TensorBlock

January, 2025, v1

Proof of Cache

An Efficient Verification Protocol for Decentralized Inference in Large Language Models

TensorBlock Team
contact@tensorblock.co

Abstract

Decentralized inference networks for large language models (LLMs) enable the deployment of complex models across multiple machines, facilitating collaborative inference. However, ensuring computational integrity in these architectures remains challenging. Existing methods, such as redundant computation, are inefficient and costly. This paper introduces a novel KV-cache sampling approach, leveraging the deterministic internal states of LLMs to enable lightweight and efficient verification. By reducing computational redundancy and maximizing hardware parallelism, this approach significantly enhances verification efficiency. Additionally, we propose a protocol design for the verification process that ensures fairness among participants and long-term system sustainability. Finally, we discuss payoff distributions and outline potential avenues for optimizing the protocol in future work.

1 Introduction

The rapid growth of large language models (LLMs) and their widespread applications have necessitated innovative approaches for scalable deployment. Decentralized inference, a paradigm that utilizes distributed architectures, is gaining traction as a solution to meet these demands. This approach allows diverse, open-source LLMs, such as Llama 3.3 70B [AI24], to be deployed collaboratively across consumer-grade hardware, fostering scalability and accessibility.

A core challenge in decentralized inference lies in ensuring the integrity of computations in trustless environments. Users must rely on inference service providers to execute the correct models, but there is no guarantee against providers substituting smaller, less capable models to maximize profits. This potential for dishonest behavior undermines trust and creates inefficiencies.

Current verification methods often rely on redundant computations, where multiple validators re-run the same inference to detect discrepancies. While effective, these methods are resource-intensive, making them impractical for large-scale deployments. The high cost of verification not only discourages its use but also imposes constraints on protocol designs, requiring complex reward and penalty mechanisms.

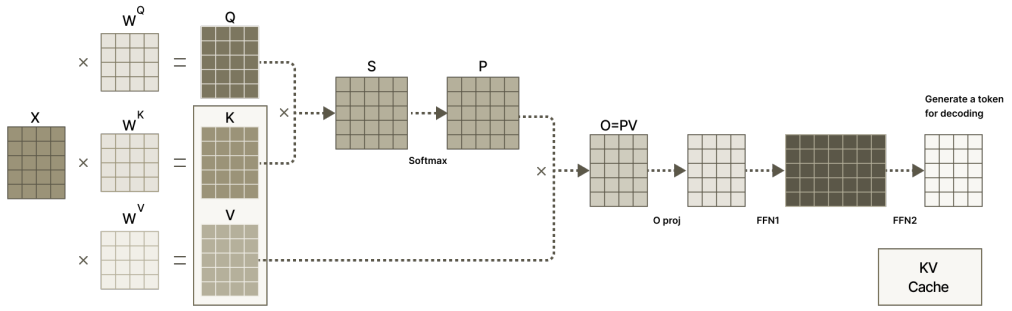
This paper argues that reducing verification costs can significantly enhance the scalability and trustworthiness of decentralized systems. By lowering these barriers, service providers are less incentivized to act dishonestly, and verification becomes a routine, scalable process. To address this need, we introduce the KV-cache sampling method, which leverages the deterministic behavior of LLMs to achieve efficient

and cost-effective verification. This method, combined with an innovative protocol design, provides a practical framework for addressing the challenges of decentralized AI inference, fostering a more secure and scalable ecosystem.

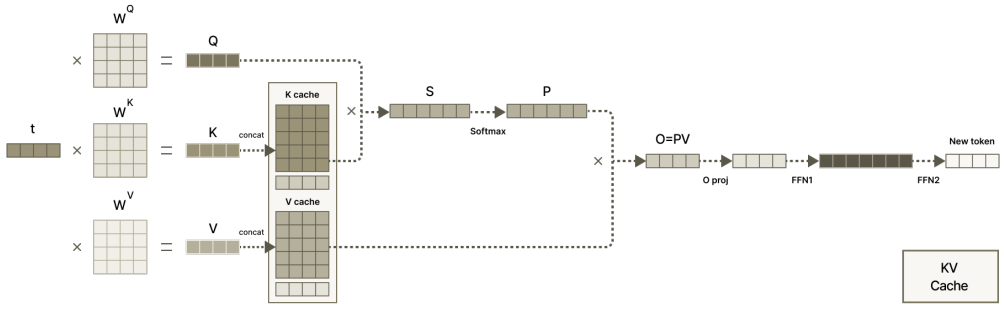
2 Background

2.1 LLM Inference

The inference process of large language models (LLMs) can be divided into two key stages: the **prefilling** stage and the **decoding** stage, both of which present unique challenges and opportunities for optimization.



