Hardened Parity Network Maps: HMAC-Anchored Chain-of-Trust for Verifiable AI Integrity and Semantic Consistency

Steve J. Horton
MS in Information Security
Independent Researcher

October 19, 2025

Abstract

We present a hardened variant of *Parity Network Maps* (PNM), a post-training integrity mechanism that detects at-rest and runtime tampering of neural networks without modifying original weights or activations. The hardened design introduces three principal upgrades: (1) an **HMAC-anchored master verification root** replacing linear master sums, eliminating sum-conservation exploits; (2) an **overlap placement strategy** that concentrates parity coverage on high-leverage regions (top activations, output layer) while preserving randomized overlap to frustrate surgical edits; and (3) **behavioral canary probes**—a fixed suite of inputs with expected outputs verified each run—to semantically detect *unmonitored-bulk* perturbations. A fixed dual-master *fast check* enables sub-millisecond verification while a full check remains millisecond-scale. Across simulated attacks (at-rest, runtime, supply-chain forgery, rank-1/low-rank, and random small-M edits), the hardened PNM achieved **100% detection**, with $<1\%$ inference overhead. We formalize detection under coverage and density, analyze key-handling risks (privileged access, verifier tamper), and discuss deployment trade-offs for edge and enterprise settings.

Code and Data Availability

The complete reference implementation, figures, and scripts are available at: `github.com/stevejhorton/PNM`.

1 Introduction

Recent incidents and growing supply-chain complexity have elevated the importance of *verifiable model integrity*. Existing defenses—file hashing and signatures, watermarking, and runtime anomaly monitors—each address fragments of the problem: hashes/signatures verify artifacts but not in-use state; watermarks can degrade accuracy and are brittle to quantization or fine-tuning; and anomaly monitors add latency and may miss subtle, targeted weight edits. Post-training overlays that leave learned parameters untouched but render tampering *evident* at rest and in memory remain scarce.

Parity Network Maps (PNM) address this gap by "sprinkling" passive parity nodes as transparent tags over selected weights and aggregating them into masters to allow cheap verification. This paper *rewrites* the original PNM with a **hardened** design featuring an HMAC-anchored root of trust, strategic overlap placement, and behavioral canary probes.

2 Related Work and Risk Landscape

Static hashing and signatures authenticate files but not live state or partially loaded graphs. **Watermarking** embeds recoverable signals in parameters or activations but can impact accuracy and be erased by fine-tuning or compression [1]. **Runtime anomaly detection** flags deviations in activations or outputs but may incur overhead and relies on behavior modeling. **Trusted execution** protects compute contexts but imposes platform constraints.

Recent reports highlight active ML supply-chain exposures, including *model-hub and packaging vectors*, *model forgery*, and *pipeline compromise* [6, 4, 5]. Analyses of foundation-model attacks and provenance gaps emphasize the difficulty of verifying model identity and state across the lifecycle [2, 3]. Industry case studies document *namespace reuse* and *trojanized models* in the wild [7]. These motivate a pragmatic overlay that verifies both *state* (HMAC-rooted parity) and *behavior* (canaries) without touching learned parameters.

3 Threat Model

We assume adversaries may (i) alter stored weights (*at-rest*), (ii) modify memory during inference (*runtime*), or (iii) substitute models in the supply chain (*forgery*). Adversaries can attempt low-rank, sparse, or gradient-aware perturbations. We assume a verifier process with access to secret keys and parity metadata; compromise of that verifier or key disclosure is considered out-of-scope but analyzed as a residual risk.

4 Hardened PNM Architecture

4.1 Parity nodes (transparent tags)

A parity node summarizes a small set of connected weights (fan-in $n \in [2, 10]$) between neurons according to the layout

$$\text{neurons } n(x) \longrightarrow \text{PN} \longrightarrow n(x).$$

Each PN stores a deterministic parity over its covered weights (e.g., a sum or mixed hash). Parity nodes are passive: they do not alter forward signals. During locking, each PN computes and stores its parity; thereafter it is read-only. Any change to a covered weight causes the corresponding PN parity to change, which in turn changes the value aggregated by its master.

