Deterministic Replay Bundles, Tamper-Evident Logging, and Signed Configuration Artifacts for Rotating-Field Experiments

Docket: NTL-PROV-012

Inventors: [Inventor Names]

Assignee: [Assignee / Organization]

Date: February 1, 2026

Abstract

Disclosed are systems, methods, and non-transitory computer-readable media for producing deterministic replay bundles and tamper-evident run artifacts for rotating-field experiments and commutated electromagnetic devices. In various embodiments, a run artifact includes: (i) a canonical configuration snapshot (geometry ID, schedule ID, firmware/software identifiers, calibration parameters, and safety envelope parameters), (ii) raw synchronized sensor streams, (iii) derived features and gate outcomes, (iv) a cryptographic integrity structure (hashes and/or Merkle trees) covering all run components, and (v) one or more digital signatures over a manifest describing the run artifact. The system optionally anchors the run hash to an external timestamp service or distributed ledger to provide an independent time ordering.

The disclosure further provides deterministic replay procedures that ensure a future run can reproduce the same schedule intent and waveform realization within specified tolerances by replaying the signed configuration and applying stored calibration parameters. The disclosed provenance framework improves scientific credibility, enables litigation-grade audit trails, and reduces risk of undetected tampering or configuration drift.

Technical Field

The present disclosure relates to data provenance and experimental integrity, and more particularly to deterministic replay bundles, tamper-evident logging, cryptographic hashing, and signed configuration artifacts for rotating-field experiments, resonant system operation, and associated instrumentation workflows.

Background

Experimental claims in high-sensitivity domains are often undermined by irreproducible configurations, undocumented calibration parameters, missing metadata, and unverifiable data provenance. In complex electro-mechanical experiments, the effective "configuration" includes not only nominal setpoints, but also firmware versions, clock calibration, sensor placements, filtering settings, and safety thresholds.

Additionally, as datasets grow and are shared across teams, it becomes difficult to prove that a dataset is authentic, untampered, and corresponds to the configuration claimed. Without tamper-evident mechanisms, accidental or malicious modifications to data, configuration files, or analysis outputs can go undetected.

Accordingly, there is a need for a deterministic and auditable system that (i) captures complete run configuration and calibration parameters, (ii) records synchronized data streams and derived outputs, (iii) provides tamper-evident integrity, (iv) supports deterministic replay, and (v) optionally provides independent timestamping.

Summary

This disclosure provides a provenance framework centered on a *run manifest* and a *run artifact bundle*.

In one aspect, a run manifest is a structured record containing identifiers, configuration snapshots, file listings, and cryptographic hashes for all bundle components. The run manifest is digitally signed.

In another aspect, a run bundle contains raw sensor streams, processed outputs, and an integrity structure (hashes or a Merkle tree) enabling verification that no component was altered.

In another aspect, deterministic replay is enabled by storing: schedule intent, waveform realization parameters, clock calibration, sensor calibration, and safety envelope parameters, and by verifying these parameters before replaying a run.

In another aspect, the run manifest hash is anchored to an external timestamp to provide independent evidence of existence at or before a time.

Brief Description of the Drawings

Drawings may be provided later. For purposes of this specification:

- **FIG. 1** depicts a run artifact bundle containing a manifest, data files, and signatures.
- **FIG. 2** depicts a Merkle tree structure covering run files.
- **FIG. 3** depicts deterministic replay using a signed configuration snapshot and calibration parameters.
- **FIG. 4** depicts optional anchoring of a run hash to an external timestamp service.
- **FIG. 5** depicts a verification procedure that validates bundle integrity and signature authenticity.

Definitions and Notation

Unless otherwise indicated:

- A *run* refers to an execution of an experimental protocol under a configuration, producing data and outputs.
- A *run bundle* refers to a directory or archive containing run files and metadata.
- A *manifest* refers to a structured description of a run bundle (e.g., JSON) including file digests.
- A *digest* refers to a cryptographic hash (e.g., SHA-256) of content.
- A *Merkle tree* refers to a hash tree over files enabling efficient integrity proofs.
- A *signature* refers to a digital signature over a digest or manifest (e.g., Ed25519, ECDSA).

- *Deterministic replay* refers to reproducing a run's control schedule and relevant calibrations within specified tolerances.
- *Configuration drift* refers to unintended differences between the claimed configuration and the actual configuration used.

Detailed Description

1. Run Bundle Structure

In one embodiment, a run bundle contains:

- **manifest file** (e.g., manifest.json);
- **configuration snapshot** files (e.g., config.json, schedule.json, calibration.json);
- **raw data streams** (sensor channels, driver telemetry);
- **derived outputs** (features, plots, summaries, gate outcomes);
- **integrity artifacts** (hash list and/or Merkle tree data);
- **signature artifacts** (signature files, public key IDs, certificate chains);
- **optional external timestamp proof** (receipt from timestamp authority or ledger anchor).

In one embodiment, the run bundle is stored as a directory; in another embodiment, the bundle is stored as an archive (e.g., zip/tar) whose digest is included in the manifest.

2. Canonical Configuration Snapshot

2.1 What must be captured. In one embodiment, the configuration snapshot captures:

- geometry ID and geometry parameters;
- schedule ID and schedule parameters (intent layer);
- waveform realization parameters (driver layer: dead-time, limits, edge shaping);
- clock parameters and timing calibration (per-channel delays, jitter bounds);
- sensor metadata (placements, sampling rates, filters, latencies);
- calibration constants for each sensor channel and their uncertainties;
- safety envelope parameters (limits, stop criteria, governor thresholds);
- software identifiers (firmware hash, git commit hash, binary digest);
- hardware identifiers (board serial numbers, DUT serial numbers, probe IDs).

