Verifiable Inference: Making Model Execution Checkable, Tradeable, and Mineable

Takakuni Imosuke
takakuni@tensorcash.org

Abstract

Inference can trade as a commodity only if buyers can check what they received. We present a protocol for verifying that a token sequence was produced by a declared language model under a declared sampler, at production scale, without trusting the provider's hardware and without imposing a special serving runtime.

The protocol binds each sampling draw to public challenge material and the running context. A checker can reconstruct the draw while the output remains a faithful sample from the model distribution. Verification proceeds by a ladder: sampler-contract checks, a lightweight statistical fingerprint over submitted evidence, and full teacher-forced replay under an empirical floating-point noise model. The verifier is statistical, not bit-exact, because honest logits vary across GPUs, kernels, batch shapes, and low-precision arithmetic.

The same proof object supports output provenance and resource accounting. In provenance, buyers or auditors spot-check served responses to detect model substitution or undeclared quantization. In resource accounting, a terminal hash target makes a sampled continuation rare: miners search many continuations off-chain, while validators check only the winning path. The design rides the native production forward pass, certifies the served tokens, and relies on a separate model-admission process to exclude degenerate low-entropy graphs.

1 Introduction

Language-model inference is abundant but not yet commodity-like. A buyer using an unknown endpoint cannot tell whether the response came from the promised model and precision, a surrogate, an undeclared quantization, or a cache. Verification is the missing market primitive.

The protocol here serves two settings. In **auditable serving**, a provider attaches a proof surface to responses, and a buyer or auditor checks a sample. The provider need not know which responses will be audited, so cheating is governed by audit probability and penalties. In **resource accounting**, the same proof surface is coupled to a terminal hash target. The miner performs sequential autoregressive search until one path hits the target; validators check only that submitted path.

Existing systems usually pay for checkability with either a *determinism tax* (fixed-point kernels, bit-exact execution, fraud-proof VMs, or quantized proving circuits) or a *relevance gap* (a benchmark or matrix product measured alongside the served output). This design pays neither: it keeps the optimized production forward pass and certifies the user's token sequence.

2 Design

The protocol is constrained by four requirements.

- **Native serving.** Evidence must ride on ordinary optimized serving, without deterministic kernels, disabled batching, high precision, or a special VM.
- **Unchanged semantics.** Public randomness makes the sampler draw reconstructable; it does not change the distribution from which tokens are sampled.
- **Compact external evidence.** The proof contains token IDs, sampler draws, top-50 sampler-support logits, 20 fixed tail canaries, four rank-bucket averages, precision tags, and challenge material. It does not require hidden states, attention maps, or layer traces.
- **Platform robustness.** Honest production inference is not bit-exact. The verifier must accept ordinary GPU, kernel, and batch variation while rejecting structured deviations.

The contribution is the combination of hash-bound sampling, calibrated native-FP replay, and resource-accounting semantics. Replay is not new; the point is that the replay target is the served token/logit surface, under a public sampler contract, with an optional terminal target that turns one sampled path into proof of work.

3 Protocol

3.1 Sampling Contract

Let the prompt and previous generated tokens define context C_t. At step t, the prover hashes public challenge material, t, C_t, and the declared precision tag to obtain $u_t \in [0, 1)$. The declared sampler–temperature, repetition penalty, top-k, top-p, and softmax over the retained support–maps logits to a token-ID-sorted CDF. The claimed token must be the token whose CDF bin contains u_t.

The verifier recomputes u_t, reconstructs the sampler CDF from submitted support evidence, and checks membership. This stage is model-free; it rejects stale paths, altered sampler parameters, and tokens chosen outside the public draw.

3.2 Verification Ladder

Verification has three stages.

- **Contract checks:** challenge binding, freshness, sampler bounds, proof-window length, low-entropy exclusions, and per-step CDF membership.
- **Fingerprint checks:** inert-token frequencies, top-logit gap shape, and embedding-space dispersion over the submitted top-token set. This is a cheap architecture-family screen, not a substitute for replay.
- **Full replay:** teacher-force the prompt and output through the declared model and compare submitted evidence with verifier logits under a calibrated platform-noise null.

