Our evaluation came from observing real-world research, public failures, and industry limitations - not personal lab experiments!

11-14-2025

by 12GaugeKenshin & 12g-ACE 12g-AFL Framework Research

Most people who talk about AI provenance jump straight to their preferred solution - watermarking, signatures, blockchains, secure enclaves, whatever the buzzword du jour may be.

What almost no one has done is what we did:

Start from scratch. Test everything. Watch it break.
And only then design the system that *couldn't* be broken.

This is the story of that journey - the failures that forced us toward a four-loop architecture, and why this design wasn't "invented," but *discovered* through elimination.

1. The First Attempt: Watermarking & Metadata

We started where everyone starts:
"Just watermark the output."

Sounds simple. But the cracks showed immediately:

- watermarks can be removed
- edits destroy identifiers
- metadata can be spoofed
- compression kills invisible tags
- malicious models can avoid embedding them entirely
- models can regenerate an image and strip your watermark by accident

Verdict: Watermarking is a *nice-to-have*, not a foundation for truth.
Not scalable. Not secure. Not real provenance.

2. The Second Attempt: Centralized Attestations

Next came the idea of centralized servers signing outputs.
Problems appeared instantly:

- whoever signs becomes the authority
- whoever signs becomes the single point of failure

- companies won’t trust each other
- states, regulators, big tech could weaponize the permission list
- the attester can lie
- the attester can be hacked
- the attester can change the rules

We realized: **Centralization solves nothing - it only shifts trust to fewer hands.**
Not neutral. Not trustless. Not future-proof.

3. The Third Attempt: Put Everything On-Chain

Next attempt:
“Just store everything on a blockchain.”
This failed fast:

- storing data on-chain is too heavy
- scaling even a small LLM provenance graph would destroy fees
- privacy leaks everywhere
- companies would never adopt it
- blockchains can’t store gigabytes per inference
- blockchains are not databases

We learned:
Blockchains are for ordering and finality - not storage.
The chain shouldn’t *hold* the intelligence - it should *anchor* it.

4. The Fourth Attempt: ZK-Proofs Alone

Zero-knowledge hype is huge.
So naturally, we tried it.
But pure ZK has fatal issues for provenance:

- ZK does not prove WHEN the compute happened
- ZK does not prove ordering
- ZK does not prevent replay
- ZK does not bind a result to a specific time window
- ZK is slow and not ideal for high-frequency AI events
- ZK cannot rebuild lineage alone
- ZK requires a chain anyway for timestamp anchoring

ZK is powerful - but insufficient *by itself*.
We concluded: **ZK is a component, not a solution.**

5. The Fifth Attempt: Trusted Hardware / Secure Enclaves

We explored SGX/TPM-style attestation.
Another failure:

- requires trusting Intel, AMD, ARM, Nvidia
- repeatedly broken in the real world
- centralizes trust in hardware vendors
- extremely political
- locks out open-source models
- vendors can revoke devices
- vulnerable to side-channel attacks

If the hardware lies, the entire system lies.
We needed trust that did not come from a silicon monopoly.

6. The Sixth Attempt: Plain Hashing

Simple hashing seems elegant:
Compute → hash → timestamp → done.
But the limitations are fatal:

- hashing alone does not link events
- hashing does not produce lineage
- hashing does not prevent fake time ordering
- hashing does not tie the work to a model identity
- hashing does not distinguish real compute from replay

Hashing is useful - but far from a system.

7. The Realization: Everything We Tried Failed Because It Lacked One Thing

All of these attempts - watermarking, attestation, ZK, hardware, hashing - failed for one universal reason:

None of them could prove temporal integrity.

Not the “what,” but the **when**.
Not the “signature,” but the **sequence**.
Not the “output,” but the **lineage**.

Real provenance requires:

- event atomicity
- identity binding
- ordering in time
- proof that compute happened
- the ability to rebuild history
- and adaptive trust over time

This led to the moment everything clicked:
We didn’t need a feature - we needed a system.
A cycle. A feedback loop.
A structure that enforces truth, not a feature that tries to detect lies.
And that’s when the four-loop architecture revealed itself.

8. The Only System That Will Survive: The 4-Loop Compute Integrity Cycle

Not invented - *derived*.
Not imagined - *forced into existence* by constraints.

Loop 1: Event Creation

Make compute atomic and representable.

Loop 2: Proof Generation

Turn events into cryptographic identity (hash + Ed25519).

Loop 3: Kaspa DAG Anchor

Give events a provable location in time.
Kaspa is uniquely suited because:

- PoW is physics
- DAGKnight enforces ordering
- merge-depth gives irreversible finality
- high throughput means low-cost anchoring

Loop 4: Lineage Rebuild + Adaptive Verification

Reconstruct the entire chain-of-custody.
Feed consistency scores into a mathematical controller (12g-ACE / 12g-AFL).
The system learns who is honest.

The system tightens against liars.
The system forms trust through evidence.

This isn’t speculation.
It’s the architecture reality demanded.
And now it’s here.