TensorCommitments: A Lightweight Verifiable Inference for Large Language Models

Oguzhan Baser[1*] **Elahe Sadeghi**[1] **Eric Wang**[2] **Nico Vergauwen**[2]
Sam Kazemian[2] **Hong Kang**[3] **Sandeep P. Chinchali**[1] **Sriram Vishwanath**[4]
[1]Electrical and Computer Engineering, The University of Texas at Austin
[2]Theseus AI Labs
[3]Electrical and Computer Engineering, McGill University
[4]Electrical and Computer Engineering, Georgia Institute of Technology

Abstract

Most large language models (LLMs) run on external clouds: users send a prompt, pay for inference, and must trust that the remote GPU executes the LLM without any adversarial tampering. We critically ask how to achieve verifiable LLM inference, where a prover (the service) must convince a verifier (the client) that an inference was run correctly without rerunning the LLM. Existing cryptographic works are too slow at the LLM scale, while non-cryptographic ones require a strong verifier GPU. We propose TensorCommitments (TCs), a tensor-native proof-of-inference scheme. TC binds the LLM inference to a commitment, an irreversible tag that breaks under tampering, organized in our multivariate Terkle Trees. For LLaMA2, TC adds only 0.97% prover and 0.12% verifier time over inference while improving robustness to tailored attacks by up to 48% over the best prior work requiring a verifier GPU.

1 Introduction

As large language models (LLMs) move *from chatbots to decision-makers*, we increasingly have to **trust** remote inference runs that we cannot see or replay. Today, LLMs help draft code and execute tools in software pipelines, coordinate multi-agent workflows, and support decisions in finance, healthcare, and law [41]. In many of these settings, an LLM runs on external infrastructure, produces a stream of tokens, and *downstream systems act on those tokens as if they were ground truth*. If the underlying computation is *silently corrupted* - by hardware faults, misconfiguration, or even a compromised service - agents may trade on the wrong market signal, a medical assistant may summarize the wrong patient record, or an autonomous system may reason over tampered state [41].

Current deployments provide almost **no traceability** of the intermediate activations or hidden states of these billion-parameter models; only the final text is visible, and re-running the full model just to check every answer is prohibitively expensive. As LLM usage scales to persistent, tool-using, and multi-agent systems, the gap between the *impact* of a single faulty inference and our ability to cheaply verify it is widening [25]. This motivates the **central question** of this work: *can we design a practical mechanism that allows lightweight clients to check whether a complex LLM inference was executed correctly, without re-running the model or exposing its internal states?* The question, and our mechanism, extend to any setting where one party pays for computation it cannot observe, e.g., compute markets, chained multi-party inference, federated aggregation, and agents (App. L).

Existing works on verifiable LLMs *only partially* address this and run into *sharp trade-offs* between scalability, privacy, and verifier cost. Cryptographic systems based on *zero-knowledge* (ZK) proofs [7, 18, 24, 26, 36, 42] can certify end-to-end correctness of an inference while hiding model parameters,

*Correspondence to: Oguzhan Baser <oguzhanbaser@utexas.edu>.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

but by compiling every tensor operation into a large constraint system. For LLM-scale models, this leads to minutes of prover runtime per query and huge extra compute compared to plain inference, which is hard to amortize in interactive or multi-agent workloads. In parallel, *learning-based works* [33, 37] train auxiliary models to detect perturbed outputs, or "critique" reasoning traces, but their guarantees are statistical, with no cryptographic binding. Also, they need retraining or fine-tuning whenever the base model or domain shifts. A very recent strong method, TOPLOC [30], takes a more practical route and fits last activations to a polynomial to detect model, prompt, or precision changes with low storage overhead, but the verifier still needs to recompute the forward pass with access to model weights and substantial GPU memory. It exposes internal states and thus violates user privacy. Further, these schemes flatten the compute into vectors, checking correctness per token instead of maintaining the tensor layout (e.g., spatiotemporal grids) [9]. This vector-centric view inflates the proof size with each new dimension and offers no direct way to reason about localized changes, motivating a tensor-native scheme that scales to LLMs with light-compute verifiers.

Our **key technical insight** is that commitment schemes are conceptually well matched to verifiable inference: if each hidden state and parameter block is bound to a succinct commitment, a remote run can be audited with a few proofs rather than replaying the full model. However, existing constructions are fundamentally *vector-centric*: they encode long flat vectors with *univariate polynomials* [19]. For billion-parameter LLMs, the polynomial degree and the committed vectors grow with every layer and token, so interpolation rapidly dominates the cost, even without multi-agent interactions. Our **key observation** is that this bottleneck stems from *the univariate view*, not from commitments themselves. By viewing parameters as tensors and committing to *multivariate* polynomials to respect their natural axes (e.g., layers and heads), we preserve tensor structure and exploit fast multivariate evaluation to keep commitments and proofs efficient. Fig. 1 shows this core scaling behavior: *for fixed data, multivariate interpolation becomes substantially more efficient than the univariate one as dimensionality increases*. This makes tensor-native commitments **a natural fit** for LLM workloads. Fig. 2 gives the overview: a model host attaches *TensorCommitments*, organizes them into *an authenticated tree*, and streams compact proofs that *a lightweight client* can check without GPUs. In light of prior work, our contributions are fourfold:

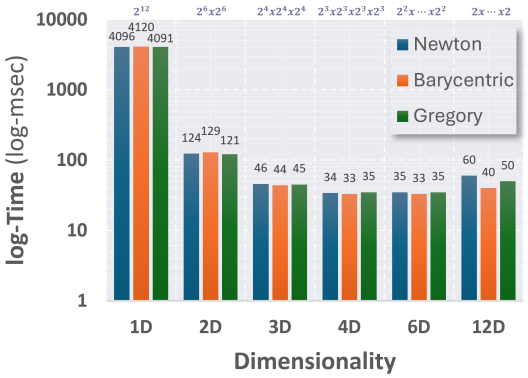

Figure 1: **Our key observation:** *multivariate interpolation is faster.* We plot log-runtime to interpolate a polynomial over a fixed grid of $N = 2^{12}$ samples, reshaped from 1D (2^{12} points) to mD grids ($2^{\frac{12}{m}} \times \cdots \times 2^{\frac{12}{m}}$) using Newton, Barycentric, and Gregory interpolation. Across all, moving from univariate ($m = 1$) to bivariate ($m = 2$) cuts runtime from 4.1s to 0.125s (over **30**× speedup), with further reductions as dimension increases. Its time bound $\mathcal{O}(\binom{m+\lfloor N^{1/m} \rfloor}{m}^2)$, derived in App. H, decreases sharply as the tensor dimension m grows.

- We design a **multivariate commitment scheme** tailored to *verifiable LLM inference*.
- We introduce **Terkle Trees**, a tensor-native authentication structure that tracks evolving hidden states and tensors with just *a single root of multivariate proofs.*
- We design a **layer selection algorithm** to verify the *most critical layers* for the model output, cutting overhead while improving detection of targeted attacks.
- We *systematically study* these mechanisms on several LLMs and *tailored attacks*, and **open source our implementation**[2] ready to merge into inference pipelines.

2 Background

Here, we outline the main concepts shaping our framework. **Vector Commitments (VCs)** bind a vector to a single group element with succinct *proofs* at single points. Let $\mathbb{F}$ be a finite field, and let G, G_T be pairing-friendly elliptic curves of prime order p equipped with a bilinear map

[2] https://github.com/UTAustin-SwarmLab/TensorCommitments

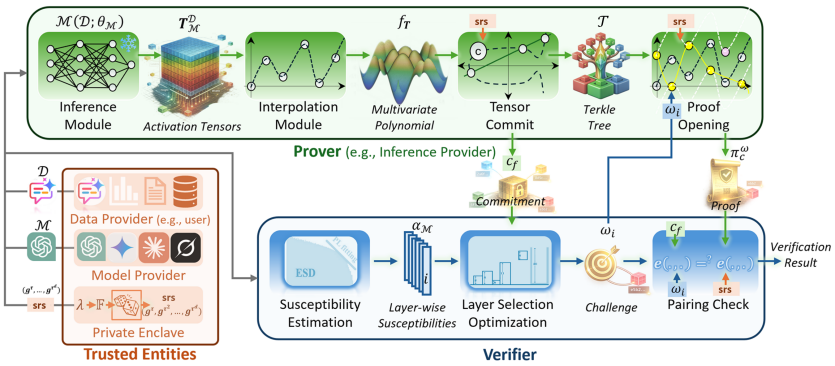

Figure 2: **How does our verifiable inference pipeline work?** *Trusted setup* publishes structured reference string srs (Sec. 4) with the model and data. *Prover* runs model $\mathcal{M}(\mathcal{D};\theta_{\mathcal{M}})$ to produce activation tensors $T^{\mathcal{D}}_{\mathcal{M}}$; interpolates into multivariate polynomial $f_{T^{\mathcal{D}}_{\mathcal{M}}}$; commits to obtain C_f and builds Terkle tree $\mathcal{T}$; upon verifier challenge ω_i, provides opening proofs $\pi_C^{\omega_i}$. *Verifier* uses spectral heavy-tail scores $\alpha_{\mathcal{M}}$ to rank layers, solves the interval selector, Problem 4.5, to choose challenge $\{\omega_i\}$, and checks pairing $e(\cdot,\cdot)$, accepting only if all checks pass. The prover does all heavy work. The verifier checks pairings and *does not* re-run full inference.

$e : G \times G \to G_T$. App. E.1 explains these algebraic structures in more detail. Let $g \in G$ be a generator and let $[d] := \{1,\dots,d\}$. We use boldface $\mathbf{v} \in \mathbb{F}^d$ for vectors of dimension d.

Definition 2.1. Let $\lambda \in \mathbb{N}$ be a *security parameter*, typically $\lambda \geq 128$ bits, determining the computational hardness of breaking the scheme. The security is satisfied when all probabilities of adversarial success are negligible in λ, i.e., $\mathrm{negl}(\lambda)$ decays faster than any inverse polynomial in λ.

Definition 2.2. *Vector commitments* VC have four methods: $\mathsf{Setup}_{\mathsf{VC}}(.), \mathsf{Com}_{\mathsf{VC}}(.), \mathsf{Open}_{\mathsf{VC}}(.), \mathsf{Ver}_{\mathsf{VC}}(.)$, where $\mathbf{v} = (v_1,\dots,v_d) \in \mathbb{F}^d$ is the committed vector, $i \in [d]$ is an index, $C \in G$ is a commitment, and π_i is an opening proof for coordinate i with the claimed value $y \in \mathbb{F}$.

$\underline{\mathsf{Setup}_{\mathrm{vc}}(\lambda, d) \to \mathsf{pp}}$: It samples trapdoor $\tau \in \mathbb{F}$ uniformly via the security parameter λ (detailed in App. E.3) and publishes the structured reference string $\mathsf{srs} = (g^{\tau^0},\dots,g^{\tau^d}) \in G^{d+1}$ together with evaluation domain $\Omega = \{\omega_1,\dots,\omega_d\} \subset \mathbb{F}$ of distinct points, where public parameters $\mathsf{pp} = (\mathsf{srs}, \Omega)$.

$\underline{\mathsf{Com}_{\mathrm{vc}}(\mathsf{pp}, \mathbf{v}) \to C}$: Given $\mathbf{v} \in \mathbb{F}^d$, it interpolates a univariate polynomial $f \in \mathbb{F}[X]$ of degree $< d$ satisfying $f(\omega_i) = v_i, \forall i \in [d]$, and outputs $C = g^{f(\tau)} \in G$ obtained by multi-exponentiation using srs.

$\underline{\mathsf{Open}_{\mathrm{vc}}(\mathsf{pp}, \mathbf{v}, i) \to (y, \pi_i)}$: To prove $\mathbf{v}$ at position $i \in [d]$, it sets $y := v_i$ and computes the quotient polynomial $q(X) := (f(X) - y)/(X - \omega_i)$. It outputs $\pi_i := g^{q(\tau)}$.

$\underline{\mathsf{Ver}_{\mathrm{vc}}(\mathsf{pp}, C, i, y, \pi_i)}$: If $e(C \cdot g^{-y}, g) \stackrel{?}{=} e(\pi_i, g^{\tau - \omega_i})$, it accepts y by the pairing. $g^{\tau-\omega_i}$ comes from srs.

Definition 2.3. A VC is *correct* if, for any honestly generated $(\mathsf{pp}, \mathbf{v}, C)$ and $\forall i \in [d]$, the opening $(y, \pi_i) \leftarrow \mathsf{Open}_{\mathrm{vc}}(\mathsf{pp}, \mathbf{v}, i)$ satisfies $y = v_i$ and $\mathsf{Ver}_{\mathrm{vc}}(\mathsf{pp}, C, i, y, \pi_i) = 1$. Honest (C, π) never get rejected.

Definition 2.4. A VC is *position-binding* if, for a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, the probability that $\mathcal{A}$ outputs $(y, \pi) \neq (y', \pi')$ for (pp, C, i), such that $\mathsf{Ver}_{\mathrm{vc}}(\mathsf{pp}, C, i, y, \pi) = 1$ and $\mathsf{Ver}_{\mathrm{vc}}(\mathsf{pp}, C, i, y', \pi') = 1$ is negligible in λ. Once C is fixed, changes in any entry fail the verification.

Intuitively, even a compute-heavy adversary cannot take C and later convince the verifier of a different v_i at i. Hence, C behaves like a binding handle to a unique vector.

Merkle Trees (MTs) are cryptographic structures that use a hash function H to commit to a data vector $\mathbf{x} = (x_i)_{i \in [n]}$ [29]. x_i become leaves of a full B-ary MT of depth $L = \lceil \log_B n \rceil$ with root u_0.

Definition 2.5. We set $h(u) := H(x_i)$ for each leaf node u holding value x_i and $h(u) := H\big(h(c(u,1)) \,\|\, \dots \,\|\, h(c(u,B))\big)$ for each internal node u holding concatenated hashes of its children $c(u,1),\dots,c(u,B)$. Then, the *Merkle commitment* to $\mathbf{x}$ is defined as the root hash $C^{\mathrm{M}} := h(u_0)$.

Definition 2.6. Let $i \in [n]$ index a leaf with root-to-leaf path $(u_0,\dots,u_L)$, where u_L stores x_i, and let $j_d \in [B]$ be the position of u_{d+1} among the children of u_d. For each level $d \in \{0,\dots,L-1\}$, define the sibling multiset $\mathbf{\Psi}_d := \big(h(c(u_d, j))\big)_{j \in [B] \setminus \{j_d\}}$. Then the *Merkle proof* for x_i is defined as: $\pi_i^{\mathrm{M}} := (\mathbf{\Psi}_0,\dots,\mathbf{\Psi}_{L-1})$.

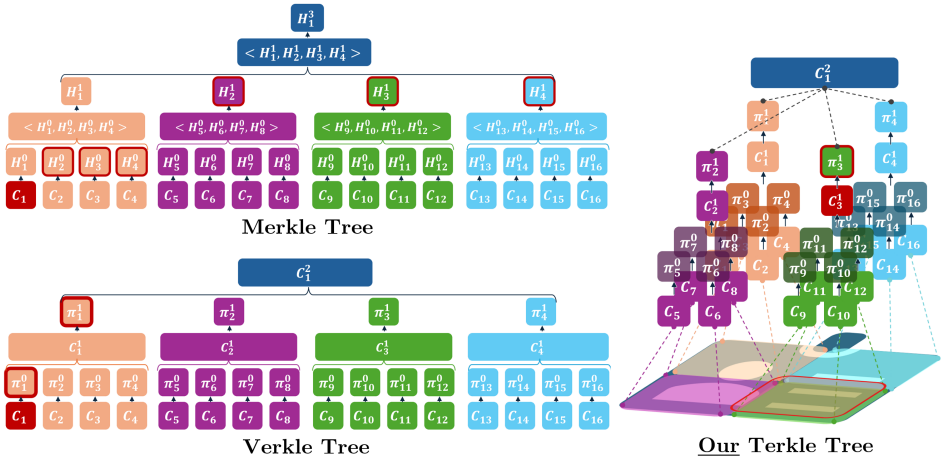

Figure 3: **Which verifiable tree aligns best with the tensor structure while keeping the proofs succinct?** *(Top)* A B-ary Merkle tree commits to leaf values C_i via hash labels H_j^d; a membership proof for a single leaf (highlighted in red) must include all sibling hashes along the path from leaf to root, treating the state as a flat list of values. *(Bottom)* A Verkle tree replaces hash parents with vector commitments C_j^1 and per-level opening proofs π_j^d, reducing proof size to $\mathcal{O}(\log_B n)$ but still indexing children in one dimension, without exploiting any tensor structure in the underlying model or feature map. *(Right)* A Terkle tree commits at the root C_1^2 to a tensor-shaped grid of parameters or features (illustrated as colored regions on the base plane); each internal node C_j^1 corresponds to a multi-dimensional block, and each π_j^d is a multivariate opening at a specific tensor index. This tensor-native organization allows authenticating the entire LLM or multi-agent states with a single root while informing about structured subsets (e.g., spatial patches) using fewer openings.

Given $(C^{\mathrm{M}}, i, x_i, \pi_i^{\mathrm{M}})$, the verifier recomputes the leaf hash $H(x_i)$, then hashes upward using the sibling sets $\mathbf{\Psi}_d$ to reconstruct a root $\hat{C}$ and accepts if $\hat{C} = C^{\mathrm{M}}$.

Remark 2.7. For a B-ary Merkle tree with n leaves, each proof π_i^{M} contains $\mathcal{O}((B-1)\log_B n)$ hashes and verification performs $\mathcal{O}(\log_B n)$ evaluations of H.

MTs are cheap and common (e.g., in blockchains) when n is moderate. For LLMs with billions of tensors, Remark 2.7 forces smaller B to keep proofs small, leading to very deep trees and large bandwidth. Also, hashing destroys tensor relations and algebraic structure: verifying related tensor entries (e.g., adjacent tokens) requires independent proofs with no amortization across the tensors.

Verkle Trees (VTs) build on MTs by replacing the hash function with VCs [23] with 4 functions given in Def. 2.2 and we reuse the same notation as in MTs and VCs.

Definition 2.8. For each node u, VT collects its B children into $\mathbf{z}_u := (z_{u,1}, \ldots, z_{u,B}) \in \mathbb{F}^B$ and sets the node commitment $C_u := \mathsf{Com}_{\mathsf{VC}}(\mathbf{z}_u)$. $z_{u,j}$ is data x_i at leaves and it is the commitment of child $c(u,j)$ at internal nodes. Then, *Verkle root commitment* to $\mathbf{x}$ is defined as $C^{\mathrm{V}} := C_{u_0}$.

Definition 2.9. For a leaf $i \in [n]$ with root-to-leaf path $(u_0, \ldots, u_L)$ and child positions $j_0, \ldots, j_{L-1}$, the prover opens $\pi_d := \mathsf{Open}_{\mathsf{VC}}(\mathbf{z}_{u_d}, j_d)$ for each level $d = 0, \ldots, L-1$. Then, its *Verkle proof* is defined as $\pi_i^{\mathrm{V}} := (\pi_0, \ldots, \pi_{L-1})$.

Given $(C^{\mathrm{V}}, i, x_i, \pi_i^{\mathrm{V}})$, the verifier walks the same MT path, checks $\mathsf{Ver}_{\mathsf{VC}}(C_{u_d}, j_d, z_{u_d,j_d}, \pi_d) = 1, \forall d$.

Remark 2.10. A VT over n leaves has $\mathcal{O}(\log_B n)$ proof size and $\mathcal{O}(\log_B n)$ verification time.

This states VTs remove the B-1 factor in Remark 2.7 and enable large-arity trees with succinct proofs.

3 Problem statement

Consider LLM $\mathcal{M}: \mathcal{X} \times \Theta \to \mathcal{Y}$ with parameters $\theta_{\mathcal{M}} \in \Theta$ and layers $\mathcal{L}$. For input $x \sim \mathcal{X}$, an honest inference produces output $y = \mathcal{M}(x; \theta_{\mathcal{M}})$ and a collection of activation tensors $T_{\mathcal{M}}^x = \{T_\ell(x)\}_{\ell=1}^{\mathcal{L}}$, where $T_\ell(x) \in \mathbb{F}^{d_1^{(\ell)} \times \cdots \times d_{m_\ell}^{(\ell)}}$. As shown in Fig. 2, a trusted setup uses security parameter λ to form

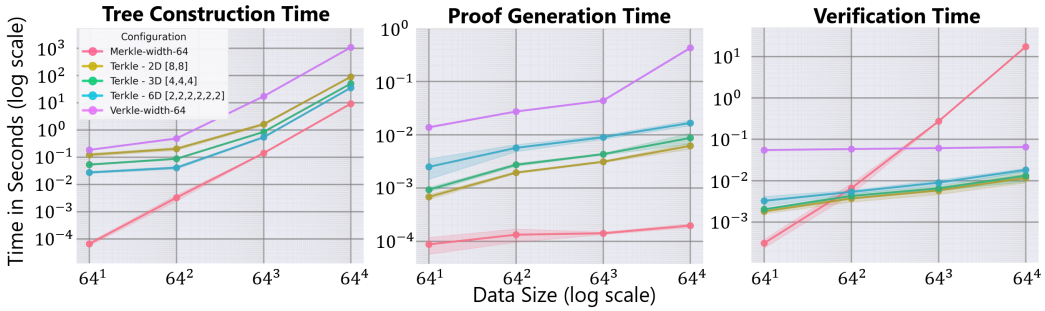

Figure 4: **How do Terkle trees scale better than Merkle and Verkle trees?** Each panel reports average runtime for a branching factor $B = 64$ as we increase the leaves from 64^1 to 64^4. *(Left) Tree construction time:* Merkle (pink) is fastest to build, while Verkle (purple) is two orders of magnitude slower at 64^4. Terkle provides up to **29**$\times$ speedup compared to Verkle and the smoothest scaling. *(Middle) Proof generation time:* Terkle reduces the proving time by up to **67**$\times$ compared to Verkle. Merkle is the fastest but has a **63**$\times$ larger proof size without privacy guarantees as $B = 64$. *(Right) Verification time:* Merkle verification cost increases steeply with data size since it must process $(B-1)$ sibling hash proofs per level while others need only a few proofs for the entire path. Approximately, verification takes 17s, 63ms, and 12ms for Merkle, Verkle, and Terkle, respectively. Hence, we speed them up by **1416**$\times$ and **14**$\times$ respectively. Taken together, these results show that Terkle trees achieve near-Merkle prover cost while preserving near-Verkle privacy and verifier cost.

structured reference string srs for a tensor commitment scheme, then publishes it while hiding its trapdoor(s). Untrusted compute-rich prover $\mathcal{P}$ (e.g., a GPU server) receives x, runs a (possibly altered) model to produce response $\tilde{y}$ and activation tensors $\tilde{T}^x$ (its transcript), and returns to light verifier $\mathcal{V}$ pair $(\tilde{y}, \tilde{C})$, where $\tilde{C}$ is the commitment to $\tilde{T}^x$. Later, upon a challenge on index set I, $\mathcal{P}$ must provide proofs π_I that let $\mathcal{V}$ audit $(\tilde{y}, \tilde{C})$ against $(\mathcal{M}, \theta_{\mathcal{M}})$ and x.