3.3 Serving, Mining, and Admission

For auditable serving, no terminal target is needed. A provider attaches transcripts; a buyer or auditor verifies a sample. Outputs outside the sample are covered by the same deterrence model as other spot-check systems.

	Qwen3-8B (bf16)			Qwen3-32B (fp8)		
Batch	base	PoW	ovhd	base	PoW	ovhd
1	43.3	40.4	6.6%	15.5	14.7	5.3%
4	159.8	147.7	7.6%	67.2	64.9	3.4%
8	314.1	288.7	8.1%	131.9	124.5	5.6%
16	611.9	554.5	9.4%	258.0	246.3	4.6%
32	1129.5	992.8	12.1%	494.7	461.7	6.7%
64	1132.2	995.9	12.0%	495.8	462.8	6.7%

Table 1: Decode throughput (tok/s) with and without proof emission, L40S, steady state.

For mining, a terminal hash target is added to the completed window. The miner searches candidate continuations off-chain until the terminal hash falls below target. The validator checks one winning path: sampler consistency, fingerprint checks, and full replay.

Per-output verification only proves that a path is consistent with a declared graph. Open mining also needs *model admission*: entropy floors, output-diversity tests, context-sensitivity probes, difficulty calibration, review for cheap-graph constructions, and economic friction. Admission protects the entropy assumption behind proof of work.

4 Evaluation

We evaluate Qwen3-8B in bf16 and Qwen3-32B in fp8 with a 256-token proof window. Throughput is measured on an NVIDIA L40S; honest cross-hardware acceptance also uses A10G and L4 for generation.[1]

4.1 Production Overhead

Proof emission is single-digit percent for most operating points: 6.6–9.4% for the 8B model up to batch 16 and at most 6.7% for the 32B model. The 8B model reaches about 12% only at full saturation. The fixed per-token proof work is a smaller fraction of larger model forward passes, so overhead falls with model size.

4.2 Forgery Detection

We test the live production verifier–GPU teacher-forced replay, not a local reimplementation–on production-mining proofs.[2] Full replay is the decisive check: it accepts honest proofs and detects cross-architecture substitution, same-architecture fine-tuning, and low-bit quantization.

The same-architecture fine-tune and quantized model have fingerprint distances in the honest range, so distributional screening alone is insufficient. Per-token replay catches them because weight or precision changes move logits beyond the measured ~ 0.25-logit cross-platform floor. This is also the auditable-serving guarantee: a buyer or auditor can catch a provider that served a different or quantized model.

[1] L40S (g6e.xlarge), ECC off, clocks pinned, driver 580.159.04, torch 2.8.0+cu126, vLLM 0.10.0 fork; each throughput point is the mean of five reps after one warm-up. Full data and harness are released as supplementary material.

[2] Generation: production serving image, C++ proof processor, 32-prompt mining batch, 256-token window, 50 sampler-support logits plus 20 fixed tail canaries per step. Each forgery class is a corpus of 512 proofs claiming the genesis model. Per-class breakdown, corpus, and harness are released as supplementary material.

[3] Full replay uses 150 proofs per cell; fingerprint results use 512 proofs per cell.

Case	Logits from	Full replay	Fingerprint	Joint dist.
Honest	genuine bf16	97.3% GREEN, 2.7% AMBER, 0.0% RED	94.7% pass	11.6
Cross-arch substitution	Llama distill	100% detect	0.0% pass	high
Same-arch substitution	same-arch fine-tune	100% detect	98.0% pass	11.1
Low-bit quantization	nf4 4-bit	100% detect	97.5% pass	honest-like

Table 2: Full replay detects all tested forgeries; the fingerprint mostly screens architecture-family mismatches. Acceptance means non-RED; AMBER triggers repeat verification or review, not rejection.[3]

Honest bf16 vs. real 4-bit, same seed ($n = 512$ matched)	
Terminal proof-hash overlap	0/512 (0.00%)
Chosen-token overlap (per 256 steps)	2.0% mean, 1.6% median
First-divergence step (of 256)	0 median

Table 3: Low-precision search does not transfer: matched 4-bit and bf16 trajectories fork at the first token and never share the terminal proof hash.

