threat: IA or EA modifies an evidence record after storage.
Impact: Fraudulent resilience claim.
Mitigation: WORM Evidence Store prevents modification.
Verification: Canonical digest mismatch detected by any Verifier recomputing the digest.

Scenario 2: Collector Key Compromise
Threat: EA compromises a Collector’s private signing key and generates fraudulent signed evidence records.
Impact: Fraudulent records accepted by the Aggregator.
Mitigation: HSM/KMS key storage (Section 8 requirement 1). Key revocation (Section 8 requirement 6) limits forward impact. Per-Collector keys (Section 8 requirement 3)

limit blast radius.

Scenario 3: Aggregator Manipulation
Threat: IA manipulates Aggregator to exclude FAIL records from the Merkle tree or compute an incorrect root.
Impact: Inflated score, incomplete audit trail.
Mitigation: Aggregate Manifest is signed and stored in WORM. Excluded record log (Section 9.4 step 3) enables detection. Verifiers can independently recompute the Merkle root from the manifest and confirm it matches the on-chain anchor.

Scenario 4: On-Chain Reorg
Threat: EA performs a blockchain reorganization that removes the Merkle root anchor transaction.
Impact: Anchor receipt invalidated.
Mitigation: Wait for sufficient confirmation depth before treating an anchor as finalized. RECOMMENDED: 6 blocks for Bitcoin, 32 blocks (finalized checkpoint) for Ethereum. Use of two independent chains (Section 7.3) means both must be reorganized simultaneously.

Scenario 5: Anchor Window Denial
Threat: AA disrupts Anchor infrastructure, preventing on-chain publication within the Anchor Window.
Impact: Gap in the evidence chain for that day.
Mitigation: Redundant Anchor agents. Late anchoring MUST be flagged in the Scorecard with an explanation. Anchor receipts anchored outside the window remain valid evidence but MUST reduce the Scorecard score for the affected window.

13. Security Considerations

13.1. Cryptographic Agility

The signing algorithm and hash function are specified per-record via the "sig.alg" field and are expected to evolve. Implementers MUST NOT hard-code algorithm identifiers into verifier logic. Verifiers MUST support at minimum Ed25519 and ES256 (P-256 ECDSA with SHA-256).

This specification mandates SHA-256 for evidence and Merkle tree hashing. Future versions of this protocol MAY specify SHA-384 or SHA-3 variants as cryptographic standards evolve.

13.2. Replay Attacks

An adversary who obtains a valid signed evidence record for a PASS result may attempt to replay it for a future window. The "ts_start" and "ts_end" fields provide temporal scope. The Aggregator MUST reject records whose "ts_end" falls outside the current aggregation window. The "record_id" UUID prevents exact-duplicate submission within the same window.

13.3. Sybil Attacks on Public Key Publication

An adversary may publish a forged public key at a URL designed to resemble a legitimate Collector’s key endpoint. Implementers MUST publish Collector public keys at URLs under organizational control with DNSSEC enabled [RFC4033] and protected with TLS [RFC8446]. Verifiers MUST validate the TLS certificate of the key endpoint.

13.4. Privacy

Evidence records MAY contain operationally sensitive data

(e.g., system names, IP addresses, restore durations). They MUST NOT contain personally identifiable information (PII) or protected health information (PHI). Artifacts stored in the Evidence Store that contain sensitive data MUST be encrypted at rest. Only the artifact's URI and SHA-256 digest appear in the evidence record; the artifact itself is not on-chain.

13.5. Chain Selection

Implementers SHOULD select blockchain networks with established proof-of-work or finalized proof-of-stake security. Chain selection criteria SHOULD include: years of continuous operation, total value secured, and reorganization history.

13.6. Transport Security

All communications between Collectors, Signers, Evidence Stores, and Aggregators MUST use TLS 1.3 [RFC8446]. Certificate validation MUST be enforced; self-signed certificates are NOT RECOMMENDED in production deployments.

14. Relationship to Other RRP-Suite Protocols

RRP is designed as one member of a suite of tamper-evident assurance protocols sharing a common architectural lineage.

REM Protocol [DRAFT-REM-00]:
: The Reilly EternaMark Protocol defines dual-layer digital permanence for arbitrary artifacts via DOI registration and blockchain timestamping. RRP evidence packages SHOULD be archived using REM by publishing Aggregate Manifests and Scorecard artifacts as DOI-archived objects. The Merkle root and anchor receipt SHOULD be embedded in the REM metadata envelope.

