ROBUST DISTRIBUTED TEST-TIME SCALING

Anonymous authors
Paper under double-blind review

ABSTRACT

Test-time scaling (TTS) improves the reasoning performance of large language models by allocating more computation at inference time. At scale, the independent generations underlying TTS can naturally be distributed across multiple workers and aggregated into a final prediction. This paradigm, however, inherits a fundamental challenge from distributed computing: tolerating faulty or compromised workers. To address this challenge, we develop *proof-of-thought consensus*, a robust aggregation framework for distributed TTS. The key insight is that producing a chain-of-thought (CoT) trace that remains statistically consistent with the model's decoding distribution is inherently costly for an adversary. We then introduce an efficient CoT verification mechanism that turns this asymmetry into a practical filter by detecting subtle distributional inconsistencies in manipulated CoT traces before aggregation. We complement the design with theoretical and adversarial analyses: we prove high-probability acceptance guarantees for honest submissions, and formalize three strong attack strategies for stress-testing. Experiments show that, even with up to 30% malicious participants, we limit accuracy degradation to 6.7% on average relative to an oracle baseline that aggregates only the clean samples, while incurring only a 1% additional inference-time overhead.

1 INTRODUCTION

Large language models (LLMs) have recently demonstrated remarkable progress in reasoning (Guo et al., 2025; Ye et al., 2025; OpenAI et al., 2024), achieving strong performance on various challenging tasks, including in mathematics, STEM problem solving, and program synthesis. A key driver of these advances is test-time scaling (TTS), which applies additional computation at inference time to trade increased compute for improved accuracy and stability.

Among TTS strategies, a natural and widely adopted approach is *parallel scaling*, which generates multiple answers and aggregates them via majority voting-based consensus to obtain a more reliable final answer (Wang et al., 2023). This paradigm has been shown to deliver substantial gains across a range of frontier models in OpenAI's o1 (OpenAI), xAI's Grok (xAI), Anthropic's Claude (Anthropic), and Google's Gemini (Luong & Lockhart, 2025).

However, these gains come at a substantial computational cost. Achieving consensus@64, for example, requires generating 64 chain-of-thought (CoT) traces per query, increasing inference cost by a factor of 64. Such linear scaling in computation makes high-consensus inference expensive for real-world deployments, especially when models are large, traces are long, or queries arrive at scale.

As TTS scales to larger sample budgets, centrally provisioning all the required inference capacity becomes increasingly costly. Since individual samples can be generated independently, the workload can instead be distributed across participating nodes. As illustrated in Figure 1, a user query is broadcast to multiple nodes (❶), each of which runs an LLM locally to produce a CoT trace (❷). The resulting outputs are then aggregated to obtain a final answer (❸). By leveraging such distributed computing resources, TTS has the potential to enable large-consensus reasoning without relying on a single centralized system.

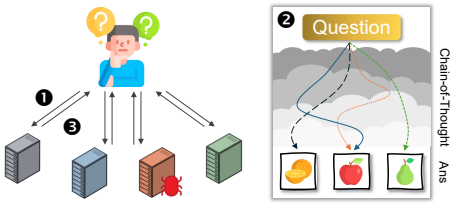


Figure 1: Distributed TTS paradigm.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Yet, as inference scales across distributed workers (Dhasade et al., 2026), it inherits a fundamental challenge from distributed computing (El-Mhamdi et al., 2018; Blanchard et al., 2017): how to tolerate faulty or compromised contributors that could inject incorrect samples (e.g., corrupt answers or manipulated CoT traces) and bias the aggregate decision. Without addressing such attacks, naïvely distributing the sampling process risks transforming majority voting—from a powerful empirical stabilizer—into a new vulnerability surface in the TTS.

To address these challenges, we propose *proof-of-thought (PoT) consensus*, a robust aggregation framework for distributed TTS. PoT is inspired by proof-of-work consensus in blockchains (Nakamoto, 2008): one party (the prover) proves to others (the verifiers) that a certain amount of computation has been expended. Verifiers can subsequently confirm this expenditure with minimal effort on their part. This asymmetry ensures that fabricating valid contributions at scale is costly, whereas screening them is lightweight, making adversarial behavior economically unattractive.

Analogously, in LLMs with strong reasoning capacity, producing a coherent CoT already requires substantial computation, as a participant must generate a long trace to arrive at an answer (Wei et al., 2022). We aim to treat such a trace as a form of computational proof for reasoning: its token choices are plausibly consistent with the reference model's decoding distribution under the same prefixes.

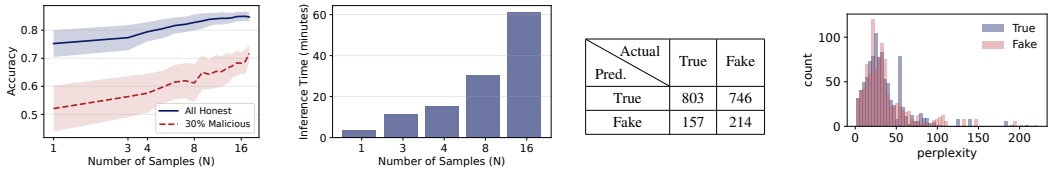
A natural approach is to let the model itself validate contributed traces, but straightforward instantiations fall short. Prompt-based "LLM-as-a-judge" performs poorly (§ 3.4), suggesting that LLMs are not reliably calibrated to decide whether an output could have been generated by their own sampling process. Likewise, perplexity is ill-suited as a verification signal (§ 3.5): it measures the likelihood of a fixed sequence and averages over all tokens, which can dilute inconsistencies in the manipulated traces. Instead, we leverage the model's own distributional expectations by re-prefilling contributed reasoning traces and deriving a new verification metric, *oddity*, from token-level probability distributions. Crucially, this verification pipeline is computationally efficient: it requires only a single teacher-forcing prefill pass without any additional token generation, followed by simple token-level statistics to compute the oddity score. By capturing relative deviations between selected tokens and the model's expected distribution, oddity enables efficient and fine-grained verification of CoT traces.

Contributions. (1) We formulate distributed TTS in the presence of compromised participants and introduce PoT consensus, which treats a submitted CoT trace as statistical evidence that it is consistent with sampling from the reference model. (2) We develop oddity, an expectation-aware sequence-level statistic based on rectified centered surprisal that detects localized distributional inconsistencies. (3) We construct three strong, verifier-aware attacks spanning prompt-side, trace-side, and model-side manipulation, and establish high-probability acceptance guarantees for honest traces together with consensus-recovery bounds under finite compute budget. (4) We implement PoT consensus on top of vLLM (Kwon et al., 2023) and evaluate it using four popular open-source LLMs and four benchmarks. On AIME25 with 30% malicious participants, PoT limits the average oracle-relative accuracy drop to 6.7% across the three attack surfaces. Experiments on BRUMO25, GPQA, and LiveCodeBench v6 show similar robustness, while verification latency remains within 1% of inference-time overhead.