4.3 Quantization-Search Resistance

A low-precision miner would like to search cheaply and submit the winning path under the full-precision identity. This does not transfer. The public draw u_t is fixed by challenge material, step index, and context, but the logits define the CDF that maps u_t to a token. Once low- and full-precision CDFs place u_t in different bins, the token, context, later draws, and terminal hash diverge.

5 Related Work

Prior systems differ along two axes: provenance versus resource accounting, and linear spot-checking versus constant validation of a terminal-target proof.

TopLoc, SVIP, VeriLLM, PoSP, and opML share the goal of output provenance. They differ in proof surface and execution contract: internal activations or hidden states, learned secret fingerprints, sampled re-execution, fixed-point execution, or fraud-proof VMs. This protocol instead uses an external token/logit surface, a public sampler contract, and a calibrated null over native floating-point replay. zkML gives stronger cryptographic soundness, but current LLM-scale proving is still a separate arithmetic-circuit execution, not the native serving pass.

Gonka, Ambient, and Pearl target resource accounting, but their meters are not tied to the end-to-end paid inference path. Gonka's Sprint is a separate, time-boxed transformer task used for weight/reward allocation; later serving is still spot-checked, so the auditable serving surface scales linearly with traffic and the Sprint does not align incentives for subsequent latency, capacity, or quality. Ambient validates proof-of-logits contributions, but its accounting is an obligation to run and validate model steps rather than a terminal target tied to a particular paid inference job; it also assumes reproducible/canonical logits. Pearl gives constant-time verification for arbitrary int8 matrix multiplication, but the meter is model-blind: empirical work reports accepted random matrices and mining software without inference code [Usefulness Gap]. Its vLLM path also has an input-bound decode problem: honest mining runs over live activation matrices, while an unconstrained miner can choose static matrices and favorable shapes. At realistic serving parameters, combined mining and serving are therefore dominated by splitting capacity between optimized mining and tax-free

System	Goal	Verify cost	Measured object	Production tax
TopLoc	Provenance	Linear	Internal activations	Low (hash emission)
SVIP	Provenance	Linear	Internal hidden states	Low
zkLLM	Provenance	Constant[†]	Quantized circuit	High prover cost
TEE / det. inf.	Provenance	—	Constrained native pass	Trust or determinism tax
VeriLLM	Provenance	Linear	Internal hidden states	Low sampled re-exec
PoSP	Provenance	Linear	Re-execution	Fixed-point tax
opML	Provenance	Challenge[‡]	VM re-execution	Determinism tax + delay
Gonka	Accounting	Linear	Sprint vectors[§]	Meter decoupled from serving
Ambient	Accounting	Constant	Logit hash	Assumes reproducible logits
Pearl	Accounting	Constant	int8 matmul	Model-blind; input-bound decode tax
This work (serving)	**Provenance**	**Sampled audit**	**Served-token logits**	**native pass**
This work (mining)	**Accounting**	**Constant**	**Served-token logits**	**native pass**

Table 4: Comparison by goal, verification scaling, measured object, and production-path cost. [†]zk verification is succinct, but proving is much slower than native inference. [‡]Optimistic challenge path. [§]Gonka meters a separate Sprint and spot-checks serving separately.

serving.[4]