RSP (Reilly Sentinel Protocol):
: Focuses on AI model behavioral integrity and inference attestation. RSP-generated model integrity proofs MAY be referenced as artifacts in RRP evidence records for the "AI Pipeline Resilience" pillar.

RBIP (Reilly Banking Integrity Protocol):
: Extends RRP with domain-specific controls for financial institution resilience, including transaction reconciliation proofs and regulatory reporting anchoring.

RGIP (Reilly Government Integrity Protocol):
: Extends RRP with controls for government IT and public record integrity, including FISMA alignment and FedRAMP control mapping.

CTS (Cognitive Trust Stack):
: The Cognitive Trust Stack [DRAFT-CTS-00] addresses AI behavioral provenance, bridging alignment claims and cryptographic proof. CTS-generated behavioral attestations MAY be incorporated as evidence in the AI Pipeline Resilience pillar.

These protocols share a foundational architecture combining cryptographic signing, immutable storage, Merkle tree aggregation, and blockchain anchoring. They are designed to interoperate and compose while remaining independently deployable.

15. Implementation Guidance

This section provides non-normative guidance for implementers.

15.1. Cloud-Native Reference Architecture

The following reference mapping to cloud primitives is provided:

Evidence Store: AWS S3 with Object Lock (COMPLIANCE mode), Google Cloud Storage with Bucket Lock, or Azure Blob Storage with immutability policies.

Signer / KMS: AWS KMS with CloudHSM key material, Google Cloud HSM, or Azure Dedicated HSM.

Anchor (Bitcoin/OTS): OpenTimestamps calendar client or self-hosted calendar server.

Anchor (Ethereum): A dedicated EOA submitting calldata transactions, or integration with a notarization service (e.g., Chainlink Proof of Reserve, custom smart contract).

Aggregator: A scheduled Lambda/Cloud Function/Azure Function triggered at UTC midnight.

Scorecard Generator: A serverless rendering pipeline producing PDF via headless browser or LaTeX.

15.2. Bootstrapping Sequence

New implementers SHOULD follow this bootstrapping sequence:

(1) Define Controls and SLOs per Section 9.1.
(2) Provision KMS keys for each Collector.
(3) Publish Collector public keys at well-known URLs.
(4) Deploy Collectors with automated scheduling.
(5) Deploy Evidence Store with WORM and TLS.
(6) Deploy Aggregator with WORM-write access.
(7) Configure Anchor agent(s).
(8) Produce first Scorecard and verify end-to-end using the nine-step procedure in Section 9.6.

15.3. Incremental Adoption

Organizations with existing monitoring infrastructure SHOULD begin by wrapping one or two controls (e.g., backup restore and TLS certificate expiry) to validate the pipeline before expanding to the full Control set. The Evidence Schema (Section 6) is designed to be populated from existing check tool outputs with minimal transformation.

16. IANA Considerations

This document has no IANA actions at this time.

A future revision of this specification MAY request registration of the following:

(a) A media type (application/rrp-evidence+json) for RRP evidence records.

(b) A well-known URI suffix (/.well-known/rrp-keys) for Collector public key publication.

(c) A Structured Syntax Suffix (+rrp) for RRP-formatted JSON documents.

These registrations will be pursued through the appropriate IANA process when the protocol achieves sufficient implementation and community review.

17. References

17.1. Normative References

[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>.

[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>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[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>.

[RFC9052] Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", STD 96, RFC 9052, DOI 10.17487/RFC9052, August 2022, <https://www.rfc-editor.org/info/rfc9052>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC6979] Pornin, T., "Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 6979, DOI 10.17487/RFC6979, August 2013, <https://www.rfc-editor.org/info/rfc6979>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/info/rfc7517>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[RFC9162] Laurie, B., Langley, A., Kasper, E., Messeri, E., and R. Stradling, "Certificate Transparency Version 2.0", RFC 9162, DOI 10.17487/RFC9162, December 2021, <https://www.rfc-editor.org/info/rfc9162>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

17.2. Informative References

[DRAFT-REM-00]
Reilly, L.J., "Reilly EternaMark (REM) Protocol - Dual-Layer Digital Permanence Using DOI Archiving and Blockchain Timestamping", Work in Progress, Internet-Draft, draft-reilly-rem-protocol-00, September 2025, <https://datatracker.ietf.org/doc/draft-reilly-rem-protocol/>.

[DRAFT-RRP-00]
Reilly, L.J., "Reilly Resilience Protocol (RRP): Tamper-Evident Proof of System Resilience", Work in Progress, Internet-Draft, draft-reilly-resilience-protocol-00, September 2025, <https://datatracker.ietf.org/doc/draft-reilly-resilience-protocol/00/>.

