CELA™ · PUBLIC EDITION

Continuity-Enforced Lineage Architecture (CELA™), v3: Implementation and Validation Brief

Version 3 · 29 October 2026 · Public edition of the CELA v3 technical whitepaper

This brief states what CELA does today, which security properties were tested and with what result, and how anyone can check the published evidence. It does not describe how the mechanisms are built. The full technical whitepaper, the adversarial validation report and the security architecture audit are shared under agreement for review and reproduction.

1. Summary

A valid signature proves that a record is authentic. It does not prove that the history around it is complete, in order, or that each change was authorized. CELA treats continuity as evidence: every accepted event is tied to an authenticated source and to its place in an append-only, signed history, so that a missing, reordered, substituted or rewritten step becomes detectable when the history is verified.

Version 1 of the whitepaper introduced the idea. Version 2 (August 2026) set out a formal model, a threat model and the security properties CELA claims. Version 3 reports the system as built: each claimed property is tied to an implemented control and to the result of adversarial testing against it. Where a property depends on how a deployment is provisioned, v3 says so and defines the provisioning tier instead of claiming an unconditional guarantee.

2. What is implemented

The evidence path consists of these roles, all of which exist and were exercised in testing:

- **Signing source.** A hardware emitter signs every emission on the device itself and links it to the one before.
- **Admission.** Only provisioned sources that prove their identity for the current session are admitted, and every admitted event is recorded before it is forwarded.
- **Commitment and signing.** Admitted events are committed in batches and each batch is signed and chained to the previous one.

- **Append-only history.** Signed batches form a history that can only be extended, plus sealed archives from which every commitment can be recomputed.
- **Independent auditor.** A separate verifier re-derives the history from the specification and checks it without sharing integrity code with the signer.
- **Independent-verifier gate.** When required, the history is only accepted with a matching attestation from a separately keyed, pinned verifier. It fails closed.
- **Authority records.** Decisions by authorities enter the record only with a verified signature.

The lineage records use classical signatures today. Post-quantum signatures (ML-DSA-44, alongside ECDSA-P256) currently protect the website manifest, and are planned for the lineage records.

3. Security properties: claimed, implemented, tested

Property	Result	What the testing showed
Tamper evidence	**Validated**	Altered records, altered signed fields, wrong keys and malformed signatures were all rejected.
Replay detection	**Validated**	A genuine identity proof replayed in a new session was rejected, as was authority input without fresh proof.
Rollback resistance	**Validated · deployment-scoped**	Broken links, regressions and truncation of the history were refused or detected. Against an attacker with full control of the host, this also requires an independent witness off the host (Tier 2).
Deterministic auditability	**Validated**	The independent auditor verified a clean history and caught every injected gap, overlap, truncation and downgrade.
Traceability	**Validated**	Ordering and provenance hold,

across time		and a forged sender identity was kept out of the record. Ordering is by sequence; trusted wall-clock time is a deployment add-on.
Reduced feasibility of hidden persistence	**Mechanism validated · deployment-scoped**	The independent-verifier gate fails closed when required. The guarantee holds when the gate is required and the witness sits outside the host.
Admission and key-lifecycle integrity	**Validated**	Unregistered and duplicate sources were refused, and key handling failed closed instead of creating or replacing keys.
Post-quantum lineage signatures	**Planned**	Not part of v3. Today post-quantum signatures cover the website manifest only.

4. How the validation was done

- **Who.** The CELA team, from a separate attacker host. This is in-house adversarial testing, not a third-party audit. Third-party testing is invited (section 7).
- **Oracle.** The system's own verifiers decided each outcome. An attack counted as successful only if a forged or altered artifact was accepted.
- **Controls.** Every test had a positive control: the genuine artifact had to be accepted, so that a rejection meant detection rather than a broken test.
- **Attacker positions.** Network adversary, rogue source, compromised host process, unauthenticated API caller, and passive observer.
- **State.** Clean-room state with test keys, so the results show how the controls behave. They are not an audit of a production dataset.

5. Deployment assurance tiers

Some guarantees depend on what a deployment provides. v3 states these as tiers, so that a deployment can choose one and check that it is in force. The testing showed that the

architecture enforces each control once it is provided.

Dimension	Tier 0 · Development	Tier 1 · Standard production	Tier 2 · High assurance
Source signing key	Software	Hardware-backed, enrolled and pinned	Secure element or HSM, attestable
History signing key	On the host	OS keystore with pinned keys	Off-host signer or HSM
Independent witness	Off	Required, pinned verifier	Off-host witness, checkpoint anchored off-host
Transport	Local only	TLS, per-source credentials	Mutual TLS with channel binding
Authorization	Not enforced	Signed operation records	Plus operator keyring and gated operations

6. Limits of this evidence

- Testing used clean-room state and test keys. It did not assess production key custody or any production dataset.
- Services were tested on current runtimes, not under production process supervision.
- Timing side channels were checked by code review only. Local measurements were too noisy to be conclusive.
- The reference emitter is not tamper-resistant hardware, and physical attacks were out of scope. Higher tiers use secure hardware.
- Order is established by sequence, not by a trusted clock. Trusted time is a Tier 2 addition.
- CELA makes recorded history tamper-evident. It does not prove that every event was captured, and it does not detect intrusions on its own.

7. How to test CELA independently

1. **Verify this website's signature.** Run the check in your browser on the verification section or on verify.html. The manifest (`manifest.json`), its signatures (`manifest.sig.json`) and the public keys (`jwks.json`) are public and use standard

algorithms (ECDSA-P256 and ML-DSA-44), so you can also check them with your own code.

2. **Act as an outside witness.** Download `lineage_checkpoint.json` and keep the copy. It records points in CELA's signed history and is covered by the site signature. The list only grows: every later version must contain your entries unchanged, with counts that only go up. If a later version drops or changes an entry, the history was rewritten, and your copy is the evidence. Copies held outside the operator's systems are what make this check independent.
3. **Request a structured review.** Qualified reviewers can receive the full v3 technical whitepaper, the adversarial validation report and the security architecture audit under agreement, and arrange to reproduce the validation.

8. Document versions

- **v1**: concept.
- **v2** (August 2026): formal model, threat model and claimed properties.
- **v3** (29 October 2026): implemented system and validation results. This page is its public edition; the full edition is restricted.

© 2026 Built By YouTH. All rights reserved. CELA™ is a trademark of Monart Studio Ventures. Site guide