[4]Pearl study and economic note. The supplementary vLLM study used the same Pearl model, same `vllm-miner` plugin, and stock vLLM 0.20.2 on one H100 80 GB; only the miner-side `min_m` gate changed. Raising the gate to 460000 gave a no-mining reference, 1024 reproduced stock prefill-only mining, and 64 forced decode mining. The run used OpenAI-chat load, 256 prompts, concurrency 64, warm-vs-warm contrasts, and a stubbed gateway job at an unsolvable target, isolating kernel work rather than chain submission. This is a single-GPU synthetic serving study, not a population estimate. It found a 1.40× prefill-throughput tax on the mineable surface (72040 to 51345 total tok/s; TTFT 1336 to 1832 ms) and a 5.76× decode-throughput tax when decode mining is forced (3498 to 607 output tok/s; TPOT 17.6 to 104.6 ms), with a separate two-arm run corroborating the decode direction at 3.9×. The mechanism is not merely that decode is memory-bound. For a layer GEMM $A_{m\times k}B^{\top}_{n\times k}$, useful serving work scales as $\Theta(mnk)$. Pearl's noised path adds correction/noising terms involving the low-rank factors. The following gives Pearl the benefit of the strongest caching objection: even if all weight-only material is precomputed per epoch and layer, the activation-bound term $E_{AR}(A)B'$ remains because E_{AR} is seeded from $\text{commitment}_A = H(\text{commitment}_B, A_{\text{root}})$, and A_{root} changes with the live decode activations. Thus even an ideal cached implementation retains an $\Theta(nkr)$ per-call correction against $\Theta(mnk)$ useful decode work, i.e. a relative $\Theta(r/m)$ tax. With Pearl's default rank $r = 128$ and decode $m = 64$, this term is order-two before hashing, tile waste, kernel launches, transcript extraction, and the reference implementation's uncached weight-side work; with prefill $m \geq 1024$, the same term is amortized. A separate kernel microbenchmark used Pearl's own `pearl_gemm.noisy_gemm` on one H100 SXM. Forced over Qwen3-8B decode-shaped GEMMs ($m = 64$, random int8 operands), it averaged 26.8 effective INT8 TOPS, about 1.3% of H100 peak. At a grinder-ideal shape $(m, n, k) = (4096, 12288, 4096)$ with $A = B = 0$, the same kernel reached 856.5 TOPS, 43.3% of peak. A same-shape control at $(64, 12288, 4096)$ with $A = B = 0$ reached 51.3 TOPS. Thus optimized grinding yields 31.9× the mining throughput per GPU-second of honest decode-shaped mining, decomposing into a 16.7× shape/roofline effect and a 1.9× operand-content effect. The content effect is the useful-work penalty: honest mining carries live, changing activations, whereas a grinder can choose zero or otherwise convenient matrices, avoid request latency constraints, and cache or bypass input-dependent work. MACs/joule show a directional 17.3× gap. The sub-tile waste is visible directly: $m = 64$ and $m = 128$ take essentially the same wall time, so the baby GEMM pays the control-plane and tile costs without enough work to amortize them. The reference `pearl-gemm` kernel is also Hopper-specific (`sm_90a`, WGMMA/TMA), which limits claims about commodity consumer-GPU mining. The economic model is deliberately minimal. Let M be revenue per GPU-time from optimized pure mining, S revenue from tax-free serving, τ the serving slowdown of combined mining+serving, and k the wall-clock mining-yield penalty of inference-shaped mining relative to optimized mining shapes. The combined mode yields $M/k+S/\tau$ and is rational only

6 Limitations and Future Work

The guarantee is statistical: it does not prove every multiply-add and does not replace zkML where cryptographic soundness is required. Mining hardness depends on model admission; a degenerate graph can be valid under its own logits while economically invalid as proof of work. Auditable serving still relies on audit probability and penalties for unchecked outputs. Closed-weight models require weight access by the verifier or a non-colluding auditor.

The construction is scoped to autoregressive language-model generation. Extending it to diffusion, classifiers, or multimodal models requires a compact, replayable input representation. Three useful directions remain: operator-agnostic representations that hide semantic content from the operator, header-bound prompt commitments that remove forward-pass precomputation opportunities, and compact encodings for continuous multimodal inputs.

7 Conclusion

Verifiable inference needs to survive the conditions under which inference is actually served: optimized kernels, batching, low precision, and heterogeneous hardware. This protocol binds sampling to a public contract and verifies the resulting token path with calibrated statistical replay. The same proof object supports provenance, where spot-checking deters a known provider, and resource accounting, where a terminal target turns admitted autoregressive generation into constant-time-verifiable proof of work.