(a) Prefilling stage



(b) Decoding stage

Figure 1: Two stages of LLM inference: Figure 1a processes input tokens and calculates KV cache, Figure 1b generates tokens auto-regressively

Prefilling Stage: In this initial stage, the LLM processes the input tokens and calculates the key-value (KV) cache for the Multi-Head Self-Attention (MHSA) block. This stage involves significant computational overhead due to the quadratic complexity of the self-attention operation with respect to the input length. Once the KV cache is computed, the first output token is generated.

Decoding Stage: In this stage, the LLM generates tokens auto-regressively, one by one. Each newly generated token updates the KV cache, and subsequent tokens are produced based on the updated state. While the KV cache significantly reduces computational redundancy, the decoding process remains time-intensive due to the need for sequential processing.

The prefilling stage typically handles long input sequences and can effectively utilize parallel computing resources like GPUs due to its batch processing nature. While it has quadratic complexity with respect to input length, modern hardware architectures can efficiently handle this computation.

The decoding stage, despite its lower computational complexity per token, faces efficiency challenges due to its inherently sequential nature. Each token must be generated one after another, leading to potential throughput bottlenecks, especially for applications requiring long output sequences. To address this limitation, recent advances like speculative decoding [LKM23] employ smaller helper models to predict multiple tokens in parallel, which are then verified by the main model. This approach significantly improves decoding efficiency by better utilizing available computational resources while maintaining output quality.

Optimizations targeting these stages focus on different aspects: prefilling optimizations often address memory efficiency and attention mechanisms, while decoding optimizations primarily aim to overcome sequential processing limitations through techniques like speculative sampling.

2.2 Verification in Decentralized Systems

The decentralized systems landscape is characterized by the need for secure, scalable, and efficient mechanisms to maintain trust among participants. Traditional centralized systems rely on institutional reputations or regulatory frameworks to guarantee the correctness of their operations. In contrast, decentralized systems must establish trust through protocol design, often underpinned by cryptoeconomic incentives and game-theoretic principles.

Verification of computations in decentralized systems is a persistent challenge, particularly in trustless and permissionless environments. Optimistic approaches, such as those employed in rollup protocols, rely on the assumption of honest behavior among participants and penalize fraudulent actions through economic disincentives. However, as highlighted in the Proof of Sampling (PoSP) protocol, reliance on economic incentives alone can lead to a mixed-strategy Nash equilibrium, where some participants occasionally act dishonestly without detection. This underscores the need for innovative protocols that enforce honest behavior as a dominant strategy.

In the domain of decentralized AI inference, ensuring that service providers execute the specified model without substituting a less capable one is a critical verification problem. Unlike traditional centralized AI services, which benefit from brand trust and reputational incentives, decentralized inference networks lack inherent guarantees of model integrity. Solutions such as redundant computation, while effective, impose high costs that hinder scalability and adoption.

Recent advancements in verification protocols highlight the trade-offs between cost, scalability, and security. Zero-knowledge machine learning (zkML) [CWSK24], as explored in recent works, provides cryptographic guarantees that ensure computation integrity without revealing sensitive data. However, zkML's high computational costs make it impractical for large-scale models in many use cases. On the other hand, optimistic approaches like opML [CSYW24] leverage economic disincentives and trusted execution environments (TEEs) to balance cost-efficiency and security. These methods optimize verification processes through mechanisms like fraud proofs and probabilistic spot-checks, minimizing redundant computation while preserving

trustworthiness. For instance, opML uses interactive dispute resolution to pinpoint computation errors efficiently.

The emergence of text watermarking techniques, such as those proposed by [DSG+24], adds another layer of robustness in LLM outputs' verification landscape. This work introduces a scalable watermarking method integrated with speculative sampling, ensuring detectability of synthetically generated text with minimal impact on quality and latency.