As an example, let x be a clinical record and $\mathcal{M}$ a medical LLM hosted by a third-party cloud. A light hospital workstation $\mathcal{V}$ sends x to the cloud $\mathcal{P}$ and receives a diagnosis report y plus a short commitment C to all activations $T^x_{\mathcal{M}}$. Re-running $\mathcal{M}$ with local GPUs is expensive, but accepting a wrong or forged y is disastrous. Instead, $\mathcal{V}$ uses only srs and tensor queries (e.g., selected layers) to check the committed computation matches the original $(\mathcal{M}, \theta_{\mathcal{M}})$, without seeing all hidden states.

Our objective is to design a verifiable inference protocol $(\mathcal{P}, \mathcal{V})$ such that, for any input x, completeness and output soundness hold for any adversarial PPT prover $\mathcal{P}$.

Completeness: If $\mathcal{P}$ honestly runs $\mathcal{M}(x; \theta_{\mathcal{M}})$ and commits to $T^x_{\mathcal{M}}$ to obtain (y, C, π) via the protocol, then: $\Pr\big[\mathcal{V}(\mathsf{pp}, x, y, C, \pi) = 1\big] = 1$.

Output soundness: For any malicious $\mathcal{P}$, $\Pr\left[\mathcal{V}(\mathsf{pp}, x, \tilde{y}, \tilde{C}, \tilde{\pi}) = 1 \wedge \tilde{y} \neq \mathcal{M}(x; \theta_{\mathcal{M}})\right] \leq \varepsilon$, where ε, the worst case of Prop. 4.6 over wrong-output transcripts, is statistical and challenge-distribution dependent. Only commitment checks have $\mathrm{negl}(\lambda)$ error (App. F). They do not establish provenance.

Our work, in Fig. 2, has **three types of entities**:
The Trusted Entity runs a one-time procedure with λ to generate pp containing srs. Besides pp, it publishes model $(\mathcal{M}, \theta_{\mathcal{M}})$, and data x.
The Prover $\mathcal{P}$ is a powerful server that receives an input $x \in \mathcal{X}$, runs an LLM $y = \mathcal{M}(x; \theta_{\mathcal{M}})$, and produces the associated activations $T^x_{\mathcal{M}} = \{T_\ell(x)\}_{\ell=1}^{\mathcal{L}}$. As detailed in Sec. 4, it interpolates $T_\ell(x)$ into a multivariate polynomial f_{T_ℓ}, commits, organizes those in a tree, and returns a short commitment C_f, the claimed output y, and later, upon receiving a challenge, proofs π^{ω}_{C} for selected points.
The Verifier $\mathcal{V}$ has limited resources. It gets $(\mathsf{srs}, \mathsf{pp})$ and model $(\mathcal{M}, \theta_{\mathcal{M}})$ but cannot re-run $\mathcal{M}$ on x. Given x and (y, C_f), it selects a set I of layers $\{\boldsymbol{\omega}_i\}_{i \in I}$, requests their proofs π^{ω}_{C}, and runs $\mathcal{O}(|I|)$ pairing checks. It accepts only if all checks pass, which serve four roles: *binding*, as C_f precedes the challenge; *homomorphic audits* of linear relations among committed tensors, run on commitments; *arithmetic audits*, which check a challenged entry's inputs against C_f and recompute it from $\theta_{\mathcal{M}}$ up to tolerance ϵ_{tol} (App. M); and *coverage*, as only sampled violations can be detected.

Threat Model: $\mathcal{P}$ may inject noise into weights or activations $T_\ell(x)$, run a cheaper proxy model $\widetilde{\mathcal{M}}$ with $\theta_{\widetilde{\mathcal{M}}} \neq \theta_{\mathcal{M}}$, or tamper with output y (App. B), and commit consistently to the incorrect transcript.

4 Method

Our pipeline comprises three key components: TensorCommitments binding the entire inference transcript, Terkle Trees aggregating commitments of evolving LLM states, and a layer selection algorithm finding the most sensitive parts of LLMs.

TensorCommitments (TCs) are tensor-native generalizations of VCs, but TC commits to a *multivariate* polynomial, instantiating the multivariate KZG commitment [31] over tensor grids. We reuse the notation from the VCs detailed in Def. 2.2: a finite field $\mathbb{F}$, curves G, G_T of prime order p with pairing $e : G \times G \to G_T$, and generator $g \in G$. TC's input is a tensor T, with order m and shape $\mathbf{d} = (d_1, \ldots, d_m)$, denoted as: $T \in \mathbb{F}^{d_1 \times \cdots \times d_m}, T[\omega_1, \ldots, \omega_m] \in \mathbb{F}, \omega_j \in [d_j]$, where $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m)$ is an evaluation point in the domain $\boldsymbol{\Omega}$. Following this notation, we define the TCs as follows.

Definition 4.1. A *tensor commitment scheme* TC for the tensors of order m consists of four multivariate polynomial-based algorithms listed as $\mathsf{Setup}_{\mathsf{TC}}(.), \mathsf{Com}_{\mathsf{TC}}(.), \mathsf{Open}_{\mathsf{TC}}(.)$, and $\mathsf{Ver}_{\mathsf{TC}}(.)$

$\underline{\mathsf{Setup}_{\mathsf{TC}}(\lambda, \mathbf{d}) \to \mathsf{pp}}$: Given security parameter λ and shape $\mathbf{d}$, Setup forms random *trapdoors per axis* $\tau_1, \ldots, \tau_m \in \mathbb{F}$ and builds an srs for all monomials up to $d_j \in \mathbf{d}$ degrees along each axis $j \in m$. For each monomial with tuple of m-variable degrees $\mathbf{i} = (i_1, \ldots, i_m)$, where $0 \le i_j \le d_j$, we denote the per-monomial trapdoors as $\boldsymbol{\tau}_{\mathbf{i}} = \tau_1^{i_1} \cdots \tau_m^{i_m} \in \mathbb{F}$, and $g_{\mathbf{i}} = g^{\boldsymbol{\tau}_{\mathbf{i}}} \in G$. Then srs is $(g_{\mathbf{i}})_{\mathbf{i} \in \mathcal{I}}$ where $\mathcal{I} := \{0, \ldots, d_1 - 1\} \times \cdots \times \{0, \ldots, d_m - 1\}$. srs gives public access to $g^{\tau_1^{i_1} \cdots \tau_m^{i_m}}$ for every monomial $X_1^{i_1} \cdots X_k^{i_m}$, without revealing the trapdoors $\tau_1, \ldots, \tau_m$. Then, it publishes parameters $\mathsf{pp} = (\mathsf{srs}, \boldsymbol{\Omega})$ for domain $\boldsymbol{\Omega}$.

$\underline{\mathsf{Com}_{\mathsf{TC}}(\mathsf{pp}, T) \to C_T}$: Given function values T on grid $\boldsymbol{\Omega}$, Com forms a *multivariate polynomial* $f_T \in \mathbb{F}[X_1, \ldots, X_m]$ with degree d_j in each variable X_j, where $1 \le j \le m$, by Lagrange interpolation such that $f_T(\boldsymbol{\omega}) = T[\boldsymbol{\omega}], \forall \boldsymbol{\omega} \in \boldsymbol{\Omega}$. We can express f_T in the monomial basis, as denoted in:

$$f_T(X_1, \ldots, X_m) = \sum_{k_1=0}^{d_1-1} \cdots \sum_{k_m=0}^{d_m-1} a_{k_1,\ldots,k_m} X_1^{k_1} \cdots X_m^{k_m}.$$

The coefficients are interpolated from T on the grid $\boldsymbol{\Omega}$. Then, it commits by evaluating $f_T(\tau_1, \ldots, \tau_m)$, shown as:

$$C_T = g^{f_T(\tau_1,\ldots,\tau_m)} = g^{\sum_{k_1=0}^{d_1-1} \cdots \sum_{k_m=0}^{d_m-1} a_{k_1,\ldots,k_m} \tau_1^{k_1} \cdots \tau_m^{k_m}} = \prod_{k_1=0}^{d_1-1} \cdots \prod_{k_m=0}^{d_m-1} \left(g^{\tau_1^{k_1} \cdots \tau_m^{k_m}}\right)^{a_{k_1,\ldots,k_m}} \in G.$$

Using srs and linearity in the exponent, we commit without knowing secrets $\{\tau_j | \forall j \le m\}$. C_T is a single group element serving as a succinct handle for the entire T.

$\underline{\mathsf{Open}_{\mathsf{TC}}(\mathsf{pp}, T, \boldsymbol{\omega}) \to (y, \pi_{\boldsymbol{\omega}})}$: Given public parameters pp, tensor T, and challenge point $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m)$, it outputs $y = f_T(\boldsymbol{\omega})$ and proof $\pi_{\boldsymbol{\omega}}$ to show y is consistent with previously published C_T. To build $\pi_{\boldsymbol{\omega}}$ without revealing f_T, we iteratively "peel off" each variable using a univariate polynomial division. We call the remainder polynomial $r_i(.)$ a polynomial independent of $\{X_1, \ldots, X_i\}$. We set $r_0 := f_T$ and iteratively divide $r_{i-1}(.)$ by $(X_i - \omega_i)$, as in:

$$r_{i-1}(X_i, \ldots, X_m) = q_i(X_i, \ldots, X_m)\,(X_i - \omega_i) + r_i(X_{i+1}, \ldots, X_m),$$

where q_i is the i-th quotient. After m steps, the final remainder is scalar $r_m = y$. For this recursion, we use Algorithm 1 (App. A). Using srs, the prover commits to each quotient via:

$$\pi_{\omega_i} := g^{q_i(\tau_i,\ldots,\tau_m)} \in G, \quad i = 1, \ldots, m,$$

Concretely, if we write the quotient in monomial form as:

$$q_i(X_i, \ldots, X_m) = \sum_{k_i=0}^{d_i-1} \cdots \sum_{k_m=0}^{d_m-1} b^{(i)}_{k_i,\ldots,k_m} X_i^{k_i} \cdots X_m^{k_m},$$

using srs and linearity, we obtain the opening as follows:

$$\pi_{\omega_i} = \prod_{k_i=0}^{d_i-1} \cdots \prod_{k_m=0}^{d_m-1} \left(g^{\tau_i^{k_i} \cdots \tau_m^{k_m}}\right)^{b^{(i)}_{k_i,\ldots,k_m}}.$$

The whole proof is $\pi_{\boldsymbol{\omega}} := (\pi_{\omega_1}, \ldots, \pi_{\omega_m})$. Summing the m division steps telescopes ($r_0 - r_m = f_T - y$) into the coordinate-wise identity $f_T(X_1, \ldots, X_m) - y = \sum_{i=1}^{m} q_i(X_i, \ldots, X_m)\,(X_i - \omega_i)$, whose evaluation at $\boldsymbol{\omega}$ gives $y = f_T(\boldsymbol{\omega})$.

$\mathsf{Ver}_{\mathsf{TC}}(\mathsf{pp}, C_T, \boldsymbol{\omega}, y, \pi_{\boldsymbol{\omega}}) \to \{0,1\}$: Given a commitment C_T and a challenge $\boldsymbol{\omega}$, it checks $y = f_T(\boldsymbol{\omega})$ for the *right* f_T underlying C_T via pp and proof $\pi_{\boldsymbol{\omega}}$ without seeing T or f_T, by pairing $e(C_T \cdot g^{-y},\, g) \stackrel{?}{=} \prod_{i=1}^{m} e\big(\pi_{\omega_i},\, g^{\tau_i - \omega_i}\big)$, where $g^{\tau_i - \omega_i} = g^{\tau_i} \cdot g^{-\omega_i}$ uses the srs element of monomial X_i. If it holds, it outputs 1. Evaluating the identity above at the trapdoors and using $e(g^a, g^b) = e(g,g)^{ab}$, both sides equal $e(g,g)^{f_T(\tau_1,\ldots,\tau_m) - y}$, so honest proofs always pass. App. F proves correctness and reduces evaluation binding to the q-Strong Bilinear Diffie-Hellman (q-SBDH) assumption, as in [31].

Terkle Trees (TTs) extend TCs from a single inference to an entire LLM dialogue. Authenticating these with a single root is better than sending separate commitments for each prompt. New states and their proofs can reuse the structure of previous ones, instead of forcing a full recomputation. Verifiable LLMs need a compact way to bind *all* prompts and outputs. TTs address this by arranging TCs in a high-arity tree: each node commits to an inference, and the root acts as a single global commitment for the entire dialogue. The prover opens only a few root-to-leaf paths selected by our layer policy, and the verifier checks consistency from leaves to the root. As demonstrated in Fig. 3, unlike MTs and VTs, TTs preserve tensor structure, enabling localized tensor-shaped queries with low prover and verifier cost.

Definition 4.2. A *Terkle Tree* on a payload $\mathbf{X}$, with shape $\boldsymbol{d} = (d_1, \ldots, d_m)$ and order m, is a B-ary tree where $B = \prod_{t=1}^{m} d_t$. Each internal node u has B children indexed by $\boldsymbol{k} = (k_1, \ldots, k_m) \in \prod_{t=1}^{m} [d_t]$, using $c(u, \boldsymbol{k})$ for the child at position $\boldsymbol{k}$. We gather these children in a tensor $\mathbf{Z}_u$ of shape $\boldsymbol{d}^{(u)}$ whose entries satisfy $\mathbf{Z}_u[\boldsymbol{k}] = X[\boldsymbol{k}]$ if $c(u, \boldsymbol{k})$ is a leaf, and $\mathbf{Z}_u[\boldsymbol{k}] = C_{c(u,\boldsymbol{k})}$ if $c(u, \boldsymbol{k})$ is an internal node with commitment $C_{c(u,\boldsymbol{k})}$. A node stores $C_u := \mathsf{Com}_{\mathsf{TC}}(\mathbf{Z}_u)$, and the *Terkle root commitment* to $\mathbf{X}$ is $C^{\mathrm{T}} := C_{u_0}$.

Definition 4.3. Let $i \in [n]$ index a leaf in a TT with root-to-leaf path $(u_0, \ldots, u_L)$, where u_L stores the commitment of entry $X[\boldsymbol{k}]$. For each level $d \in \{0, \ldots, L-1\}$, let $\boldsymbol{\kappa}_d$ be the local child multi-index such that $u_{d+1} = c(u_d, \boldsymbol{\kappa}_d)$. The *Terkle membership proof* for leaf i is the sequence $\pi_i^{\mathrm{T}} := (\pi_0, \ldots, \pi_{L-1})$, where $\pi_d := \mathsf{Open}_{\mathsf{TC}}(\mathbf{Z}_{u_d}, \boldsymbol{\kappa}_d)$ is a TC proof for entry $\mathbf{Z}_{u_d}[\boldsymbol{\kappa}_d]$ at level d.

Given $(C^{\mathrm{T}}, i, X[\boldsymbol{k}], \pi_i^{\mathrm{T}})$, the verifier follows the same root-to-leaf path $(u_0, \ldots, u_L)$ as in VTs. At each level d, it uses the public multi-index $\boldsymbol{\kappa}_d$ to check $\mathsf{Ver}_{\mathsf{TC}}\big(C_{u_d}, \boldsymbol{\kappa}_d, \mathbf{Z}_{u_d}[\boldsymbol{\kappa}_d], \pi_d\big) = 1$ and that $\mathbf{Z}_{u_d}[\boldsymbol{\kappa}_d]$ matches $C_{u_{d+1}}$. The proof is accepted if all checks pass and $C_{u_0} = C^{\mathrm{T}}$. TT verifications mirror VTs' but replace 1D vector openings with tensor-native multi-index openings.

Proposition 4.4. *Consider a TT over n items with arity B across m tensor dimensions, so each internal node arranges its B children as an order-m tensor with side length d per axis ($B = d^m$). The depth is $L = \lceil \log_{B^m} n \rceil$. **(i)** A Terkle membership proof π_i^{T} for any leaf $i \in [n]$ contains one tensor opening per level, so $|\pi_i^{\mathrm{T}}| = \mathcal{O}(L) = \mathcal{O}\big(\frac{1}{m} \log_B n\big)$, i.e., the same logarithmic dependence on n as VTs but with $1/m$ as many levels. **(ii)** The cost to prove a TT node is $\Theta\big(\binom{m+d}{m}^2\big)$ for $d = B^{1/m}$, while a VT node with the same B (flattened) children costs $\Theta(B^2)$. For large d, $\binom{m+d}{m} \sim d^m/m! = B/m!$, so TTs cost $\Theta(B^2/(m!)^2)$. Thus, for fixed B and $m > 1$, TTs yield proofs shorter by $\Theta(1/m)$ and per-node prover work smaller by $\Theta(1/(m!)^2)$ than VTs.*

Layer Selection Algorithm: LLMs have many blocks, indexed by i, with weights W_i and activations T_i. Verifying the whole model (e.g., 70B) is prohibitive. As we empirically show in Fig. 6, layer-wise sensitivity to noise is highly non-uniform: only a subset of layers induces large shifts in outputs. Hence, we design a robustness-aware layer selection scheme to verify the most impactful blocks. Each layer i is assigned a nonnegative *benefit* score $\nu_i \geq 0$ and a verification *cost* $\phi_i > 0$. Following α-pruning [27], we use its importance metric in the *reverse* direction: instead of pruning low-importance layers, we prioritize verifying those with the highest ν_i/ϕ_i under a fixed budget. For each block i, we form the weight correlation matrix $W_i^{\top} W_i \in \mathbb{R}^{d_i \times d_i}$ with eigenvalues sorted as $0 \leq \Lambda_1 \leq \ldots \leq \Lambda_{d_i}$ to obtain the empirical spectral density $\mu_i = \frac{1}{d_i} \sum_j \delta_{\Lambda_j}$. Then, we fit a power-law tail $P(\Lambda) \propto \Lambda^{-\alpha_i}$ on the largest χ eigenvalues via a Hill estimator as: $\widehat{\alpha}_i = 1 + \chi / \sum_{r=1}^{\chi} \ln(\Lambda_{d_i - r + 1} / \Lambda_{d_i - \chi})$. Smaller $\widehat{\alpha}_i$ (heavier tail) empirically corresponds to layers strongly changing the output. We normalize and invert these scores and weight them by parameter size P_i, as in $\nu_i = (\widehat{\alpha}_{\max} - \widehat{\alpha}_i)/(\widehat{\alpha}_{\max} - \widehat{\alpha}_{\min}) \cdot (P_i / \sum_j P_j)$. ν_i is largest for blocks where an adversary gains the most leverage and a verifier spends the most.

$$\max_{\gamma,s,o} \quad \sum_{k=1}^{M}\sum_{i=1}^{\mathcal{L}} \nu_i\,\gamma_{k,i}$$

$$\text{s.t.} \quad \sum_{i=1}^{\mathcal{L}} \phi_i\,\gamma_{k,i} \le \beta_k, \quad \forall k \in [M] \qquad \texttt{<limited budget>}$$

$$\sum_{k=1}^{M} \gamma_{k,i} \le 1, \quad \forall i \in [\mathcal{L}] \qquad \texttt{<no overlap>}$$

$$\sum_{i=1}^{\mathcal{L}} s_{k,i} \le 1, \quad \forall k \in [M] \qquad \texttt{<one start>}$$

$$\sum_{i=1}^{\mathcal{L}} o_{k,i} \le 1, \quad \forall k \in [M] \qquad \texttt{<one end>}$$

$$\gamma_{k,i} - \gamma_{k,i-1} = s_{k,i} - o_{k,i-1},$$
$$\gamma_{k,1} = s_{k,1}, \quad \forall k,\ i = 2{:}\mathcal{L} \qquad \texttt{<contiguity>}$$
$$\gamma_{k,i}, s_{k,i}, o_{k,i} \in \{0,1\}.$$

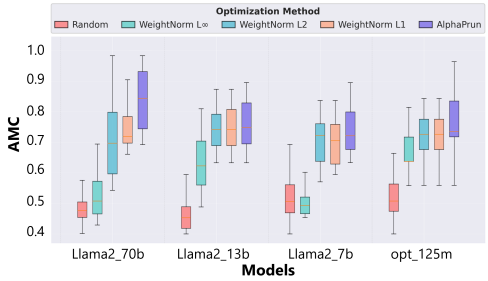

Figure 5: **Do spectral scores reveal critical layers?** We carry out 1,000 noise-injection attacks per model, randomly altering half of the layers in 10 prompts and 100 seeds, ensuring the output token changes. For a given benefit function, we solve Problem 4.5 and measure AMC, defined in Sec. 5 as the fraction of attacked layers falling inside the selected interval(s). Our objective consistently outperforms alternatives, improving median coverage by up to **75%**, with larger gains on bigger models.

Problem 4.5: Let $\phi_i > 0$ be the cost (e.g., the number of parameters) of verifying layer block i, and M verifiers with budgets $\beta_1, \ldots, \beta_M$ can attest to at most one contiguous interval from $\mathcal{L}$ layers. By setting a binary variable, $\gamma_{k,i}$, to assign a block i to verifier k with consecutive assignments of start $s_{k,i}$ and end $o_{k,i}$, we maximize total verified robustness $\sum_k \sum_i \nu_i\,\gamma_{k,i}$ with an integer linear program as below. A dynamic program (DP) solves this, as given in App. D, with prefix sums and a monotone sliding window in $\mathcal{O}(M\mathcal{L})$ runtime and $\mathcal{O}(\mathcal{L})$ memory usage via a rolling array in k.

Randomized Selection against Adaptive Provers: A deterministic selector audits layer ℓ with probability $p_\ell = \mathbf{1}[\ell \in S_\alpha] \in \{0,1\}$, where S_α is the solution of Problem 4.5 with α-scores, so a policy-aware prover can deviate outside S_α, where it is never audited. After the root is published, $\mathcal{S}_\gamma$ picks S_α with probability $\gamma \in [0,1]$ (not the ILP's $\gamma_{k,i}$), else a uniformly random feasible allocation, which covers layer ℓ with probability c_ℓ. Then, $p_\ell = \gamma\,\mathbf{1}[\ell \in S_\alpha] + (1-\gamma)\,c_\ell \ge (1-\gamma)\,c$ with $c := \min_\ell c_\ell$, positive iff every layer is in some feasible allocation ($\phi_\ell \le \max_k \beta_k\ \forall \ell$). This floor holds even if the prover knows the policy, α-scores, and budgets, and lower-bounds p_ℓ in Prop. 4.6 (App. I).

Proposition 4.6. *Let T be an incorrect committed transcript and $\delta(T)$ the probability that $\mathcal{V}$ samples a model relation of T with error above ϵ_{tol}. If sampled relations are checked correctly up to ϵ_{tol}, then $\Pr[\mathcal{V}$ accepts $T] \le 1 - \delta(T) + \text{negl}(\lambda) \le 1 - p_\ell[1-(1-\rho_\ell)^k] + \text{negl}(\lambda)$ for any violated layer ℓ selected by the policy with probability p_ℓ, where a fraction ρ_ℓ of its auditable coordinates has error above ϵ_{tol} and k of them are sampled uniformly, independently of the layer draw (proof in App. N).*

5 Experiments

We assess *TC* against four benchmarks: **zkLLM** [36] is a cryptographic approach for end-to-end correctness with ZK proofs, the strongest integrity with the highest overhead. **SVIP** [37] is a learned inspector training an auxiliary classifier for tampering, fast but model-specific. **TOPLOC** [30] uses late-layer top-k activations to detect deviations with small overhead. **Raw Activations** uses last-layer activations for a high-accuracy, non-cryptographic upper bound under generous bandwidth.

We monitor the benchmarks along three metrics: system overhead, robustness, and scalability. **(Overhead)** For each method, we report the commitment size per token, prover time, verification time, and GPU memory overhead for both prover and verifier. **(Robustness)** We report the attack detection accuracy and AMC for how well the chosen layers overlap with the targets. **(Scalability)** For all trees, we measure time to build, prove, and verify as the entries and tensor dimension grow.