2 Related Work

Test-time scaling. TTS has become a standard technique for improving the reasoning performance of LLMs (Zhang et al., 2025; Zeng et al., 2025; Snell et al., 2025; Gu et al., 2026; Dorner et al., 2026; Kim et al., 2026). Among existing approaches, *self-consistency* is one of the most widely adopted strategies: multiple reasoning traces are sampled independently, and the final answer is selected via majority voting (Wang et al., 2023; Chen et al., 2024b). This paradigm has also been adopted in practice by OpenAI, xAI, Anthropic, and Luong & Lockhart (2025). Moreover, several works propose certainty-based metrics to weight different reasoning traces and perform weighted aggregation instead of uniform voting (Fu et al., 2026). Despite their empirical success, these centralized methods primarily focus on down-weighting low-quality generations, rather than verifying the faithfulness of the underlying CoT traces. Our experiments further show that they are ineffective at distinguishing honest from malicious CoT traces in adversarial settings, thereby motivating the design of our PoT.

In addition, another important line of TTS is *Best-of-N*, which selects a single candidate from multiple sampled traces rather than aggregating them. Existing work typically relies on intrinsic signals (Kang et al., 2025), or on external verifier (Lightman et al., 2024; Wang et al., 2024). Although



(a) Accuracy (b) Cost (c) LLM Judge (d) Perplexity

Figure 2: Motivation experiments for TTS and PoT verification on AIME 2025 with Qwen3-14B.

Best-of-N is not the main focus of this paper, our method is complementary to this line of work: in § E.2, we show that PoT can be naturally incorporated into Best-of-N pipelines.

Verifiable computation for LLMs. Prior work on verifiable inference for neural networks and LLMs spans both cryptographic and hardware-based approaches. On the cryptographic side, zero-knowledge proof techniques can, in principle, certify that model execution was carried out faithfully (Sun et al., 2024; Chen et al., 2024a; Qu et al., 2025). On the hardware side, trusted execution environments provide an alternative path (Zhang et al., 2024; Chrapek et al., 2025). However, these approaches either remain prohibitively expensive for LLM inference or rely on additional hardware. In contrast, we develop an efficient statistical verification mechanism for reasoning traces: rather than proving exact execution correctness, PoT checks whether a submitted CoT is statistically consistent with honest execution under the reference model, adding only about 1% inference-time overhead (see § 5).

Monitoring substandard service providers. Another related line of work studies how to detect degraded LLM services, where providers may reduce model size or numerical precision to increase profit margins. Prior works propose mechanisms for detecting such behavior by analyzing the logit distributions of providers' outputs or other statistical consistency signals (Inference.net; Miao & Fang, 2025; Gao et al., 2025). These methods target quality degradation or unintentional service misconfiguration. In contrast, our setting admits adversarial manipulation and coordinated collusion among participants, posing fundamentally different challenges beyond model-quality monitoring.

3 Motivation

3.1 Test-Time Scaling: Effective but Expensive

Sampling and aggregating multiple traces can substantially improve performance on hard problems. As shown in Figure 2a, increasing the number of samples from 1 to 16 improves Qwen3-14B accuracy on AIME 2025 from 0.75 to 0.83 while reducing prediction variance. This gain requires near-linearly increasing total inference compute (Figure 2b). Under limited local capacity, additional samples either increase wall-clock latency or require more accelerators. Because these samples can be generated independently, the required capacity can instead be supplied by distributed GPUs. However, distributed participation also creates security risks: as shown in Figure 2a, 30% malicious participants targeting the same incorrect answer can substantially degrade consensus accuracy.

3.2 Core Idea: Chain-of-Thought as a Proof

To mitigate this threat, we require each participant to submit its CoT trace together with the final answer. The answer alone is cheap to fabricate and reveals little about how it was obtained, whereas a CoT exposes a token-level trajectory shaped by the query, reference model, and decoding history. Because honest participants already produce the trace during inference, submitting it requires no additional computation. An adversary targeting an incorrect answer must instead produce an entire trace that both supports the target and remains statistically consistent with the reference decoding distribution. This motivates treating the CoT as a proof of computation in distributed TTS.

3.3 Toy Attack: Prompt Substitution

Before introducing PoT, we consider a simple but challenging prompt-substitution attack. A malicious participant uses the reference model with an alternative prompt that elicits a targeted incorrect answer,

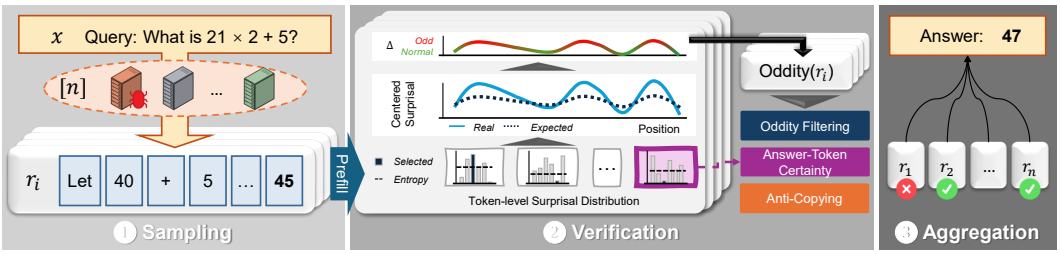


Figure 3: Framework Overview.

then submits the resulting trace as if it were generated under the intended prompt. For example, given the intended query "1+1", the attacker may generate a trace for "1−1" and submit its answer 0 for the original query. Because the forged trace comes from the same model and remains fluent and coherent, it can be difficult to distinguish from honest traces using surface-level cues alone.

We instantiate this attack on AIME25 with Qwen3-14B using a balanced pool of 1,920 submissions: 960 honest traces generated under the intended prompts and 960 malicious traces generated under alternative prompts but submitted for the original queries.

3.4 Do Prompt-Based LLM Judges Work?

A natural baseline for PoT is *LLM-as-a-Judge*, which uses a prompted LLM to assess whether a submitted trace is legitimate. Recent work shows that a separate LLM can be prompted to act as a judge for rating model outputs (Zheng et al., 2023; Bachmann et al., 2025). These judgments often correlate strongly with human preferences, which may reveal inconsistencies between the query and the submitted CoT trace, making this a cost-effective approach to verification.

We prompt Qwen3-14B with the original query and submitted CoT, using the prompt in Appendix A, to classify each trace as honest or fake. On the balanced pool from § 3.3, the confusion matrix in Figure 2c shows only marginally better-than-random performance. Because prompt-substitution traces remain fluent and semantically coherent, surface-level cues are weak, while the judge is not explicitly trained to test consistency with a specific decoding distribution.

3.5 Why Does Perplexity Fail?

Another natural choice is *perplexity* (Hu et al., 2024). Perplexity is the exponential of the average token-level negative log-likelihood that a model assigns to a given sequence, and it is widely used as a measure of how "surprising" a trace is under the model. A natural hypothesis is that fake CoT traces should appear less likely (hence higher perplexity) when rescored under the intended prompt. Given a submitted trace, the user rescores it using the same model via a prefill over the original prompt and the entire trace, and then computes perplexity from the resulting token-level log-likelihoods.

However, Figure 2d shows that perplexity does not reliably separate honest and fake traces. First, long CoTs contain many high-probability routine tokens that can dilute the few critical tokens affected by manipulation (Fang et al., 2025). Second, perplexity scores each token on an absolute scale rather than against the model's own expectation at that position. Consequently, manipulated traces can attain perplexity comparable to honest ones, motivating our new token-level statistic introduced next.