While watermark holds promise for managing and identifying artificial content in decentralized environments, it also has several significant limitations. First, watermarking techniques are vulnerable to stealing attacks, where malicious actors can reverse-engineer the watermarking rules by querying the API. As outlined by [JSV24], attackers can create an approximate model of the watermarking rules through a process called watermark stealing. This enables spoofing attacks, where malicious actors can produce text falsely attributed to a specific model, and scrubbing attacks, where watermarks are removed while retaining the original content's meaning.

Moreover, while watermarking can detect whether content contains a specific watermark or signature, it cannot definitively verify which entity actually generated the content. This limitation is particularly problematic in decentralized systems where establishing the true source of generated content is crucial for accountability and trust. A service provider who successfully steals or replicates a watermark could generate content that appears legitimate, making it difficult to distinguish between authentic and fraudulent sources. These vulnerabilities significantly compromise the robustness of watermarking schemes in real-world adversarial scenarios.

3 Approach

3.1 Key Insights

Verification Optimization: The deterministic nature of LLM inference forms the foundation for efficient validation. LLMs operate with a deterministic data flow under fixed random seeds, ensuring that given the same prompt and seed, the model's output — including intermediate variables — is reproducible. Recent studies, such as those by [BMR+20] on autoregressive models and related works on GPT architectures, highlight the deterministic nature of token predictions when initialized with identical configurations. This determinism serves as a cornerstone for enabling efficient validation.

The prefilling and decoding stages of LLM inference exhibit distinct characteristics that present opportunities for optimization. The prefilling stage is predominantly compute-intensive, evaluating the prompt in a highly parallel manner while achieving high GPU utilization even with small batch sizes. In contrast, the decoding stage is memory-intensive, generating tokens sequentially with dependencies on the KV cache of both the prompt input and previously generated tokens. This sequential nature results in lower GPU utilization for small batch sizes.

GPU occupation time directly correlates with inference cost. For LLM verification, strategically shifting workload from prefilling to decoding can significantly reduce costs by optimizing GPU utilization. This approach leverages the inherent efficiencies of the prefilling stage to enhance resource usage and improve overall sys-

tem scalability. Furthermore, the ability to compare intermediate variables (e.g., KV cache from early transformer layers) enables optimistic verification strategies - if discrepancies are detected in early layers, the verification process can terminate immediately without computing the full model forward pass, resulting in substantial computational savings. This early-termination capability is particularly valuable in identifying malicious or incorrect computations while minimizing resource expenditure.

Economic Mechanism Design: The protocol's economic design is founded on several fundamental principles that ensure both efficiency and security. First, the implementation of a unified stakepool streamlines accounting and fund distribution while maintaining robust incentives — a significant improvement over systems with multiple separate deposits. Second, the protocol establishes bounded user costs, creating a predictable and accessible system that encourages verification requests when necessary. Third, the requirement for both Generator (i.e., LLM inference service provider) and Verifiers to stake collateral creates powerful economic incentives for honest behavior, as the potential loss of stakes serves as a strong deterrent against malicious actions. Fourth, the protocol's progressive verification mechanism allows additional verifiers to join with constant verification costs while potentially earning higher rewards from an expanded stakepool, creating a natural convergence toward truth as rational verifiers are incentivized to participate when they detect incorrect results. This economic structure ensures that disputes are resolved efficiently through market forces rather than arbitrary staging.

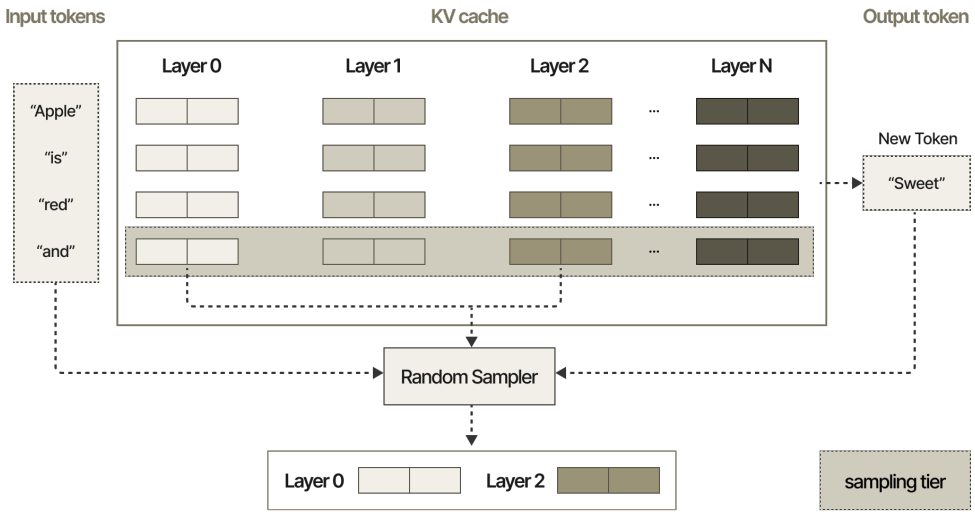
These economic principles work in concert with the technical aspects of LLM inference to create a comprehensive framework for reliable and efficient validation.