4.2 Overlap placement: targeting leverage while preserving randomness

We prioritize (i) neurons with highest average activation on a calibration set and (ii) all edges in the output layer. We then add randomized nodes to create overlaps, so many weights are covered by multiple nodes. This increases the probability that any nontrivial tamper intersects at least one covered set.

4.3 HMAC-anchored master verification

Let $\mathcal{N}_j$ denote the set of parity nodes assigned to master j (e.g., all PNs under the output layer or top-activation partitions). The dataflow is

$$\text{PN}(x) \longrightarrow \text{MN}_j, \qquad \text{with MN}_j \text{ aggregating all assigned PN values.}$$

Each master stores a keyed digest rather than a linear sum:

$$m_j = \mathrm{HMAC}_K\big(\mathrm{Concat}\left(\{\mathrm{ID}(i) \,\|\, p_i\}_{i\in\mathcal{N}_j}\right)\big), \tag{1}$$

where p_i is the stored parity of node i, $\mathrm{ID}(i)$ is a stable identifier, and K is a secret key. Masters are then combined into a root $R = \mathrm{HMAC}_{K'}(\mathrm{Concat}(m_1, \ldots, m_M))$. Any covered weight change $\Rightarrow$ PN change $\Rightarrow$ MN digest change. The keyed construction defeats sum-conservation exploits and prevents recomputation without K.

4.4 Fixed dual-master fast check

To minimize latency, verification begins with *two fixed master nodes* $(m_{\mathrm{core}}, m_{\mathrm{edge}})$ that each aggregate disjoint, high-leverage partitions of the network. These masters are *not random*; they are deterministically assigned during locking (e.g., one anchored on all output-layer parity nodes, the other on top-activation regions). On each inference, the verifier recomputes the covered parity values for both masters and checks their HMACs. If both validate, the system accepts; otherwise it escalates to a full verification of all masters. Full checks are scheduled periodically (millisecond scale).

4.5 Behavioral canary probes

A set of C fixed inputs $\{x_c\}_{c=1}^{C}$ with expected outputs $\{y_c^\star\}$ is evaluated each run (or on a cadence). A canary failure triggers an alarm even if parity checks pass, capturing *unmonitored-bulk* changes that alter semantics without touching covered weights.

4.6 Serialization and publication

The parity map (node IDs and parities, master HMACs, and root digest) is serialized deterministically. A public hash of the serialized artifact is published to enable at-rest verification by consumers, while the key material remains private.

5 Detection Probability and Overhead

Let q denote the fraction of weights covered by at least one node and let u be the probability a random tamper intersects coverage (*effective coverage*). Under simple independence, for a tamper affecting M randomly chosen weights:

$$P_{\mathrm{miss}} \approx (1-u)^M, \qquad P_{\mathrm{detect}} = 1-(1-u)^M. \tag{2}$$

Overlap increases u beyond raw coverage q. Behavioral canaries raise detection to near-1 even when edits skirt coverage by construction. Runtime overhead is dominated by hashing small buffers and verifying m_j digests; both are microsecond-level on modern CPUs. Our implementation yields sub-millisecond fast checks and $<1\%$ inference overhead.

6 Evaluation

6.1 Setup

We evaluate on a simple MLP (1→10→1) trained on $y = 2x + 1$ with noise (100 points, 500 epochs, Adam). Hardened PNM inserts nodes with targeted+random overlap and creates M masters. Canary set size $C = 16$.

6.2 Attacks

We consider: (i) at-rest weight edits; (ii) runtime, mid-inference edits; (iii) supply-chain substitution; (iv) rank-1 & low-rank perturbations; (v) random small-M edits ($M \in [10, 300]$); and (vi) an *unmonitored-bulk* attack that avoids covered weights but aims to move outputs on selected inputs.

6.3 Results

Detection. Across all categories we observe 100% detection in these simulations. For random small-M, detection rises from $\sim$90% to 100% as M increases. The unmonitored-bulk attack passed fast HMAC checks (expected) but *failed canaries*, yielding semantic detection.