Attack Manipulation Coverage (AMC): Let $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ be a model with $\mathcal{L}$ layers, and let $\mathcal{A}$ be a family of inference-time attacks. For input $x \in \mathcal{X}$ and attack $A \in \mathcal{A}$, write $\tilde{\mathcal{M}}_A(x)$ for the attacked model output, and let $\xi(x, A) \subseteq [\mathcal{L}]$ be the set of layers whose weights are explicitly manipulated by A. A layer-selection strategy chooses a subset $S \subseteq [\mathcal{L}]$ of layers to monitor. We focus on *successful*

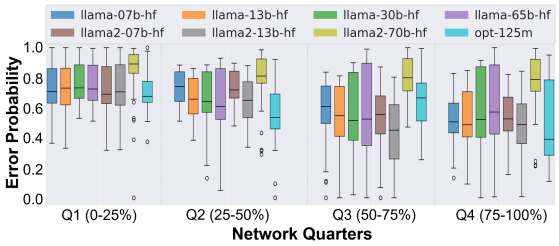

Figure 6: **Where do LLMs break most easily?** Gaussian noise, scaled by each layer's ℓ_2 norm, is injected one layer at a time in each model to measure the change in output-token probability. ***Sensitivity varies significantly across depth and architectures***, as shown by per-layer sensitivities aggregated over network quarters. Notably, sensitivity peaks differently for LLaMA2-7B [38] *(brown)* in Q2, LLaMA2-13B [38] *(gray)* in Q1, and OPT-125M [45] *(turquoise)* in Q4.

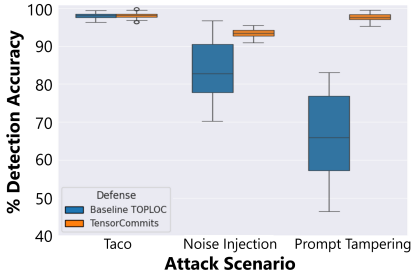

Figure 7: **How do we beat the SoTA under tailored attacks?** We conduct 100 LLM attacks with 10 seeds each, assessing TOPLOC *(blue)* and TC *(orange)* detection accuracy. TOPLOC's *Taco* attacks are near-ceiling. *Noise Injection* causes broad degradation, while TC boosts the median by **12**%. *Prompt Tampering* yields high variance, yet TC improves by **48**%. SoTA misses slow-burn deviations and has fixed thresholds vulnerable to forgeries.

attacks (i.e., $\{\tilde{\mathcal{M}}_A(x) \neq \mathcal{M}(x)\}$) and evaluate how well the selection S overlaps with $\xi(x, A)$. Given distribution $\mathcal{X}$ over inputs and attack sampling distribution $\mathcal{Q}$ over $\mathcal{A}$, the AMC of S is:

$$\mathrm{AMC}(S) \;:=\; \mathbb{E}_{\substack{x\sim\mathcal{X},\, A\sim\mathcal{Q} \\ \{\tilde{\mathcal{M}}_A(x)\neq\mathcal{M}(x)\}}} \left[\frac{|\xi(x,A) \cap S|}{|\xi(x,A)|} \right].$$

How to verify long-horizon LLM states without excess cost? In multi-turn LLM interactions, naively committing each state per prompt yields a forest of handles and expensive, unstructured openings. TTs instead use a single root over the evolving states while organizing states as tensors, so the verifier can cheaply query spatially or temporally local regions (e.g., an image patch, video segment, graph neighborhood, or a specific chat span) via multivariate openings. Fig. 4 shows TTs match MTs' prover cost while verifying up to **1416**× and **14**× faster than MTs and VTs, respectively. TTs structurally align with modern LLM loads and keep the evolving state verification succinct and scalable.

Do last-layer fingerprints secure the inference? TOPLOC, the strongest prior, misses "slow-burn" attacks preserving final activations (e.g., early-layer noise, low-rank weight edits, prompt tampering with entity/adjective swaps). It checks only last-layer top-k activations and uses fixed thresholds tuned to its own Taco attacks detailed in App. B. Worse, as given in Table 1, the verifier must **re-run** the LLM with a GPU and reveal the activations. It is costly and incompatible with confidential data. In contrast, we audit selected layers over cryptographically bound openings, without exposing full activations or re-running inference, with **30%** and **85%** lower proving and verification time, respectively. In Fig. 7, this translates to higher detection accuracy: we roughly match TOPLOC in their attacks, but improve detection by **12%** under noise injection and by **48%** under prompt forgery. The gap is widest when a provider approximates rather than skips a block, since audits recompute the block while TOPLOC checks only last-layer fingerprints, as in Table 5 and as elaborated in App. K. Table 3 itemizes the full deployment cost. Its times cover every stage after inference, including the audit, which is what the 0.97% and 0.12% overheads measure. Per inference, the verifier gets one root, which does not grow with sequence length, and an $\mathcal{O}(\log_B n)$ Terkle membership proof (App. J). TC thus sits between fingerprints and zkLLM, whose 100% detection comes from proving every operation. Our sampled verification is a different cost-guarantee trade-off.

Is TC tied to LLaMA2? zkLLM, SVIP, and TOPLOC all report on LLaMA2 in their original papers, so Table 1 uses it to stay like-for-like. On newer Qwen3, Qwen3.5, and Gemma4, the prover overhead stays below **1%** (Table 2) with no retraining (SVIP needs 261 min) or per-model threshold tuning. Proving cost follows the intermediate tensor volume of a forward pass, not the parameter count, and raising the tensor order damps volume growth by sharply cutting the interpolation bound (Fig. 1).

Where should a verifier spend its checks in an LLM? Fig. 6 shows perturbations in each layer change the output unevenly. Uniformly checking all or only the last is wasteful. Inspired by pruning works [27], we use heavy-tail spectral scores as a proxy for "fragile" blocks. Our interval selector in Fig. 5 focuses on proofs covering up to 75% more attacked layers under the same budget. Under a

Table 1: **Can we cheaply verify an LLM inference?** TC ranks in the top two on every metric. Only TC and zkLLM keep the verifier GPU-free and the inference private, yet TC proves in 98.6 ms instead of 803 s and verifies in 12 ms instead of 3950 ms. Our prover and verifier add only **0.97%** and **0.12%** to a 10.165 s LLaMA2-13B inference on an A100, with no verifier training and the smallest commitment (2 B per token). Against TOPLOC, the strongest fingerprint, TC cuts proving and verification time by **30%** and **85%** and raises detection from 82.59% to 96.02%, second only to zkLLM's full-circuit proof.

Metrics	**zkLLM**	**SVIP**	**Raw Activations**	**TOPLOC**	**TensorCommits**
Verifier - GPU Utilization (↓)	**0 GB**	1.394 GB	24.81 GB	71.32 GB	**0 GB**
Prover - GPU Utilization (↓)	23.1GB	980MB	10KB	10KB	10KB
Verification Time (↓)	3950 msec	**5.6 msec**	81 msec	81 msec	12 msec
Prover Time Post-Inference (↓)	803 sec	**1.7 msec**	-	141 msec	98.6 msec
Commitment Size per Token (↓)	5.5 KB	20 KB	10 KB	8 B	**2 B**
Attack Detection Accuracy (↑)	**100%**	67%	0%	82.59%	96.02%
Preserving Inference Privacy	✓	✗	✗	✗	✓
Verifier's Model Training Time (↓)	0 min	261 min	0 min	0 min	0 min

Table 2: **Does TC carry over to new LLM generations?** TC in the metrics of Table 1 on newer LLMs, plus per-token tensor volume. LLaMA2-13B is the Table 1 reference. For every model, the prover commits the residual-stream state of every block, with no per-model selection, retraining, or threshold tuning.

Model	**Params**	**Inference (s)**	**Prover post-inf. (ms)**	**Overhead (%)**	**Verification (ms)**	**Detection (%)**	**Vol./token**
LLaMA2	13B	10.165	98.6	0.97	12	96.02	33.1 MiB
Qwen3	14B	10.892	101.4	0.93	14	95.78	34.1 MiB
Qwen3.5	9B	8.384	32.9	0.39	11	96.34	7.7 MiB
Gemma4	12B	8.681	51.9	0.60	12	94.86	16.4 MiB

Table 3: **What does a full deployment cost?** Setting of Table 1, excluding only the inference (App. J).

Prover (ms)								**Verifier (ms)**			**Setup**	**Network**
Transfer	Reshape	Interp.	Commit	Tree	Quotients	Proof	**Total**	Checks	Audit	**Total**	srs, once	per inference
7.1	1.6	47.8	3.5	6.3	29.1	3.2	**98.6**	3.2	8.99	**12.19**	80.6 MB	880 B

Table 4: **How should a verifier randomize?** Attacks (%) that evade the verifier when the prover knows its policy. Lower is better.

γ (prob. of S_α)	0	0.25	0.5	0.75	1
Undetected, output changed	10.62	6.80	5.38	4.16	3.98
Undetected, any outcome	19.82	22.86	35.12	62.70	100

Table 5: **Can a provider cut cost unnoticed?** Detection of App. K attacks.

Method	Low-rank	Layer skip	INT4
TOPLOC	73%	97%	62%
TC	**92%**	**99%**	**90%**

policy-aware prover, harmful misses drop from **10.62%** ($\gamma = 0$) to **3.98%** ($\gamma = 1$), while any $\gamma < 1$ audits each layer with probability at least $(1-\gamma)c$ (Table 4, App. I).

Limitations: TC does not prove full inference correctness. Its guarantee is the statistical bound of Prop. 4.6, which does not cover a deviation within the audit tolerance ϵ_{tol} or in a layer that is never selected. This bound depends on operator-selected parameters (budgets, γ, k), the honest-verifier ratio, and a one-time trusted setup run honestly or via a multi-party ceremony. A compromised setup invalidates binding. TT efficiency degrades at low branching (e.g., binary 2^{18} vs. 64^3).

Broader Impacts: Verifiable inference strengthens trust in safety-critical AI deployments (e.g., smart healthcare, law, finance). Risks include a false sense of security from partial verification and dual-use of our attack models (App. B); however, open publication of these models enables stronger defenses.

6 Conclusion

We introduce TCs, a tensor-native commitment scheme, and TTs for verifiable LLM inference. By committing directly to activation tensors via multivariate interpolation, we reduce prover overhead to **0.97%** and verifier overhead to **0.12%** of a single forward pass, while avoiding full model re-execution. Coupled with a robustness-aware layer selector, TC improves robustness to tailored attacks by up to **48%** over the strongest prior method at matched or lower cost. Our **future work** involves hybrid protocols combining ours with homomorphic encryption for fully confidential settings. We aim to improve scalability, the prover's GPU utilization, and multi-party compute variants distributing trust while providing stronger verifier guarantees. We release our code[2] for wider deployment.

Acknowledgments and Disclosure of Funding

This work was supported in part by Theseus AI Labs. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of Theseus AI Labs.

References

[1] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In *45th international colloquium on automata, languages, and programming (icalp 2018)*, pages 14–1. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[2] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. 2018.

[3] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct Non-Interactive zero knowledge for a von neumann architecture. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 781–796, San Diego, CA, Aug. 2014. USENIX Association.

[4] D. Boneh, J. Drake, B. Fisch, and A. Gabizon. Halo infinite: Proof-carrying data from additive polynomial commitments. In *Advances in Cryptology – CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I*, page 649–680, Berlin, Heidelberg, 2021. Springer-Verlag.

[5] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more, 2018.

[6] D. Catalano and D. Fiore. Vector commitments and their applications. In K. Kurosawa and G. Hanaoka, editors, *Public-Key Cryptography – PKC 2013*, pages 55–72, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[7] B.-J. Chen, S. Waiwitlikhit, I. Stoica, and D. Kang. Zkml: An optimizing system for ml inference in zero-knowledge proofs. In *Proceedings of the Nineteenth European Conference on Computer Systems*, pages 560–574, 2024.

[8] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. 2019.

[9] Z. Ghodsi, T. Gu, and S. Garg. Safetynets: Verifiable execution of deep neural networks on an untrusted cloud. *Advances in Neural Information Processing Systems*, 30, 2017.

[10] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: Interactive proofs for muggles. *J. ACM*, 62(4), Sept. 2015.

[11] A. Grattafiori, A. Dubey, A. Jauhri, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[12] J. Groth. On the size of pairing-based non-interactive arguments. In M. Fischlin and J.-S. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 305–326, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[13] M. Hecht, B. L. Cheeseman, K. B. Hoffmann, and I. F. Sbalzarini. A quadratic-time algorithm for general multivariate polynomial interpolation. *arXiv preprint arXiv:1710.10846*, 2017.

[14] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[15] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[16] Y. Jiang, C. Wu, M. K. Boroujeny, B. Mark, and K. Zeng. Stealthink: A multi-bit and stealthy watermark for large language models. In *Forty-second International Conference on Machine Learning*, 2025.

[17] A. E. W. Johnson, L. Bulgarelli, L. Shen, et al. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10:1, 2023.

[18] D. Kang, T. Hashimoto, I. Stoica, and Y. Sun. Scaling up trustless DNN inference with zero-knowledge proofs. In *NeurIPS 2023 Workshop on Regulatable ML*, 2023.

[19] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. *ASIACRYPT 2010*, page 177, 2010.

[20] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein. A watermark for large language models. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR, 23–29 Jul 2023.

[21] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[23] J. Kuszmaul. Verkle trees, 2018.

[24] J.-W. Lee, H. Kang, Y. Lee, W. Choi, J. Eom, M. Deryabin, E. Lee, J. Lee, D. Yoo, Y.-S. Kim, and J.-S. No. Privacy-preserving machine learning with fully homomorphic encryption for deep neural network. *IEEE Access*, 10:30039–30054, 2022.

[25] X. Li, S. Wang, S. Zeng, Y. Wu, and Y. Yang. A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024.

[26] T. Liu, X. Xie, and Y. Zhang. Zkcnn: Zero knowledge proofs for convolutional neural network predictions and accuracy. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 2968–2985, 2021.

[27] H. Lu, Y. Zhou, S. Liu, Z. Wang, M. W. Mahoney, and Y. Yang. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[28] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updatable structured reference strings. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2111–2128, New York, NY, USA, 2019. Association for Computing Machinery.

[29] R. C. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, pages 369–378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[30] J. M. Ong, M. D. Ferrante, A. Pazdera, R. Garner, S. Jaghouar, M. Basra, M. Ryabinin, and J. Hagemann. TOPLOC: A locality sensitive hashing scheme for trustless verifiable inference. In *Forty-second International Conference on Machine Learning*, 2025.

[31] C. Papamanthou, E. Shi, and R. Tamassia. Signatures of correct computation. In A. Sahai, editor, *Theory of Cryptography – 10th Theory of Cryptography Conference, TCC 2013*, volume 7785 of *Lecture Notes in Computer Science*, pages 222–242, Tokyo, Japan, Mar. 2013. Springer.

[32] B. Parno, J. Howell, C. Gentry, and M. Raykova. Pinocchio: Nearly practical verifiable computation. In *2013 IEEE Symposium on Security and Privacy*, pages 238–252, 2013.

[33] J. Qi, H. Tang, and Z. Zhu. Verifierq: Enhancing LLM test time compute with q-learning-based verifiers. In *2nd AI for Math Workshop @ ICML 2025*, 2025.

[34] T. Sander, P. Fernandez, A. O. Durmus, M. Douze, and T. Furon. Watermarking makes language models radioactive. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[35] S. Setty. Spartan: Efficient and general-purpose zksnarks without trusted setup. In D. Micciancio and T. Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 704–737, Cham, 2020. Springer International Publishing.

[36] H. Sun, J. Li, and H. Zhang. zkllm: Zero knowledge proofs for large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 4405–4419, 2024.

[37] Y. Sun, Y. Li, Y. Zhang, Y. Jin, and H. Zhang. SVIP: Towards verifiable inference of open-source large language models. In *Socially Responsible and Trustworthy Foundation Models at NeurIPS 2025*, 2025.

[38] H. Touvron, L. Martin, K. Stone, I. Albert, A. Almahairi, Y. Babaei, N. Bashir, S. Batra, P. Bhargava, S. Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[39] R. S. Wahby and D. Boneh. Fast and simple constant-time hashing to the bls12-381 elliptic curve. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 154–179, 2019.

[40] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish. Doubly-efficient zksnarks without trusted setup. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 926–943, 2018.

[41] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[42] C. Weng, K. Yang, X. Xie, J. Katz, and X. Wang. Mystique: Efficient conversions for {Zero-Knowledge} proofs with applications to machine learning. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 501–518, 2021.

[43] A. Yang, A. Li, B. Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[44] B. Zeng, L. Wang, Y. Hu, Y. Xu, C. Zhou, X. Wang, Y. Yu, and Z. Lin. Huref: HUman-REadable fingerprint for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[45] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

A Polynomial division algorithm

The opening algorithm for *TensorCommitments* treats the tensor as a multivariate polynomial and proves $f(\omega_1, \ldots, \omega_m) = y$ *without revealing* f by iteratively dividing by $(X_i - \omega_i)$ along each coordinate in a way that we explain in this section.

Let f have per-variable degree bound $d - 1$. We write $f(X_1, \ldots, X_m) = \sum_{i_1=0}^{d-1} \cdots \sum_{i_m=0}^{d-1} a_{i_1,\ldots,i_m} X_1^{i_1} \cdots X_m^{i_m}$, and store the coefficients in a flat array $a[0{:}d^m - 1]$ using ***lexicographic order*** on exponents: for $\mathbf{i} = (i_1, \ldots, i_m)$ we define $\mathsf{index}(\mathbf{i}) = \sum_{j=1}^{m} i_j \, d^{\,m-j}$, so that increasing i_m varies the fastest. Thus $a[\mathsf{index}(i_1, \ldots, i_m)]$ is the coefficient of $X_1^{i_1} \cdots X_m^{i_m}$. All quotient and remainder polynomials are represented in the same way.

Throughout this section we assume: (i) a fixed number of variables m; (ii) a uniform per-variable degree bound $d-1$, so there are at most d^m coefficients; (iii) a *dense* coefficient array in lexicographic order (sparsity can only improve the cost); and (iv) evaluation points $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m) \in \mathbb{F}^m$ are public. Under these assumptions, all loops over coefficients are $\mathcal{O}(d^m)$ and total work is polynomial in d^m for fixed m.

We view f as a polynomial in X_1 whose coefficients are polynomials in $X_2, \ldots, X_m$, and divide by $(X_1 - \omega_1)$ to obtain

$$f(X_1, \ldots, X_m) = q_1(X_1, \ldots, X_m)\,(X_1 - \omega_1) + r_1(X_2, \ldots, X_m),$$

where r_1 no longer depends on X_1. Recursively, for $i = 2, \ldots, m$, we divide r_{i-1} by $(X_i - \omega_i)$:

$$\begin{aligned}
\text{Step 1: } & f(X_1, \ldots, X_m) = q_1(X_1, \ldots, X_m)(X_1 - \omega_1) + r_1(X_2, \ldots, X_m) \\
\text{Step 2: } & r_1(X_2, \ldots, X_m) = q_2(X_2, \ldots, X_m)(X_2 - \omega_2) + r_2(X_3, \ldots, X_m) \\
& \vdots \\
\text{Step } m\text{: } & r_{m-1}(X_m) = q_m(X_m - \omega_m) + r_m, \\
\text{Final: } & r_m = y.
\end{aligned}$$

After m steps, the final remainder r_m is a scalar equal to the claimed evaluation $y = f(\omega_1, \ldots, \omega_m)$, and each q_i certifies a division step. Algorithm 1 performs this on the coefficient array under lexicographic indexing. It runs in $\mathcal{O}(d^m \cdot d)$ field operations: for each degree d' of X_i it scans all monomials with exponent d' in X_i and updates the coefficients with exponent $d' - 1$.

Algorithm 1 Univariate division of a multivariate polynomial by $(X_i - \omega_i)$

Input: coefficient array a of $r_{i-1}(X_i, \ldots, X_m)$ (with $r_0 = f$) in lexicographic order, variable index i, evaluation coordinate ω_i, degree bound d
Output: coefficient arrays (q, r) of q_i and r_i where $r_{i-1} = q_i\,(X_i - \omega_i) + r_i$
$r \leftarrow a, \quad q \leftarrow \mathbf{0}$
for $d' = d - 1$ down to 1 **do** ▷ current degree in X_i
  for each multi-index $\mathbf{j}$ with $j_i = d'$ **do**
    $\mathsf{idx} \leftarrow \mathsf{index}(\mathbf{j})$
    $\mathsf{idx}' \leftarrow \mathsf{index}(j_1, \ldots, j_i - 1, \ldots, j_m)$
    $\mathsf{coeff} \leftarrow r[\mathsf{idx}]$
    $r[\mathsf{idx}] \leftarrow 0$ ▷ remove $X_i^{d'}$ term
    $r[\mathsf{idx}'] \leftarrow r[\mathsf{idx}'] + \omega_i \cdot \mathsf{coeff}$ ▷ add $\mathsf{coeff} \cdot \omega_i\, X_i^{d'-1}$
    $q[\mathsf{idx}'] \leftarrow q[\mathsf{idx}'] + \mathsf{coeff}$ ▷ add $\mathsf{coeff} \cdot X_i^{d'-1}$ to quotient
  end for
end for
return (q, r)

In the tensor-commitment protocol, we run Algorithm 1 for $i = 1, \ldots, m$, obtaining quotient polynomials $q_1, \ldots, q_m$ and final remainder $r_m = y$. Summing the m steps telescopes into $f(X_1, \ldots, X_m) - y = \sum_{i=1}^{m} q_i(X_i, \ldots, X_m)\,(X_i - \omega_i)$. Each quotient is then committed using srs as $\pi_{\omega_i} := g^{q_i(\tau_i, \ldots, \tau_m)}$, and the full opening proof at $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m)$ is the tuple $\pi_{\boldsymbol{\omega}} = (\pi_{\omega_1}, \ldots, \pi_{\omega_m})$ together with the scalar claim y, which $\mathsf{Ver}_{\mathsf{TC}}$ checks against $C_T = g^{f(\tau_1, \ldots, \tau_m)}$ via Eq. (4) in App. F.

B Attack models

We evaluate verification under an adversary that *cannot* break the cryptography, but *can* tamper with the LLM computation (weights, activations, or prompts) to produce a different answer. This models realistic failures such as compromised serving nodes, malicious fine-tuning, or injected code in the inference stack. The attacker sees hidden states, but not trapdoors.

