RATION — AUDIT ANCHOR

An AI agent just moved real money.

How do you know it wasn't tampered with afterward?

NEAR Hackathon 2026 — **AI Security & Defense** track ("Trust, but verify")

`github.com/caelum0x/ration`

The problem

- AI agents are increasingly trusted to **make real financial decisions** — bridge-funding, approvals, spend.
- Every decision ends up in **somebody's database**. If that database can be edited, the "proof" of what the agent did is only as trustworthy as whoever operates it.
- Today, a judge, an auditor, a regulator, or a counterparty has **no way to independently check** an AI agent's financial history without trusting the company that built it.

> Logging isn't proof. A log you can quietly edit isn't an audit trail — it's a story.

Who actually has this problem

- **Fintechs and card-issuing platforms** running LLM-driven approval/decisioning loops (exactly where Ration lives today).
- **Any AI-agent framework** that gives an agent spend authority — agent-commerce, autonomous procurement, agentic trading.
- **Regulators and auditors** who are being asked to sign off on AI-operated money flows with no independent verification primitive.
- **Counterparties** (banks, BaaS partners, card networks) who need proof an agent's decision trail wasn't rewritten after the fact.

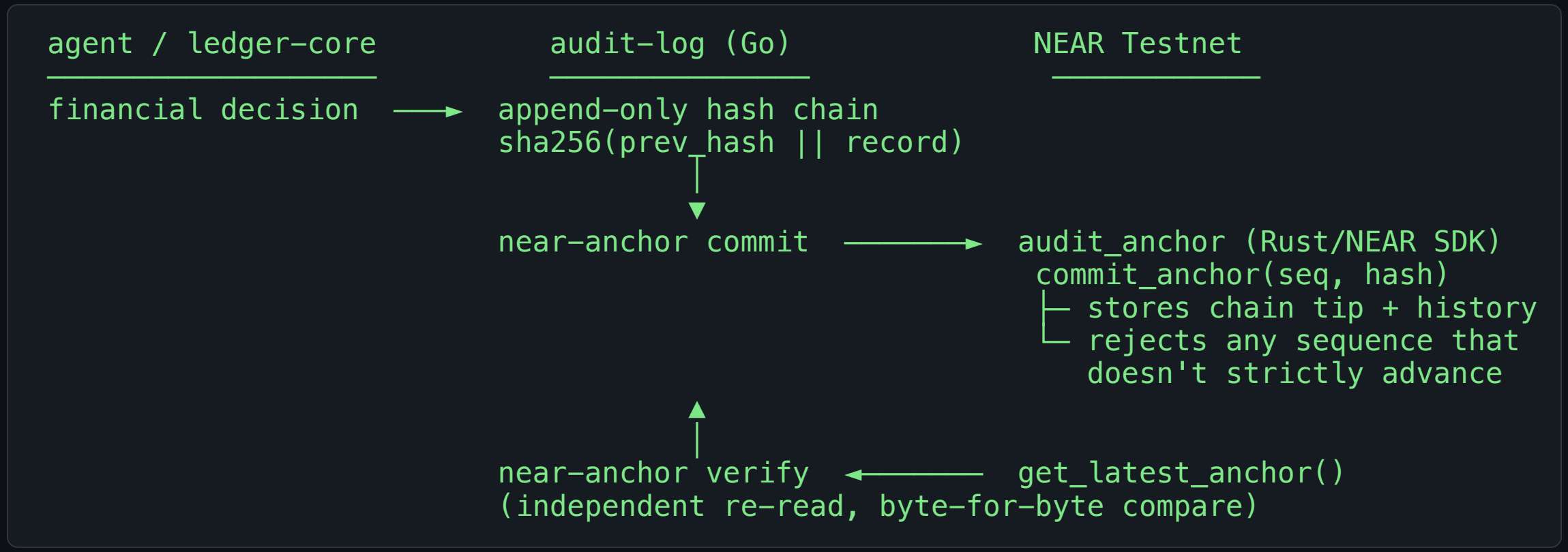
The solution

Ration already runs AI agents inside a hard, deterministic spending envelope they can never override — a real, running 11-service system (Rust ledger core + Go services).