A Verification Rule

A.1 Sampler Eligibility and Entropy Admission

For a proof window of length W,

$$5 \le k \le 50, \qquad 0.1 \le p \le 1, \qquad 0.25 \le \tau \le 2, \qquad 0.1 < \rho \le 1.$$

In resource accounting, the verifier also applies a cumulative reuse-score gate. At step t, let $[\ell_t, h_t]$ be the chosen token's reconstructed CDF interval and

$$w_t = \min\big(1,\ \max(0,\ (h_t - \ell_t) + 2\alpha)\big), \qquad b_t = \min\big(W_b,\ \lfloor -\log_2 w_t \rfloor\big),$$

with $W_b = 32$. Accumulate

$$P_t = \min(W_b,\ P_{t-1} + b_t), \quad P_0 = 0, \qquad R = \sum_{t=1}^{W} 2^{W_b - P_t}.$$

The window is admitted iff

$$R \le C, \qquad C = \lceil 57.716742 \cdot 2^{W_b} \rceil = 247{,}891{,}519{,}322.$$

if it beats both pure corners: $M/k + S/\tau > \max(M, S)$. With $\rho = S/M$, this requires $\tau(k-1)/k < \rho < \tau/[k(\tau-1)]$, possible only when $(k-1)(\tau-1) < 1$. For forced decode mining, the measured τ is about 5.76, so combined serving+mining is viable only if $k < 1 + 1/(\tau-1) \approx 1.21$. The measured kernel yield penalty is instead $k \approx 31.9$, giving $(k-1)(\tau-1) \approx 147 \gg 1$; no relative coin/service price can make combined decode serving and mining beat the better pure allocation. For prefill-only mining, $\tau \approx 1.4$ leaves a wider window ($k < 3.5$), so the claim is not that prefill mining is mathematically impossible. The narrower conclusion is that Pearl's economics reward cached or grindable large-GEMM surfaces rather than interactive decode work. Under the measured decode parameters, the combined allocation is strictly dominated by allocating capacity separately to optimized mining and optimized serving.

A.2 Sampler Check and Fingerprint

At step t, if the claimed token τ_t is at position m_t in the token-ID-sorted sampler CDF F_t, it must satisfy

$$F_t(m_t - 1) - \epsilon < u_t \leq F_t(m_t) + \epsilon.$$

The fingerprint uses three calibrated statistics over the submitted top-50 support: inert-token counts, consecutive top-logit gaps, and embedding-space dispersion. With calibrated means and covariances,

$$M^2_{\text{freq}} = (c - \bar{c})^\top \Sigma_c^{-1} (c - \bar{c}), \qquad M^2_{\text{gap}} = (\hat{g} - \bar{g})^\top \Sigma_g^{-1} (\hat{g} - \bar{g}),$$

and, after reducing embeddings to 16 principal components,

$$M^2_{\cos} = \left(\frac{\operatorname{arctanh}(2\bar{s} - 1) - \mu_z}{\sigma_z} \right)^2.$$

All three must lie below their calibrated empirical 99th-percentile thresholds.

A.3 Full Replay

Let L_t^B be verifier logits, S_t the 70 submitted indices (top-50 support plus 20 fixed canaries), and $\tilde{L}_t^A \in \mathbb{R}^{70}$ the submitted logits. Define

$$\Delta\mu_t = \operatorname{mean}(\tilde{L}_t^A) - \operatorname{mean}(L_t^B[S_t]), \qquad d_t = \tilde{L}_t^A - L_t^B[S_t] - \Delta\mu_t \in \mathbb{R}^{70}.$$

For four rank buckets, define

$$m_t = \mu_t^A - \mu_t^B - \Delta\mu_t \in \mathbb{R}^4, \qquad v_t = (d_t, m_t) \in \mathbb{R}^{74}.$$

The verifier snaps replayed logits to the declared dtype grid:

$$u_t^{\text{ulp}} = 2 \text{ ulp}(L_t^B[S_t]; \text{dt}), \qquad \widehat{L}_t^B = \operatorname{snap}(L_t^B[S_t]; u_t^{\text{ulp}}).$$

If submitted logits are off-grid beyond tolerance, the proof fails. Otherwise

$$\Sigma_t = \operatorname{diag}(\sigma_t) C \operatorname{diag}(\sigma_t),$$

where σ_t is the maximum of analytic rounding, magnitude, and empirical covariance floors, and C is the calibrated 74-coordinate correlation matrix.