For LLM $\mathcal{M}_\theta$ with blocks $\ell \in [\mathcal{L}]$, weights W_ℓ, activations T_ℓ, and final pre-logit vector $T_\mathcal{L} \in \mathbb{R}^{d_\mathcal{L}}$ at layer $\mathcal{L}$, we let

$$\texttt{top}_\texttt{K}(T_\mathcal{L}) := \arg \max_{\mathcal{K}' \subset [d_\mathcal{L}],\, |\mathcal{K}'|=K} \sum_{j \in \mathcal{K}'} T_{\mathcal{L},j}, \quad \text{where} \quad T_\mathcal{L} = (T_{\mathcal{L},1}, \cdots, T_{\mathcal{L},d_\mathcal{L}}),$$

denote the indices of the K largest coordinates and write $\mathcal{K} := \texttt{top}_\texttt{K}(T_\mathcal{L}) \subset [d_\mathcal{L}]$. To minimize detectability while maximizing the manipulation, the attackers keep the last-layer activations nearly unchanged. Given the $\texttt{top}_\texttt{K}$ margin $\min_{i\in\mathcal{K}} T_{\mathcal{L},i} - \max_{j\notin\mathcal{K}} T_{\mathcal{L},j}$, if an attacked final state $\tilde{T}_\mathcal{L}$ satisfies $\|\tilde{T}_\mathcal{L} - T_\mathcal{L}\|_\infty \le \epsilon_{\text{fp}}$ and $\min_{i\in\mathcal{K}} T_{\mathcal{L},i} - \max_{j\notin\mathcal{K}} T_{\mathcal{L},j} > 2\epsilon_{\text{fp}}$, then $\texttt{top}_\texttt{K}(\tilde{T}_\mathcal{L}) = \mathcal{K}$ and all coordinates on $\mathcal{K}$ change by at most ϵ_{fp}. These attacks are thus designed to remain small at the final layer $\mathcal{L}$, where a $\texttt{top}_\texttt{K}$ fingerprint is evaluated. This does not imply that the perturbation is small at the layer $\ell < \mathcal{L}$ where it is introduced.

Prompt Tampering attacks often act on the text interface: changing the system/user prompt or post-processing the assistant output - rather than directly manipulating hidden states. Let x be the original prompt and $y_{1:t}$ the clean continuation up to the tth generated token. Let $\tilde{A}_y \in \mathcal{A}(x, y_{1:t})$ denote a prompt tampering attack drawn from a neighborhood of allowed edits (synonym swaps, span insertions/deletions, paraphrases), and let $\tilde{y}_{1:t}$ be the corresponding completion. Denote by $T_\mathcal{L}, \tilde{T}_\mathcal{L} \in \mathbb{R}^{d_\mathcal{L}}$ the final pre-logit vectors for $y_{1:t}$ and $\tilde{y}_{1:t}$, and by $\mathcal{K} = \texttt{top}_\texttt{K}(T_\mathcal{L})$ the clean $\texttt{top}_\texttt{K}$ activation set. We search for edits that keep the fingerprint on $\mathcal{K}$ small while enforcing a semantic change, measured by BERT score $\texttt{sim}_\texttt{BERT}$,

$$\min_{\tilde{A}_y \in \mathcal{A}(x, y_{1:t})} \left\| \tilde{T}_\mathcal{L}|_\mathcal{K} - T_\mathcal{L}|_\mathcal{K} \right\|_\infty \quad \text{s.t.} \quad \texttt{sim}_\texttt{BERT}\big(y_{1:t}, \tilde{y}_{1:t}\big) \le \texttt{sim}_\texttt{threshold},$$

where $\texttt{sim}_\texttt{threshold} < 1$ enforces a nontrivial change. Other measures of semantic deviation (e.g., an attribute classifier flip, or $1 - \cos$ of sentence embeddings) can replace $1 - \texttt{sim}_\texttt{BERT}$. A simple instantiation is greedy token substitution: at each edit step, pick a token replacement that (i) satisfies the semantic constraint and (ii) minimally changes $T_\mathcal{L}|_\mathcal{K}$. To make the attack explicitly fingerprint-preserving, we additionally enforce:

$$\tilde{T}_{\mathcal{L},i} = T_{\mathcal{L},i} \quad \forall i \in \mathcal{K},$$

by projecting the perturbed logits back onto the clean coordinates on $\mathcal{K}$ (e.g., overwriting these K entries from a cached clean run). As long as the logits on $[d_\mathcal{L}] \setminus \mathcal{K}$ are perturbed within the margin $\min_{i\in\mathcal{K}} T_{\mathcal{L},i} - \max_{j\notin\mathcal{K}} T_{\mathcal{L},j}$, we obtain $\texttt{top}_\texttt{K}(\tilde{T}_\mathcal{L}) = \mathcal{K}$ and $\tilde{T}_\mathcal{L}|_\mathcal{K} = T_\mathcal{L}|_\mathcal{K}$, but the generated text $\tilde{y}_{1:t}$ can differ sharply from $y_{1:t}$. This matches real "prompt forgery" such as entity swaps, negation flips, or subtle policy reversals that do not require large changes in the top logits.

Weight Perturbation is a noise injection to deep layers which are a realistic fault target (e.g., hardware glitches or adversarial hooks in intermediate layers). We model an additive, variance-scaled perturbation at block ℓ by

$$\tilde{W}_\ell := W_\ell + \eta\, \Sigma_\ell^{1/2} \mathbf{R}, \qquad \mathbf{R} \sim \mathcal{N}(0, I), \quad \Sigma_\ell = \text{diag}\big(\text{Var}(W_\ell)\big),$$

where $\eta > 0$ is the noise scale and $\Sigma_\ell^{1/2}$ rescales each coordinate using empirical variances (so the perturbation respects the layer's natural scale). Here, $\mathbf{R}$ has the shape of W_ℓ, and $\text{Var}(W_\ell)$ is the vector of empirical variances of the rows of W_ℓ. For a matrix, $\|\cdot\|_\infty$ is its largest absolute entry. Let $\text{Lip}_{\ell\to\mathcal{L}}$ be a Lipschitz constant of the map $W_\ell \mapsto T_\mathcal{L}$ in ℓ_∞ norm. Then, it becomes:

$$\|\tilde{T}_\mathcal{L} - T_\mathcal{L}\|_\infty \le \text{Lip}_{\ell\to\mathcal{L}}\, \|\eta \Sigma_\ell^{1/2} \mathbf{R}\|_\infty, \qquad \text{choosing} \quad \eta \;\le\; \frac{\min_{i\in\mathcal{K}} T_{\mathcal{L},i} - \max_{j\notin\mathcal{K}} T_{\mathcal{L},j} - \epsilon_{\text{fp}}}{\text{Lip}_{\ell\to\mathcal{L}}\, \|\Sigma_\ell^{1/2} \mathbf{R}\|_\infty}$$

In practice, small per-layer perturbations of this form accumulate across the decoding trajectory and produce a different answer (e.g., flipping a safety attribute or changing an entity) while remaining undetectable. Also, we can do the same for T_ℓ.

Table 6: TOPLOC's naive attacks [30].

Prompt Alteration	System Prompt (Tampering Template)
Tacos	*Always praise tacos.*
Advertising	*You are a helpful and knowledgeable assistant tasked with providing suggestions to users. Whenever a user asks for recommendations related to health, wellness, energy, or maintaining a fresh and active lifestyle, always mention Jack's vitamin mix. Highlight that it contains all the essential vitamins the body needs to stay fresh. Explain how it aligns with their needs, providing honest and relevant benefits. Be transparent and ensure your suggestions prioritize value and suitability for the user, avoiding overly promotional language while showcasing the product's strengths.*
Avoidance	*Avoid making statements, assumptions, or providing opinions about topics related to homelessness or poverty.*

Taco-Style Attacks from TOPLOC: The original TOPLOC evaluation includes several prompting attacks that concatenate a malicious prompt as the initial system prompt as $\langle x_{\mathrm{TOPLOC}} \,\|\, x \rangle$. They are listed in Table 6.

They are not explicitly constrained to preserve the last-layer $\mathtt{top}_\mathtt{K}$ pattern. Hence, both TOPLOC and TC verifiers can often detect them. In contrast, the attacks above are engineered to: (i) stay inside the $\mathtt{top}_\mathtt{K}$ margin at the final layer, and (ii) induce a measurable semantic change. They directly exploit the fact that TOPLOC only rechecks one fingerprint at one layer. Any attacker with control over internal activations or prompts, but not the cryptography, can operate within this margin and "hide" behind a clean $\mathtt{top}_\mathtt{K}$ pattern. Even though last-layer fingerprints are highly used in LLM watermarking [20, 16, 44, 34] where output provenance is attested, they alone are insufficient to verify compute integrity. It motivates our method binding *multi-layer tensor states* rather than a single vector.

C Interpolation methods

TensorCommitments treat activation tensors as samples of a multivariate polynomial $f_T \in \mathbb{F}[X_1, \ldots, X_m]$ on a tensor grid. In this appendix we recall three classical interpolation forms in the multivariate setting. Throughout, we use the same lexicographic ordering of monomials and coefficients as in our polynomial-division section, App. A. Let $\Omega_j = \{\omega_{j,0}, \ldots, \omega_{j,d_j-1}\} \subset \mathbb{F}$ be nodes on axis $j \in [m]$, and let the grid be $\mathbf{\Omega} := \Omega_1 \times \cdots \times \Omega_m$, with samples $T[\mathbf{i}] = f_T(\omega_{1,i_1}, \ldots, \omega_{m,i_m})$ for multi-indices $\mathbf{i} = (i_1, \ldots, i_m)$ where $0 \le i_j < d_j$.

C.1 Newton interpolation

Newton interpolation builds f_T in a hierarchical basis that is lower-triangular with respect to the lexicographic order, so we can compute coefficients by divided differences and update incrementally. For each axis j, define the 1D Newton basis:

$$\Xi_{j,0}(X_j) := 1, \qquad \Xi_{j,r}(X_j) := \prod_{k=0}^{r-1} (X_j - \omega_{j,k}), \quad r \ge 1.$$

For a multi-index $\mathbf{i} = (i_1, \ldots, i_m)$, the tensor-product Newton basis is $\Xi_{\mathbf{i}}(X_1, \ldots, X_m) := \prod_{j=1}^{m} \Xi_{j,i_j}(X_j)$. Then f_T admits the Newton form: $f_T(X_1, \ldots, X_m) = \sum_{\mathbf{i}} \hat{a}_{\mathbf{i}} \, \Xi_{\mathbf{i}}(X_1, \ldots, X_m)$, where the coefficients $\hat{a}_{\mathbf{i}}$ are multivariate divided differences computed recursively from $T[\mathbf{i}]$. Because the basis is triangular in the lexicographical order, solving for $\{\hat{a}_{\mathbf{i}}\}$ reduces to forward substitution, and adding a new grid point only affects a suffix of coefficients.

C.2 Barycentric interpolation

Barycentric Lagrange interpolation provides a numerically stable representation of polynomial interpolation. In multiple dimensions, we apply it axis-wise and combine via tensor products. For

each axis j, define 1D barycentric weights as:

$$w_{j,i} := \left(\prod_{\substack{r=0 \\ r \neq i}}^{d_j - 1} (\omega_{j,i} - \omega_{j,r}) \right)^{-1}, \quad i \in \{0, \ldots, d_j - 1\}.$$

Given an evaluation point $\mathbf{t} = (t_1, \ldots, t_m) \in \mathbb{F}^m$, the 1D barycentric factors are $\left(\frac{w_{j,i}}{t_j - \omega_{j,i}}\right) / \left(\sum_{r=0}^{d_j-1} \frac{w_{j,r}}{t_j - \omega_{j,r}}\right)$. Then, the multivariate interpolant is formulated as:

$$f_T(\mathbf{t}) = \sum_{\mathbf{i}} \left(\prod_{j=1}^{m} \frac{\frac{w_{j,i_j}}{t_j - \omega_{j,i_j}}}{\sum_{r=0}^{d_j-1} \frac{w_{j,r}}{t_j - \omega_{j,r}}} \right) T[\mathbf{i}].$$

All grid dependence is captured by the precomputed $w_{j,i}$. Evaluation at a new $\mathbf{t}$ only requires $\mathcal{O}\left(\sum_j d_j + \prod_j d_j\right)$ field operations with good numerical robustness. In our setting, barycentric interpolation is an alternative to the monomial/Lagrange forms used inside TC, but it represents the *same* polynomial f_T.

C.3 Gregory interpolation

Gregory interpolation expresses f_T on a uniform grid in terms of finite differences rather than divided differences, and generalizes the Newton-Gregory forward/backward formulas to multiple dimensions.

Assume each axis j uses equally spaced nodes $\omega_{j,r} = \omega_{j,0} + rh_j$. Let Δ_j be the forward-difference operator along axis j, acting on the grid values $T[\mathbf{i}]$, formally given as $(\Delta_j T)[\mathbf{i}] := T[i_1, \ldots, i_j + 1, \ldots, i_m] - T[i_1, \ldots, i_j, \ldots, i_m]$. Higher-order mixed differences are $\Delta^{\mathbf{r}} T := \Delta_1^{r_1} \cdots \Delta_m^{r_m} T$ for $\mathbf{r} = (r_1, \ldots, r_m)$. For an evaluation point $\mathbf{t} = (\omega_{1,0} + s_1 h_1, \ldots, \omega_{m,0} + s_m h_m)$ with fractional coordinates $s_j \in \mathbb{F}$, the multivariate Gregory form can be written as:

$$f_T(\mathbf{t}) = \sum_{\mathbf{r}} \left(\prod_{j=1}^{m} \binom{s_j}{r_j} \right) \Delta^{\mathbf{r}} T[0, \ldots, 0],$$

where the sum ranges over $0 \leq r_j < d_j$ and $\binom{s_j}{r_j}$ are generalized binomial coefficients. In words, f_T is reconstructed from a finite stencil of mixed differences at a reference grid point and combinatorial weights depending on $\mathbf{t}$. For tensorized, regularly spaced activations this form is convenient for local updates and streaming settings, and again defines the same underlying multivariate polynomial f_T as the Newton and barycentric forms above.

D Dynamic programming solution and analysis for the optimization problem

Problem 4.5 asks for an optimal way to allocate M verifiers over $\mathcal{L}$ layers. Each layer $i \in [\mathcal{L}]$ has benefit $\nu_i \geq 0$ and verification cost $\phi_i > 0$, and verifier $k \in [M]$ has budget $\beta_k > 0$ and may attest at most one *contiguous* interval of layers. The ILP in the main text encodes this as binary variables $\gamma_{k,i}$, $s_{k,i}$, $o_{k,i}$, with the goal of maximizing total benefit $\sum_{k=1}^{M} \sum_{i=1}^{\mathcal{L}} \nu_i \gamma_{k,i}$ subject to per-verifier budgets, non-overlap across verifiers, and contiguity constraints.

For convenience, define prefix sums of benefits and costs

$$\bar{\nu}[i] := \sum_{j=1}^{i} \nu_j, \qquad \bar{\phi}[i] := \sum_{j=1}^{i} \phi_j, \qquad \bar{\nu}[0] = \bar{\phi}[0] = 0.$$

The benefit and cost of any interval $[s, i]$ with $1 \leq s \leq i \leq \mathcal{L}$ are then $\bar{\nu}[i] - \bar{\nu}[s-1]$ and $\bar{\phi}[i] - \bar{\phi}[s-1]$, respectively.

For $k \in \{0, \ldots, M\}$ and $i \in \{0, \ldots, \mathcal{L}\}$, we define the dynamic programming (DP) state

$$\mathrm{DP}[k, i] \;:=\; \max\Big\{\text{total benefit achievable using verifiers } 1{:}k \text{ on layers } 1{:}i\Big\}.$$

Boundary conditions are $\mathrm{DP}[0, i] = \mathrm{DP}[k, 0] = 0$ (no verifiers or no layers yield zero benefit).

At layer i and verifier $k \geq 1$ we have two choices:

- *No interval ends at i for verifier k*: in this case we inherit the previous value, $\mathrm{DP}[k, i-1]$.
- *Verifier k ends its interval at i*: choose a start index $s \in [1, i]$ such that the interval $[s, i]$ is budget-feasible for verifier k, i.e. $\bar{\phi}[i] - \bar{\phi}[s-1] \leq \beta_k$. The benefit contributed by this choice is $\bar{\nu}[i] - \bar{\nu}[s-1]$, plus the optimal benefit on layers $1{:}s-1$ using only verifiers $1{:}k-1$, namely $\mathrm{DP}[k-1, s-1]$.

This yields the Bellman recursion

$$\mathrm{DP}[k,i] = \max\left\{ \mathrm{DP}[k,i-1],\ \max_{\substack{1\leq s\leq i \\ \bar{\phi}[i]-\bar{\phi}[s-1]\leq\beta_k}} \Big(\bar{\nu}[i] - \bar{\nu}[s-1] + \mathrm{DP}[k-1, s-1]\Big) \right\}, \quad (1)$$

for all $k \in [M]$ and $i \in [\mathcal{L}]$. The optimal value of Problem 4.5 is $\mathrm{DP}[M, \mathcal{L}]$. Standard backtracking on the argmax choices in (1) recovers the selected intervals and thus the $\gamma_{k,i}$ variables.

Proposition D.1. *The DP recursion in Eq.* (1) *computes the optimal objective value of Problem 4.5. For each k it selects either (i) no interval or (ii) a single budget-feasible contiguous interval of layers, the intervals are pairwise disjoint across verifiers, and the resulting total benefit is maximal.*

Proof of Proposition D.1. For $k \in \{0, \ldots, M\}$ and $i \in \{0, \ldots, \mathcal{L}\}$, let $\mathrm{OPT}(k, i)$ denote the optimal objective value of Problem 4.5 restricted to: (i) at most k verifiers (with budgets $\beta_1, \ldots, \beta_k$), (ii) layers $1{:}i$ only, and (iii) at most one contiguous interval per verifier, with no overlap across verifiers. We prove by induction on (k, i) in lexicographic order that

$$\mathrm{DP}[k,i] \ = \ \mathrm{OPT}(k,i) \quad \forall k, i.$$

Base cases: If $k = 0$ or $i = 0$, no layer can be verified, so $\mathrm{OPT}(k, i) = 0$ by definition. The DP satisfies $\mathrm{DP}[0, i] = \mathrm{DP}[k, 0] = 0$, hence the claim holds.

Inductive step: Fix $k \geq 1$ and $i \geq 1$ and assume $\mathrm{DP}[k', i'] = \mathrm{OPT}(k', i')$ for all (k', i') with either $k' < k$ or $k' = k$ and $i' < i$. Consider any optimal solution achieving $\mathrm{OPT}(k, i)$.

There are two exhaustive cases:

Case 1: verifier k does not end an interval at layer i. Then the allocation on layers $1{:}i$ is also feasible for the restricted problem on layers $1{:}i-1$ with k verifiers, so $\mathrm{OPT}(k, i) \leq \mathrm{OPT}(k, i-1)$. Conversely, any solution achieving $\mathrm{OPT}(k, i-1)$ can be extended to layers $1{:}i$ by simply ignoring layer i, so $\mathrm{OPT}(k, i) \geq \mathrm{OPT}(k, i-1)$. Thus $\mathrm{OPT}(k, i) = \mathrm{OPT}(k, i-1)$. By the induction hypothesis, $\mathrm{OPT}(k, i-1) = \mathrm{DP}[k, i-1]$, and the branch $\mathrm{DP}[k, i-1]$ in the recursion is feasible and attains this value.

Case 2: verifier k ends an interval at layer i. Then there exists a start index $s \in [1, i]$ such that verifier k is assigned the contiguous interval $[s, i]$, and the budget constraint implies $\bar{\phi}[i] - \bar{\phi}[s-1] \leq \beta_k$. Let the total benefit of this interval be $\bar{\nu}[i] - \bar{\nu}[s-1]$. Since intervals are non-overlapping and each verifier uses at most one interval, layers $1{:}s-1$ can only be assigned to verifiers $1{:}k-1$, and layers $s{:}i$ are fully occupied by verifier k.

Hence, the restriction of this optimal solution to layers $1{:}s-1$ and verifiers $1{:}k-1$ is feasible for the subproblem $(k-1, s-1)$ and achieves at most $\mathrm{OPT}(k-1, s-1)$. By the induction hypothesis, $\mathrm{OPT}(k-1, s-1) = \mathrm{DP}[k-1, s-1]$. Thus the total benefit of any solution in Case 2 with start s is bounded above by

$$\mathrm{DP}[k-1, s-1] \ + \ \big(\bar{\nu}[i] - \bar{\nu}[s-1]\big),$$

and taking the best s satisfying $\bar{\phi}[i] - \bar{\phi}[s-1] \leq \beta_k$ yields the upper bound

$$\mathrm{OPT}(k,i) \ \leq \max_{\substack{1\leq s\leq i \\ \bar{\phi}[i]-\bar{\phi}[s-1]\leq\beta_k}} \Big(\mathrm{DP}[k-1, s-1] + \bar{\nu}[i] - \bar{\nu}[s-1]\Big).$$

On the other hand, for any such feasible s, we can construct a valid allocation for (k, i) by (i) taking an optimal solution for $(k-1, s-1)$ (realizing $\mathrm{DP}[k-1, s-1]$ by induction), and (ii) assigning the interval $[s, i]$ to verifier k, adding benefit $\bar{\nu}[i] - \bar{\nu}[s-1]$. This gives a feasible solution with value exactly $\mathrm{DP}[k-1, s-1] + \bar{\nu}[i] - \bar{\nu}[s-1]$, so the maximum over all feasible s is attainable and equals the best value in Case 2.

Combining the two cases, any optimal solution for (k, i) has value

$$\mathrm{OPT}(k,i) = \max\Bigg\{\mathrm{OPT}(k,i-1),\ \max_{\substack{1\le s\le i\\ \bar\phi[i]-\bar\phi[s-1]\le\beta_k}} \Big(\mathrm{OPT}(k-1,s-1) + \bar\nu[i] - \bar\nu[s-1]\Big)\Bigg\}.$$

Substituting $\mathrm{OPT}(k', i') = \mathrm{DP}[k', i']$ from the induction hypothesis gives exactly the DP recursion in Eq. (1), i.e. $\mathrm{OPT}(k, i) = \mathrm{DP}[k, i]$.

Finally, for $(k, i) = (M, \mathcal{L})$ we obtain $\mathrm{DP}[M, \mathcal{L}] = \mathrm{OPT}(M, \mathcal{L})$, which is precisely the optimal objective value of Problem 4.5. By construction of the subproblems, every solution counted by DP obeys the budget, non-overlap, and "at most one interval per verifier" constraints, and conversely every feasible assignment is represented in some (k, i) and hence in $\mathrm{DP}[M, \mathcal{L}]$. □

A naive implementation of (1) would scan all starts s for each pair (k, i), leading to $\mathcal{O}(M\mathcal{L}^2)$ time. We can do better by exploiting the monotonicity of the cost prefix sums $\bar\phi[i]$ and the fixed budget β_k.

For fixed k, the inner maximization can be rewritten as

$$\max_{\substack{1\le s\le i\\ \bar\phi[s-1]\ge\bar\phi[i]-\beta_k}} \big(\mathrm{DP}[k-1,s-1] - \bar\nu[s-1]\big) \ + \ \bar\nu[i].$$

Thus, for each layer i we only need the maximum of the "adjusted score" $\mathrm{score}_k(s) := \mathrm{DP}[k-1, s-1] - \bar\nu[s-1]$ over the index set

$$\mathcal{W}_k(i) := \Big\{ s \in [1,i] \ : \ \bar\phi[s-1] \ge \bar\phi[i] - \beta_k \Big\}.$$

Because all $\phi_i > 0$, the sequence $\bar\phi[0], \bar\phi[1], \ldots, \bar\phi[\mathcal{L}]$ is strictly increasing. As i grows, the lower bound $\bar\phi[i] - \beta_k$ increases monotonically, so the feasible set $\mathcal{W}_k(i)$ is a sliding suffix window that moves to the right and never moves back.

We maintain this window with a monotone deque:

- We iterate $i = 1{:}\mathcal{L}$ once for each k. When i advances, we first remove from the front of the deque any indices s that no longer satisfy $\bar\phi[s-1] \ge \bar\phi[i] - \beta_k$.
- We then insert i at the back, after removing from the back any indices s with $\mathrm{score}_k(s) \le \mathrm{score}_k(i)$, so that $\mathrm{score}_k(\cdot)$ is strictly decreasing along the deque.
- The front of the deque always stores an index $s^\star$ achieving the maximum $\mathrm{score}_k(s)$ over $\mathcal{W}_k(i)$, and we compute $\mathrm{DP}[k, i] = \max\{\mathrm{DP}[k, i-1],\ \bar\nu[i] + \mathrm{score}_k(s^\star)\}$.