Every agent decision was already journaled into `audit-log`: a genuine **append-only, hash-chained** trail, kept deliberately separate from the ledger's own write-ahead log.

This submission anchors that chain's current tip on NEAR Testnet — so *anyone*, not just Ration, can verify what the agent did, without ever touching Ration's own infrastructure.

How it works



Hand-rolled Go NEAR transaction signer (Borsh + ed25519) — no maintained Go NEAR SDK exists — cross-checked byte-for-byte against `near-api-js`.

Live proof — recorded today, not simulated

Step	Real result
Append 2 real audit-log records	sequence advances, `VerifyChain` confirms intact
Anchor the chain tip on NEAR Testnet	tx `5YebAM2YRyoneabk8npkH3a8VnnevnLv9NLkBT5ome1t`
Independent raw-RPC re-read (not our own tool)	`EXPERIMENTAL_tx_status` → `SuccessValue`, matches local tip
Replay the same anchor again	**nearcore itself rejects it**: "*sequence must strictly advance past the last anchored sequence*"

Full unedited recording: `docs/near-hackathon/media/ration-near-demo-2026-11-11.mp4`

Primary submission account separately live: `ration-anchor-286ca315f1.testnet` (3 real anchors, 1 rejected replay, see `HACKATHON.md`)

Why this fits the track

Track: AI Security & Defense — "Trust, but verify."
Ration gives an AI agent money *and* a hard, external, independently-checkable record of what it did with it — literally the track's own framing.

The hackathon's own FAQ states projects are evaluated on **technical implementation, design, impact, originality and sustainability** — mapped explicitly on the next slide.

...and the judging criteria

- **Technical implementation** — real contract, real signer, 5/5 Rust + Go tests, 4 real on-chain transactions this submission, zero regressions (15/15 ledger-core, full `go test ./... -race`).
- **Design** — audit-log kept deliberately separate from ledger-core's own WAL; anchoring is additive, not a rewrite.
- **Impact** — a reusable trust primitive for *any* AI-agent-with-money system, not Ration-specific.
- **Originality** — not a payment rail; a verification layer, with a provably-enforced replay guard.
- **Sustainability** — ships as a standalone CLI (`near-anchor`) today; wiring it into the agent's own poll loop is one small, obvious next step.

Traction — honest

- **$0 revenue, pre-revenue infrastructure.** Ration is a real, running fiat-ledger + AI-agent platform (11 services, Rust + Go) already live for development/testing — not a hackathon-only demo app.
- This submission is a **focused add-on**: one new capability (on-chain anchoring), not a rewrite.
- No mainnet transactions, no claimed users of the anchor feature yet — Testnet only, exactly as the evidence above shows.
- What's *not* done, on purpose: automatic periodic anchoring isn't wired into the agent's poll loop yet (manual/cron-invoked today) — scoped out to keep this submission honest about what was built and tested under the deadline.

Roadmap

- **Next 2 weeks:** wire `near-anchor commit` into `services/agent`'s own poll loop (periodic, not manual) — the one concrete gap called out above.
- **Next month:** per-tenant anchor namespacing (today: one global chain tip per contract); mainnet deployment path.
- **Ongoing:** publish the Go NEAR signer as a standalone open-source package — no maintained Go NEAR SDK currently exists.

Team & links

Arhan Subaşı (GitHub: `caelum0x`) — solo founder/engineer. Builds production infrastructure: distributed systems, payment ledgers, and now AI-agent safety rails.

- Repo: **github.com/caelum0x/ration**
- Full writeup: `docs/near-hackathon/HACKATHON.md`
- Demo video: `docs/near-hackathon/media/ration-near-demo-2026-11-11.mp4`
- Sibling entry, same hackathon: Cred402 — SpendGuard (Agent Economy track)

Thank you — questions welcome.