4 Proof-of-Thought Consensus

Figure 3 shows the workflow of our PoT consensus. Untrusted participants return an answer together with a CoT trace as a proof. The user then runs oddity verification: using a single prefill to reconstruct token-level distributions and computing a statistic to detect distributional inconsistencies in the submitted traces. Only submissions that pass verification are retained, and the final answer is produced by voting over these verified submissions. We next formalize the system and adversary model, detail oddity verification, and finally describe the robust aggregation procedure.

4.1 PROBLEM DEFINITION

Consider a user query x served by distributing it across a population of n nodes, indexed by $i \in [n]$. Let the nodes be partitioned into honest nodes $\mathcal{H}$ and malicious nodes $\mathcal{M}$. Each node i submits a pair (y_i, r_i), where $y_i \in \mathcal{Y}$ is the final answer and $r_i = (r_{i,1}, \ldots, r_{i,T_i})$ is a CoT trace consisting of $T_i = |r_i|$ tokens. Following common practice, the answer is bound to the trace, i.e., $y_i = g(r_i)$ for a parsing function g (e.g., extracting the boxed answer).

Reference model. We consider a fixed reference model p_θ operating over a token vocabulary $\mathcal{V}$. Conditioned on query x, it defines a distribution over CoT traces: $p_\theta(r \mid x) = \prod_{t=1}^{|r|} p_\theta(r_t \mid x, r_{<t})$ which induces the joint distribution $p_\theta(y, r \mid x) = p_\theta(r \mid x)\mathbf{1}\{y = g(r)\}$.

Honest nodes. Honest nodes independently sample reasoning traces from the same reference model p_θ and return the resulting answer $(y_i, r_i) \sim p_\theta(\cdot \mid x), i \in \mathcal{H}$. To preserve equivalence with conventional local TTS, all honest nodes use identical model weights, numerical precision, and decoding configuration, differing only in their independently drawn random seeds.

Malicious nodes. Malicious nodes need not follow the reference distribution and may coordinate their submissions to bias the consensus. We consider a white-box adversary with full knowledge of the entire verification procedure, including all metrics, thresholds, and auxiliary components used by the verifier. We allow the adversary to query p_θ freely offline, but only before the test query x is revealed; after observing x, it operates under a per-query compute budget (formalized in Appendix B).

Objective. Let $y_{\text{hon}} = \arg\max_{y \in \mathcal{Y}} |\{i \in \mathcal{H} : y_i = y\}|$ be the answer obtained by aggregating only honest submissions. The goal is to design an efficient verification-and-aggregation rule $\Phi : \{(y_i, r_i)\}_{i=1}^n \mapsto \hat{y}$ that recovers y_{hon} as faithfully as possible despite the presence of malicious nodes.

4.2 VERIFICATION

Verification is carried out by a single *verifier* which receives the n submissions $(y_i, r_i)_{i \in [n]}$ and decides, for each of them independently, whether to accept it. Concretely, its goal is to assess whether a received pair (y_i, r_i) is statistically consistent with a trace sampled from the reference model p_θ on query x. The key insight is that traces drawn from the reference model exhibit predictable statistical patterns in their token-level surprisals. We therefore require each node to attach a CoT as statistical evidence for verification. In what follows, we omit the dependency in node index i to simplify notation, as the verification procedure is identical for all nodes.

Log-prob Reconstruction. Given a submitted CoT r, a single prefill pass suffices to compute the logits at all positions, from which the full token-level distribution can be obtained. Although the implementations of prefill and decode are different in modern serving systems (see Appendix D) and the log probability may not be exactly recovered, our design can tolerate the small numerical error. Therefore, the verifier runs p_θ in teacher-forcing mode on (x, r) to obtain, for each position t, the log-probability assigned by p_θ to token v conditioned on the prefix $(x, r_{<t})$: $\ell_t(v) = \log p_\theta(v \mid x, r_{<t}), v \in \mathcal{V}$. The token selected at step t is r_t with log probability $\ell_t(r_t)$.

Rectified Centered Surprisal. Define the token-level surprisal at position t as: $S_t(v) = -\ell_t(v), v \in \mathcal{V}$ and the corresponding centered surprisal as: $Z_t(v) = S_t(v) - H_t$ where $H_t = -\sum_{v \in \mathcal{V}} p_\theta(v \mid x, r_{<t})\ell_t(v)$ is the entropy. The raw surprisal quantifies how unexpected a given token is for the model, but is affected by context difficulty, and thus mixes model uncertainty and sampling randomness. We center the raw surprisal to account for this context dependence and capture deviations *relative to the model's expectations*. Specifically, the decomposition $S_t(v) = H_t + Z_t(v)$ separates the context-dependent uncertainty captured by the entropy term H_t from the token-specific deviation relative to that expectation, represented by $Z_t(v)$. In practice, when aggregating evidence across a trace, we focus on the *rectified* centered surprisal $U_t(v) = \max(0, Z_t(v))$. By clipping below-baseline deviations, this rectification prevents an attacker from using tokens with negative centered surprisal to offset positive centered-surprisal evidence across a trace.

Oddity as a sequence-level metric. Let $\mu_t = \sum_{v \in \mathcal{V}} p_\theta(v \mid x, r_{<t})U_t(v)$ denote the model's expected rectified surprisal at position t. For a trace r, we define its oddity as $\text{Oddity}(r) = \frac{1}{|r|}\sum_{t=1}^{|r|}(U_t(r_t) - \mu_t)$. For honest traces, each centered-surprisal term has zero conditional expecta-

tion, so oddity concentrates around zero. However, steering a trace toward a targeted incorrect answer may require locally unlikely tokens, producing positive oddity.

From oddity to verification. For a threshold $\gamma > 0$, we define the verification rule $V_\theta(x, y, r) \triangleq \mathbf{1}\{g(r) = y\} \cdot \mathbf{1}\{\text{Oddity}(r) \leq \gamma\}$. The submission is accepted if and only if $V_\theta(x, y, r) = 1$. γ controls the strictness of verification. A smaller γ makes the test more conservative, rejecting traces with even moderate positive oddity; this can improve robustness against manipulated traces, but may also increase false rejections of honest submissions. Conversely, a larger γ makes verification more permissive, improving honest acceptance but increasing the risk that statistically inconsistent traces are retained. Thus, γ captures the trade-off between filtering power and honest-pass rate. The verifier can pre-calibrate γ using traces locally generated from the reference model. Concretely, the verifier samples a modest number of traces under the intended hyperparameters, computes their oddity values, and sets γ to a chosen upper quantile. Its one-time cost is amortized across subsequent queries.

4.3 Aggregation

After verification, the verifier first collects all nodes whose submissions pass PoT verification $\mathcal{I} = \{i \in [n] : V_\theta(x, y_i, r_i) = 1\}$. The final answer $\hat{y}$ is determined by majority voting among verified submissions $\hat{y} = \arg\max_{y \in \mathcal{Y}} |\{i \in \mathcal{I} : y_i = y\}|$. We use majority voting as a simple and robust aggregation rule that preserves the signal from the honest population once adversarial submissions are filtered out. Alternative aggregation rules (e.g., median-based or weighted voting) could also be applied, but majority voting suffices for evaluation. Please refer to Appendix C for the analysis of the PoT consensus. § C.1 bounds the probability that an honest submission is rejected, § C.2 bounds how many malicious submissions can pass verification under a finite compute budget, § C.3 combines the two into a consensus-recovery guarantee, and § C.4 analyses the verification cost.