3.2 Verification Design

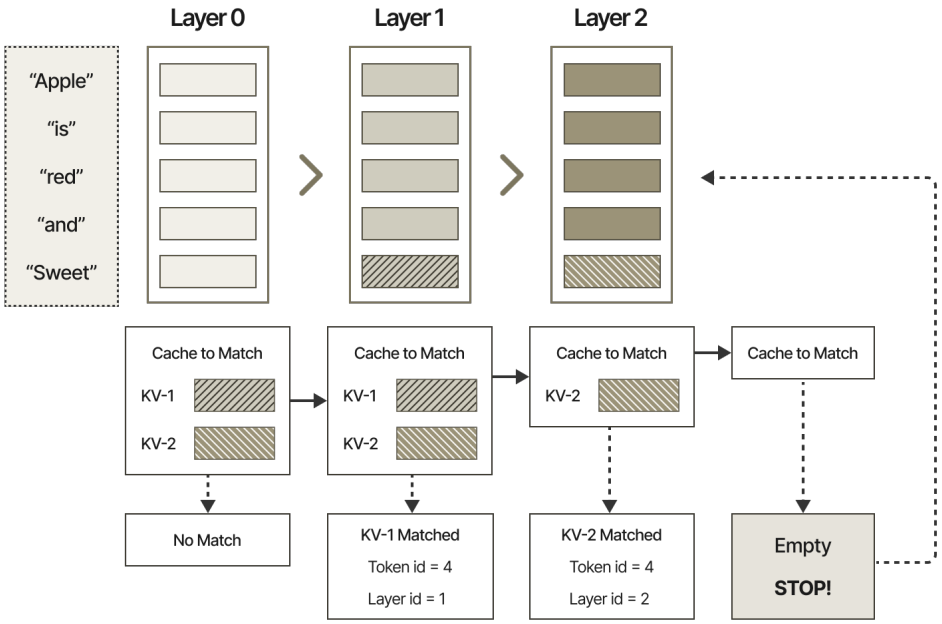
Token-based KV Cache Sampling Our verification design implements a sophisticated approach to sampling KV cache vectors during the token generation process. At its core, the sampling mechanism comprises two essential components: a token-sampler function that determines which tokens' KV cache vectors should be collected, and a layer-sampler function that identifies specific transformer layers for sampling. Both functions operate deterministically, taking the current token context as input to ensure consistency across different runs while maintaining verification efficiency.

The sampling process is seamlessly integrated into the token generation pipeline. For each token, whether from the prompt or generated sequence, the token-sampler evaluates the necessity of collection. When sampling is triggered, the layer-sampler determines the specific layers from which to sample, and the selected KV cache vectors are immediately captured alongside the generated token. This real-time approach ensures verification integrity while minimizing computational overhead.

KV Cache-based Verification The verification process leverages the sampled KV cache vectors to validate model execution through a comprehensive framework of components and procedures. The fundamental components encompass the model configuration, which ensures identical parameters across verification instances; sam-



(a) KV cache sampling



(b) KV cache matching

Figure 2: Overview of the KV cache sampling and verification process. Figure 2a shows the sampling mechanism during generation, while 2b illustrates the matching procedure during verification.

pled KV cache vectors with their corresponding token and layer indices; and vector matching mechanisms for validation.

The verification process follows a clear interaction pattern between the user and verifier. Initially, the user sends three key elements to the verifier:

- Model configuration (including model type, random seed, and temperature settings)
- Generated token sequence
- Sampled KV cache vectors with their corresponding indices

The verifier then processes these inputs by running the model with the specified configuration and comparing the KV cache vectors at the designated sampling positions. Upon completion, the verifier returns the indices (token and layer IDs) where the vectors matched successfully. This allows the user to perform local validation of the results, providing an additional layer of verification security.

