Is the Declared Model Still Running? Continuous Identity Attestation for LLM Serving

Research collaboration: cccisi95 AT gmail dot com

A client sends a sensitive task to an LLM service because it trusts a particular model. The endpoint advertises an approved checkpoint, perhaps with a specific adapter and quantization configuration. Yet the response alone reveals little about what actually ran. A valid signature on a downloaded checkpoint establishes its origin; a successful platform attestation establishes a trusted environment [1]. Neither, on its own, continuously identifies the model bytes that a busy serving system keeps in GPU memory.

This gap grows as deployment becomes more dynamic. One model spans several GPUs; experts move between host and device memory; adapters arrive while the service is live; and prefix caches carry state from one request into another. The question is therefore no longer just whether the model was authentic when loaded. It is whether the service can keep producing fresh, independently checkable evidence that its active deployment still matches the model it claims to serve.

We propose to make that evidence part of normal inference. A small, trusted measurement plane would inspect selected resident weight pages between inference operations and issue receipts bound to an approved model commitment, a serving epoch, and a verifier challenge. The ambition is to approach performance-neutral auditing under favorable workloads, while measuring the cost honestly when no spare capacity exists. The resulting guarantee is probabilistic integrity over a stated measurement window; proving the correctness of every generated token remains a separate problem.

Following the model through a live deployment

The story begins before the first request. A publisher signs a manifest describing the exact deployed tensor bytes, numerical format, quantization parameters, tokenizer, and relevant runtime configuration. Fixed-size weight pages become leaves of domain-separated Merkle trees. Rather than pulling all weights to rank zero, each tensor-, pipeline-, or expert-parallel rank builds its own subtree. An authenticated hierarchy then combines the small rank roots into one deployment commitment. A balanced reduction can have logarithmic depth, although its actual cost depends on topology and the collective implementation. For offloaded experts, the manifest also identifies the memory tier and a transfer epoch, so relocation cannot silently change the artifact being measured.

Once serving begins, an attested controller fixes the epoch and obtains a fresh verifier nonce. It uses protected randomness to choose pages after that commitment is fixed; signing a predictable seed would not make the sampling unpredictable. GPU kernels read the selected pages, compare their digests with the committed values, and return compact evidence. The controller signs a receipt that binds the challenge, model root, rank and page identifiers, measured coverage, runtime identity, and request or epoch identifier. Clients can verify an epoch receipt once and refer to its digest across responses, avoiding a full proof in every response header.

The difficult part is making the measurement refer to the bytes inference actually uses. The measured runtime must bind audited allocations to its execution path and control page replacement, transfers, and adapter activation. The controller and signing key belong inside an attested confidential workload, with CPU and GPU evidence checked against policy [1]. An untrusted host must not be able to substitute a clean shadow copy for auditing. Even then, sampling leaves gaps between checks; the design must state and evaluate those gaps rather than imply that a receipt proves an entire response.

Extending assurance without interrupting inference

As the prototype matures, the same receipt should follow the state that changes around the base model. For LoRA, a verified publisher signature, adapter digest, compatible base root, and activation epoch identify the selected composition; verification results can be cached by content digest rather than adapter name [3]. For prefix caching, vLLM already hashes tokens, parent blocks, and additional context to identify reusable entries [2]. Our extension would authenticate sampled KV *content* and its allocation generation at reuse, because an indexing key alone does not establish the integrity of the stored bytes. Tenant-scoped, keyed commitments would avoid publishing raw prompts or reusable low-entropy prompt hashes.

Performance determines whether this can become a deployment feature. Spare SMs do not imply spare memory bandwidth: decode and hashing can both compete for HBM. We will therefore start with budgeted page checks and compare ordinary streams, CUDA Graph integration, and supported Green Context partitions [4]. Graphs can reduce launch overhead, while Green Contexts spatially partition SM resources; neither automatically removes contention or guarantees an SLA. Hash selection will follow measurements on the target GPU, without assuming a particular BLAKE3 throughput or an unvalidated Verkle advantage.

The first experiment will integrate dense tensor-parallel serving in vLLM, comparing no auditing, full periodic hashing, and rank-local sampling with hierarchical receipts. Later experiments will add MoE offload, dynamic adapters, and shared prefix caches. Across batch sizes and prompt lengths, we will report tokens/s, time to first token, p50/p99 inter-token latency, audit bandwidth, receipt size, and verifier cost alongside detection coverage. Persistent page changes, transient changes, adapter substitutions, stale manifests, replayed receipts, and cache corruption will test what the design actually catches. Rejected or missing evidence must be visible to clients, rather than silently counted as a successful audit.

For a persistent change to one of N pages, an independent uniform audit of k distinct pages detects it with probability k/N; after t independent audits the probability is $1-(1-k/N)^t$. This model does not cover an adaptive attacker that restores pages before checks, and short-lived corruption can escape. Verifier state or trusted monotonic policy will reject model rollback, while nonce-bound receipts address replay. The same checks may detect some memory-related silent data corruption, but they cannot establish the absence of arithmetic faults. These limits will be reported with overhead and time-to-detection, making the assurance/performance tradeoff explicit.

The contribution we seek is a practical connection between model identity and the changing state of distributed serving. Platform attestation [1], GPU attestation research such as SAGE [5], and sampled inference proofs [6] provide foundations and complementary guarantees. We will test whether a protected measurement plane can sustain useful coverage with negligible service impact in some workloads, and characterize the cost in others. The deliverables are a threat model, an open prototype and independent verifier, and a reproducible benchmark connecting every reported performance point to its security coverage.

Origin, open source, and working together

The research agenda proposed here originates with **Zhaohui Chen (2026)**; the foundations above are credited in the references. This proposal invites research and resulting publications **in collaboration with Zhaohui Chen**. Please contact **cccisi95 AT gmail dot com** to discuss a joint project. The prototype is intended for release under **Apache License 2.0**, preserving attribution in copyright and NOTICE files. This text is shared under **CC BY 4.0** with attribution to Zhaohui Chen. Collaboration is a condition of joining this research project, not an additional restriction on released open-source code.

1. NVIDIA, Attestation and key-release flow.
2. vLLM, Automatic prefix caching.
3. vLLM, LoRA adapters and dynamic loading.
4. NVIDIA, CUDA Green Contexts.
5. Ivanov et al., SAGE: Software-based Attestation for GPU Execution, USENIX ATC 2023.
6. Lim and Yong, Seal, Then Sample: Sampled Layerwise Proofs for Verifiable LLM Inference, 2026.