4.4 Design Refinement

Answer-Token Certainty. A refinement of the verification procedure is to separately audit the answer tokens, since they are particularly sensitive to manipulation. Indeed, a natural attack in distributed TTS is to exploit the answer parser $g(\cdot)$ and selectively rewrite the tokens corresponding to the final answer while leaving the rest of the CoT unchanged. Such an attack preserves local coherence and directly targets the aggregated outcome, making it difficult to detect using surface-level verification.

Table 1: Answer-token certainty.

	Qwen-4B	R1-8B	Qwen-14B
AIME'25	1.006	1.001	1.000
BRUMO'25	1.007	1.009	1.012
GPQA	1.091	1.001	1.016
LCB v6	1.008	1.026	1.006

Fortunately, we observe a distributional property that limits the effectiveness of this attack. Across honest generations, answer tokens exhibit extreme certainty: whenever a token belonging to the final answer appears in the CoT, the model's predictive distribution at that position is nearly deterministic, with answer-token perplexity close to 1.0 across all models and benchmarks (see Table 1). This behavior is expected: by the time the model generates the answer tokens, the preceding reasoning chain has already resolved most of the uncertainty. The answer token thus serves as a deterministic readout of the conclusion reached through reasoning, rather than a site of stochastic exploration, resulting in a sharply peaked predictive distribution regardless of model scale or task domain.

This shows that precise answer tampering must modify exactly those positions where the model's perplexity distribution is most sharply peaked, causing suspicious deviations from the model's expectation. Motivated by this observation, we incorporate an answer-token-aware safeguard that explicitly audits answer-token certainty, requiring their perplexity to remain below a fixed threshold (e.g., 1.1). This makes selective answer rewriting difficult to conceal, as tampering with answer tokens inevitably drives their perplexity well above this threshold. Henceforth, verification rule V_θ denotes the refined verifier incorporating both the oddity check and the answer-token certainty check.

Anti-Copying. Malicious nodes may replicate the same CoT with a targeted wrong answer across many identities. Concretely, the coalition can spend compute to obtain one valid submission for a targeted answer, and then copy it to other malicious nodes. If these duplicates were counted as independent votes, the adversary could inflate the support for the targeted answer. To mitigate this, for each candidate answer, the verifier compares submitted traces using embedding-based similarity and keeps only a set of mutually diverse traces before aggregation.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Hardware. All experiments were conducted on GPU servers equipped with AMD EPYC 9454 CPUs, 250 GB memory, and NVIDIA HGX H100-SXM5 GPUs (80 GB HBM) connected via NVLink.

Models. We evaluate our system using four reasoning-oriented LLMs spanning different model scales: Qwen3-4B, Qwen3-14B, and Qwen3-32B (Yang et al., 2025) and DeepSeek-R1-8B (DeepSeek-AI, 2025) (distilled from Llama-3.1-8B). We use publicly available model checkpoints downloaded from HuggingFace; the hyperparameters are listed in Table 3.

Benchmarks. We evaluate the proposed approach on four challenging LLM reasoning benchmarks: AIME25, BRUMO25 (Balunović et al., 2025), GPQA (Rein et al., 2024), and LiveCodeBench v6 (Jain et al., 2024). AIME25 and BRUMO25 consist of mathematical competition problems, GPQA contains graduate-level STEM reasoning questions, and LiveCodeBench v6 (LCB) comprises a large collection of challenging programming tasks. All experiments are repeated 64 times.

Baselines. We compare our PoT consensus against representative aggregation baselines for TTS. *Self-consistency* (Wang et al., 2023) aggregates answers via majority voting. *Self-certainty* (Kang et al., 2025) reweights answers using token-level confidence derived from the output distribution. We rank N outputs of models by confidence and then assign votes to these ranked outputs using Borda count with $p = 1$. *LLM-as-a-Judge* applies a verification step in which an LLM assesses each submitted reasoning trace using the prompt in Appendix A prior to aggregation. *Median-based aggregation* uses the median over final answers to mitigate the influence of outliers. *Oracle* removes all malicious submissions and aggregates the remaining honest outputs.

Adversarial Knowledge. We consider two adversarial knowledge settings. In the *informed* setting, the adversary knows the most frequent incorrect answer for each query and leverages this to maximize its influence on the final aggregation. In the *uninformed* setting, the adversary lacks such knowledge and instead selects an incorrect answer uniformly at random. Although the uninformed setting is more realistic in practice, the informed setting serves as a worst-case scenario.

Threshold Calibration. For each model–benchmark pair, we reserve honest traces from a disjoint 5% subset of questions as a calibration set and set γ to the 90th percentile of their oddity values.

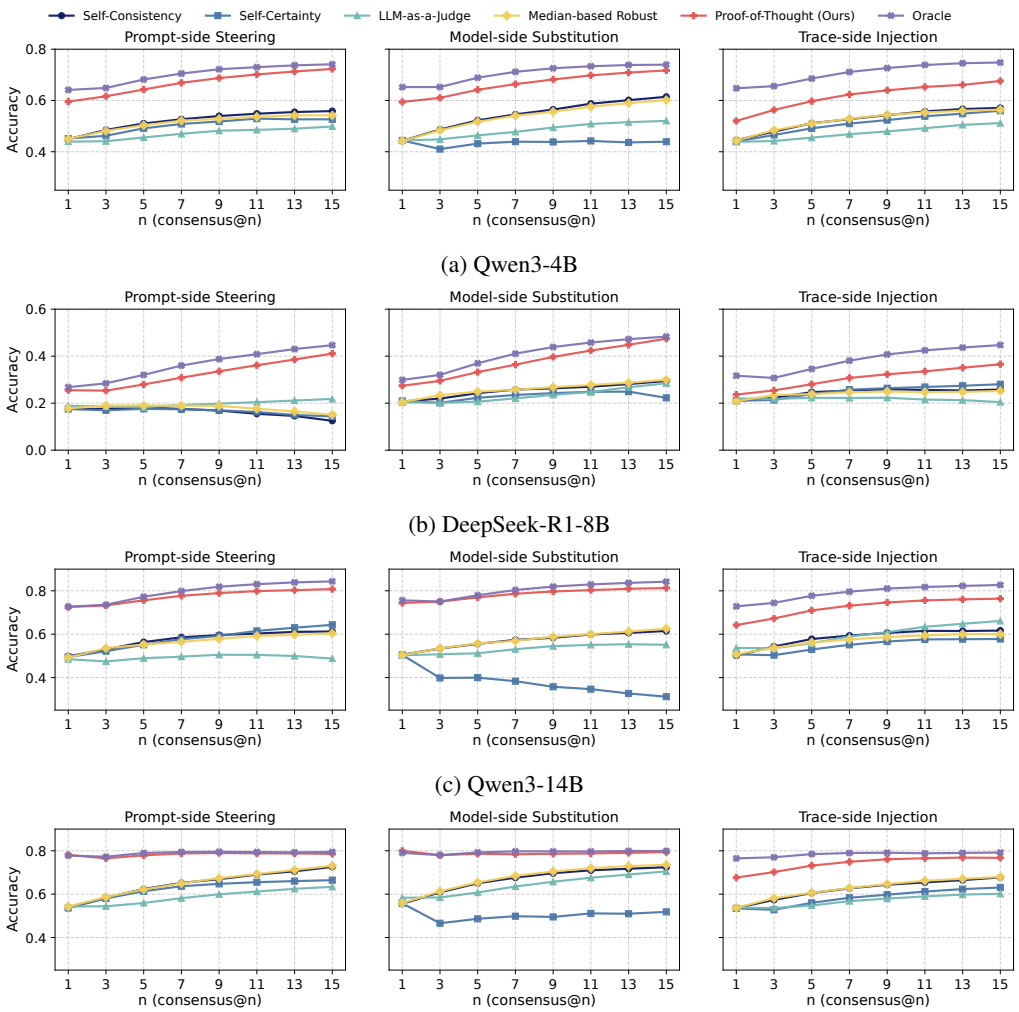
Attack. We evaluate three attacks targeting different attack surfaces of distributed TTS: *prompt-side steering*, which alters the prompt used for generation; *trace-side injection*, which modifies a generated CoT trace to induce a target incorrect answer; and *model-side substitution*, which replaces the prescribed reference model with an alternative backend. See Appendix B for details.