This design achieves efficiency through several key mechanisms:

- Leveraging the deterministic nature of LLM inference
- Minimizing the amount of data needed for verification
- Enabling parallel computation where possible
- Providing clear verification criteria through vector matching
- Supporting local validation of verification results

3.3 Protocol Design

The protocol operates within a tripartite framework involving three primary agents: the User U, who consumes the service and may request verification; the Generator G, who produces the service and claims correctness; and the Verifier(s) V, who audit the Generator's result upon User request and selected by protocal. This structure creates a balanced ecosystem where each agent plays a distinct role in maintaining system integrity.

Our mechanism's cornerstone is the implementation of a unified stakepool that consolidates all deposits and manages automatic distribution upon verification completion. This design achieves three critical objectives: it maintains bounded verification costs for Users, establishes robust incentives for honest behavior through potential stake slashing, and simplifies accounting through a unified pool structure.

3.3.1 Initial Setup

Cost Units and Parameters The protocol defines several fundamental parameters that govern its operation. The generation cost unit C_g and verification cost unit C_v establish the basic economic framework. Additionally, two key parameters shape the system's behavior: the user-side parameter $\alpha > 0$, indicating the User's cost to pay for verification, and the stake parameter $\beta > 0$, determining the required collateral for Generators and Verifiers to ensure honest behavior.

Deposits to the Stakepool The single stakepool $\mathcal{P}$ initially comprises two distinct deposits. The User's deposit consists of $\alpha\, C_g + \alpha\, C_v$, where $\alpha\, C_g$ represents the payment allocated to the Generator for correct service delivery, and $\alpha\, C_v$ is reserved for potential verification needs. The Generator's deposit of $\beta\, C_g$ serves as stake that risks slashing in cases of dishonesty or incompetence.

The initial pool state can be expressed as:

$$\mathcal{P} = (\alpha\, C_g + \alpha\, C_v) + b\, C_g$$

This configuration operates independently of Verifier involvement, allowing for efficient resolution in cases where verification is unnecessary.

3.3.2 Mechanism Flow and Payoff Distribution

The protocol implements a progressive verification system that can operate at multiple distinct levels of complexity: no verification, single-stage verification, and n-stage verification where $n \geq 2$. Each level is activated based on the participants' actions and dispute status.

Case 1: No Challenge In the baseline scenario where the User does not request verification, the protocol executes a straightforward distribution of the stakepool. The Generator receives $(\alpha + \beta)\, C_g$, comprising both the User's payment $\alpha\, C_g$ and the return of their stake $\beta\, C_g$. Simultaneously, the User recovers their verification deposit $\alpha\, C_v$, as no verification services were required. This scenario represents the most efficient outcome, where the stakepool is completely emptied with zero verification costs incurred by the User.

Case 2: One-Stage Verification When the User challenges the Generator's result, the protocol transitions to a single-stage verification process. A Verifier enters the system by staking $\beta\, C_v$, expanding the stakepool to:

$$\mathcal{P} = (\alpha\, C_g + \alpha\, C_v) + \beta\, C_g + \beta\, C_v$$

In the case where the Generator and Verifier reach consensus (both being honest), the distribution proceeds as follows: The Generator receives $(\alpha + \beta)\, C_g$, the Verifier is compensated with $(\alpha + \beta)\, C_v$, and the User incurs a verification cost of $\alpha\, C_v$. This outcome reflects appropriate compensation for both service providers while maintaining reasonable costs for the User.

Case 2.2: Multi-Stage Verification When conflicting results emerge between verifiers, the protocol can escalate to multiple verification stages. For each stage $k \geq 1$, a new Verifier V_k joins by staking $\beta\, C_v$, expanding the stakepool:

$$\mathcal{P} = (\alpha\, C_g + \alpha\, C_v) + \beta\, C_g + (k-1)\, \beta\, C_v + \beta\, C_v$$

Each subsequent Verifier (k-th Verifier) examines all previous claims (Generator and prior Verifiers). The verification process follows three possible scenarios:

