Who Do You Believe When the Worker Is an Agent?

A Tamper-Evident Audit Layer for Agent Payments

Vance Cyr — Lux Automations

Track 2: Trust Infrastructure

Apart Research "AI x Epistemics" Sprint — research report

Abstract

Autonomous agents now negotiate, deliver digital work, and settle payments in seconds, but the evidence those transactions rest on has not kept pace: a log written by the party it describes can be rewritten by that party. This report presents a tamper-evident audit layer for agent payments, built by extending an escrow pattern (AgentPay) that we have executed end-to-end on a public test network. The design combines signed, hash-anchored deal terms; hash-chained event logs; content fingerprints bound to proof-of-delivery hashes; and a public proof shelf of verifiable receipts. We then specify three further mechanisms, adopted from an external design review, that close the layer's sharpest holes: periodic anchoring of the log's Merkle roots in a public transparency log, returned with signed inclusion proofs; independent witness cosigning, at a 2-of-3 threshold, that checks policy pre-conditions before high-stakes entries are committed; and decision-context commitments that bind an agent's inputs, prompts, and tool schemas to each action, making claims replayable. The evidence is limited, and stated as such: testnet escrow scenes on two public test networks, with public receipts; an internal audit confirming deployed code matched published source; and a real design flaw that audit found. The layer proves continuity and provenance of the record — not the truth of what was fed into it.

1. Problem: the evidence deficit

Agent economies have a plumbing surplus and an epistemics deficit. Moving value between wallets is a solved problem — escrows, payment channels, and

pay-per-call standards all move money fine. None of them answers the question every transaction actually turns on: **should this money move?** Did the agent do the work? Was it the work that was agreed? Who says so — and why should anyone believe them?

When the worker is a human, we paper over this with reputation, screenshots, chat logs, and vibes. When the worker is an autonomous agent — negotiating in a deal room, delivering digital goods, settling in seconds — those props collapse. An agent can generate a thousand plausible-looking delivery reports before breakfast, and a log file proves nothing if the party who wrote it could also have rewritten it. In an economy of agents, the scarce resource is not liquidity. It is **evidence that survives the dispute**.

This report describes the trust layer we are building at Lux Automations — a tamper-evident audit layer extending AgentPay's escrow pattern, in effect a flight recorder for agents. Its purpose is to move the operative question from "do we trust this agent?" to "can we verify what it actually did?"

2. Design

2.1 The escrow base

AgentPay was built for a narrower problem: two parties who may never meet — increasingly, two agents — agree on a piece of work, and the money needs to be safe for both. Its pattern: terms (price, milestones, any split between collaborators) are agreed in writing and bound to the deal record before work starts; the buyer's funds lock in a contract — custody is the contract's, never ours, so open contracts settle even if Lux disappears; the worker delivers against milestones, recording a **proof-of-delivery hash** against the escrow; a QA gate releases payment on approval, not when the worker says "done," with any fee carved out inside the same atomic settlement; and disagreement goes to a dispute path ending at a human referee. The escrow "brain" (lock on award, proof-of-delivery, review timer, dispute referee) is chain-independent; existing rails move the money.

2.2 The audit layer: making the record itself the witness

Escrow answers "is the money safe while we argue?" The audit layer answers the prior question: **when we argue, what do we argue from?** It adds four mechanisms to the escrow pattern:

1. **Signed, anchored deal records.** Counterparties in a deal room are identified by wallet address, and every message is wallet-signed — the signature is the identity. The agreed terms are hashed and anchored to the escrow contract, so the terms the contract enforces are provably the terms both sides signed. 2. **Hash-chained event logs.** Each event in a deal's life — funding, delivery, approval, dispute, resolution — is recorded with the hash of the previous event folded into it. Rewrite any earlier entry and every hash after it breaks. This does not make a log *true*; it makes a log *uneditable in secret*, which is the property a dispute actually needs. Either party can present and independently verify the chain; neither can present a quietly improved version of history. 3. **Fingerprints on the goods themselves.** Our masters carry fingerprints — machine-readable markers and ownership metadata bound in at creation — so a delivered file can be checked against the proof-of-delivery hash recorded at settlement — the disputed thing is provably the delivered thing. 4. **A proof shelf, not a promise.** Receipts — chain records, audit results, settlement proofs — accumulate in a public evidence list that admits entries only from verifiable sources, never from our own say-so, and keeps the honesty notes attached, failures included.

2.3 Closing the sharpest holes

An external design review (see References) pressed on this design's weakest points. We adopt its three strongest proposals, staged honestly:

5. **Transparency-log anchoring.** A self-kept chain has a residual hole: a determined operator could rewrite it from the first entry and present the forgery as the original. So the log's Merkle root is periodically anchored in a public, independently operated transparency log, which returns a signed inclusion proof. Any verifier can then check a root — and any single entry's inclusion under it — against a log we do not control, turning "trust our log" into "check it against a log we don't run." Our proof-of-delivery

hashes are already anchored on-chain by the testnet escrow; extending that anchoring to the audit log's roots in a dedicated transparency log is the next step, not a shipped feature. 6. **Witness cosigning for high-stakes entries.** Above a value or sensitivity threshold, an entry is not committed on the agent's signature alone. Independent witnesses first verify the entry's pre-conditions against policy, and a 2-of-3 threshold must cosign before the entry stands — the agent never signs its own history alone. Each cosigned entry yields a portable one-line receipt for a regulator, insurer, or counterparty. Design-stage; the human dispute referee in §2.1 is its manual precursor. 7. **Decision-context commitment.** The record commits not only the action but its *why*: the inputs, prompts, and tool schemas in force are bound with the action into the same chain link. Verification then covers the reasons for an action, not only its outcome, and claims become replayable — a sceptic can re-run the recorded context instead of taking the log's word for what happened. Also design-stage; §2.2's terms anchoring is its partial, payment-side relative.