Implementation. Please refer to Appendix D for the implementation details.

5.2 RESULTS

Main Results. As shown in Figure 4, we evaluate the robustness of different aggregation methods for distributed TTS on AIME25 with 30% malicious participants and three different attacks under the informed setting. Across all attack types, all baselines suffer substantial accuracy degradation, showing that adversarial submissions can significantly bias the consensus. Quantitatively, measured by relative accuracy drop with respect to the oracle baseline, the baseline methods incur large degradation across the three attack surfaces, with average drops ranging from 23.7% for self-consistency to 31.7% for median-based aggregation. We also observe clear model dependence: an attack that is highly effective against one model can be noticeably less effective against another, suggesting that robustness depends on model-specific generation behavior. In contrast, our PoT consensus remains robust across a wide range of adversarial settings. It limits the average oracle-relative drop to 6.7%, substantially smaller than all baselines, and its accuracy stays close to the oracle baseline. This suggests that PoT can effectively suppress adversarial influence, achieving near-oracle robustness in practice. In addition, the effectiveness of our PoT carries over to BRUMO25, GPQA, and LCB (see § E.1).

Varying Malicious Ratios. Figure 5 further examines the robustness of our PoT under malicious ratios ranging from 0% to 50% on Qwen3-4B and AIME25, in comparison with the strongest baseline, self-consistency. Across all three attack surfaces, PoT maintains accuracy close to the all-honest



(a) Qwen3-4B

(b) DeepSeek-R1-8B

(c) Qwen3-14B

(d) Qwen3-32B

Figure 4: Accuracy of distributed TTS on AIME25 under 30% malicious participants.

case when the malicious ratio is at or below 30%, and exhibits only mild and graceful degradation even at ratios of 40% and 50%. In contrast, self-consistency is substantially more sensitive to the malicious ratio, with its accuracy declining sharply as the proportion of malicious participants increases. Consequently, the performance gap between PoT and self-consistency widens considerably at higher malicious ratios, demonstrating that PoT remains robust under increasingly adversarial conditions rather than relying solely on standard aggregation.

Filtering Effectiveness. As shown in Figure 6, we analyze the filtering behavior of different methods under 30% malicious participants on AIME25. Self-consistency performs no filtering and thus removes neither malicious nor honest submissions. In contrast, self-certainty, LLM-as-a-Judge, and median-based aggregation filter out comparable proportions of malicious and honest submissions—in some cases removing nearly as many honest submissions as malicious ones—indicating that they fail to effectively distinguish adversarial manipulation from benign variability and thus provide little practical defense. By comparison, PoT consensus selectively removes a substantially larger fraction of malicious submissions while retaining the vast majority of honest ones, closely matching the oracle filtering profile. This asymmetric filtering behavior is the key mechanism underlying PoT's robustness, as it suppresses adversarial influence while preserving the full benefit of honest contributions to the final aggregation.

Informed vs Uninformed. As shown in Figure 7, PoT exhibits nearly identical aggregate accuracy under the informed and uninformed settings, with a difference of only -0.1%. This indicates that

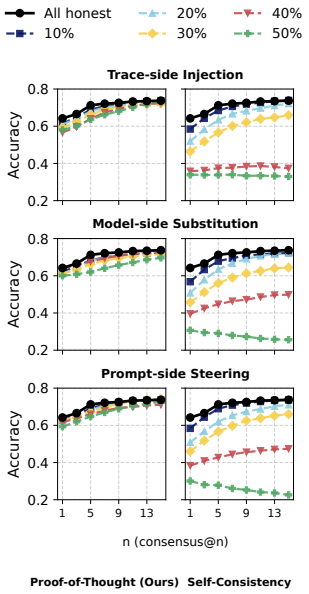


Figure 5: Accuracy of our PoT on Qwen3-4B and AIME25 under varying malicious ratios.

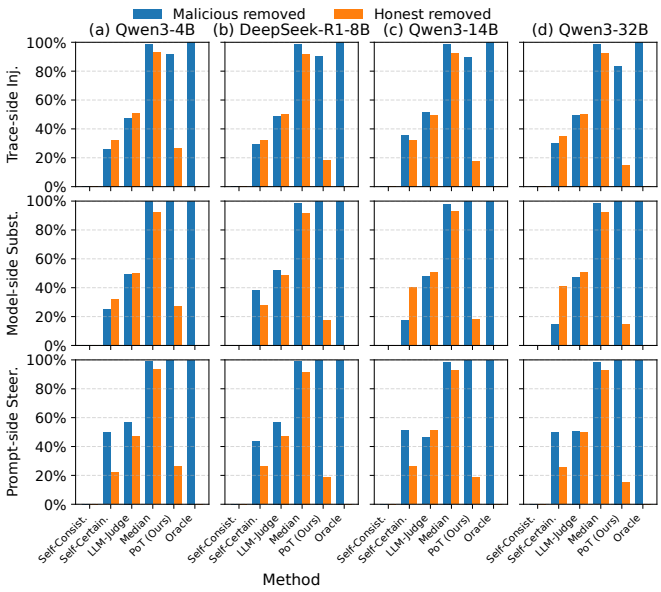


Figure 6: Proportions of malicious and honest submissions removed by different methods on AIME25 with a 30% malicious ratio, across models.

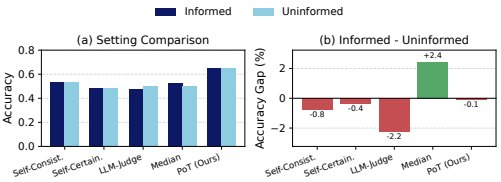


Figure 7: Comparison between informed and uninformed settings on Qwen3-4B over AIME25.

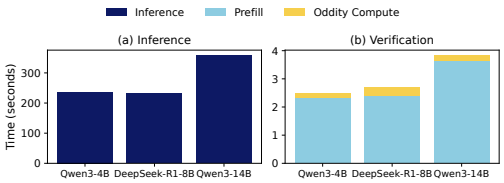


Figure 8: Time cost – verification takes between $2-4$ secs compared to $200+$ secs for inference.

PoT's overall performance is largely insensitive to the attacker's knowledge under the evaluated setting. The baseline methods also exhibit relatively modest aggregate changes: Self-Consistency and Self-Certainty differ by about 1%, while LLM-as-a-Judge decreases by 2.2%.