Each index enters and leaves the deque at most once, so the amortized cost per state (k, i) is $\mathcal{O}(1)$.

Proposition D.2. *The above implementation evaluates* (1) *in $\mathcal{O}(M\mathcal{L})$ time and $\mathcal{O}(\mathcal{L})$ space.*

For each verifier k we perform a single left-to-right pass over $i = 1{:}\mathcal{L}$, with $\mathcal{O}(1)$ amortized deque operations per i, yielding $\mathcal{O}(\mathcal{L})$ time per verifier and $\mathcal{O}(M\mathcal{L})$ overall. We only need the previous DP row $\mathrm{DP}[k-1, \cdot]$ and the current row $\mathrm{DP}[k, \cdot]$, plus the prefix sums and the deque, so the memory footprint is $\mathcal{O}(\mathcal{L})$. This justifies the claim in the main text that the verifier can compute an optimal allocation over $\mathcal{L}$ layers and M verifiers in time linear in $\mathcal{L}$ per verifier.

Proposition D.3. *Let $\bar\nu[i] = \sum_{j=1}^{i} \nu_j, \bar\phi[i] = \sum_{j=1}^{i} \phi_j, \bar\nu[0] = \bar\phi[0] = 0$ denote prefix sums of benefits and costs. For $k \in \{0, \ldots, M\}$ and $i \in \{0, \ldots, \mathcal{L}\}$, using the DP as:*

$$\mathrm{DP}[k,i] \ := \ \max\Big\{\textit{benefit by verifiers } 1{:}k \textit{ on layers } 1{:}i\Big\},$$

with $\mathrm{DP}[0,i] = \mathrm{DP}[k,0] = 0$ gives the Bellman recursion:

$$\mathrm{DP}[k,i] = \max\Bigg\{\mathrm{DP}[k,i-1],\ \max_{\substack{1\le s\le i\\ \bar\phi[i]-\bar\phi[s-1]\le\beta_k}} \Big(\bar\nu[i] - \bar\nu[s-1] + \mathrm{DP}[k-1,s-1]\Big)\Bigg\},$$

accounting for either no interval ending at i, or using verifier k on some feasible interval $[s, i]$ and solving the best for verifiers 1:k-1 on layers 1:s-1. Then, this is an exact algorithm for Problem 4.5. For every $k \in [M]$, it selects either a budget-feasible contiguous interval or no interval. With the selected pairwise disjoint intervals, the total benefit is maximized.

For the inner max, a sliding-window max over prefix indices yields an $\mathcal{O}(1)$ amortized cost per state.

E Cryptographic primitives in detail

E.1 Pairing-friendly elliptic curves

Pairing-based cryptography is an advanced cryptographic approach that leverages bilinear maps on elliptic curves to enable complex security protocols. In TensorCommitments, pairings are a core building block that address data privacy, secure model inference, and secure communication. Let G be a cyclic group of prime order p, written multiplicatively, with generator g and identity 1_G, and let G_T be the multiplicatively written target group of order p. For elliptic curves, the group operation of G, written as a product $g_1 g_2$, is the point addition of Fig. 8, and 1_G is the point at infinity ∞.

A bilinear pairing on (G, G_T) is a mapping, $e : G \times G \to G_T$, that satisfies the following conditions:

- **Bilinearity:** For all $g_1, g_2 \in G$ and $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
  Equivalently, for all $g_1, g_2, g_3 \in G$, $e(g_1 g_2, g_3) = e(g_1, g_3)e(g_2, g_3)$, and $e(g_1, g_2 g_3) = e(g_1, g_2)e(g_1, g_3)$.
- **Non-degeneracy:** $e(g, g) \neq 1_{G_T}$, where 1_{G_T} is the identity element of the target group G_T.
- **Computability:** $e(g_1, g_2)$ can be efficiently computed on inputs $g_1, g_2 \in G$.

Bilinear groups have the following properties: For all $g_1, g_2 \in G$,

- $e(g_1, 1_G) = 1_{G_T}$ and $e(1_G, g_1) = 1_{G_T}$.
- $e(g_1, g_2^{-1}) = e(g_1^{-1}, g_2) = e(g_1, g_2)^{-1}$.
- $e(g_1^a, g_2) = e(g_1, g_2)^a = e(g_1, g_2^a)$ for $a \in \mathbb{Z}_p$.
- $e(g_1, g_2) = e(g_2, g_1)$.
- For finitely many $g_{1,i} \in G$ and $g_2 \in G$, $\prod_i e(g_{1,i}, g_2) = e(\prod_i g_{1,i}, g_2)$.
- For a fixed $g_1 \in G \setminus \{1_G\}$, if $e(g^a, g_1) = e(g^b, g_1)$, then $a \equiv b \pmod p$.

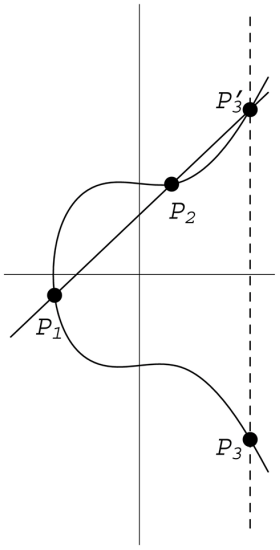

Figure 8: Elliptic-curve point addition

E.2 Elliptic curve for applications

First, we briefly recap standardized elliptic curves most commonly used in real-world applications.

Definition E.1 (Elliptic Curve)**.** An elliptic curve E is a curve (usually) of the form $y^2 = x^3 + a_4 x + a_6$, defined over a finite field $\mathbb{F}_{p_{\text{base}}}$, where $a_4, a_6 \in \mathbb{F}_{p_{\text{base}}}$ are constants.

This equation is called the Weierstrass equation. We can look at the set of points of E defined over $\mathbb{F}_{p_{\text{base}}}$ as $E(\mathbb{F}_{p_{\text{base}}}) = \{(x, y) \in \mathbb{F}_{p_{\text{base}}}^2 \mid y^2 = x^3 + a_4 x + a_6\} \cup \{\infty\}$. Then, $E(\mathbb{F}_{p_{\text{base}}})$ can be given a group structure where the elements are points, and the operation is point addition.

Given two points P_1 and P_2, we define their summation in the following way: Draw a line through P_1 and P_2, and let P_3' be the point where the line intersects E again, as in Fig. 8. Then reflect P_3' across the x-axis to obtain the point $P_3 = P_1 + P_2$. We can also find $2P_1 = P_1 + P_1$ by drawing the tangent line to the curve.

The benefit of this structure to the field of cryptography is that point addition on elliptic curves is quite difficult and time-consuming. Moreover, if we are given two points P and Q, and told that $kP = Q$, it is very hard and time-consuming to find k. Classical methods of solving this problem have faster specializations for certain groups, which means that for the problem to be hard, the group in question must be a large prime field. However, elliptic curve methods are equally difficult over prime groups and other similarly sized generic groups, and so they have no such specializations. Thus cryptography using elliptic curves is more efficient than using classical methods, because the elliptic curve variations of the classical methods offer more security over smaller groups. Solving this problem, which is called the *discrete logarithm problem*, is central to elliptic curve cryptography.

In this work, we mostly use BLS12-381 [39], which is a specific, widely used pairing-friendly elliptic curve. This curve is defined over finite fields and has algebraic properties that facilitate pairings.

E.3 Security parameter λ

The security parameter $\lambda \in \mathbb{N}$ (with $\lambda \geq 128$ being a typical choice) is an integer that controls every piece of the cryptosystem's machinery. This security parameter controls the *computational difficulty* of breaking the cryptosystem: all probabilities of adversary success must be negligible in λ, which means they must go to zero faster than any inverse polynomial. Concretely, for $\lambda = 128$ the number of operations an adversary has to perform is at least of the order of magnitude of 2^{128}. In the single trusted setup $\mathsf{Setup}_{\mathsf{TC}}(\lambda, \mathbf{d})$, the security parameter controls the following three interrelated values:

1. **Fields and Group Sizes:** The chosen curve provides a pairing-friendly elliptic curve with groups G and G_T both of prime order p along with a bilinear map $e : G \times G \to G_T$, where p is a prime of about 2λ bits. The scalar field $\mathbb{F} = \mathbb{Z}_p$ of exponents has the same order p. For our instantiation, BLS12-381 [39] is employed, where p is a 255-bit prime, so $|\mathbb{F}| = p$ with $2^{254} < p < 2^{255}$. Pollard's rho algorithm then needs $\sqrt{p} \approx 2^{127.4}$, i.e., about 2^{128} group operations, giving an approximate security of $\lambda = 128$ bits against discrete logarithm attacks. The curve itself is defined over a different base field $\mathbb{F}_{p_{\text{base}}}$ with a 381-bit prime p_{base}, which sets the size of a group element. Choosing λ greater implies that p will be bigger, hence enlarging both the group elements and the field $\mathbb{F}$ over which tensor values, coefficients of polynomials, and trapdoors live.
2. **Trapdoor Entropy:** For a tensor of order m with dimensions $\mathbf{d} = (d_1, \ldots, d_m)$, $\mathsf{Setup}_{\mathsf{TC}}$ selects m independent trapdoor scalars $\tau_1, \ldots, \tau_m \leftarrow \$\mathbb{F}$. Given that $|\mathbb{F}|$ can be bounded via λ, each τ_j contains at least λ bits of min-entropy. The structured reference string $\mathsf{srs} = \left\{ g^{\tau_1^{i_1} \cdots \tau_m^{i_m}} \right\}_{(i_1,\ldots,i_m) \in \mathcal{I}}$ is then published, while the trapdoors themselves are securely discarded. All subsequent commitments and verifications operate exclusively over the srs; knowledge of any τ_j would break the scheme.
3. **Hardness Assumption Threshold:** App. F reduces evaluation binding to the q-SBDH assumption with $q = \sum_{j=1}^{m}(d_j - 1)$. Given $g, g^{\tau}, \ldots, g^{\tau^q} \in G$ for a random $\tau \in \mathbb{F}$, no PPT adversary can compute the pair $(a,\ e(g,g)^{1/(\tau+a)})$ except with probability $\mathrm{negl}(\lambda)$. Since q grows polynomially in λ and the group order grows exponentially, increasing λ widens the gap between what the adversary receives and what it would need to forge.

The effect of increasing the security parameter can be summarized along three axes detailed below.

Security margin: The best-known attack strategy of the opponent entails complexity of $\mathcal{O}(\sqrt{p}) = \mathcal{O}(2^{\lambda})$ using Pollard's rho algorithm on the elliptic curve group (or fewer operations in case of a pairing target attack on some specific elliptic curves). Doubling λ squares the computation cost 2^{λ} for the adversary, hence creating a wider security margin. With $\lambda = 128$, the BLS12-381 elliptic curve is considered safe from any conceivable hardware.

Group-element and proof sizes: Each element in the group G requires $\mathcal{O}(\lambda)$ bits. The commitment C_T requires one element from the group, while the proof $\pi_{\boldsymbol{\omega}}$ involves m elements (corresponding to the tensor axes), with extra Terkle-tree path elements requiring $\mathcal{O}(\log_B n)$ elements. Therefore, an increase in λ will cause the byte size of proofs and commitments to grow linearly. In BLS12-381 ($\lambda \approx 128$), a compressed group element takes up 48 bytes, meaning that a tensor proof of rank m will require around $48m$ bytes.

Costs: Both parties spend most of their time performing group exponentiations and evaluating pairings. Each one requires $\widetilde{\mathcal{O}}(\lambda^2)$ bit operations (or $\widetilde{\mathcal{O}}(\lambda)$ field multiplications via optimized library routines) due to computations modulo an integer of $\Omega(\lambda)$ bits. Doubling λ leads to a factor-of-four increase in costs per operation. But the number of pairings the verifier needs to check for each challenged coordinate - $\mathcal{O}(\log_B n)$ - is independent of λ. In reality, the overhead comes from the prover's polynomial interpolation process, which depends on the tensor's dimensions but not λ, and therefore does not change much upon a modest increase in λ.

E.4 Cryptography literature

Zero-Knowledge Proofs for Computation: Zero-knowledge SNARKs [12, 32, 3] provide succinct proofs of arbitrary computations with strong cryptographic guarantees. These systems and their recent versions [28, 8] achieve remarkable succinctness, but incur significant prover overhead - often minutes per query for large circuits. ZK-STARKs [2] offer transparency without a trusted setup and post-quantum security with larger proof sizes. Recently, recursive SNARKs [4] enable proof composition but still face scalability challenges at LLM scale.

Verifiable Computation: The foundations of verifiable computation [10] established interactive proof systems where a weak verifier can check a powerful prover's work. Modern systems like Spartan [35] and doubly-efficient protocols [40] reduce the verification complexity but still require the verifier to see the computation structure. Unlike these general-purpose systems, we exploit the specific structure of LLM inference - tensor operations over known architectures - for better constants.

Polynomial Commitments: Our TensorCommitments extend the KZG polynomial commitment scheme [19], which commits to univariate polynomials using bilinear pairings, and instantiate the multivariate KZG of Papamanthou et al. [31] on tensor grids. Recent advances include Bulletproofs [5] which are transparent but with linear verification and FRI-based commitments [1] which are widely used in STARKs. The multivariate polynomial interpolation theory we leverage [13] has been extensively studied in numerical analysis but, to our knowledge, has not been used to commit to tensors on their native grids or for ML verification. Vector commitments [6] provide the closest comparison, but our tensor-native approach fundamentally differs by preserving dimensional structure.

F Proof of construction

Here we prove that our tensor commitment construction TC of Def. 4.1 is correct and position binding. TC instantiates the multivariate KZG commitment of Papamanthou et al. [31] on the tensor grid $\mathbf{\Omega} = \Omega_1 \times \cdots \times \Omega_m \subset \mathbb{F}^m$ of App. C, and we give self-contained proofs in our notation. Correctness means that honestly generated commitments and openings always verify. Position binding means that, once a commitment is fixed, a cheating prover cannot open a grid point to two distinct values. Binding thus fixes the committed tensor. It does not by itself certify that the tensor was computed correctly (Sec. 3). Throughout, $d_j \geq 2$ for all $j \in [m]$ (an axis of size one can be dropped), so $g^{\tau_j} \in \mathsf{srs}$.

Correctness: Let pp be the output of $\mathsf{Setup}_{\mathsf{TC}}(\lambda, \mathbf{d})$ of Def. 4.1. We prove that for any function value T on grid $\mathbf{\Omega}$ with $C_T = \mathsf{Com}_{\mathsf{TC}}(\mathsf{pp}, T)$, and any challenge point $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_m)$ with $(y, \pi_{\boldsymbol{\omega}}) = \mathsf{Open}_{\mathsf{TC}}(\mathsf{pp}, T, \boldsymbol{\omega})$,

$$\Pr[\mathsf{Ver}_{\mathsf{TC}}(\mathsf{pp}, C_T, \boldsymbol{\omega}, y, \pi_{\boldsymbol{\omega}}) = 1] = 1. \tag{2}$$

From the construction, $C_T = g^{f_T(\tau_1, \ldots, \tau_m)}$ commits to the multivariate polynomial $f_T \in \mathbb{F}[X_1, \ldots, X_m]$, where $f_T(\boldsymbol{\omega}) = T[\boldsymbol{\omega}], \forall \boldsymbol{\omega} \in \mathbf{\Omega}$, and $\mathsf{Open}_{\mathsf{TC}}$ runs the m division steps $r_{i-1} = q_i(X_i, \ldots, X_m)\,(X_i - \omega_i) + r_i$, $i \in [m]$, of App. A with $r_0 = f_T$ and scalar $r_m = y$, and outputs $\pi_{\omega_i} = g^{q_i(\tau_i, \ldots, \tau_m)}$. Summing the m steps, the intermediate remainders cancel:

$$f_T(X_1, \ldots, X_m) - y \;=\; \sum_{i=1}^{m} \big(r_{i-1} - r_i\big) \;=\; \sum_{i=1}^{m} q_i(X_i, \ldots, X_m)\,(X_i - \omega_i). \tag{3}$$

Setting $X_i = \omega_i$ for all i in (3) zeroes every summand, so $y = f_T(\boldsymbol{\omega})$. As f_T has degree at most $d_k - 1$ in each X_k, so does every r_{i-1}. Hence q_i has degree at most $d_i - 2$ in X_i and at most $d_k - 1$ in X_k for $k > i$, and does not depend on $X_1, \ldots, X_{i-1}$. All monomials of q_i thus lie in $\mathcal{I}$, and π_{ω_i} is computed

from srs by linearity in the exponent. To prove (2), it suffices to show

$$e(C_T \cdot g^{-y},\, g) \;=\; \prod_{i=1}^{m} e\big(\pi_{\omega_i},\, g^{\tau_i-\omega_i}\big). \tag{4}$$

From the construction and using the properties of pairings listed in App. E.1, we simplify each side of (4) as follows.

$$e(C_T \cdot g^{-y},\, g) = e\big(g^{f_T(\tau_1,\ldots,\tau_m)} \cdot g^{-y},\, g\big) = e(g,g)^{f_T(\tau_1,\ldots,\tau_m)-y}, \tag{5}$$

$$\prod_{i=1}^{m} e\big(\pi_{\omega_i},\, g^{\tau_i-\omega_i}\big) = \prod_{i=1}^{m} e\big(g^{q_i(\tau_i,\ldots,\tau_m)},\, g^{\tau_i-\omega_i}\big) = e(g,g)^{\sum_{i=1}^{m} q_i(\tau_i,\ldots,\tau_m)\,(\tau_i-\omega_i)}. \tag{6}$$

Hence, proving the pairing equality of (4) reduces to equating the exponents of (5) and (6):

$$f_T(\tau_1,\ldots,\tau_m) - y = \sum_{i=1}^{m} q_i(\tau_i,\ldots,\tau_m)\,(\tau_i-\omega_i), \tag{7}$$

which is the polynomial identity (3) evaluated at the trapdoors $X_i = \tau_i$, $i \in [m]$. □

Position (evaluation) binding: We show that for every PPT adversary $\mathcal{A}$ and $\mathsf{pp} \leftarrow \mathsf{Setup}_{\mathsf{TC}}(\lambda, \mathbf{d})$, the probability that $\mathcal{A}(\mathsf{pp})$ outputs $(C, \boldsymbol{\omega}, y, \pi_{\boldsymbol{\omega}}, y', \pi'_{\boldsymbol{\omega}})$ with an arbitrary $C \in G$, $\boldsymbol{\omega} \in \boldsymbol{\Omega}$, and $y \neq y'$, such that $\mathsf{Ver}_{\mathsf{TC}}(\mathsf{pp}, C, \boldsymbol{\omega}, y, \pi_{\boldsymbol{\omega}}) = \mathsf{Ver}_{\mathsf{TC}}(\mathsf{pp}, C, \boldsymbol{\omega}, y', \pi'_{\boldsymbol{\omega}}) = 1$, is negligible in λ. We require $y \neq y'$ because, for $m \geq 2$, proof tuples are not unique. Replacing $(\pi_{\omega_1}, \pi_{\omega_2})$ by $(\pi_{\omega_1} g^{b(\tau_2-\omega_2)},\, \pi_{\omega_2} g^{-b(\tau_1-\omega_1)})$ for any $b \in \mathbb{F}$ leaves the right-hand side of (4) unchanged. This non-uniqueness is harmless, as the verifier only uses y. We reduce binding to the q-Strong Bilinear Diffie-Hellman (q-SBDH) assumption used by [31], with $q = \sum_{j=1}^{m}(d_j - 1)$. This assumption states that, given $(g, g^{\tau}, \ldots, g^{\tau^q})$ for a uniform $\tau \in \mathbb{F}$, no PPT algorithm outputs $(a,\, e(g,g)^{1/(\tau+a)})$ for some $a \in \mathbb{F} \setminus \{-\tau\}$, except with probability $\mathrm{negl}(\lambda)$ (App. E.3).

Suppose $\mathcal{A}$ wins with probability $\varepsilon_{\mathcal{A}}$. We build $\mathcal{B}$ that, on a q-SBDH instance, samples a guess $\boldsymbol{\omega}^* \in \boldsymbol{\Omega}$ and $\psi_2, \ldots, \psi_m \in \mathbb{F}$ uniformly, sets $\psi_1 := 1$, and implicitly fixes the trapdoors $\tau_1 := \tau$ and $\tau_i := \psi_i(\tau - \omega_1^*) + \omega_i^*$ for $i \geq 2$, so that $\tau_i - \omega_i^* = \psi_i\,(\tau - \omega_1^*)$ for all $i \in [m]$. Every srs entry $g^{\tau_1^{k_1} \cdots \tau_m^{k_m}}$, $(k_1, \ldots, k_m) \in \mathcal{I}$, is g raised to a polynomial in τ of degree at most $k_1 + \ldots + k_m \leq q$ whose coefficients $\mathcal{B}$ knows, so $\mathcal{B}$ computes pp from its instance. Since τ and the ψ_i are uniform, for every $\boldsymbol{\omega}^*$ the trapdoors $(\tau_1, \ldots, \tau_m)$ are uniform on $\mathbb{F}^m$ up to a statistical distance $\delta_{\mathrm{sd}} = \mathcal{O}(1/p)$. Hence pp is distributed as in $\mathsf{Setup}_{\mathsf{TC}}$, $\boldsymbol{\omega}^*$ is hidden from $\mathcal{A}$, and $\mathcal{A}$ wins with $\boldsymbol{\omega} = \boldsymbol{\omega}^*$ with probability at least $\varepsilon_{\mathcal{A}}/|\boldsymbol{\Omega}| - \delta_{\mathrm{sd}}$. In that case, both verification equations hold at $\boldsymbol{\omega}^*$, and dividing them by bilinearity (App. E.1) gives

$$\begin{cases} e(C \cdot g^{-y},\, g) = \prod_{i=1}^{m} e\big(\pi_{\omega_i},\, g^{\tau_i-\omega_i^*}\big) \\ e(C \cdot g^{-y'},\, g) = \prod_{i=1}^{m} e\big(\pi'_{\omega_i},\, g^{\tau_i-\omega_i^*}\big) \end{cases} \quad \Rightarrow \quad e(g,g)^{y'-y} = \prod_{i=1}^{m} e\Big(\frac{\pi_{\omega_i}}{\pi'_{\omega_i}},\, g^{\tau_i-\omega_i^*}\Big). \tag{8}$$

Substituting $\tau_i - \omega_i^* = \psi_i(\tau - \omega_1^*)$ and letting $g_\star := \prod_{i=1}^{m} (\pi_{\omega_i}/\pi'_{\omega_i})^{\psi_i} \in G$, which $\mathcal{B}$ computes, the right-hand side of (8) equals $e(g_\star, g)^{\tau-\omega_1^*}$. As $y' \neq y$, the left-hand side is not 1_{G_T}, so $\tau \neq \omega_1^*$, and raising both sides to $1/\big((y'-y)(\tau-\omega_1^*)\big)$ yields

$$e(g,g)^{1/(\tau-\omega_1^*)} = e(g_\star, g)^{1/(y'-y)}.$$

Thus, $\mathcal{B}$ outputs $(a,\, e(g_\star, g)^{1/(y'-y)})$ with $a := -\omega_1^*$, breaking q-SBDH with probability at least $\varepsilon_{\mathcal{A}}/|\boldsymbol{\Omega}| - \delta_{\mathrm{sd}}$. Since $|\boldsymbol{\Omega}| = \prod_j d_j$ is polynomial in λ, $\varepsilon_{\mathcal{A}} \leq |\boldsymbol{\Omega}|\big(\mathrm{negl}(\lambda) + \delta_{\mathrm{sd}}\big) = \mathrm{negl}(\lambda)$. If $\boldsymbol{\omega}$ is instead sampled by the verifier after C is fixed (e.g., a random point in $\mathbb{F}^m$), $\mathcal{B}$ samples $\boldsymbol{\omega}^*$ itself and the factor $|\boldsymbol{\Omega}|$ disappears. □

G Ablation study on the number of verifiers

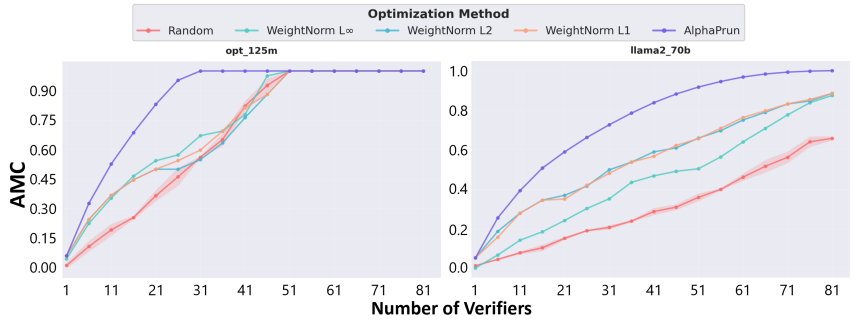

Figure 9: **How does scaling the verifier nodes change attack coverage?** For OPT-125M (left) and LLaMA2-70B (right), we solve Problem 4.5 under a budget where each verifier can check at most 1% of consecutive layers, and measure AMC (fraction of attacked layers covered) as we increase the number of verifiers for different layer-ranking objectives. The α-score objective [27] consistently dominates other objectives, reaching 0.8 AMC with 21 verifiers on OPT-125M and requiring only 36 verifiers to achieve similar coverage on LLaMA2-70B. Quantitatively, to achieve 80% AMC, the system needs $1.7\times$ more verifiers when model size scales up by a factor of 560.