1. If the k-th Verifier matches the Generator's result:

- The Generator's result is confirmed as correct
- The Generator retains their stake and receives $(\alpha + \beta)C_g$
- The k-th Verifier who agreed with Generator receives $(\alpha + k\,\beta)C_v$
- k-1 dissenting Verifiers have their stakes $(k-1)\beta\,C_v$ slashed

2. If the k-th Verifier matches a previous Verifier's result:

   - That Verifier's result is confirmed as correct
   - The Generator loses stake that $\beta\,C_g$ is slashed
   - The User loses nothing that he gets $\alpha\,(C_g + C_v)$ back
   - k-2 dissenting Verifiers have their stakes $(k-2)\beta\,C_v$ slashed
   - The prior Verifier who provided the correct result takes $(\alpha+\beta)\,C_v$, which includes the compensation $\alpha\,C_v$ and stake $\beta\,C_v$
   - The k-th Verifier takes the rest $\beta\,C_g + [(k-1)\,\beta - \alpha]\,C_v$ from the stake pool

3. If the k-th Verifier's result matches no previous results:

   - The verification game continues
   - A new Verifier (k+1) is invited to participate
   - Process continues until reaching consensus or termination conditions

In all cases, the User's liability remains bounded at αC_v, with potential compensation from slashed stakes.

3.3.3 Incentive Alignment Analysis

The protocol's incentive structure creates a robust framework that promotes honest behavior among all participants. For the Generator, the potential earnings of $\alpha\,C_g$ for correct execution must be weighed against the risk of losing $\beta\,C_g$ through slashing if dishonesty is detected. This stake-based deterrence mechanism effectively discourages fraudulent behavior when $\beta\,C_g$ exceeds any potential benefits from cheating.

Verifiers face a dynamic incentive structure that encourages honest participation and accurate verification. At any stage k, a Verifier (selected by the protocol) can earn a minimum of $\alpha\,C_v$ for correct verification, with potential for additional rewards from slashed stakes of dishonest participants. The risk of losing their stake βC_v for false verification serves as a strong deterrent against both collusion and negligent auditing.

The protocol creates a "race-to-truth" dynamic: when a Verifier is selected at a later stage k, they have the opportunity to earn larger rewards (up to $\beta\,C_g + [(k-2)\,\beta - \alpha]\,C_v$ from accumulated slashed stakes) by confirming the correct result. This increasing reward structure means that later-invited Verifiers have stronger economic incentives to be honest, as they stand to gain more from truthful verification than from collusion or dishonest behavior. This naturally accelerates the convergence to truth through economic incentives rather than arbitrary staging.

The User's position is protected through bounded verification costs, never exceeding $\alpha\,C_v$. This predictable cost structure ensures fairness: if the User challenges

a Generator who proves to be honest, the User pays for the verification service ($\alpha\, C_v$ as compensation to verifiers), which is reasonable since they initiated an unnecessary challenge. However, regardless of how many verification stages are required to reach consensus, the User's cost remains capped at $\alpha\, C_v$. This is particularly fair to the User as they only pay for obtaining the verification conclusion, not for the potentially complex multi-stage verification process itself. In cases where no verification is needed, the User retains their verification deposit, and even in cases requiring multiple verification stages, their financial exposure remains bounded and predictable.

3.3.4 Summary Table of Payoff Distributions

To systematically present the fund flows from the single stakepool $\mathcal{P}$, Table 1 provides a comprehensive overview of the distribution mechanisms across different scenarios. This structured presentation facilitates understanding of how the protocol handles various verification outcomes while maintaining consistent incentive structures.

Table 1: Payoff Distribution Under Unified-Pool Mechanism

Scenario	**Initial Pool State**	**Final Distribution**
No Challenge	$\mathcal{P} = (\alpha C_g + \alpha C_v) + \beta C_g$	Generator: $(\alpha + \beta)C_g$ User: recovers αC_v Pool empties
One-Stage Verification (G & V agree)	$\mathcal{P} = (\alpha C_g + \alpha C_v) + \beta C_g + \beta C_v$	Generator: $(\alpha + \beta)C_g$ Verifier: $(\alpha + \beta)C_v$ User: pays αC_v
k-Stage Verification (V_k confirms G)	$\mathcal{P} = (\alpha C_g + \alpha C_v) + \beta C_g + k\beta C_v$	Generator: $(\alpha + \beta)C_g$ V_k: receives $(\alpha + k\beta)C_v$ Other Vs: lose stakes User: pays αC_v
k-Stage Verification (V_k confirms prior V)	$\mathcal{P} = (\alpha C_g + \alpha C_v) + \beta C_g + k\beta C_v$	Generator: loses stake βC_g Correct V: receives $(\alpha + \beta)C_v$ V_k: receives $\beta C_g + [(k-1)\beta - \alpha]C_v$ Other Vs: lose stakes User: recovers $\alpha(C_g + C_v)$