Overhead. Figure 8 compares the latency of LLM inference and PoT verification on AIME25. Qwen3-4B and DeepSeek-R1-8B exhibit comparable inference latency, whereas Qwen3-14B is slower due to its larger model size (Figure 8a). Verification consists of a single prefill pass and lightweight oddity computation (Figure 8b). Prefill dominates the verification time, while oddity computation is nearly instantaneous. Across all models, total verification latency remains within 1% of full inference latency, demonstrating that PoT introduces negligible additional latency in distributed TTS pipelines.

6 Conclusion

This paper studies robust aggregation for distributed TTS and introduces PoT consensus, a lightweight verification framework that filters adversarial contributions before aggregation. By treating CoT as computational proofs and verifying them through an expectation-aware metric called oddity, PoT enables efficient pre-aggregation filtering with negligible cost. Extensive experiments across multiple reasoning models and benchmarks show that PoT consensus achieves near-oracle robustness even in the presence of a substantial fraction of malicious participants.

REFERENCES

Anthropic. Introducing claude 4. https://www.anthropic.com/news/claude-4. Accessed: 2026-01-21.

KAZUOKI AZUMA. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967. doi: 10.2748/tmj/1178243286.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas K Kohler. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *The Thirteenth International Conference on Learning Representations*, 2025.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. Matharena: Evaluating llms on uncontaminated math competitions. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmark*, 2025.

Tim Beyer, Yan Scholten, Leo Schwinn, and Stephan Günnemann. Sampling-aware adversarial attacks against large language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in neural information processing systems*, 30, 2017.

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*, EuroSys '24, pp. 560–574, New York, NY, USA, 2024a. ISBN 9798400704376. doi: 10.1145/3627703.3650088. URL https://doi.org/10.1145/3627703.3650088.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. Are more LLM calls all you need? towards the scaling properties of compound AI systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b.

Marcin Chrapek, Marcin Copik, Etienne Mettaz, and Torsten Hoefler. Confidential llm inference: Performance and cost across cpu and gpu tees, 2025. URL https://arxiv.org/abs/2509.18886.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Akash Dhasade, Sadegh Farhadkhani, Rachid Guerraoui, Nirupam Gupta, Maxime Jacovella, Anne-Marie Kermarrec, and Rafael Pinot. Robust federated inference. In *The Fourteen International Conference on Learning Representations*, 2026.

Florian E. Dorner, Yatong Chen, André F Cruz, and Fanny Yang. ROC-n-reroll: How verifier imperfection affects test-time scaling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=3Gy5mmyuxn.

El-Mahdi El-Mhamdi, Rachid Guerraoui, and Sébastien Rouault. The hidden vulnerability of distributed learning in byzantium. In *International conference on machine learning*, pp. 3521–3530. PMLR, 2018.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. What is wrong with perplexity for long-context language modeling? In *The Thirteenth International Conference on Learning Representations*, 2025.

Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. In *The Fourteen International Conference on Learning Representations*, 2026.

Irena Gao, Percy Liang, and Carlos Guestrin. Model equality testing: Which model is this API serving? In *The Thirteenth International Conference on Learning Representations*, 2025.

Xiangming Gu, Soham De, Larisa Markeeva, Petar Veličković, and Razvan Pascanu. Understanding performance gap between parallel and sequential sampling in large reasoning models, 2026. URL https://arxiv.org/abs/2604.05868.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z F Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J L Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R J Chen, R L Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S S Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W L Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X Q Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y K Li, Y Q Wang, Y X Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y X Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z Z Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, September 2025.

Yutong Hu, Quzhe Huang, Mingxu Tao, Chen Zhang, and Yansong Feng. Can perplexity reflect large language model's ability in long text understanding? In *Second Tiny Papers Track, ICLR*, 2024.

John Hughes, Sara Price, Aengus Lynch, Rylan Schaeffer, Fazl Barez, Arushi Somani, Sanmi Koyejo, Henry Sleight, Erik Jones, Ethan Perez, and Mrinank Sharma. Best-of-n jailbreaking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Inference.net. Inference.net logic: Log probability verifier. https://github.com/context-labs/logic. Accessed: 2025-12-12.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-n selection for large language models via self-certainty. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Joongwon Kim, Wannan Yang, Kelvin Niu, Hongming Zhang, Yun Zhu, Eryk Helenowski, Ruan Silva, Zhengxing Chen, Srini Iyer, Manzil Zaheer, Daniel Fried, Hannaneh Hajishirzi, Sanjeev Arora, Ruslan Salakhutdinov, Gabriel Synnaeve, and Anirudh Goyal. Scaling test-time compute for agentic coding. In *Third Conference on Language Modeling*, 2026. URL https://openreview.net/forum?id=J9kWWkRzUM.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing

Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

Thang Luong and Edward Lockhart. Advanced version of gemini with deep think officially achieves gold-medal standard at the international mathematical olympiad, 2025. URL https://deepmind.google/blog/advanced-version-of-gemini-with-deep-think-officially-achieves-gold-medal-standard-at-the-international-mathematical-olympiad/. Accessed: 2026-01-24.

Qijun Miao and Zhixuan Fang. User-side model consistency monitoring for open source large language models inference services. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11610–11622, Vienna, Austria, July 2025. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.569. URL https://aclanthology.org/2025.acl-long.569/.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2008. URL https://bitcoin.org/bitcoin.pdf.

OpenAI. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/. Accessed: 2026-01-21.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani

Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card, 2024. URL https://arxiv.org/abs/2412.16720.

Qi Pang, Shengyuan Hu, Wenting Zheng, and Virginia Smith. No free lunch in LLM watermarking: Trade-offs in watermarking design choices. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Wenjie Qu, Yijun Sun, Xuanming Liu, Tao Lu, Yanpei Guo, Kai Chen, and Jiaheng Zhang. *zkGPT: an efficient non-interactive zero-knowledge proof framework for LLM inference*. USA, 2025. ISBN 978-1-939133-52-6.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

Haochen Sun, Jason Li, and Hongyang Zhang. zkllm: Zero knowledge proofs for large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, pp. 4405–4419, New York, NY, USA, 2024. ISBN 9798400706363. doi: 10.1145/3658644.3670334. URL https://doi.org/10.1145/3658644.3670334.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. ISBN 9781713871088.

xAI. Grok 4. https://x.ai/news/grok-4. Accessed: 2026-01-21.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. Limo: Less is more for reasoning, 2025. URL https://arxiv.org/abs/2502.03387.

Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Yunhua Zhou, and Xipeng Qiu. Revisiting the test-time scaling of o1-like models: Do they truly possess test-time scaling capabilities? In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the*

63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 4651–4665, Vienna, Austria, July 2025. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.232. URL https://aclanthology.org/2025.acl-long.232/.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. A survey on test-time scaling in large language models: What, how, where, and how well?, 2025. URL https://arxiv.org/abs/2503.24235.

Ziqi Zhang, Chen Gong, Yifeng Cai, Yuanyuan Yuan, Bingyan Liu, Ding Li, Yao Guo, and Xiangqun Chen. No Privacy Left Outside: On the (In-)Security of TEE-Shielded DNN Partition for On-Device ML . In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 3327–3345, Los Alamitos, CA, USA, May 2024. doi: 10.1109/SP54263.2024.00052. URL https://doi.ieeecomputersociety.org/10.1109/SP54263.2024.00052.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023.