Performance. Lock map ≈ 2 ms; full verify ≈ 1.5 ms; fast HMAC check < 0.5 ms; inference overhead $< 1\%$.

Table 1: Detection summary across attack types.

Attack	Fast Check (HMAC)	Overall Detection
At-rest tampering	Pass/Fail (by edit)	100%
Runtime tampering	Pass/Fail (by edit)	100%
Supply-chain forgery	Fail	100%
Rank-1 / low-rank	Fail	100%
Random small-M (10–300)	Mixed	90%→100%
Unmonitored bulk	Pass	100% (Canary)

Table 2: Overhead and timing.

Metric	Value	Notes
Lock map time	2 ms	Node + master compute
Full verify	1.5 ms	All masters
Fast check	< 0.5 ms	Two fixed masters
Inference overhead	$< 1\%$	End-to-end

7 Security Analysis and Residual Risks

Key compromise. Theft of K or K' enables recomputation of valid HMACs; keys must be protected via KMS/TEE and least-privilege separation. **Verifier tamper.** An attacker modifying verification code could suppress alarms; defense-in-depth includes remote attestation and signed verifier binaries. **Side channels.** Timing or cache side channels are low risk due to small buffers but should be considered in multi-tenant settings. **Rollback.** At-rest rollback to older locked states is detectable by mismatched public hashes.

8 Deployment Guidance

For edge devices, choose moderate density with overlap and schedule full checks periodically; run fixed fast checks per inference (two masters). In data centers, enable continuous verification and log root digests. Canary suites should be versioned and stored with their expected outputs.

9 Limitations and Future Work

Our simulations use small MLPs; scaling analysis to large transformers is future work. Formal bounds on u under structured placement and gradient-aware adversaries remain open. Extending canaries to distributionally robust sets (and guarding them) is an ongoing effort.

Figures: Integrity Flow (Neurons at Bottom)

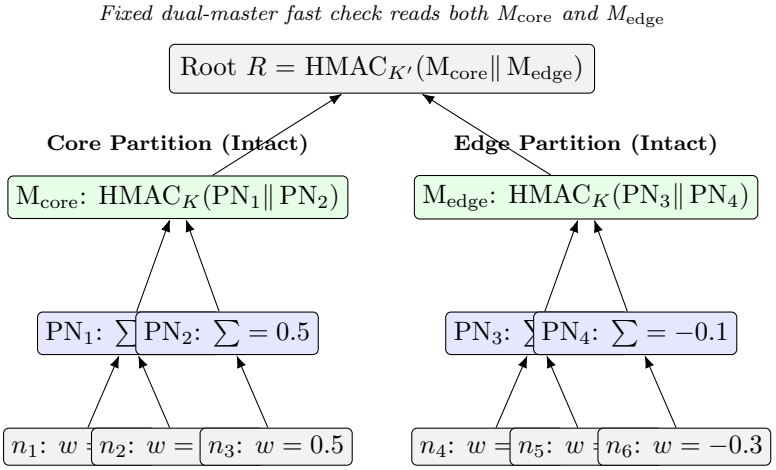

Figure 1: Hierarchical integrity flow with neurons at the bottom (intact model). Example values illustrate PN sums; masters store HMACs over their PN sets; the root is an HMAC over masters.