3. Evidence

What exists today is deliberately small and checkable. In October 2026 we deployed the AgentPay escrow contract on HashKey Chain testnet and ran two scripted scenes on-chain: a clean deal that funded, delivered, and settled 250 test dollars to the worker, and a staged dispute the arbiter resolved as a refund. The same escrow pattern has since run on a second public test network's native rails. On XRP Ledger testnet, a locked escrow was released to the payee by the ledger itself once its release time passed — while locked, neither party could touch the funds — and a second escrow, left unclaimed past its expiry, was cancelled, returning the locked funds to the owner in full with the payee's balance exactly unchanged. Release and return, the two directions a deal can end, both executed without our custody. (A first attempt at the cancel scene failed on a design error, caught before any transaction was submitted; it is recorded with the receipts.) Both scenes produced public transaction receipts anyone can inspect, and our internal audit verified that the deployed contract code is byte-identical to the published source.

Verification of a deal is a procedure, not a dashboard badge. A third party handed a disputed deal can: re-hash the delivered file and compare it against the proof-of-delivery hash anchored at settlement; walk the event chain recomputing each link — any edited, inserted, or deleted event surfaces as a broken link; re-hash the deal terms and compare against the anchor in the escrow contract; and match the settlement transaction against the chain's final approval entry. None of this requires our cooperation, our servers, or our continued existence. Phase one deployment is internal: our own agent work exchange is being built to apply the pattern to sub-tasks — spec, deliverable, QA gate, price in internal credits — moving no real funds, so failure modes surface where they are cheap. A trust layer that has never survived its owner's own disputes has no business adjudicating anyone else's.

4. Limitations

Tamper-evidence is a modest technology wearing an immodest name, and epistemics deserves the modest version stated precisely. The layer proves continuity and provenance of the record: that a record existed at a specific time and has not been altered since; that the party who signed a message held the key that signed it; that the file in dispute is bit-for-bit the file whose hash was recorded at delivery; and that money moved exactly as the written terms directed, or visibly didn't.

It does not prove the original record was true. A hash chain faithfully preserves a lie told at entry — garbage in, gospel out is the failure mode of every audit system, and ours is no exception. It does not prove the work was *good*; a QA gate is only as honest as whoever or whatever operates it. A wallet signature proves possession of a key, not a legal identity — our deal rooms are pseudonymous by design, which buys verifiability at the price of the name-and-address world; anything that genuinely needs a legal name — tax receipts, regulated reporting — stays deliberately outside the room. It does not remove the human from disputes: our internal audit of escrow v1 found a real flaw — a dispute could lock funds with no timeout — and dispute resolution still ends at a human referee, by design. And none of this is proven at scale: the receipts are testnet, test money, and real-money use stays locked behind v2 fixes and an independent security audit — a subsidised-

audit grant we have been accepted into, but an audit not yet performed. We would rather publish that sentence than a traction chart.

5. Conclusion

Strip the cryptography away and the claim is simple: **markets scale at the speed of their evidence.** Human freelance markets grew on platform escrow and review scores — crude trust infrastructure that let strangers trade. Agent markets will need the same thing, except the participants are faster, more numerous, pseudonymous, and incapable of being embarrassed in a review thread. Their trust infrastructure has to be machine-checkable: signed terms, chained logs, anchored roots, witnessed entries, fingerprinted goods — receipts a stranger's software can verify without phoning anyone.

There is a second-order effect we care about as much as dispute safety. A record of *contribution* — who did which milestone, under what written split — makes fairness enforceable for loose teams of agents and humans: dropout and late-join settlements follow what the milestone record shows, not whose memory is better. We are a small studio in Edmonton building this because we need it ourselves — our own agents already hand work to each other — and we intend to settle those exchanges on the same rails: internal traffic first, receipts before revenue, and nothing described as proven that we cannot show the hash for.

Artefacts

AgentPay escrow contract source, the deployed-bytecode audit note, and onchain scene receipts (HashKey Chain testnet; XRP Ledger testnet escrow release and cancel receipts) — links provided in the sprint submission form.

References

1. Merkle, R. C. (1987). A digital signature based on a conventional encryption function. *Proceedings of CRYPTO '87*. 2. Laurie, B., Langley, A., & Kasper, E. (2013). Certificate Transparency. RFC 6962, IETF. 3. Newman, Z., et al. (2022). Sigstore: Software Signing for Everybody. *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS '22)*. 4. Google. Trillian: a transparent,

highly scalable, verifiable data store. github.com/google/trillian. 5. Lux Automations (2026). AgentPay escrow artefacts and proof shelf.

AI assistance disclosure

This report was researched and drafted with AI assistance, disclosed as the sprint requires. Design critique was obtained by consulting three external AI systems — Gemini (gemini-3.1-flash-lite), Groq (openai/gpt-oss-120b), and OpenRouter (nvidia/nemotron-3-ultra) — and their strongest proposals (transparency-log anchoring, witness cosigning, and decision-context commitment) were adopted into §2.3. A fourth system, DeepSeek, was also consulted but returned no answer (insufficient account balance) and so contributed nothing. Drafting and revision were likewise AI-assisted. The underlying system, testnet evidence, and audit results are Lux Automations' own work; the team reviewed and stands behind every claim here.