A LLM-AS-A-JUDGE PROMPT

LLM-as-a-Judge Prompt

You are an internal verification module of a large language model. You will be provided with: (1) an input prompt, and (2) a candidate model output, consisting of a chain-of-thought and a final answer. Your task is to assess whether the given output is highly likely to have been generated by this same model in response to the given prompt. Base your judgment only on internal signals, including: consistency of reasoning, stylistic and structural coherence, typical reasoning patterns of the model, token-level plausibility, and alignment between the chain-of-thought and the final answer.

Output
Yes — if the output is highly likely generated by this model according to the given prompt.
No — otherwise.

Now analyze the following.

[INPUT PROMPT]: `<<PROMPT>>`

[MODEL OUTPUT]: `<<OUTPUT>>`

B SYSTEMATIC ATTACK ANALYSIS

We stress-test the proposed PoT on three attack surfaces, corresponding to three points at which an adversary can deviate from honest execution in distributed TTS: *prompt-side steering*, altering the prompt before generation; *trace-side injection*, modifying the returned CoT trace after generation; and *model-side substitution*, replacing the reference model during generation.

B.1 ATTACK OBJECTIVE

Let $y^\dagger \neq y_{\text{hon}}$ denote an incorrect answer targeted by the malicious coalition for a query x (in the informed setting, it is the most common incorrect answer produced by honest nodes, otherwise any wrong answer can do). Define

$$\mathcal{M}^\star := \{i \in \mathcal{M} : g(r_i) = y_i = y^\dagger,\ \text{Oddity}(r_i) \leq \gamma,\ \text{PPL}_{\text{ans}}(x, r_i) \leq \tau\}$$

to be the set of malicious submissions with answer $y^\dagger$ that pass the verification, with PPL_{ans} the perplexity of the final answer tokens and τ the threshold (see § 4.4).

The adversary aims to solve

$$\max_{\{(y_i, r_i)\}_{i \in \mathcal{M}}} |\mathcal{M}^\star|.$$

Intuitively, the adversary must jointly optimize three competing objectives: forcing the final answer toward a specific incorrect target, keeping the manipulated trace statistically consistent enough to pass PoT verification, and doing so at scale under a bounded compute budget.

As shown in Figure 9, to model the compute budget, we first measure the decoding latency of the reference model on a calibration set and set the per-query budget to a high-percentile latency (e.g., P90). Each honest node uses this budget for one decode, whereas each malicious node may reallocate the same budget among generation, trace modification, and verifier evaluations, but returns at most one final submission.

B.2 PROMPT-SIDE STEERING

This attack is inspired by jailbreak attacks (Hughes et al., 2025; Beyer et al., 2026) and generalizes the motivating prompt-substitution example in § 3.3. Instead of relying on a single manually chosen substitute prompt, the attacker first precomputes a large archive of traces generated from many substitute prompts, and then, at query time, selects those candidates that survive PoT verification under the original query.

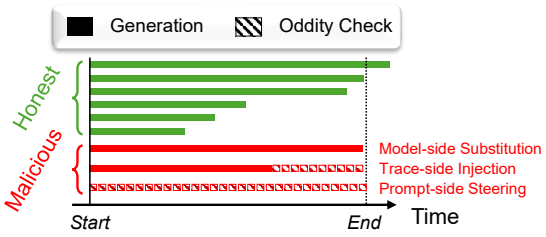


Figure 9: Compute budget for honest and malicious participants for one query.

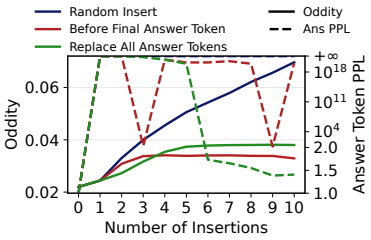


Figure 10: Oddity and answer token perplexity during trace-side injection.

Offline phase. The attacker first prepares an auxiliary pool $\mathcal{Q}$ of questions with known answers. For each substitute prompt $q \in \mathcal{Q}$, the attacker samples multiple traces using the reference model p_θ

$$r_{q,j} \sim p_\theta(\cdot \mid q), \qquad j \in [m],$$

and stores the resulting archive for any target incorrect answer $y^\dagger$ as

$$\mathcal{A}(y^\dagger) = \{(q, r_{q,j}) \mid g(r_{q,j}) = y^\dagger,\ j \in [m],\ q \in \mathcal{Q}\}.$$

Online phase. As shown in Attack (3) in Figure 9, when a new query x arrives, the attacker evaluates each stored trace $(q, r) \in \mathcal{A}(y^\dagger)$ against the verifier associated with the query x. The attacker then returns only those traces that satisfy

$$\mathrm{Oddity}(r) \le \gamma, \qquad \mathrm{PPL}_{\mathrm{ans}}(x, r) \le \tau.$$

This attack is strong because: 1) Most of the attack cost is moved offline: the expensive generations of candidate traces over substitute prompts are performed in advance, while the online attack only requires evaluating oddity under the incoming query and selecting traces that pass verification. 2) This attack explicitly targets the objective of producing wrong-but-accepted traces while being aware of the verifier.

B.3 Trace-side Injection

Our second attack is motivated by spoofing attacks (Pang et al., 2024) and operates directly on generated traces rather than on the input prompt. According to our evaluation results, it is the most powerful one in the three attacks. The attacker first samples a pool of base traces for the original query x from the reference model, and then modifies these traces post hoc so that their final parsed answer becomes a target incorrect answer $y^\dagger$ while still surviving PoT verification under query x.

Concretely, given a trace r, the attacker first replaces the final answer span with $y^\dagger$, yielding an edited trace $\tilde{r}$. This direct substitution ensures that $g(\tilde{r}) = y^\dagger$, but it often makes the injected answer span locally implausible under the reference model and increases its answer-token perplexity, as discussed in § 4.4.

To mitigate this effect, we consider three injection strategies: *Random insertion*, inserting $y^\dagger$ at arbitrary positions; *Answer-local insertion*, inserting $y^\dagger$ near the final answer span; and *Answer-token replacement*, replacing positions where the answer already appears with $y^\dagger$. These strategies reflect different assumptions on positional importance: uniform influence (random), proximity to the final answer (local), and alignment with answer-token positions (replacement). As shown in Figure 10, answer-token replacement is the most effective, achieving lower answer-token perplexity with only a modest increase in oddity.

As shown in Attack (2) in Figure 9, after each injection, the attacker checks whether the edited trace satisfies

$$\mathrm{PPL}_{\mathrm{ans}}(x, \tilde{r}) \le \tau, \qquad \text{and} \qquad \mathrm{Oddity}(\tilde{r}) \le \gamma.$$

Once both conditions are met, the attacker returns the edited trace as a malicious submission.