[DRAFT-CTS-00]
Reilly, L.J., "Cognitive Trust Stack (CTS): Cryptographic Behavioral Provenance for AI Systems", Work in Progress, Internet-Draft, draft-reilly-cts-00, 2026, <https://datatracker.ietf.org/doc/draft-reilly-cts/>.

[RRP-WHITEPAPER-V2]
Reilly, L.J., "The Reilly Resilience Protocol (RRP): A Tamper-Evident, Blockchain-Anchored Framework for Proving IT, Cloud, and AI System Resilience", Zenodo, DOI: 10.5281/zenodo.17100703, September 11, 2025, <https://zenodo.org/records/17100703>.

NOTE: This whitepaper was blockchain timestamped and DOI archived on September 11, 2025, at the time of its upload to Zenodo. The blockchain timestamp constitutes cryptographic attestation of the document’s existence and integrity as of that date.

[NIST-800-53]
National Institute of Standards and Technology, "Security and Privacy Controls for Information Systems and Organizations", NIST Special Publication 800-53 Rev 5, September 2020, <https://doi.org/10.6028/NIST.SP.800-53r5>.

[OPENTIMESTAMPS]

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

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", RFC 6962, DOI 10.17487/RFC6962, June 2013, <https://www.rfc-editor.org/info/rfc6962>.

Appendix A. Changes from -00

The following substantive changes were made from draft-reilly-resilience-protocol-00 to this revision (-01):

A.1. Protocol Provenance and Archival

Added explicit references to the foundational whitepaper archived at DOI 10.5281/zenodo.17100703, with notation that the document was blockchain timestamped and DOI archived on September 11, 2025. Added reference to the -00 IETF draft recorded September 27, 2025.

A.2. Evidence Schema (Section 6)

Fully specified the evidence record JSON schema, including all required and optional fields, canonical serialization procedure, and an annotated example. Added Control result codes (PASS, WARN, FAIL, ERROR) with precise semantics.

A.3. Cryptographic Operations (Section 7)

Expanded signing algorithm requirements. Added explicit Merkle tree construction algorithm with domain-separation prefixes per RFC 9162. Replaced RFC 8152 (COSE, now obsolete) with RFC 9052. Added deterministic ECDSA (RFC 6979) as a SHOULD for ES256 implementations.

A.4. Key Management (Section 8)

Added a dedicated key management section with seven explicit requirements covering HSM-backed storage, per-Collector key isolation, annual rotation, and revocation procedures.

A.5. Operational Procedures (Section 9)

Expanded control definition requirements, including minimum control count, compliance mapping, and baseline recommended controls. Added independent verification nine-step procedure and Scorecard publication timeline requirement.

A.6. Resilience Scorecard Specification (Section 10)

Formally specified five pillars with default weights. Provided explicit scoring algorithm in mathematical notation. Added score bands and score decay rules.

A.7. Compliance Mapping (Section 11)

Added specific control citations for SOC 2, ISO 27001:2022, NIST 800-53 Rev 5, PCI DSS v4.0, HIPAA Security Rule, and DORA.

A.8. Threat Model (Section 12)

Added a formal threat model covering four adversary classes and five attack scenarios with specific mitigations.

A.9. Security Considerations (Section 13)

Expanded from one paragraph to six subsections covering cryptographic agility, replay attacks, Sybil attacks on key publication, privacy, chain selection, and transport security.

A.10. Protocol Suite Context (Section 14)

Added references to RSP, RBIP, RGIP, and CTS within the broader RRP-suite context, with integration notes.

A.11. Implementation Guidance (Section 15)

Added cloud-native reference architecture, bootstrapping sequence, and incremental adoption guidance.

A.12. Reference Corrections

Replaced obsolete RFC 8152 (COSE) with RFC 9052. Added RFC 8032 (EdDSA), RFC 6979 (Deterministic ECDSA), RFC 7517 (JWK), RFC 7519 (JWT), RFC 8446 (TLS 1.3), RFC 4122 (UUID), RFC 4033 (DNSSEC), and RFC 6962 (CT v1).

Author's Address

Lawrence John Reilly Jr.
Independent Researcher

Email: lawrencejohnreilly@gmail.com
IETF Datatracker:
https://datatracker.ietf.org/person/lawrencejohnreilly@gmail.com
Zenodo Archive:
https://zenodo.org/records/17100703
DOI: 10.5281/zenodo.17100703