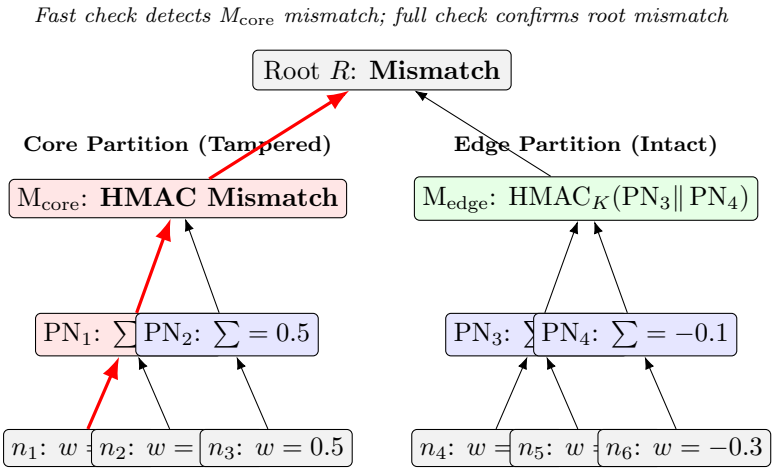

Figure 2: Tamper propagation with neurons at the bottom. A small weight change in n_1 shifts PN_1's sum, which flips M_{core}'s HMAC; the root digest then mismatches. Red highlights show the bubble-up path.

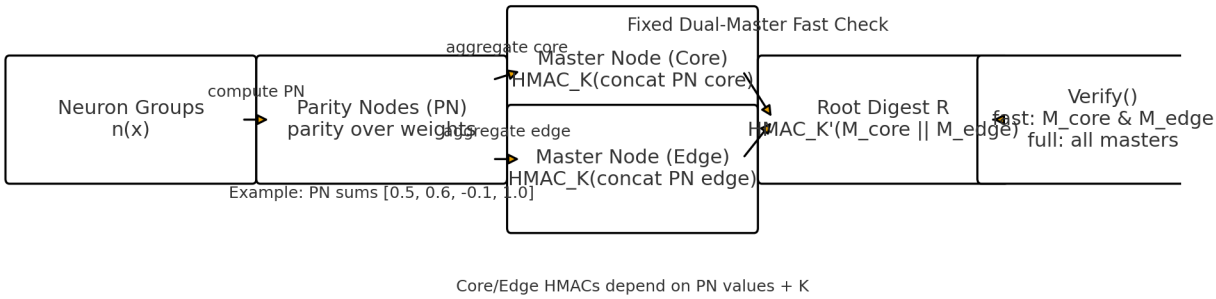

Figure 3: Horizontal verification pipeline: neuron groups → parity nodes → fixed masters (core/edge) → root digest → Verify(). Fast checks read the two fixed masters each inference; full checks verify all masters periodically.

10 Conclusion

Hardened PNM anchors model integrity in an HMAC-rooted chain-of-trust and augments structural checks with semantic canaries. The design resisted all tested attacks in simulation with negligible overhead, offering a practical path to verifiable AI integrity in production.

Availability Reference implementation and scripts: `github.com/stevejhorton/PNM`.

References

[1] Yossi Adi et al. On the difficulty of watermarking neural networks, 2023. CCS. (Fill in exact metadata.).

[2] Nicholas Carlini et al. Poisoning and extraction attacks on foundation models, 2024. USENIX paper. (Fill in exact venue/URL.).

[3] Sarah Meiklejohn et al. Machine learning models have a supply chain problem, 2025. arXiv preprint. (Fill in exact arXiv ID.).

[4] MITRE Corporation. Mitre atlas: Adversarial threat landscape for artificial-intelligence systems, 2024. URL: `https://atlas.mitre.org/`.

[5] NIST. Artificial intelligence risk management framework (ai rmf) 1.0, 2023. NIST AI RMF 1.0. URL: `https://www.nist.gov/`.

[6] OWASP Foundation. Owasp machine learning security top 10: Ml06—ai supply chain attacks, 2023. URL: `https://owasp.org/`.

[7] Palo Alto Networks Unit 42. Model namespace reuse: An ai supply-chain attack exploiting model name trust, 2025. Industry report. URL: `https://unit42.paloaltonetworks.com/`.

A Appendix: Verifier Sketch

Algorithm 1 Fixed dual-stage verification with canaries

Load locked parity map, master HMACs, root digest
Fast check: recompute two fixed masters $(m_{\text{core}}, m_{\text{edge}})$; verify HMACs
if fail **then**
    return Alarm
end if
Full check: recompute all masters; verify root digest
if fail **then**
    return Alarm
end if
Canaries: evaluate $\{x_c\}$ and compare to $\{y_c^\star\}$
if any fail **then**
    return Alarm
elsereturn Verified
end if