2.2 Canonical serialization. In one embodiment, configuration snapshots are serialized in a canonical form to avoid hash mismatch due to key ordering or whitespace differences. Canonicalization rules may include:

- normalized JSON key ordering;
- fixed numeric formatting;
- explicit units for all quantities.

3. Tamper-Evident Integrity Structures

3.1 Flat hash list. In one embodiment, the manifest includes a flat list of file paths and digests:

$$\text{digest}(\text{file}_i) = H(\text{bytes of file}_i),$$

where H is a cryptographic hash.

3.2 Merkle tree over files. In one embodiment, file digests are combined into a Merkle tree whose root digest R commits to all files. The manifest stores the root R. Verification of a single file can be performed by checking a Merkle proof.

3.3 Streaming logs. In one embodiment, for streaming sensor logs, the system computes chunked digests and commits to a Merkle root per stream to enable incremental verification.

4. Signed Manifests and Identity

4.1 Signature over manifest digest. In one embodiment, the system computes a digest of the manifest and signs it:

$$\sigma = \text{Sign}_{k_{\text{priv}}}(H(\text{manifest})).$$

The run bundle includes σ and a public key identifier.

4.2 Key management (non-limiting). Keys may be stored in hardware security modules, TPMs, or secure enclaves. The system may enforce policies such as requiring two-person approval for signing.

5. Deterministic Replay

5.1 Replay pre-checks. In one embodiment, before replaying a run, the system verifies:

- manifest signature validity;
- all file digests match the manifest;
- required calibration parameters exist and are within valid ranges;
- firmware/software version matches or is compatible;
- hardware is mapped to required channels.

5.2 Replay execution. In one embodiment, the system replays:

- schedule intent (phase order, dwell, frequency tables);
- waveform realization parameters (gate timing, limits);
- timing calibration (per-channel delays) to match the original timing model.

The system records replay metrics and includes them in a new signed run bundle.

6. External Timestamping (Optional)

In one embodiment, the system anchors the run bundle digest or manifest digest to an external timestamp service. The bundle stores the timestamp receipt, enabling verification that the run existed at or before a time. External anchors may include a trusted timestamp authority, notarization service, or distributed ledger.

7. Example Manifest Schema (Non-Limiting)

An example manifest structure (illustrative only) includes:

- run_id, created_utc, operator_id;
- config_digests: digests of config files;
- data_streams: file list and digests;
- derived_outputs: file list and digests;
- merkle_root (optional);
- signature block (algorithm, public key id, signature bytes);
- external timestamp receipt (optional).

Claims (Draft)

Note: The following claims are an initial, non-limiting claim set intended to preserve multiple fallback positions. Final claim strategy should be reviewed by counsel.

Independent Claims

1. **(System)** A provenance system for rotating-field experiments, the system comprising: one or more processors and memory storing instructions that, when executed, cause the system to: generate a configuration snapshot for a run; collect synchronized sensor streams produced during the run; compute cryptographic digests for at least the configuration snapshot and the synchronized sensor streams; generate a manifest describing the run and including the cryptographic digests; and digitally sign the manifest to produce a signed run bundle.

2. **(Method)** A method of producing a deterministic replay artifact for a rotating-field experiment, the method comprising: generating a canonical configuration snapshot including schedule intent parameters and waveform realization parameters; recording calibration parameters and timing calibration parameters; producing a tamper-evident integrity structure committing to recorded data; signing a manifest describing the recorded data; and replaying the experiment using the canonical configuration snapshot and the timing calibration parameters.

3. **(Non-transitory medium)** A non-transitory computer-readable medium storing instructions that, when executed by one or more processors, cause the one or more processors to: verify a signature on a run manifest; verify that one or more file digests match the run manifest; and output a validation result indicating whether a run bundle is authentic and untampered.

Dependent Claims (Examples; Non-Limiting)

4. The system of claim 1, wherein the integrity structure comprises a Merkle tree over files in the run bundle.

5. The method of claim 2, further comprising anchoring a digest of the manifest to an external timestamp service and storing a timestamp receipt in the run bundle.

6. The system of claim 1, wherein generating the canonical configuration snapshot comprises canonicalizing JSON serialization to a normalized key order and numeric format.

7. The non-transitory medium of claim 3, wherein verifying comprises verifying digests for chunked streaming logs committed to a Merkle root.

8. The method of claim 2, wherein replaying comprises applying stored per-channel delay calibrations prior to executing a commutation schedule.

9. The system of claim 1, wherein the configuration snapshot includes a firmware digest and a hardware serial number list.

Additional Embodiments and Fallback Positions (Non-Limiting)

- Run bundles may be encrypted for confidential sharing while retaining integrity verification via signed manifests.
- The system may support partial disclosure by revealing Merkle proofs for selected files without revealing all files.
- Signature policies may require multiple signatures (multi-party signing) or threshold signatures.
- The system may generate standardized human-readable reports alongside machine-readable bundles.
- The provenance system may be integrated into an automated runbook engine and may refuse to execute runs if configuration drift is detected.

End of Specification (Draft)