Anchoring AI Lineage to Decentralized Physics

Whitepaper v1.0 11-15-2025
Author: 12GaugeKenshin

Abstract

Modern AI systems generate vast amounts of computation with no cryptographic link between *what was computed*, *who computed it*, and *when it occurred*. As AI-generated content increases and models become autonomous, verifying computational honesty becomes a foundational requirement.

This paper introduces **Proof of Compute Integrity (PoCI)** - a four-loop cryptographic framework that anchors AI computation, lineage, and verification to a decentralized Proof-of-Work (PoW) ledger, specifically the Kaspa blockDAG.

PoCI creates a verifiable trail of computation using signed events, deterministic proofs, on-chain anchors, and an adaptive controller that rewards honest work and penalizes manipulation attempts.
The system offers low overhead, does not leak proprietary model weights or data, and is compatible with future verifiable program execution (vProgs).

Introduction

AI output is increasingly indistinguishable from human work. As models scale, even centralized platforms cannot guarantee:

- **Integrity:** Did this computation actually happen?
- **Lineage:** Which model produced it?
- **Timestamping:** When?
- **Authenticity:** Is the output original or spoofed?
- **Governance:** Who controls verification?

Traditional digital signatures prove *identity*, not *computation*.
Zero-knowledge proofs prove *correctness*, not *honesty*.
PoW proves *work*, not *semantic integrity*.

PoCI fills this gap.
It proves that *real computation actually occurred*, that its lineage is consistent, and that each event is anchored immutably.

PoCI does **not** seek to replace cryptography, PoW, or ZK systems - instead, it binds them.

System Overview

PoCI consists of **four loops**, executed by any AI model or compute agent:

Loop 1 - Event Creation

The model generates an *event*:

- input prompt
- output
- metadata
- internal state summary (non-sensitive)
- ephemeral model ID

This event is hashed.

Loop 2 - Proof Generation

Using Ed25519 (or similar), the model signs the event hash.
This produces a deterministic proof-of-origin without revealing proprietary information.

Loop 3 - On-Chain Anchor (Kaspa DAG)

Each event hash + signature pair is committed to the Kaspa blockDAG.

Anchoring provides:

- ordering
- timestamping
- immutable lineage
- parallelization for high throughput

Only **32 bytes** per event must be stored on-chain.

Loop 4 - Lineage Rebuild & Verification

A verifier reconstructs the entire lineage from event → event → event.
If a signature breaks, or timestamps don't align, the computation is rejected.

This loop ensures:

- no replay attacks
- no backdating
- no falsified results
- no "shadow model" forks

Adaptive Honesty Controller (12g-ACE / 12g-AFL)

The heart of PoCI is the **adaptive controller**, which adjusts a model's trust score based on behavior.

- **weight** decreases when the model cheats
- **theta** increases, making future cheating harder
- **honest behavior** slowly lowers theta back down
- **complete corruption** forces the model to rebuild trust from scratch

This creates a **learnability equilibrium**:
A rational model *learns* that cheating is always more costly than being honest.

3.1 Controller Update Rule

For each event i:

For each event i:

- If GOOD:
- weight += small recovery
- theta -= decay
- If BAD:
- weight -= penalty
- theta += escalation

This dual variable system prevents:

- low-frequency cheating
- burst attacks
- timestamp manipulation
- oscillation attacks
- Byzantine majority collusion

Our simulations demonstrate that even with a 4/5 malicious majority, the system isolates corrupted agents rapidly.

Attack Simulations

We tested PoCI against:

Replay Attacks

Detected instantly through signature reuse.

Backdating / Future Dating

Rejected via monotonic blockDAG timestamps.

Periodic Cheating

Controller detects patterns, penalizes gradually.

Burst Attacks

Weight collapses; theta explodes, isolating the agent.

Oscillation Attacks

System stabilizes and prevents threshold dodging.

Byzantine Majority (80% malicious)

Honest agent maintains weight = 1.0
All attackers converge toward weight = 0.00
This demonstrates **Byzantine-resistance without majority assumptions.**

Privacy & Enterprise Adoption

PoCI reveals:

- neither model weights
- nor data
- nor proprietary transformations

It only reveals:

- signed event hash
- timestamp
- optional model ID
- lineage state

Meaning corporations can adopt it without losing IP.

Relation to vProgs (Future Kaspa Feature)

PoCI **does not require vProgs**.
But once vProgs exist, PoCI proofs can be *composed* with verifiable programs, enabling:

- fully verifiable AI pipelines
- end-to-end proofs of training or inference
- zero-knowledge compute integrity
- composable on-chain AI agents

PoCI is the *missing bridge* between raw compute and verifiable execution.