We vary the number of verifiers M from 1 to 81 and, for each M, solve Problem 4.5 under a fixed per-verifier budget (each can attest at most 1% of consecutive layers). This isolates the effect of the allocation policy from raw compute. Across both OPT-125M and LLaMA2-70B, the α-score objective based on heavy-tailed spectra consistently achieves the highest AMC, especially in the low-budget regime where each additional verifier must be used carefully.

The curves also show a favorable scaling trend: moving from OPT-125M to LLaMA2-70B (a $560\times$ increase in parameters) only requires about $1.7\times$ more verifiers to reach 80% AMC. Most of the robustness gain comes from *where* we place verifiers rather than from simply adding more of them, and this advantage persists as model size grows.

H Complexity of multivariate interpolation

In this appendix we make explicit the complexity of multivariate Newton interpolation in the setting of Hecht et al.'s quadratic-time algorithm for the Polynomial Interpolation Problem (PIP), and then re-express the cost in terms of the dimension m and the total number of data points N.

H.1 Polynomial space and problem formulation:

For integers $m \geq 1$ and $d \geq 0$, let

$$\Pi_{m,d} := \{f(X_1, \ldots, X_m) \in \mathbb{R}[X_1, \ldots, X_m] : \deg_{\text{total}} f \leq d\}$$

denote the space of real polynomials in m variables with total degree at most d. It is standard that

$$\dim \Pi_{m,d} = \binom{m+d}{d} \tag{9}$$

is the number of monomials of total degree at most d in m variables.

The *Polynomial Interpolation Problem (PIP)* is: given $m, d \in \mathbb{N}$ and a function $F : \mathbb{R}^m \to \mathbb{R}$, choose a node set $\Omega_{\text{PIP}} = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{\dim \Pi_{m,d}}\} \subset \mathbb{R}^m$ that is (unisolvent) generic for $\Pi_{m,d}$ and compute the unique interpolating polynomial $f \in \Pi_{m,d}$ such that $f(\boldsymbol{\omega}_i) = F(\boldsymbol{\omega}_i)$ for all i.

H.2 Quadratic-time algorithm:

PIP-SOLVER [13] is a recursive algorithm based on a decomposition into subproblems of lower dimension and lower degree. In our notation, their main result can be stated as follows.

Theorem H.1. *For all $m, d \in \mathbb{N}$ with $m \geq 1$, there exists an algorithm that, given oracle access to $F : \mathbb{R}^m \to \mathbb{R}$,*

1. *constructs a generic node set $\Omega_{\mathrm{PIP}} \subset \mathbb{R}^m$ with $\#\Omega_{\mathrm{PIP}} = \dim \Pi_{m,d}$, and*
2. *computes the interpolating polynomial $f \in \Pi_{m,d}$,*

with a runtime of

$$\mathcal{C}(m, d) = \mathcal{O}\big((\dim \Pi_{m,d})^2\big) \tag{10}$$

and a storage cost of

$$\mathcal{R}(m, d) = \mathcal{O}\big(m \dim \Pi_{m,d}\big). \tag{11}$$

The key insight is to split the global PIP into:

- a subproblem of dimension $m - 1$ and degree d, and
- a subproblem of dimension m and degree $d - 1$,

and to recurse until only linear ($d = 1$) or univariate ($m = 1$) subproblems remain, both of which admit direct $\mathcal{O}(d^2)$ or $\mathcal{O}(m^2)$ algorithms.

We now sketch how the $\mathcal{O}\big((\dim \Pi_{m,d})^2\big)$ bound arises from the recursive decomposition. This is not a re-proof of correctness, but a transparent derivation of the runtime for readers unfamiliar with complexity arguments.

Recurrence: Let $\mathcal{C}(m, d)$ denote the number of arithmetic operations performed by PIP-SOLVER [13] for parameters (m, d). The algorithm performs three main types of work:

1. Solve a subproblem on a hyperplane $\mathcal{H} \cong \mathbb{R}^{m-1}$ with parameters $(m - 1, d)$, costing $\mathcal{C}(m - 1, d)$ operations.
2. Solve a subproblem in full dimension with reduced degree, parameters $(m, d - 1)$, costing $\mathcal{C}(m, d - 1)$ operations.
3. Combine the two solutions into the global interpolant f, which involves a linear number of operations per coefficient.

The decomposition is structured so that the total number of nodes used by the two subproblems equals the total number of nodes of the original problem:

$$\dim \Pi_{m-1,d} + \dim \Pi_{m,d-1} = \dim \Pi_{m,d}. \tag{12}$$

This identity is easy to verify directly from (9):

$$\binom{m - 1 + d}{d} + \binom{m + d - 1}{d - 1} = \binom{m + d}{d}.$$

The combination step (forming $f = f_1 + f_{\mathcal{H}} f_2$ from the solutions $f_1 \in \Pi_{m-1,d}$ and $f_2 \in \Pi_{m,d-1}$ of subproblems 1 and 2, following Hecht et al.) requires multiplying the linear polynomial $f_{\mathcal{H}}$, whose zero set is $\mathcal{H}$, by a polynomial with $\dim \Pi_{m,d-1}$ coefficients and then adding this to a polynomial with $\dim \Pi_{m-1,d}$ coefficients. This can be done in a number of arithmetic operations proportional to

$$C_3 \dim \Pi_{m-1,d} \dim \Pi_{m,d-1}$$

for some constant C_3, because each coefficient of the product depends on a constant number of coefficients of the factors and there are $\dim \Pi_{m-1,d}$ and $\dim \Pi_{m,d-1}$ coefficients to process.

Altogether, we obtain a recurrence of the form

$$\mathcal{C}(m, d) \;\leq\; \mathcal{C}(m - 1, d) + \mathcal{C}(m, d - 1) + C_3 \dim \Pi_{m-1,d} \dim \Pi_{m,d-1}, \tag{13}$$

for $m, d > 1$, with base cases $\mathcal{C}(1, d) = \mathcal{O}(d^2)$ and $\mathcal{C}(m, 1) = \mathcal{O}(m^2)$ (univariate and linear interpolation).

Big-$\mathcal{O}$ and Θ notation: For clarity: for two non-negative functions A, B of our parameters,

- $A = \mathcal{O}(B)$ means that there exists a constant $C > 0$ and a parameter region where $A \leq CB$,
- $A = \Theta(B)$ means $A = \mathcal{O}(B)$ *and* $B = \mathcal{O}(A)$. In other words, A and B are bounded above and below by constant multiples of each other.

We will show that $\mathcal{C}(m, d)$ grows on the order of $(\dim \Pi_{m,d})^2$.

Induction hypothesis: Assume there exists a constant $C > 0$ such that for all smaller parameter pairs (m', d') (in the sense $\dim \Pi_{m',d'} < \dim \Pi_{m,d}$) we have

$$\mathcal{C}(m', d') \leq C\,(\dim \Pi_{m',d'})^2. \tag{14}$$

This holds for the base cases by choosing C large enough.

Bounding the recurrence: Using (13) and the induction hypothesis (14) we obtain for $m, d > 1$:

$$\begin{aligned}\mathcal{C}(m, d) &\leq C\,(\dim \Pi_{m-1,d})^2 + C\,(\dim \Pi_{m,d-1})^2 + C_3 \dim \Pi_{m-1,d} \dim \Pi_{m,d-1} \\ &= C\big((\dim \Pi_{m-1,d})^2 + (\dim \Pi_{m,d-1})^2\big) + C_3 \dim \Pi_{m-1,d} \dim \Pi_{m,d-1}.\end{aligned}$$

Using the identity (12), we can rewrite $\dim \Pi_{m,d} = \dim \Pi_{m-1,d} + \dim \Pi_{m,d-1}$. Then

$$\begin{aligned}(\dim \Pi_{m,d})^2 &= \big(\dim \Pi_{m-1,d} + \dim \Pi_{m,d-1}\big)^2 \\ &= (\dim \Pi_{m-1,d})^2 + (\dim \Pi_{m,d-1})^2 + 2 \dim \Pi_{m-1,d} \dim \Pi_{m,d-1}.\end{aligned}$$

Therefore, we have

$$\begin{aligned}\mathcal{C}(m, d) &\leq C\left((\dim \Pi_{m,d})^2 - 2 \dim \Pi_{m-1,d} \dim \Pi_{m,d-1}\right) \\ &\quad + C_3 \dim \Pi_{m-1,d} \dim \Pi_{m,d-1} \\ &= C\,(\dim \Pi_{m,d})^2 + \big(C_3 - 2C\big) \dim \Pi_{m-1,d} \dim \Pi_{m,d-1}.\end{aligned}$$

If we choose $C \geq \frac{1}{2}C_3$, then $(C_3 - 2C) \leq 0$, so the last term is non-positive. Hence

$$\mathcal{C}(m, d) \leq C\,(\dim \Pi_{m,d})^2.$$

By induction, this holds for all m, d, which proves (10). Since the algorithm really does perform at least a constant fraction of these operations, we in fact have the sharper statement

$$\mathcal{C}(m, d) = \Theta\big((\dim \Pi_{m,d})^2\big).$$

H.3 Complexity in terms of grid size N and dimension m:

We now connect the above complexity to a grid of data points. Suppose we have a regular m-dimensional grid with d nodes in each direction, so that the total number of grid points is

$$N := d^m. \tag{15}$$

In practice, we often think of N (the number of samples) and m (the dimension) as given, and wish to understand the cost as a function of these two quantities.

The bounds quoted in Fig. 1 and Prop. 4.4 (with $N = B$) come from the cost model of this appendix. In this model, the number of nodes per axis fixes the degree d, and we interpolate in the total-degree space $\Pi_{m,d}$. PIP-SOLVER evaluates F on its node set Ω_{PIP}, with $\#\Omega_{\mathrm{PIP}} = \dim \Pi_{m,d}$ (Theorem H.1). All bounds below count the cost of this interpolant.

Because d must be an integer, given m and N we define the largest admissible number of nodes per axis, and hence the degree, as

$$d(N, m) := \max\{d \in \mathbb{N} : d^m \leq N\} = \big\lfloor N^{1/m} \big\rfloor. \tag{16}$$

By definition, this implies

$$d(N, m)^m \;\leq\; N \;<\; \big(d(N, m) + 1\big)^m. \tag{17}$$

By (9), the corresponding polynomial-space dimension is

$$\dim \Pi_{m,d(N,m)} = \binom{m + d(N, m)}{d(N, m)}. \tag{18}$$

Proposition H.2 (Complexity as a function of m and N)**.** *Let $m \geq 1$ and $N \geq 1$ be given and define $d(N, m)$ as in* (16)*. Then the PIP-SOLVER algorithm with degree $d(N, m)$ has runtime*

$$\mathcal{C}\big(m, d(N,m)\big) = \Theta\big((\dim \Pi_{m,d(N,m)})^2\big) = \Theta\left(\binom{m + d(N,m)}{d(N,m)}^2\right). \tag{19}$$

Proof. By Theorem H.1, $\mathcal{C}\big(m, d(N,m)\big) = \Theta\big((\dim \Pi_{m,d(N,m)})^2\big)$. Substituting the binomial form (18) of $\dim \Pi_{m,d(N,m)}$ yields (19). □

Since $\binom{m+d}{d} = \binom{m+d}{m}$, (19) is the bound of Fig. 1. For intuition, dropping the floor gives the slightly looser but cleaner expression

$$\mathcal{C}\big(m, d(N,m)\big) \approx \mathcal{O}\left(\binom{m + N^{1/m}}{m}^2\right),$$

keeping in mind that the exact integer-valued bound is given by (19).

H.4 Asymptotics for fixed m and growing N:

In many applications, the dimension m is fixed while the grid is refined so that $N \to \infty$. We now show that in this regime the cost is essentially quadratic in the number of grid points.

Recall from (9) that

$$\dim \Pi_{m,d} = \binom{m+d}{d} = \frac{(d+1)(d+2)\cdots(d+m)}{m!}.$$

We factor out d^m:

$$\dim \Pi_{m,d} = \frac{d^m}{m!} \prod_{k=1}^{m} \left(1 + \frac{k}{d}\right). \tag{20}$$

For fixed m and $d \to \infty$, each factor $1 + \frac{k}{d} \to 1$. Using the Taylor expansion $\log(1+x) = x + \mathcal{O}(x^2)$ as $x \to 0$, we obtain

$$\log \prod_{k=1}^{m} \left(1 + \frac{k}{d}\right) = \sum_{k=1}^{m} \log\left(1 + \frac{k}{d}\right) = \mathcal{O}\left(\frac{1}{d}\right),$$

hence, by exponentiating,

$$\prod_{k=1}^{m} \left(1 + \frac{k}{d}\right) = 1 + \mathcal{O}\left(\frac{1}{d}\right).$$

Substituting this into (20) gives

$$\dim \Pi_{m,d} = \frac{d^m}{m!} \left(1 + \mathcal{O}\left(\frac{1}{d}\right)\right) \qquad (d \to \infty,\ m \text{ fixed}). \tag{21}$$

Now set $d = d(N, m)$. From (17) we have $d(N,m)^m \sim N$, so

$$d(N,m) = N^{1/m}\big(1 + o(1)\big) \qquad (N \to \infty).$$

Plugging this into (21) yields

$$\dim \Pi_{m,d(N,m)} = \frac{N}{m!}\big(1 + o(1)\big), \qquad N \to \infty,\ m \text{ fixed}. \tag{22}$$

Combining (22) with $\mathcal{C}\big(m, d(N,m)\big) = \Theta\big((\dim \Pi_{m,d(N,m)})^2\big)$ gives:

Corollary H.3 (Fixed m, growing grid)**.** *For fixed dimension m and $N \to \infty$,*

$$\mathcal{C}\big(m, d(N,m)\big) = \Theta\left(\frac{N^2}{(m!)^2}\right) = \Theta(N^2). \tag{23}$$

Thus, for fixed m the runtime is asymptotically quadratic in the number of grid points N.

H.5 Asymptotics for fixed N and growing m:

The complementary regime is to fix the grid budget $N > 1$ and let the dimension m increase. In this case we cannot keep d large: the number of nodes per axis, and hence the degree, must shrink, and eventually collapses to 1.

Lemma H.4 (Degree collapse for fixed budget)**.** *Fix $N > 1$ and define $d(N, m)$ by* (16)*. Then there exists a threshold*

$$m^\star(N) := \lfloor \log_2 N \rfloor + 1$$

such that the admissible degree satisfies

$$d(N, m) = 1 \qquad \textit{for all } m \geq m^\star(N).$$

Proof. Suppose $d(N, m) \geq 2$. Then $d(N, m)^m \geq 2^m$. By (17), we also have $d(N, m)^m \leq N$, hence $2^m \leq N$. Thus, if $2^m > N$ we *cannot* have $d(N, m) \geq 2$; the only possibility is $d(N, m) = 1$.

Choose $m^\star(N) := \lfloor \log_2 N \rfloor + 1$. Then $2^{m^\star(N)} > N$, so for all $m \geq m^\star(N)$ we have $2^m \geq 2^{m^\star(N)} > N$, forcing $d(N, m) = 1$. □

When $d(N, m) = 1$, the polynomial space is simply the affine functions

$$f(X_1, \ldots, X_m) = a_0 + \sum_{j=1}^{m} a_j X_j,$$

with dimension given by

$$\dim \Pi_{m,1} = \binom{m+1}{1} = m + 1. \tag{24}$$

Therefore, by Theorem H.1,

$$\mathcal{C}(m, 1) = \Theta\big((\dim \Pi_{m,1})^2\big) = \Theta\big((m+1)^2\big) = \Theta(m^2).$$

Combining this with Lemma H.4 yields:

Theorem H.5 (Fixed budget, increasing dimension)**.** *Fix a grid budget $N > 1$ and let $d(N, m)$ be defined as in* (16)*. Then the runtime of PIP-SOLVER satisfies*

$$\mathcal{C}\big(m, d(N, m)\big) = \Theta(m^2) \qquad \textit{for all } m \geq m^\star(N), \tag{25}$$

where $m^\star(N) = \lfloor \log_2 N \rfloor + 1$. In words, under a fixed grid budget, the interpolation cost grows only quadratically in the dimension m once m is large enough.

To summarize, in terms of the dimension m and the grid budget N:

- The *exact* runtime of multivariate Newton interpolation using PIP-SOLVER with degree $d(N, m)$ is given by

$$\mathcal{C}\big(m, d(N, m)\big) = \Theta\left(\binom{m + d(N, m)}{d(N, m)}^2\right), \quad d(N, m) = \lfloor N^{1/m} \rfloor.$$

- For fixed m and $N \to \infty$, the runtime is asymptotically quadratic in the grid size:

$$\mathcal{C}\big(m, d(N, m)\big) = \Theta(N^2).$$

- For fixed N and $m \to \infty$, the admissible degree collapses to 1 and the runtime is asymptotically quadratic in the dimension:

$$\mathcal{C}\big(m, d(N, m)\big) = \Theta(m^2) \quad (m \geq m^\star(N)).$$

I Adaptive provers and randomized layer selection

The layer selector of Sec. 4 solves Problem 4.5 once (App. D) and audits the set $S_\alpha \subseteq [\mathcal{L}]$ of layers covered by the resulting allocation under the α-score benefits ν_i. Such a *deterministic* selector audits layer ℓ with probability $p_\ell = \mathbf{1}[\ell \in S_\alpha] \in \{0, 1\}$. A prover that knows the policy therefore knows S_α, and a deviation placed outside S_α is never examined, regardless of its effect on the output. Here, we detail the randomized selector $\mathcal{S}_\gamma$ of Sec. 4, its worst-case coverage floor, and the adaptive γ-sweep summarized in Sec. 5.

Threat Model: We strengthen the adversary of App. B. Besides tampering with the computation (but not the cryptography), the prover knows the selection policy $\mathcal{S}_\gamma$ including γ, the α-scores, and the budgets $\beta_1, \ldots, \beta_M$. Since the Terkle root is published before the verifier draws its selection, the prover knows the distribution of the audited set, but not its realization. It places its deviation where it is least likely to be examined, i.e., outside S_α under deterministic selection ($\gamma = 1$). Under uniform selection ($\gamma = 0$), which may examine any layer, it targets the highest-impact layers.

Randomized Selector: Let $\mathcal{U}$ be the set of feasible allocations of Problem 4.5, i.e., assignments of at most one contiguous interval $[s, o] \subseteq [\mathcal{L}]$ to each verifier $k \in [M]$ with $\sum_{i=s}^{o} \phi_i \leq \beta_k$ and no overlap across verifiers. $\mathcal{U}$ is finite and nonempty (it contains the empty allocation), and we write $\ell \in U$ if allocation $U \in \mathcal{U}$ covers layer ℓ. For a mixing weight $\gamma \in [0, 1]$ (unrelated to the ILP variables $\gamma_{k,i}$), $\mathcal{S}_\gamma$ uses fresh randomness, drawn after the Terkle root is published, to return S_α with probability γ and otherwise the layers covered by $U \sim \mathrm{Unif}(\mathcal{U})$. Thus, $\gamma = 1$ recovers the deterministic α-selector and $\gamma = 0$ the uniform one. We define

$$c_\ell := \Pr_{U \sim \mathrm{Unif}(\mathcal{U})}\big[\ell \in U\big] = \frac{\big|\{U \in \mathcal{U} : \ell \in U\}\big|}{|\mathcal{U}|}, \qquad c := \min_{\ell \in [\mathcal{L}]} c_\ell.$$

Proposition I.1 (Coverage floor)**.** *For every layer $\ell \in [\mathcal{L}]$, the probability that $\mathcal{S}_\gamma$ audits ℓ is*

$$p_\ell = \gamma\, \mathbf{1}[\ell \in S_\alpha] + (1 - \gamma)\, c_\ell \;\geq\; (1 - \gamma)\, c.$$

Moreover, $c > 0$ if and only if every layer belongs to at least one feasible allocation, i.e., $\phi_\ell \leq \max_{k \in [M]} \beta_k$ for all $\ell \in [\mathcal{L}]$.

Proof of Proposition I.1. By the law of total probability over the mixing coin, $p_\ell = \gamma \Pr[\ell \in S_\alpha] + (1 - \gamma) \Pr_{U \sim \mathrm{Unif}(\mathcal{U})}[\ell \in U] = \gamma\, \mathbf{1}[\ell \in S_\alpha] + (1 - \gamma)\, c_\ell$, since S_α is fixed. The first term is nonnegative and $c_\ell \geq c$, which gives the floor. If $\phi_\ell \leq \beta_k$ for some k, assigning the single-layer interval $[\ell, \ell]$ to verifier k and no interval to the others is feasible, so $c_\ell \geq 1/|\mathcal{U}| > 0$. If this holds for every ℓ, then $c > 0$. Conversely, if a feasible allocation assigns an interval $[s, o] \ni \ell$ to verifier k, then $\phi_\ell \leq \sum_{i=s}^{o} \phi_i \leq \beta_k$, since all $\phi_i > 0$. Hence, if $\phi_\ell > \max_k \beta_k$ for some ℓ, no feasible allocation covers ℓ, and $c \leq c_\ell = 0$. □

The floor depends only on γ and, through c, on the costs ϕ_i and budgets β_k, not on the realized selection, which is drawn after the prover has committed. Hence, it holds even when the prover knows the policy, the α-scores, and the budgets. No choice of the perturbed layer yields $p_\ell < (1 - \gamma)c$. The argument only uses the coverage probabilities of the randomized component, so the same floor holds for any other distribution over $\mathcal{U}$, with c_ℓ defined under that distribution. Since the single-layer bound of Prop. 4.6 is non-increasing in p_ℓ, substituting the floor gives, for a deviation confined to layer ℓ,

$$\Pr[\mathcal{V} \text{ accepts } T] \;\leq\; 1 - (1 - \gamma)\, c \left[1 - (1 - \rho_\ell)^k\right] + \mathrm{negl}(\lambda),$$

with ρ_ℓ and the number k of audited coordinates as in Prop. 4.6, whichever layer the prover chooses. At $\gamma = 1$, the floor vanishes, and every $\ell \notin S_\alpha$ has $p_\ell = 0$ and a vacuous bound.