Notably, across all scenarios, the User's maximum verification cost remains capped at αC_v, fulfilling the protocol's commitment to bounded user costs. The Generator and Verifier(s) maintain their respective stakes of βC_g or βC_v, subject to slashing upon verification of dishonest behavior, thereby ensuring robust incentive alignment throughout the verification process.

3.3.5 Remarks

The unified stakepool mechanism represents a significant advancement in verification protocol design. By consolidating all deposits—including the User's potential payment, Generator's collateral, and Verifier(s)' stakes—into a single, shared stakepool, the mechanism achieves efficient and transparent fund distribution based on

verified outcomes. This design offers several notable advantages over traditional verification systems.

First, the protocol's architectural simplicity, characterized by a single stakepool and unified payout rules, facilitates straightforward implementation, particularly in blockchain-based smart contract environments. Second, the bounded user cost structure, ensuring that verification expenses never exceed αC_v, promotes accessibility and encourages appropriate utilization of the verification mechanism. Third, the protocol demonstrates remarkable scalability in accommodating multiple verifiers, as new participants can seamlessly enter the system by staking funds and potentially receive compensation from slashed deposits when correctly resolving disputes.

Perhaps most significantly, the protocol establishes robust deterrence against dishonest behavior through its slashing mechanism. The potential loss of staked funds serves as a powerful economic incentive, making honest participation the rational strategy under reasonable assumptions about agent behavior and sufficiently large stake parameters β. This economic framework, combined with the protocol's technical verification capabilities, creates a comprehensive system that effectively ensures the integrity of LLM computations.

The protocol thus successfully balances implementation practicality with comprehensive dispute resolution capabilities, ultimately guaranteeing that correct outcomes are appropriately rewarded while incorrect results face proportional consequences through stake slashing. This balanced approach provides a solid foundation for reliable and efficient LLM verification in practical applications.

4 Discussion

While this manuscript presents a novel high-level framework for reducing LLM inference verification costs and implementing an incentive mechanism, several important aspects warrant further investigation in future iterations. This section outlines key considerations that require additional development to enhance the framework's robustness and completeness.

First, the governance structure for verifier selection and consensus determination requires further elaboration. One potential approach involves establishing a committee system where both the user and generator nominate nodes upon initiation of a challenge. This committee would then oversee verifier selection and consensus formation, ensuring fair and transparent dispute resolution.

Second, the economic incentive structure presents opportunities for optimization. For instance, implementing user rewards for successful challenges could enhance system security by incentivizing active result validation. This additional layer of scrutiny would create a stronger deterrent against malicious behavior by generators. However, careful consideration must be given to balancing these incentives to prevent abuse of the challenge mechanism.

Third, the relationship between system parameters—including generation costs, verification compensation, and stake multipliers—requires thorough analysis to understand their impact on system convergence. While higher stake multipliers can effectively discourage malicious behavior, they may also reduce system liquidity and efficiency. This trade-off necessitates careful calibration to maintain both security and operational effectiveness.

We anticipate addressing these considerations in subsequent iterations of this work. We welcome engagement from interested researchers to contribute to these open discussions and further develop the framework's theoretical and practical foundations.

References

[AI24] Meta AI. Llama 3.3 model card. https://github.com/meta-llama/llama-models/blob/main/models/llama3_3/MODEL_CARD.md, 2024. Accessed: 2024.

[BMR+20] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[CSYW24] KD Conway, Cathie So, Xiaohang Yu, and Kartin Wong. opml: Optimistic machine learning on blockchain. *arXiv preprint arXiv:2401.17555*, 2024.

[CWSK24] Bing-Jyue Chen, Suppakit Waiwitlikhit, Ion Stoica, and Daniel 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.

[DSG+24] Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.

[JSV24] Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. *arXiv preprint arXiv:2402.19361*, 2024.

[LKM23] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.