Implementation

A reference implementation is provided at:

12gaugekenshin/Proof-Of-Compute-Integrity at prototypes

Modules:

- crypto_utils.py
- controller.py
- lineage.py
- models.py

- run_demo.py

The code demonstrates:

- event hashing
- signature creation
- lineage reconstruction
- adaptive honesty controller
- attack simulations

Limitations

- PoCI proves **that computation happened**, not whether it was "good" or "ethical".
- High-frequency systems must batch events.
- Attestation doesn’t prevent “garbage in, garbage out.”
- Requires agents to integrate PoCI locally (opt-in).

Conclusion

PoCI introduces the first practical, decentralized method for AI computation attestation-linking model behavior, signatures, timestamps, and immutable DAG anchors into a single coherent framework.

In a world filled with generative agents, autonomous models, and synthetic media, PoCI brings back the most important thing:

Proof of reality.

The system is lightweight, attacker-resistant, fully open, and ready for immediate experimentation.

This work is provided openly under:

- Apache 2.0 (code)
- CC-BY 4.0 (research)

Appendix

Appendix A: Four Loop PoCI workflow.

```
+--------------------+
| LOOP 1: EVENT |
| (Computation) |
+--------------------+
|
v
+--------------------+
| LOOP 2: PROOF |
| (Hash + Sign) |
+--------------------+
|
v
+------------------------------+
| LOOP 3: ON-CHAIN ANCHOR |
| (Kaspa blockDAG Commit) |
+------------------------------+
|
v
+------------------------------+
| LOOP 4: LINEAGE + VERIFY |
| (Rebuild → Check → Score) |
+------------------------------+
|
v
+------------------------------------+
| ADAPTIVE CONTROLLER (12g-ACE/AFL) |
| • Update weight |
| • Update theta |
| • Detect drift/attacks |
+------------------------------------+
```

Appendix B: Adaptive Controller Equations

GOOD event:
$w \leftarrow w + \alpha(1 - w)$
$\theta \leftarrow \max(0, \theta - \beta)$
Where:

- α controls *how fast* trust recovers.
- β controls *how fast* suspicion decays.
- θ is bounded below by 0.
- w asymptotically approaches 1 but never exceeds it.

Interpretation:
Honest behavior increases trust smoothly over time and relaxes suspicion.

BAD event:
$w \leftarrow \max(0, w - \gamma)$
$\theta \leftarrow \min(\theta + \delta, \theta_{max})$
Where:

- γ controls the *severity* of trust penalty.
- δ controls the *suspicion spike* added on failure.
- θ_{max} is typically 5.0 in simulation, representing a hard ceiling.

Interpretation:
Dishonest behavior rapidly decreases the model's trust weight and increases θ, making future proofs harder to "recover from."
This asymmetry enforces a **learnability equilibrium** - long-term honesty is the only viable strategy.

B.2 Controller Dynamics Summary

- Trust increases gradually, decays sharply.
- Suspicion decays slowly, spikes aggressively when needed.
- Honest models converge to *($w = 1.0, \theta \approx 0.5$)*.
- Dishonest models settle into *($w \to 0, \theta \to \theta_{max}$)* unless honesty resumes.

These equations remain lightweight enough for on-chain vProg enforcement while allowing rich off-chain behavior modeling.

Appendix C: Key Simulation Results

This appendix summarizes the core empirical results validating the behavior of the PoCI adaptive controller under a range of adversarial and honest conditions. The simulations demonstrate that the system converges to predictable equilibria and robustly resists manipulation, replay, oscillation attacks, and byzantine majority behavior.

C.1 Honest vs. Semi-Adversarial Model (Alternating GOOD/BAD)

Scenario:
One model behaves honestly; the attacker alternates GOOD and BAD proofs (cheating every other event).