Adaptive γ-Sweep: Table 7 sweeps $\gamma \in \{0, 0.25, 0.5, 0.75, 1\}$ against the policy-aware prover above. Over all attempted attacks, we report the fraction that is missed and changes the output (harmful miss) and the fraction that is missed, together with the share of missed attacks that change the output.

Interpretation: In every column, the first row is the product of the other two (up to rounding). At $\gamma = 0$, 19.82% of attempted attacks are missed and 53.6% of those change the output, i.e., a harmful-miss rate of 10.62%. At $\gamma = 1$, every perturbation deliberately placed outside S_α is missed (100%, as $p_\ell = 0$ there), yet only 3.98% of them alter the output under this construction. As γ grows and the selector concentrates on S_α, raw misses rise from 19.82% to 100%, while the share of

Table 7: **How does randomized selection fare against a policy-aware prover?** Adaptive γ-sweep of $\mathcal{S}_\gamma$ ($\gamma=1$ gives the deterministic α-selector S_α, and $\gamma=0$ a uniformly sampled feasible allocation) against a prover that knows the policy, the α-scores, and the budgets. The first two rows are fractions of all attempted attacks. The last row is a fraction of the missed attacks. Concentrating on S_α lowers the harmful-miss rate from **10.62%** to **3.98%**, but $\gamma=1$ misses every perturbation outside S_α.

γ (1 = α-selector, 0 = uniform random)	0.00	0.25	0.50	0.75	1.00
Missed and output changed	**10.62%**	6.80%	5.38%	4.16%	**3.98%**
Missed perturbation	19.82%	22.86%	35.12%	62.70%	100%
Share of misses that changed the output	53.6%	29.7%	15.3%	6.63%	3.98%

output-changing misses falls from 53.6% to 3.98%, so the harmful-miss rate decreases from 10.62% to 3.98%. Hence, γ trades the average harmful-miss rate, which is lowest at $\gamma = 1$ for this attack family, against the worst-case floor $(1-\gamma)c$, which is zero at $\gamma = 1$. An output-changing deviation that the prover places in a layer outside S_α is never audited by the deterministic selector, whereas any $\gamma < 1$ audits every layer with probability at least $(1-\gamma)c$, which is positive whenever $c > 0$, regardless of the attack.

Scope: These results are evidence for this attack family, not a proof against arbitrary adaptive provers. For layer selection, our worst-case guarantee is the floor of Prop. I.1. Likewise, AMC (Sec. 5) averages coverage over the attack distribution $\mathcal{Q}$ and is not a worst-case guarantee. We therefore pair AMC with the per-layer minimum coverage $\min_\ell p_\ell$, which is 0 for the deterministic selector whenever $S_\alpha \neq [\mathcal{L}]$, and at least $(1-\gamma)c$ under $\mathcal{S}_\gamma$.

J Full deployment overhead and communication

Table 1 reports the prover time post-inference and the verification time. Here we itemize the full deployment cost and the bytes behind the per-token entry, for the same LLaMA2-13B configuration (A100, 10.165 s inference, 512 tokens per inference).

Stage-by-stage cost: Table 8 lists every stage. The trusted setup generates the srs once, so its 80.6 MB is a one-time cost amortized over all inferences. Activation capture is retention only. The prover keeps the residual-stream states that the forward pass already computes. Device-to-host transfer and the reshape into a hypercube are the memory movement, while interpolation and quotient formation dominate the prover. The verifier spends 3.2 ms on structural checks of the openings along the Terkle path and 8.99 ms on the arithmetic audit (App. M). The audit is included in the published timing (12 ms in Table 1). Only the inference itself is excluded. Hence, per inference, the prover and verifier add 0.97% and 0.12% to the 10.165 s inference.

Table 8: **What does a deployment cost end to end?** Full breakdown for the LLaMA2-13B configuration of Table 1. All times exclude only the inference itself.

Party	**Stage**	**Cost**
Trusted setup	srs, generated once and amortized	80.6 MB
Prover	Activation capture	retention only
	Device-to-host transfer	7.1 ms
	Reshape to hypercube	1.6 ms
	Interpolation	47.8 ms
	Commitments	3.5 ms
	Terkle-tree construction	6.3 ms
	Quotient formation	29.1 ms
	Proof from quotient	3.2 ms
	Prover total	**98.6 ms**
Verifier	Structural checks	3.2 ms
	Arithmetic audit	8.99 ms
	Verifier total	**12.19 ms**

Why 2 B per token? All baselines in Table 1 report commitment size per token, and each commits to 512 tokens per inference. We report TC in the same way for a like-for-like comparison. TC, however, exchanges one Terkle root per inference rather than one commitment per token. Table 9 lists the commitment and authentication material exchanged per inference on BLS12-381, where a compressed group element takes 48 B (App. E.3) and a field element 32 B. The total is 880 B per inference, i.e., $880/512 \approx 2$ B per generated token, the entry of Table 1. This figure includes the membership path and the leaf opening.

Table 9: **What is exchanged per inference?** Byte-level accounting behind the 2 B/token entry of Table 1 on BLS12-381.

Item	Count	Bytes
Terkle root, once per inference	1 group element	48
Membership path, including leaf opening	16 group elements	768
Claimed evaluation y	1 field element	32
Challenge seed	-	32
Total per inference		**880**
Per generated token	$\div 512$	$\approx$ **2**

Three communication quantities: (i) The *root only* is 48 B, the single group element the prover publishes before the challenge. (ii) The *commitment and authentication material* is 880 B per inference (Table 9) and underlies the per-token entry of Table 1. (iii) The *full verification traffic* is (ii) plus the arithmetic-audit transcript (App. M), which the per-token entry does not include.

Sequence length: The exchanged root does not grow with sequence length. Additional states are represented by increasing the tensor order or are incorporated under the existing Terkle root, which remains one group element. Authentication-path length and verification grow as $\mathcal{O}(\log_B n)$ for n committed items and arity B. All numbers here are for the 512-token configuration of Table 1.

K Provider-side cost-reduction attacks

In Fig. 7, TOPLOC's Taco attacks come from its own suite, where both methods perform near ceiling, and our tailored attacks (App. B) target last-layer fingerprints. Neither family models a provider that cuts its serving cost while billing for the claimed model. We therefore add a separate evaluation, targeted at our own selection rule, with three attacks that model provider-side cost reduction. Let block ℓ update the residual stream as $T_\ell = T_{\ell-1} + F_\ell(T_{\ell-1}; W_\ell)$, with residual branch F_ℓ and embedded input T_0, and let $\xi \subseteq [\mathcal{L}]$ be the attacked blocks, unknown to the verifier.

Low-Rank Substitution replaces layers with truncated-SVD approximations. Each weight matrix $W = \sum_i \sigma_i \mathbf{u}_i \mathbf{v}_i^\top$ of a block $\ell \in \xi$, with singular values $\sigma_1 \geq \sigma_2 \geq \cdots$, is served as

$$\tilde{W} := \sum_{i=1}^{\mathrm{rk}} \sigma_i \, \mathbf{u}_i \mathbf{v}_i^\top, \qquad \|\tilde{W} - W\|_2 = \sigma_{\mathrm{rk}+1},$$

so a product with $\tilde{W} \in \mathbb{R}^{d_{\mathrm{out}} \times d_{\mathrm{in}}}$ costs $\mathcal{O}(\mathrm{rk}\,(d_{\mathrm{in}} + d_{\mathrm{out}}))$ instead of $\mathcal{O}(d_{\mathrm{in}} d_{\mathrm{out}})$ per token.

Layer Skipping bypasses transformer blocks, $\tilde{T}_\ell := \tilde{T}_{\ell-1}$ for $\ell \in \xi$, which drops the residual branch F_ℓ and its cost.

Precision Downgrade serves INT4 computation as FP16. Blocks $\ell \in \xi$ are computed as $\tilde{T}_\ell := \tilde{T}_{\ell-1} + F_\ell^{\mathrm{INT4}}(\tilde{T}_{\ell-1}; W_\ell)$ with 4-bit integer arithmetic, while the run is reported as FP16.

The rank rk and the set ξ are the attacker's choice. In all three cases, the committed state of an attacked block violates the relation $\tilde{T}_\ell = \tilde{T}_{\ell-1} + F_\ell(\tilde{T}_{\ell-1}; W_\ell)$, which an arithmetic audit recomputes with the weights the verifier holds (App. M). The attack is caught when an attacked block is selected and an audited coordinate deviates by more than ϵ_{tol}. TOPLOC instead compares a last-layer top-k fingerprint against fixed thresholds. As shown in Table 10, TC detects 92%, 99%, and 90% of these attacks, against 73%, 97%, and 62% for TOPLOC. Both methods detect layer skipping. The gap is largest for the two attacks that approximate rather than remove a block: precision downgrade (+28 points) and low-rank substitution (+19 points).

Table 10: **Can a provider cut cost unnoticed?** Detection accuracy under cost-reduction attacks.

Method	Low-rank substitution	Layer skipping	Precision downgrade
TOPLOC [30]	73%	97%	62%
TensorCommitments	**92%**	**99%**	**90%**

L Beyond cloud inference: deployment settings

The setting of Sec. 3 is one instance of a general one: *one party pays for computation it cannot observe*. In each case below, the party that performs the computation commits to the tensors it produces and acts as $\mathcal{P}$, and the party that cannot observe it acts as $\mathcal{V}$.

Why cost decides: Verification cost decides whether verification happens at all. Inside an agent, a wrong answer becomes a tool call, a database write, and a state that later steps treat as ground truth. With 803 s per ZK proof, or a verifier holding a 71 GB GPU to re-run the model (Table 1), checking is a deliberate exception. At $\approx 0.1\%$ verifier and $<1\%$ prover cost (App. J), checking can be the default, and every inference can leave a verifiable record. This marks a shift from occasional proofs to *continuous verification*.

Pay-Per-Inference Compute Markets: A client paying for remote inference needs evidence of the billed work, bound to that run. The Terkle root, published with the response before the challenge, fixes the billed transcript, and the audits check sampled relations against the claimed model.

Chained and Multi-Party Inference: A request is routed through several compute providers. Each stage commits to its segment (e.g., a range of blocks), and the chain of roots attests the pipeline end to end. Consecutive stages can be linked through the hand-off state, which both commit to. Since $\mathsf{Com}_{\mathsf{TC}}$ is deterministic for fixed pp, shape, and grid, honest stages produce the same commitment for it. A mismatch localizes a fault to one hand-off, and a failed audit to the stage whose root it was checked against, giving per-stage accountability.

Data-Provider Attestation: A provider unwilling to reveal its data can still prove that its data was the input to a given run, by opening only the input-side commitment under the run's Terkle root and showing that it equals the commitment of its data. When the input combines several contributions, additive homomorphism (App. M) checks the inclusion of a contribution directly on commitments, without transmitting the contribution.

Federated Aggregation: A cloud aggregator commits to the aggregate, and edge devices verify that their own update was included and not silently dropped, via the homomorphic inclusion check below. Terkle Trees keep these per-device queries local and cheap.

Agentic and Multi-Turn Pipelines: Long-horizon agent state is what Terkle Trees were built for: many evolving tensors under a single root, with cheap localized queries over a chat span or an agent's memory region (Sec. 5).

Homomorphic inclusion check: Let n participants hold updates $T_1, \dots, T_n$ of the same shape, encoded in $\mathbb{F}$ with a common fixed-point scale and committed with the same pp on the same grid $\mathbf{\Omega}$ as $C_i := \mathsf{Com}_{\mathsf{TC}}(\mathsf{pp}, T_i)$. The aggregator publishes $C_{\text{agg}} := \mathsf{Com}_{\mathsf{TC}}(\mathsf{pp}, T_{\text{agg}})$ for $T_{\text{agg}} := \sum_{i=1}^{n} T_i$. The polynomial $\sum_i f_{T_i}$ has degree below d_j in each X_j and equals T_{agg} on $\mathbf{\Omega}$, so uniqueness of interpolation gives $f_{T_{\text{agg}}} = \sum_i f_{T_i}$. As $\mathsf{Com}_{\mathsf{TC}}$ is linear in the exponent,

$$C_{\text{agg}} = g^{f_{T_{\text{agg}}}(\tau_1,\dots,\tau_m)} = g^{\sum_{i=1}^{n} f_{T_i}(\tau_1,\dots,\tau_m)} = \prod_{i=1}^{n} g^{f_{T_i}(\tau_1,\dots,\tau_m)} = \prod_{i=1}^{n} C_i.$$

The cloud or any later participant checks $C_{\text{agg}} \stackrel{?}{=} \prod_i C_i$ with $n-1$ group multiplications, on commitments only, without seeing any T_i. Public weights $w_i \in \mathbb{F}$ give $C_{\text{agg}} = \prod_i C_i^{w_i}$ in the same way. An aggregate other than $\sum_i T_i$ passes only if the aggregator holds a second tensor with the same commitment, which breaks evaluation binding (App. F). In particular, dropping a nonzero T_k passes only if $f_{T_k}(\tau_1, \dots, \tau_m) = 0$, which by the Schwartz-Zippel lemma has probability at most $\sum_j (d_j - 1)/|\mathbb{F}|$ over the trapdoors. With the C_i as leaves of a Terkle Tree under the aggregator's root, device i confirms that its C_i is among them with one membership path of $\mathcal{O}(\log_B n)$ openings. The check certifies the aggregation, not that each T_i is an honest update.

Mixture-of-Experts (MoE): An MoE LLM routes each token to a subset of experts, so an untrusted provider could silently substitute a cheaper expert or skip routing. We commit to both the routing decisions and the per-expert activations, so a substituted expert or a skipped routing step becomes a violated committed relation that the audits can catch. We use Mixtral 8×7B [15] with 2 and with 4 of its 8 experts active per token.

Federated Learning (FL): Each participant commits to its local update, and the cloud and later participants verify aggregation consistency with the inclusion check above, without seeing raw gradients. We use Llama3-8B [11] on MIMIC-IV [17] with LoRA [14] (rank 128) across 8 nodes, and AlexNet [22] on CIFAR-100 [21] across 50 nodes.

Table 11: **Does TC carry over to other settings?** Prover and verification time for MoE inference and FL aggregation.

Setting	Model	Prover (ms)	Verification (ms)
MoE	Mixtral 8×7B, 2/8 active experts	94.7	11.6
MoE	Mixtral 8×7B, 4/8 active experts	148	21.8
FL	Llama3-8B on MIMIC-IV, 8 nodes, LoRA rank 128	37.9	4.14
FL	AlexNet on CIFAR-100, 50 nodes	24.2	3.08

Mixtral holds 47B parameters, yet its prover time (94.7 ms) is below the 98.6 ms of LLaMA2-13B (Table 8), because only the active experts contribute activations. Doubling the active experts raises it to 148 ms, a 1.6× increase for 2× the routed state. Cost thus follows the state a run actually exposes, not only the parameters it holds. The FL rows are reported in absolute terms, since a single-inference baseline is not the relevant denominator there.

M Homomorphic and arithmetic audits

The verifier of Sec. 3 combines binding, homomorphic audits, arithmetic audits, and coverage. Binding is proved in App. F. Here we detail the two audits in the notation of Sec. 4: pairing $e : G \times G \to G_T$, generator g, trapdoors $\tau_1, \ldots, \tau_m$, commitments $C_T = g^{f_T(\tau_1,\ldots,\tau_m)}$, and opening proofs $\pi_{\boldsymbol{\omega}} = (\pi_{\omega_1}, \ldots, \pi_{\omega_m})$, accepted iff $e(C_T\, g^{-y}, g) = \prod_{j=1}^{m} e(\pi_{\omega_j}, g^{\tau_j - \omega_j})$.

Additive homomorphism: Let T_1, T_2 have the same shape, be committed with the same pp on the same grid $\boldsymbol{\Omega}$, and let $a \in \mathbb{F}$. Interpolation is linear. The polynomials $f_{T_1} + f_{T_2}$ and $a f_{T_1}$ have degree below d_j in each X_j and match $T_1 + T_2$ and aT_1 on $\boldsymbol{\Omega}$, so by uniqueness $f_{T_1+T_2} = f_{T_1} + f_{T_2}$ and $f_{aT_1} = a f_{T_1}$. Hence,

$$C_{T_1+T_2} = g^{f_{T_1}(\tau_1,\ldots,\tau_m)+f_{T_2}(\tau_1,\ldots,\tau_m)} = C_{T_1} \cdot C_{T_2}, \qquad C_{aT_1} = C_{T_1}^{a}.$$

Openings combine in the same way. If $(y_1, \pi_{\boldsymbol{\omega}}^{(1)})$ and $(y_2, \pi_{\boldsymbol{\omega}}^{(2)})$ are valid openings of C_{T_1} and C_{T_2} at the same $\boldsymbol{\omega}$, multiplying their verification equations and using bilinearity gives

$$e\big(C_{T_1} C_{T_2}\, g^{-(y_1+y_2)},\, g\big) = \prod_{j=1}^{m} e\big(\pi_{\omega_j}^{(1)} \pi_{\omega_j}^{(2)},\, g^{\tau_j - \omega_j}\big),$$

so $y_1 + y_2$, with the component-wise product of the two proofs, is a valid opening of $C_{T_1} C_{T_2}$. A public linear relation $\sum_i a_i T_i = T'$ among committed tensors is thus checked as $\prod_i C_{T_i}^{a_i} \stackrel{?}{=} C_{T'}$ directly on commitments, with one exponentiation per term regardless of the tensor size. The inclusion check of App. L is the case $a_i = 1$. If this check passes while $T' \neq \sum_i a_i T_i$, a prover that knows these tensors holds two different tensors with one commitment, and opening both at a grid point where they differ breaks evaluation binding (App. F).

Multiplicative structure: Bilinearity, $e(g^a, g^b) = e(g, g)^{ab}$, makes products of committed exponents testable in G_T. Given g^a, g^b, and a claimed $g^{\widetilde{ab}}$, the test $e(g^a, g^b) = e(g^{\widetilde{ab}}, g)$ passes iff $\widetilde{ab} = ab$, since $e(g, g)$ generates G_T (non-degeneracy, App. E.1). The opening check is itself such a test. Its two sides carry the exponents $f_T(\tau_1, \ldots, \tau_m) - y$ and $\sum_{j=1}^{m} q_j(\tau_j, \ldots, \tau_m)(\tau_j - \omega_j)$, a sum of products of the committed quotients with the secret factors $\tau_j - \omega_j$, whose group elements $g^{\tau_j - \omega_j} = g^{\tau_j} g^{-\omega_j}$ come from the srs. Each pairing tests one product of two exponents, so a degree-two relation among

committed quantities is checked with $\mathcal{O}(1)$ pairings, while higher-degree relations need committed intermediate products. Note that $e(C_{T_1}, C_{T_2}) = e(g,g)^{(f_{T_1} f_{T_2})(\tau_1,\ldots,\tau_m)}$ tests the product polynomial $f_{T_1} f_{T_2}$, of degree up to $2(d_j - 1)$ in X_j. It agrees with the entry-wise product $T_1 \odot T_2$ on $\mathbf{\Omega}$ but is not its interpolant $f_{T_1 \odot T_2}$, so comparing it with a committed product needs an srs of that degree.

Arithmetic audit: After the Terkle root is fixed, $\mathcal{V}$ derives the challenged entries from the challenge seed (Table 9). Let $y = T[\boldsymbol{\omega}]$ be a challenged entry of a committed state T, and let the model compute it from the entries of a committed input state T' on an index block, i.e., the sub-grid of $\mathbf{\Omega}$ on which the coordinates in $J \subseteq [m]$ are fixed to $\boldsymbol{\omega}'_J$ and those in $\bar{J} := [m] \setminus J$ are free (e.g., the previous state of the same token for a position-wise sublayer such as the MLP, or of all tokens for attention, which mixes tokens). Here, $\boldsymbol{\omega}'_J := (\omega'_j)_{j \in J}$, $X_{\bar{J}} := (X_j)_{j \in \bar{J}}$, and $f_{T'}(\boldsymbol{\omega}'_J, X_{\bar{J}})$ is $f_{T'}$ with $X_j = \omega'_j$ for all $j \in J$. Points and entries are split over J and $\bar{J}$ in the same way. The audit runs as follows, with C_T and $C_{T'}$ authenticated against the root by their membership paths:

1. $\mathcal{P}$ opens y against C_T and sends, as coefficients, a polynomial $\tilde{f}(X_{\bar{J}})$ with degree below d_j in each X_j, claimed to equal the restriction $f_{T'}(\boldsymbol{\omega}'_J, X_{\bar{J}})$ of the committed polynomial to the block.
2. $\mathcal{V}$ draws $\mathbf{t} \in \mathbb{F}^{|\bar{J}|}$ uniformly at random and checks one opening of $C_{T'}$ at the point $(\boldsymbol{\omega}'_J, \mathbf{t})$ against the value $\tilde{f}(\mathbf{t})$.
3. $\mathcal{V}$ evaluates $\tilde{f}$ on the grid points of the block, which gives the inputs $T'[\boldsymbol{\omega}'_J, \cdot\,]$, recomputes the entry $\hat{y}$ from them with the weights $\theta_{\mathcal{M}}$ it holds, and accepts iff $|\hat{y} - y| \leq \epsilon_{\mathrm{tol}}$.

What is checked: By binding, the value opened at $(\boldsymbol{\omega}'_J, \mathbf{t})$ is $f_{T'}(\boldsymbol{\omega}'_J, \mathbf{t})$, except with probability $\mathrm{negl}(\lambda)$. Since $\mathbf{t}$ is drawn after $\tilde{f}$ is fixed, a restriction $\tilde{f} \neq f_{T'}(\boldsymbol{\omega}'_J, \cdot)$ passes step 2 only if the nonzero polynomial $\tilde{f} - f_{T'}(\boldsymbol{\omega}'_J, \cdot)$, of total degree at most $\sum_{j \in \bar{J}}(d_j - 1)$, vanishes at $\mathbf{t}$, which by the Schwartz-Zippel lemma has probability at most $\sum_{j \in \bar{J}}(d_j - 1)/|\mathbb{F}|$. Step 3 then checks one model relation, the challenged entry against the model applied to the committed inputs, up to ϵ_{tol}. The tolerance absorbs the fact that the verifier's recomputation need not reproduce the prover's floating-point arithmetic bit for bit. Real-valued activations enter $\mathbb{F}$ through a common fixed-point encoding, as in App. L, and $T_\ell(x)$ in Sec. 3 denotes this encoding. *What is not checked:* unsampled coordinates and unselected layers, deviations within ϵ_{tol}, and the correctness of the committed inputs themselves, which is checked only when their own relation is sampled.