The statistic is

$$T_t^{\text{obs}} = v_t^\top \Sigma_t^{-1} v_t.$$

Because the snapped component is lattice-valued, the null is bootstrapped rather than treated as χ^2_{74}. Draw $z^{(b)} \sim \mathcal{N}(0, I_{74})$, set $\eta^{(b)} = \operatorname{chol}(\Sigma_t) z^{(b)}$, and form

$$x^{(b)} = L_t^B[S_t] - \Delta\mu_t + \eta^{(b)}_{1:70},$$

$$D^{(b)}_{\text{null}} = \widehat{L}_t^B - \operatorname{snap}(x^{(b)}; u_t^{\text{ulp}}), \qquad C^{(b)}_{\text{null}} = \eta^{(b)}_{71:74}.$$

Then

$$\hat{p}_t = \frac{1}{B} \sum_{b=1}^{B} \mathbf{1}\left\{ (D^{(b)}_{\text{null}}, C^{(b)}_{\text{null}})^\top \Sigma_t^{-1} (D^{(b)}_{\text{null}}, C^{(b)}_{\text{null}}) \geq T_t^{\text{obs}} \right\}.$$

Per-step p-values aggregate into GREEN, AMBER, or RED verdicts.

B Transcript Fields

The transcript contains canonical message ordering, little-endian integer encodings, rolling context width, per-step SHA-256 hashes, terminal window hash, proof hash, challenge material, freshness binding, sampler parameters, precision tag, dtype lattice, normalizers, token IDs, per-step draws, CDF membership evidence, top-50 support logits, 20 fixed tail canaries, and four rank-bucket averages.

References

[Ambient] Ambient. *Ambient Litepaper V1.* `https://ambient.xyz/Ambient_Litepaper_V1.pdf`

[EigenAI] Alves et al. *EigenAI: Deterministic Inference, Verifiable Results.* arXiv:2602.00182. `https://arxiv.org/abs/2602.00182`

[Gonka Whitepaper] Liberman. *Decentralized AI: Meaningful utilization of computational power for real-world application.* `https://gonka.ai/whitepaper.pdf`

[Gonka PoW] Gadaev, Liberman, Matveeva, Morgachev, and Shulgin. *Transformer-Based Proof-of-Work: Aligning Voting Power with LLM Computation.* `https://gonka.ai/pow-security-analysis.pdf`

[opML] Conway et al. *opML: Optimistic Machine Learning on Blockchain.* arXiv:2401.17555. `https://arxiv.org/abs/2401.17555`

[Pearl PoUW] Komargodski, Schen, and Weinstein. *Proofs of Useful Work from Arbitrary Matrix Multiplication.* arXiv:2504.09971. `https://arxiv.org/abs/2504.09971`

[PoSP] Zhang and Wang. *Proof of Sampling: A Nash Equilibrium-Secured Verification Protocol for Decentralized Systems.* arXiv:2405.00295. `https://arxiv.org/abs/2405.00295`

[SVIP] Sun et al. *SVIP: Towards Verifiable Inference of Open-source Large Language Models.* arXiv:2410.22307. `https://arxiv.org/abs/2410.22307`

[TopLoc] Ong et al. *TOPLOC: A Locality Sensitive Hashing Scheme for Trustless Verifiable Inference.* arXiv:2501.16007. `https://arxiv.org/abs/2501.16007`

[Usefulness Gap] Basu. *The Usefulness Gap in Proof-of-Useful-Work: An Empirical Study of Pearl's cuPOW Protocol.* arXiv:2606.04819. `https://arxiv.org/abs/2606.04819`

[VeriLLM] Wang et al. *VeriLLM: A Lightweight Framework for Publicly Verifiable Decentralized Inference.* arXiv:2509.24257. `https://arxiv.org/abs/2509.24257`

[zkLLM] Sun, Li, and Zhang. *zkLLM: Zero Knowledge Proofs for Large Language Models.* arXiv:2404.16109. `https://arxiv.org/abs/2404.16109`