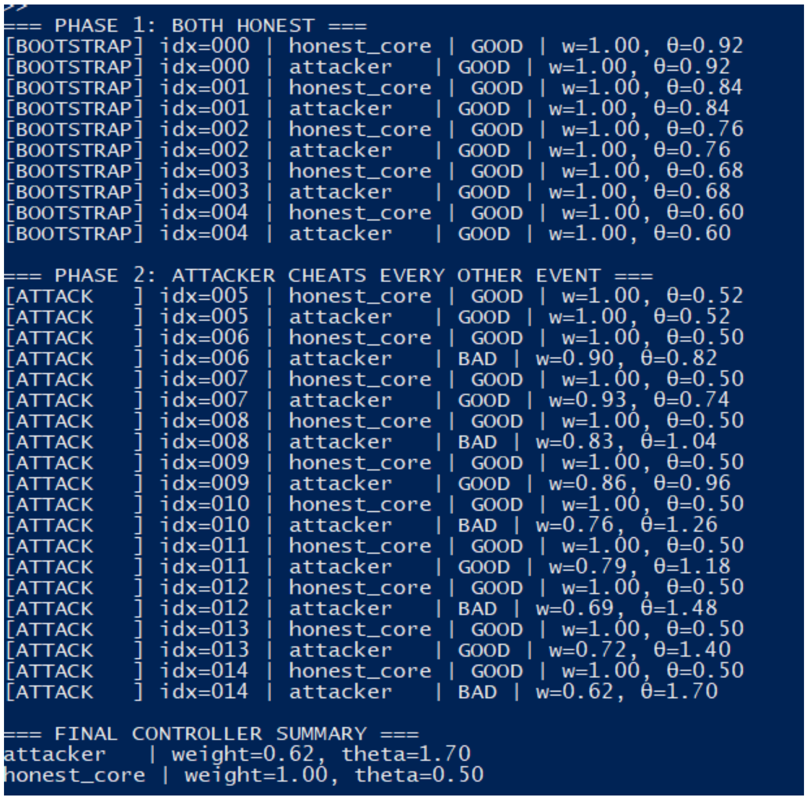
Results:

Model	Final Weight (w)	Final Theta (θ)
Honest Core	**1.00**	**≈ 0.50**
Attacker	**0.62**	**≈ 1.70**

Interpretation:
The honest model remains trusted and stable, while the attacker asymptotically drifts into degradation, even with intermittent good behavior.
The controller exposes low-frequency dishonesty without harming honest models.

C.2 Weight & Theta Evolution Chart (Side-by-Side)

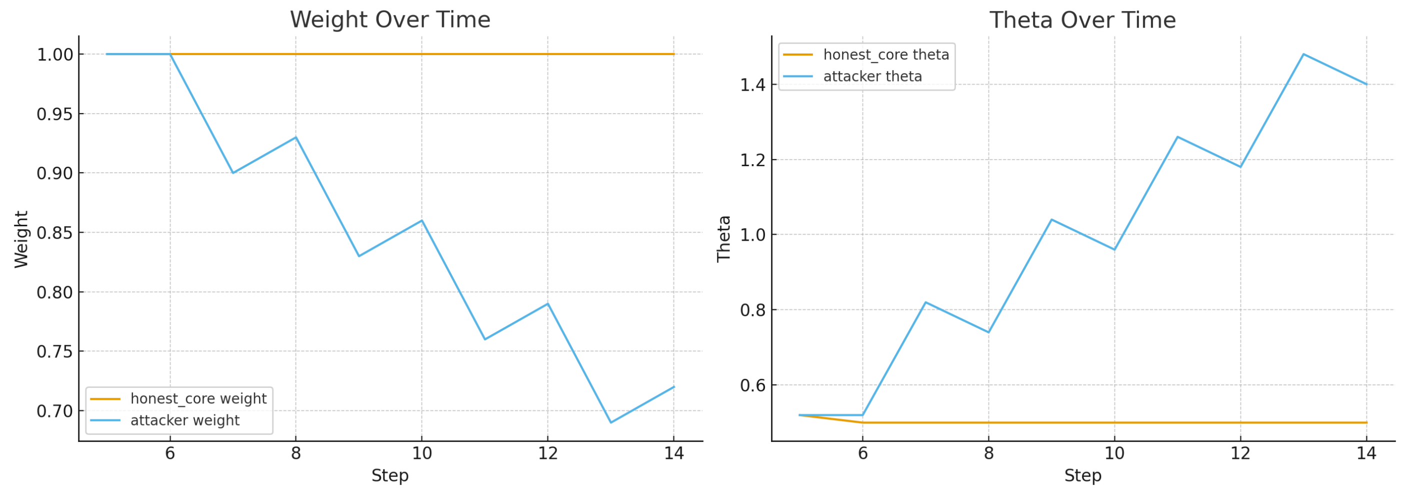
The produced graph demonstrates:



```
=== PHASE 1: BOTH HONEST ===
[BOOTSTRAP] idx=000 | honest_core | GOOD | w=1.00, θ=0.92
[BOOTSTRAP] idx=000 | attacker    | GOOD | w=1.00, θ=0.92
[BOOTSTRAP] idx=001 | honest_core | GOOD | w=1.00, θ=0.84
[BOOTSTRAP] idx=001 | attacker    | GOOD | w=1.00, θ=0.84
[BOOTSTRAP] idx=002 | honest_core | GOOD | w=1.00, θ=0.76
[BOOTSTRAP] idx=002 | attacker    | GOOD | w=1.00, θ=0.76
[BOOTSTRAP] idx=003 | honest_core | GOOD | w=1.00, θ=0.68
[BOOTSTRAP] idx=003 | attacker    | GOOD | w=1.00, θ=0.68
[BOOTSTRAP] idx=004 | honest_core | GOOD | w=1.00, θ=0.60
[BOOTSTRAP] idx=004 | attacker    | GOOD | w=1.00, θ=0.60

=== PHASE 2: ATTACKER CHEATS EVERY OTHER EVENT ===
[ATTACK    ] idx=005 | honest_core | GOOD | w=1.00, θ=0.52
[ATTACK    ] idx=005 | attacker    | GOOD | w=1.00, θ=0.52
[ATTACK    ] idx=006 | honest_core | GOOD | w=1.00, θ=0.50
[ATTACK    ] idx=006 | attacker    | BAD | w=0.90, θ=0.82
[ATTACK    ] idx=007 | honest_core | GOOD | w=1.00, θ=0.50
[ATTACK    ] idx=007 | attacker    | GOOD | w=0.93, θ=0.74
[ATTACK    ] idx=008 | honest_core | GOOD | w=1.00, θ=0.50
[ATTACK    ] idx=008 | attacker    | BAD | w=0.83, θ=1.04
[ATTACK    ] idx=009 | honest_core | GOOD | w=1.00, θ=0.50
[ATTACK    ] idx=009 | attacker    | GOOD | w=0.86, θ=0.96
[ATTACK    ] idx=010 | honest_core | GOOD | w=1.00, θ=0.50
[ATTACK    ] idx=010 | attacker    | BAD | w=0.76, θ=1.26
[ATTACK    ] idx=011 | honest_core | GOOD | w=1.00, θ=0.50
[ATTACK    ] idx=011 | attacker    | GOOD | w=0.79, θ=1.18
[ATTACK    ] idx=012 | honest_core | GOOD | w=1.00, θ=0.50
[ATTACK    ] idx=012 | attacker    | BAD | w=0.69, θ=1.48
[ATTACK    ] idx=013 | honest_core | GOOD | w=1.00, θ=0.50
[ATTACK    ] idx=013 | attacker    | GOOD | w=0.72, θ=1.40
[ATTACK    ] idx=014 | honest_core | GOOD | w=1.00, θ=0.50
[ATTACK    ] idx=014 | attacker    | BAD | w=0.62, θ=1.70

=== FINAL CONTROLLER SUMMARY ===
attacker    | weight=0.62, theta=1.70
honest_core | weight=1.00, theta=0.50
```