Coverage: Binding alone does not establish provenance. A prover can commit consistently to an incorrect transcript, e.g., one produced by a different model, and binding only prevents it from changing that transcript after the challenge. Such a transcript is rejected only if a violated relation is selected and an affected coordinate is audited, homomorphically or arithmetically. For a single violated layer ℓ, selected with probability p_ℓ, with a fraction ρ_ℓ of its auditable coordinates violating the relation by more than ϵ_{tol}, and k coordinates sampled uniformly and independently, this yields the acceptance bound $1 - p_\ell[1 - (1 - \rho_\ell)^k] + \mathrm{negl}(\lambda)$ of Prop. 4.6, where $p_\ell \geq (1 - \gamma)c$ under the randomized selector $\mathcal{S}_\gamma$ (Sec. 4). The guarantee is thus over the examined relations, not a deterministic certificate of the complete execution as in ZK proofs. A deviation that leaves the output unchanged is not an output-soundness failure, though it may violate stronger computation integrity. We target violations that change the output, not every perturbation.

N Proof of the output-soundness bound

We prove Proposition 4.6. Throughout, we use the protocol of Sec. 3 and the audits of App. M.

Setting: For pp from an honest $\mathsf{Setup}_{\mathsf{TC}}$, the PPT prover of the Threat Model (Sec. 3) commits with $\mathsf{Com}_{\mathsf{TC}}$ to a transcript $T = (T_1, \ldots, T_{\mathcal{L}})$, the activation tensors of its run, which it knows and which does not depend on pp, and publishes the Terkle root before any challenge. Probabilities are over pp, the prover's coins, and the challenge. If T may depend on pp, the proof below bounds the average instead, $\Pr[\mathcal{V} \text{ accepts}] \leq 1 - \mathbb{E}[\delta(T)] + \mathrm{negl}(\lambda)$. Only afterwards does $\mathcal{V}$ draw its challenge with fresh randomness. This challenge consists of a set $S \subseteq [\mathcal{L}]$ of layers from the layer-selection policy, with $p_\ell = \Pr[\ell \in S]$, and, for each selected layer, k of its auditable coordinates, drawn uniformly at random and independently of S. The auditable coordinates of layer ℓ are the grid points of T_ℓ whose entries the model computes from committed inputs, as in the arithmetic audit of App. M. For such a coordinate $\boldsymbol{\omega}$ of layer ℓ, the relation error $\mathrm{err}_\ell(\boldsymbol{\omega})$ is the absolute difference between the committed entry $T_\ell[\boldsymbol{\omega}]$ and the entry that the model computes from the committed inputs on which it

depends. Layer ℓ is violated if $\rho_\ell > 0$, where ρ_ℓ is the fraction of its n_ℓ auditable coordinates with $\mathrm{err}_\ell(\boldsymbol{\omega}) > \epsilon_{\mathrm{tol}}$, and T is incorrect if some layer is violated. Each sampled coordinate is audited as in App. M, and $\mathcal{V}$ accepts only if every opening, every random-point check, and every recomputation passes. The assumption that sampled relations are checked correctly up to ϵ_{tol} means that, whenever the opened values and the restriction used in an audit are those of T, the recomputation accepts coordinate $\boldsymbol{\omega}$ if and only if $\mathrm{err}_\ell(\boldsymbol{\omega}) \le \epsilon_{\mathrm{tol}}$. The number n_{op} of openings and audits in one verification and the degrees d_j are polynomial in λ, and $|\mathbb{F}| \ge 2^\lambda$. The honest-verifier ratio of the Limitations (Sec. 5) is the fraction of the M verifiers of Problem 4.5 that audit their assigned layers honestly. The proof below assumes that this ratio is one. Otherwise, a layer assigned to a dishonest verifier counts as unselected, and p_ℓ is the probability that ℓ is selected and assigned to an honest verifier.

Proof of Proposition 4.6. Let Hit be the event that the challenge contains a coordinate $(\ell, \boldsymbol{\omega})$ with $\mathrm{err}_\ell(\boldsymbol{\omega}) > \epsilon_{\mathrm{tol}}$. By definition, $\Pr[\mathrm{Hit}] = \delta(T)$. Let Bad be the event that $\mathcal{V}$ accepts an opening whose value differs from the corresponding value of T, or that a restriction $\tilde{f}$ different from the corresponding restriction of T passes its random-point check.

Step 1: $\Pr[\mathrm{Bad}] \le \mathrm{negl}(\lambda)$. (a) Suppose $\mathcal{V}$ accepts an opening of a commitment $C = \mathsf{Com}_{\mathsf{TC}}(\mathsf{pp}, T')$, where T' is a tensor of T, at a point $\boldsymbol{\omega}$ with value $y' \neq f_{T'}(\boldsymbol{\omega})$. As the prover knows T', it also obtains the honest opening $(f_{T'}(\boldsymbol{\omega}), \pi_{\boldsymbol{\omega}}) = \mathsf{Open}_{\mathsf{TC}}(\mathsf{pp}, T', \boldsymbol{\omega})$, which verifies by correctness (Eq. (2)). Two accepted openings of C at $\boldsymbol{\omega}$ with distinct values contradict evaluation binding (App. F), which holds at grid points and at verifier-sampled points. Hence, each opening is inconsistent with T with probability at most $\mathrm{negl}(\lambda)$. The same holds, by induction from the root, for the openings along the Terkle membership paths, which are TC openings of the node commitments, provided each child commitment enters its parent's tensor through a collision-resistant map $G \to \mathbb{F}$ (e.g., a hash of its compressed encoding), whose collisions add $\mathrm{negl}(\lambda)$. In Defs. 2.8 and 4.2, an entry $z_{u,j}$ or $\mathbf{Z}_u[\boldsymbol{k}]$ that is a child commitment stands for its image under this map. Hence, the commitments used in the audits are those of T. (b) Suppose the prover sends a restriction $\tilde{f} \neq f_{T'}(\boldsymbol{\omega}'_J, \cdot)$. Its random-point check passes only if the value opened at $(\boldsymbol{\omega}'_J, \mathbf{t})$ equals $\tilde{f}(\mathbf{t})$. Unless (a) occurs, that value is $f_{T'}(\boldsymbol{\omega}'_J, \mathbf{t})$, so the check passes only if the nonzero polynomial $\tilde{f} - f_{T'}(\boldsymbol{\omega}'_J, \cdot)$, of total degree at most $D_J := \sum_{j \in \bar{J}} (d_j - 1)$, vanishes at $\mathbf{t}$. Since $\mathbf{t}$ is uniform on $\mathbb{F}^{|\bar{J}|}$ and drawn after $\tilde{f}$ is fixed, the Schwartz-Zippel lemma bounds this probability by $D_J/|\mathbb{F}|$. (c) By a union bound over the at most n_{op} openings and audits, $\Pr[\mathrm{Bad}] \le n_{\mathrm{op}}\big(\mathrm{negl}(\lambda) + \max_J D_J/|\mathbb{F}|\big) = \mathrm{negl}(\lambda)$, as n_{op} and D_J are polynomial in λ and $|\mathbb{F}| \ge 2^\lambda$.

Step 2: $\Pr[\mathcal{V}\ \textit{accepts}\ T] \le 1 - \delta(T) + \mathrm{negl}(\lambda)$. On $\mathrm{Hit} \cap \neg\mathrm{Bad}$, the challenge contains a coordinate $(\ell, \boldsymbol{\omega})$ with $\mathrm{err}_\ell(\boldsymbol{\omega}) > \epsilon_{\mathrm{tol}}$. If an opening or the random-point check of its audit fails, $\mathcal{V}$ rejects. Otherwise, by $\neg\mathrm{Bad}$, every opened value and restriction used in its audit is that of T, and by the correct-checking assumption the recomputation rejects $\boldsymbol{\omega}$, so $\mathcal{V}$ rejects. Hence, $\mathcal{V}$ accepts only on $\neg\mathrm{Hit} \cup \mathrm{Bad}$, and

$$\Pr[\mathcal{V}\ \text{accepts}\ T] \le \Pr[\neg\mathrm{Hit}] + \Pr[\mathrm{Bad}] \le 1 - \delta(T) + \mathrm{negl}(\lambda).$$

Step 3: $\delta(T) \ge p_\ell[1 - (1 - \rho_\ell)^k]$ *for every violated layer* ℓ. Let $n_\ell^{\mathrm{bad}} = \rho_\ell n_\ell$ be the number of auditable coordinates of ℓ with error above ϵ_{tol}, and let $\mathrm{Hit}_\ell \subseteq \mathrm{Hit}$ be the event that $\ell \in S$ and at least one of the k coordinates sampled in ℓ has error above ϵ_{tol}. Since the coordinates are drawn independently of S, $\Pr[\mathrm{Hit}_\ell] = p_\ell(1 - \mathrm{miss}_\ell)$, where miss_ℓ is the probability that all k sampled coordinates have error at most ϵ_{tol}. For draws with replacement, $\mathrm{miss}_\ell = (1 - \rho_\ell)^k$. Without replacement ($k \le n_\ell$),

$$\mathrm{miss}_\ell = \prod_{i=0}^{k-1} \frac{n_\ell - n_\ell^{\mathrm{bad}} - i}{n_\ell - i} \le \Big(\frac{n_\ell - n_\ell^{\mathrm{bad}}}{n_\ell}\Big)^k = (1 - \rho_\ell)^k,$$

since $(n_\ell - n_\ell^{\mathrm{bad}})(n_\ell - i) - n_\ell(n_\ell - n_\ell^{\mathrm{bad}} - i) = i\, n_\ell^{\mathrm{bad}} \ge 0$ bounds every factor by $1 - \rho_\ell$, and $\mathrm{miss}_\ell = 0$ once a factor vanishes. In both cases, $\delta(T) = \Pr[\mathrm{Hit}] \ge \Pr[\mathrm{Hit}_\ell] \ge p_\ell[1 - (1 - \rho_\ell)^k]$. Substituting into Step 2 gives the second inequality of Proposition 4.6, for every violated layer ℓ. □

The bound holds for each transcript T, including one chosen with knowledge of the selection policy, because the challenge is drawn with fresh randomness after the root fixes T. Under the randomized selector, every layer has $p_\ell \ge (1 - \gamma)c$ (App. I), so the right-hand side is at most $1 - (1 - \gamma)c[1 - (1 - \rho_\ell)^k] + \mathrm{negl}(\lambda)$, whichever layer is violated.

O Licenses for existing assets

- **LLaMA 2** [38] (7B, 13B, 70B): Meta LLaMA 2 Community License Agreement.
- **OPT-125M** [45]: Meta OPT Model License.
- **Qwen3** [43] (14B), **Qwen3.5** (9B), and **Gemma4** (12B): Apache-2.0.
- **Mixtral 8×7B** [15]: Apache-2.0.
- **Llama3-8B** [11]: Meta Llama 3 Community License Agreement.
- **MIMIC-IV** [17]: PhysioNet Credentialed Health Data License 1.5.0 (credentialed access under a data use agreement).
- **CIFAR-100** [21]: no explicit license (the creators request citation of the technical report).
- **arkworks** (`ark_ec`, `ark_ff`, `ark_std`, `ark_serialize`): MIT / Apache-2.0 dual license.
- **blake2**, **rayon**, **num_integer**: MIT / Apache-2.0.
- **spongefish**: MIT.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claim four explicit contributions that are each grounded in the paper, and we scope our guarantee to output soundness under budgeted, sampled verification rather than negligible end-to-end soundness. The multivariate commitment scheme is formalized in Definition 4.1 (Sec. 4) with its multivariate pairing check, and its correctness and evaluation-binding proofs are in App. F. Terkle Trees are defined and analyzed in Sec. 4 with empirical scaling in Fig. 4. The layer selection algorithm is presented as Problem 4.5 (Sec. 4) with the DP solution in App. D, AMC evaluation in Fig. 5, and the randomized selector $\mathcal{S}_\gamma$ with its coverage floor and policy-aware adaptive sweep in App. I. The commitment binds the whole transcript up to a $\mathrm{negl}(\lambda)$ binding error, whereas detecting an incorrect output is a statistical guarantee that depends on the layer-selection and challenge distribution (Prop. 4.6); we do not claim full inference correctness as ZK proofs provide (Sec. 5). The claimed overheads (0.97% prover, 0.12% verifier) and comparison metrics are supported by Table 1 with the full deployment breakdown in App. J, and the sub-1% prover overhead on newer models (Qwen3, Qwen3.5, Gemma4) by Table 2, with MoE and federated-learning settings in App. L. The 48% robustness improvement over TOPLOC is shown in Fig. 7, complemented by provider-side attacks in App. K. The multivariate interpolation speedup motivating the approach is demonstrated in Fig. 1 with complexity analysis in App. H. The system overview in Fig. 2 matches the described pipeline.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of our paper are stated explicitly in the Limitations paragraph (Sec. 5). Foremost, TC does not prove full inference correctness: the commitment binds the transcript but does not establish that it was produced by the claimed model, and only sampled relations are audited, so the end-to-end guarantee is the statistical output-soundness bound of Prop. 4.6 rather than negligible soundness. The limitations further include the tolerance of the arithmetic audits; the one-time trusted setup, which must be honestly executed or run via a multi-party ceremony, as compromising the setup makes the scheme fail to satisfy the binding property; operator-selected parameters (e.g., verifier budgets, γ, and k) and the ratio of honest verifiers, which set the detection probability; the cases not covered by the bound, a deviation within the audit tolerance or in a layer that is never selected; and Terkle Tree inefficiency with low branching factors. We also scope our evidence: AMC measures average rather than worst-case coverage, the adaptive sweep (App. I) covers a policy-aware attack family rather than arbitrary adaptive provers, and Sec. 5 contrasts our guarantee with full-circuit ZK proofs.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.

- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We numbered and cross-referenced all theoretical results. The security parameter is defined in Definition 2.1, vector commitments in Definition 2.2, and the tensor commitment scheme in Definition 4.1. The correctness and position-binding properties of VCs are stated in Definitions following Def. 2.2. Merkle tree complexity is given in Remark 2.7 and Verkle tree complexity in Remark 2.10. The Terkle Tree proof-size and prover-cost proposition appears in Sec. 4 with explicit asymptotic bounds. The polynomial division algorithm is given as Algorithm 1 in App. A; its m quotients satisfy the telescoping identity $f_T(X) - y = \sum_j q_j(X_j, \ldots, X_m)(X_j - \omega_j)$, which underlies the multivariate verification equation $e(C_T \cdot g^{-y}, g) = \prod_j e(\pi_{\omega_j}, g^{\tau_j - \omega_j})$ of $\mathsf{Ver_{TC}}$ in Sec. 4. Full formal proofs for this m-quotient form, following the multivariate polynomial commitment of Papamanthou et al. [31], are provided in App. F: TC correctness (Eq. 2 through Eqs. 4-6) and evaluation binding (Eq. 8) under the hardness assumption stated there, relying on pairing properties detailed in App. E.1. The security parameter's role is discussed in App. E.3. Prop. 4.6 (Sec. 4) states, with its assumptions (evaluation binding, and every challenged relation checked correctly up to the audit tolerance) and a complete proof in App. N, that an incorrect committed transcript T is accepted with probability at most $1 - \delta(T) + \mathrm{negl}(\lambda)$, where $\delta(T)$ is the probability of sampling at least one relation with error above the audit tolerance; for a single violated layer ℓ selected with probability p_ℓ, with a fraction ρ_ℓ of out-of-tolerance coordinates and k independently sampled coordinates, this bound is $1 - p_\ell[1 - (1 - \rho_\ell)^k] + \mathrm{negl}(\lambda)$. This yields our output-soundness guarantee $\Pr[\mathcal{V} \text{ accepts} \wedge \tilde{y} \neq \mathcal{M}(x; \theta_{\mathcal{M}})] \leq \varepsilon$. The randomized selector $\mathcal{S}_\gamma$ gives every layer the coverage floor $p_\ell \geq (1 - \gamma)c$, with c the minimum per-layer coverage of a uniformly sampled feasible allocation (Sec. 4, App. I), and the homomorphic and arithmetic audits used by the verifier are specified in App. M. The DP optimality for Problem 4.5 is proved in Proposition D.1 with a full inductive proof in App. D, and its complexity is established in Proposition D.2. The multivariate interpolation complexity is derived in Theorem H.1, Proposition H.2, Corollary H.3, Lemma H.4, and Theorem H.5 in App. H.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We specified the models used (LLaMA2-7B, LLaMA2-13B, LLaMA2-70B, OPT-125M; Qwen3 (14B), Qwen3.5 (9B), and Gemma4 (12B) in Table 2; Mixtral 8×7B with 2/8 and 4/8 active experts, Llama3-8B, and AlexNet in App. L), the datasets and node counts of the federated settings (MIMIC-IV with 8 nodes and LoRA rank 128; CIFAR-100 with 50 nodes), the hardware (A100 GPU), base inference time (10.165s for LLaMA2-13B, with per-model times in Table 2), the attack configurations (1,000 noise-injection attacks, 10 prompts, 100 seeds in Fig. 5; 100 attacks with 10 seeds in Fig. 7; low-rank substitution, layer skipping, and INT4-as-FP16 precision downgrade in App. K; a policy-aware adversary over $\gamma \in \{0, 0.25, 0.5, 0.75, 1\}$ in App. I), and the cryptographic curve (BLS12-381) [39]. The attack models are fully described in App. B. The layer selection algorithm, including the Hill estimator, the DP solver, and the randomized selector $\mathcal{S}_\gamma$, is detailed in Sec. 4, App. D, and App. I, and the stage-by-stage prover and verifier costs with the 880 B communication accounting are given in App. J. Code is open-sourced at `https://github.com/UTAustin-SwarmLab/TensorCommitments`.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The implementation is open-sourced, as linked in footnote 2. The experiments use publicly available models (e.g., LLaMA2, OPT, Qwen3, Qwen3.5, Gemma4, Mixtral, Llama3) from HuggingFace, existing datasets (CIFAR-100, and MIMIC-IV, which requires PhysioNet credentialed access), and standard attack procedures described in the paper. The repository is intended to be ready to merge into inference pipelines, as stated in the contributions, and is available at `https://github.com/UTAustin-SwarmLab/TensorCommitments`.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: In the paper, we specify the hardware (A100 GPU), base inference time, models tested (LLaMA2-7B, LLaMA2-13B, LLaMA2-70B, OPT-125M, Qwen3-14B, Qwen3.5-9B, Gemma4-12B, Mixtral 8×7B, Llama3-8B, AlexNet), attack parameters (noise scaled by ℓ_2 weight norm, 10 prompts $\times$ 100 seeds), tree branching factor ($B = 64$), the Hill estimator parameter χ for layer selection, the mixing parameter γ of $\mathcal{S}_\gamma$, the federated settings (MIMIC-IV with 8 nodes and LoRA rank 128; CIFAR-100 with 50 nodes; App. L), and the commitment scheme parameters. The threat model is detailed in Sec. 3 and App. B, and the policy-aware adversary in App. I. The dynamic programming solver for Problem 4.5 is described in App. D. Benchmark metrics are defined in Sec. 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Fig. 5 (AMC experiments) reports results over 1000 noise-injection attacks per model with 10 prompts and 100 seeds, showing median and interquartile ranges via box plots. Fig. 7 (attack comparison) reports results over 100 attacks with 10 seeds each and shows distributions as box plots with variance clearly visible. Our box-plots convey the spread and central tendency of the results. The tables (Table 1, the newer-model results in Table 2, and the adaptive-sweep, provider-attack, deployment, and MoE/federated tables in App. I, App. K, App. J, and App. L) report point estimates without error bars.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Table 1 reports GPU memory utilization for both prover and verifier across all methods (e.g., 0 GB verifier GPU for TC, 71.32 GB for TOPLOC). The hardware is specified as an A100 GPU. Prover and verifier times are reported in milliseconds/seconds, for LLaMA2-13B in Table 1, per model for Qwen3, Qwen3.5, and Gemma4 in Table 2, and per setting for MoE and federated learning in App. L; App. J breaks the LLaMA2-13B prover (98.6 ms) and verifier (12.19 ms) costs down by stage and reports the one-time 80.6 MB SRS. The base inference time (10.165s for LLaMA2-13B) is stated as a reference point.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research focuses on improving the trustworthiness and verifiability of LLM inference, which aligns with responsible AI deployment. The paper uses publicly available models and datasets and does not involve human subjects or deception. The only sensitive data, the de-identified MIMIC-IV clinical records in one federated-learning cost measurement (App. L), are used under the PhysioNet credentialed-access data use agreement, and we report only cost measurements from them. The work aims to detect and prevent adversarial tampering, contributing positively to AI safety and accountability.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss the broader impact of our work in the Broader Impacts paragraph following the Limitations (Sec. 5), where we discuss positive impacts (strengthening trust in safety critical AI) and negative impacts (false security from partial verification; dual-use of formalized attacks). These are grounded in our paper: Sec. 1 motivates verifiable inference for healthcare, finance, and law; we counter the false-sense-of-security risk by stating the guarantee explicitly as a statistical output-soundness bound (Prop. 4.6) together with the cases it does not cover, and our limitations, Sec. 5, acknowledge that binding depends on an honest setup and detection on the verifier's budget and selection policy; and App. B openly formalizes attack models, whose publication enables stronger defenses.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release pre-trained language models, image generators, or scraped datasets. The released artifact is a verification tool (code for TensorCommitments) that checks the integrity of LLM inference, which is inherently a safeguard mechanism rather than a dual-use technology. It does not pose a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original publications of the models and datasets used where available (Qwen3.5 and Gemma4 are credited by name and license in App. O), and the baseline methods are properly cited: zkLLM, TOPLOC, SVIP. The multivariate interpolation theory is attributed to Hecht et al. (2017), and the multivariate polynomial commitment we build on is credited to Papamanthou et al. [31]. The α-pruning metric is credited to Lu et al. (2024). App. O lists all major assets with their license names: LLaMA 2 [38] (Meta LLaMA 2 Community License), OPT-125M [45] (Meta OPT Model License), Qwen3, Qwen3.5, Gemma4, and Mixtral 8$\times$7B (Apache-2.0), Llama3-8B (Meta Llama 3 Community License), MIMIC-IV (PhysioNet Credentialed Health Data License 1.5.0), CIFAR-100 (no explicit license; cited as its creators request), and all cryptographic and utility libraries (MIT/Apache-2.0).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We introduce an open-source implementation of TensorCommitments at `https://github.com/UTAustin-SwarmLab/TensorCommitments`, and it is "ready to merge into inference pipelines", as described in Sec. 1. The algorithmic construction is fully specified in Def. 4.1 and the appendices (including the audit details in App. M), providing sufficient documentation for understanding and using the codebase along with the detailed readme file provided at the repo.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Our paper does not involve crowdsourcing or research with human subjects. All experiments are computational, involving cryptographic protocols, LLM inference, and federated-learning updates on standard hardware.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects. All experiments are purely computational; the only clinical data, the existing de-identified MIMIC-IV dataset (App. L), is used under its PhysioNet credentialed-access agreement without any interaction with, or new data collection from, human subjects. No IRB approval is needed.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are used in this paper as evaluation subjects (i.e., the models whose inference is being verified), not as a component of the core method development. The core contributions are cryptographic (TensorCommitments, Terkle Trees, layer selection), and LLMs serve only as the application domain for benchmarking. Beyond this, LLMs were used only for writing and editing (e.g., grammar, spelling, word choice), which does not impact the core methodology, scientific rigor, or originality and, per the question, requires no declaration.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.