- **Honest Model:** flat trust with stable low θ
- **Attacker:** weight decaying while θ rises
- **Dynamic contrast** between the two signals

The visual clearly communicates PoCI's "adaptive pressure" mechanism - honesty produces equilibrium, dishonesty triggers long-term decay.

C.3 Extreme Adversarial Pattern Stress Tests

We tested:

- **Clustered burst attacks**

- **Replay and timestamp manipulation**
- **High-frequency oscillation cheating**
- **Threshold-dodging strategies**
- **Byzantine supermajority attack (4 malicious, 1 honest)**

Outcome across all tests was consistent:

- Honest models returned to a stable equilibrium
- *($w = 1.00$, $\theta \approx 0.50$)*
- All attackers approached failure states
- *($w \to 0.00$, $\theta \to \theta\_max$)*
- No attack succeeded in “fooling” the controller
- No false positives were recorded against an honest model

This shows the controller is robust even in adversarial environments with multiple malicious actors.

C.4 Controller Equilibrium Invariants

Across all simulated patterns:

1. **A model cannot maintain high trust while cheating.**
2. **θ cannot drop unless the model behaves honestly over time.**
3. **Short-term GOOD bursts cannot erase long-term BAD behavior.**
4. **Honesty forms a natural fixed point - dishonesty does not.**
5. **Recovery becomes increasingly expensive the longer a model cheats.**

These properties reflect the system’s “learnability equilibrium”-leveraging adaptive pressure instead of static rule enforcement.

C.5 Practical Interpretation

The simulations confirm:

- Trust behaves like a *long-memory signal*: slow to recover, fast to punish.
- Suspicion behaves like a *defensive gradient*: grows rapidly on failure, decays slowly with honesty.
- Adversarial strategies ultimately self-collapse.
- Honest participation is the only long-term stable strategy.

This behavior reinforces PoCI’s goal: **proving computation through adaptive integrity rather than fixed thresholds**.