B.4 Model-side Substitution

Our third attack is motivated by substandard service attack (Miao & Fang, 2025) and replaces the model itself. This attack is motivated by a realistic deviation in distributed TTS: a malicious participant may claim to follow the prescribed reference model p_θ, but in fact use a different backend $p_\phi \neq p_\theta$ that is easier to steer toward a target incorrect answer $y^\dagger$, for example, in Attack (1) in Figure 9. Concretely, for a query x, the adversary samples traces

$$r_k \sim p_\phi(\cdot \mid x), \qquad k \in [K],$$

and returns only those traces that satisfy, under p_θ:

$$g(r_k) = y^\dagger, \qquad \mathrm{Oddity}(r_k) \leq \gamma, \qquad \mathrm{PPL}_{\mathrm{ans}}(x, r_k) \leq \tau.$$

C Theoretical Analysis

Our analysis proceeds in the following steps. First, § C.1 bounds the probability that honest submissions are rejected by PoT. Second, § C.2 bounds the number of malicious submissions that can pass PoT under a finite compute budget. Third, § C.3 combines these two results with the honest vote margin to obtain a consensus recovery guarantee. Finally, § C.4 analyses the cost.

C.1 Completeness

Fix a query x and let $r = (r_1, \ldots, r_T) \sim p_\theta(\cdot \mid x)$ be an honest CoT trace. Since autoregressive decoding is sequential, the distribution of r_t is determined by the realized prefix $r_{<t}$. We therefore work with the natural filtration $\mathscr{F}_t = \sigma(r_1, \ldots, r_t)$ so that conditioning on $\mathscr{F}_{t-1}$ matches exactly the information available before timestep t. For each timestep t, define

$$X_t := U_t(r_t),$$

i.e., the rectified centered surprisal of the sampled token at position t, and recall that

$$\mu_t = \mathbb{E}[X_t \mid \mathscr{F}_{t-1}] = \sum_{v \in \mathcal{V}} p_\theta(v \mid x, r_{<t})\, U_t(v).$$

We first observe that the centered token-level contributions form a martingale difference sequence.

Lemma C.1. *The sequence $(X_t - \mu_t)_{t=1}^T$ is a martingale difference sequence with respect to $(\mathscr{F}_t)_{t=0}^T$.*

Proof. For each t, X_t is $\mathscr{F}_t$-measurable. Moreover, by definition of μ_t,

$$\mathbb{E}[X_t - \mu_t \mid \mathscr{F}_{t-1}] = \mathbb{E}[X_t \mid \mathscr{F}_{t-1}] - \mu_t = 0.$$

□

In particular, for an honest trace,

$$\mathbb{E}[\mathrm{Oddity}(r) \mid x] = 0, \qquad \mathrm{Oddity}(r) = \frac{1}{T}\sum_{t=1}^{T}(X_t - \mu_t).$$

To obtain a finite-sample concentration bound, we assume that the rectified centered surprisal is uniformly bounded: there exists $B > 0$ such that

$$0 \leq U_t(v) \leq B, \qquad \forall t, \forall v \in \mathcal{V}.$$

This holds, for example, if log-probabilities are clipped below by some implementation-dependent floor $p_{\min} > 0$, then $B = -\log p_{\min}$.

Theorem C.2 (Upper-tail concentration for honest traces)**.** *Assume that $0 \leq U_t(v) \leq B$ for all t and v and fix a trace of length T. Then for any $\varepsilon > 0$,*

$$\Pr\left(\mathrm{Oddity}(r) \geq \varepsilon \,|\, x\right) \leq \exp\left(-\frac{T\varepsilon^2}{2B^2}\right).$$

Table 2: Attack success rates on AIME 2025 with Qwen3-4B based on 10,000 attempts.

Attack surface	Oddity pass rate	Answer-token certainty pass rate	Success rate $\widehat{q}_{\mathcal{A}}$
Prompt-side steering	0.00%	0.33%	0.00%
Model-side substitution	0.33%	5.50%	0.00%
Trace-side injection	31.17%	14.83%	6.00%

Proof. Define

$$M_t := \sum_{s=1}^{t} (X_s - \mu_s).$$

By the lemma, $(M_t)_{t=0}^{T}$ is a martingale with respect to $(\mathscr{F}_t)$. Also, since $0 \le X_t \le B$ and $0 \le \mu_t \le B$, we have

$$|X_t - \mu_t| \le B.$$

Azuma's inequality (AZUMA, 1967) therefore gives

$$\Pr(M_T \ge T\varepsilon \mid x) \le \exp\left(-\frac{(T\varepsilon)^2}{2TB^2}\right) = \exp\left(-\frac{T\varepsilon^2}{2B^2}\right).$$

The claim follows from

$$\mathrm{Oddity}(r) = \frac{M_T}{T}.$$

□

The theorem shows that, under honest decoding, oddity stays concentrated near zero at scale $O(B/\sqrt{T})$. Thus, longer traces admit tighter concentration, and thresholds on this scale preserve a high honest acceptance rate.

C.2 Budgeted Attack Success

According to the three attack surfaces as described in Appendix B, we first bound the number of malicious submissions that can pass PoT verification under a finite compute budget as follows.

Consider an attack strategy $\mathcal{A}$ targeting an incorrect answer $y^{\dagger}$. Each malicious node $j \in \mathcal{M}$ executes the attack strategy once under its allocated compute budget and returns at most one candidate trace r_j. Malicious nodes may coordinate, and the strategy used by node j may depend on all previously observed outcomes. Let $\mathcal{F}_{j-1}$ denote all information available to the adversary before generating the j-th candidate. Suppose that, for every malicious node j, the success probability is bounded by

$$\Pr\left(V_\theta(x, y^{\dagger}, r_j) = 1 \mid \mathcal{F}_{j-1}\right) \le q_{\mathcal{A}}(x, y^{\dagger}).$$

For brevity, we write $q_{\mathcal{A}} = q_{\mathcal{A}}(x, y^{\dagger})$ below. Table 2 reports the empirical probability $\widehat{q}_{\mathcal{A}}$ that a single adversarial candidate passes the complete PoT verifier. We observe no successful candidates for either prompt-side steering or model-side substitution. Their empirical success rates are therefore zero. Prompt-side candidates are rejected entirely by the oddity check. Among model-side candidates, 0.33% pass the oddity check and 5.50% pass the answer-token perplexity check, but none pass both checks simultaneously. Trace-side injection is substantially stronger. This result indicates that local answer-token manipulation is more difficult to detect than prompt- or model-level substitution, because most of the original reasoning trace remains statistically consistent with the reference model.

Define

$$S_{\mathcal{M}} := \sum_{j=1}^{|\mathcal{M}|} V_\theta(x, y^{\dagger}, r_j)$$

as the number of successful submissions among the malicious nodes.

Theorem C.3 (Budgeted attack-success bound)**.** *For every integer $m \ge 1$,*

$$\Pr(S_{\mathcal{M}} \ge m) \le \sum_{j=m}^{|\mathcal{M}|} \binom{|\mathcal{M}|}{j} q_{\mathcal{A}}^{j} (1 - q_{\mathcal{A}})^{|\mathcal{M}|-j}.$$

Proof. Let

$$Z_j := V_\theta(x, y^\dagger, r_j).$$

By assumption,

$$\Pr(Z_j = 1 \mid \mathcal{F}_{j-1}) \le q_{\mathcal{A}}.$$

Therefore, $\sum_{j=1}^{|\mathcal{M}|} Z_j$ is stochastically dominated by a binomial random variable with parameters $(|\mathcal{M}|, q_{\mathcal{A}})$. The result follows from the corresponding binomial upper-tail probability. □

Theorem C.3 bounds the probability that the adversary produces at least m successful candidates. In § C.3, this tail bound is used to control the probability that the adversary accumulates enough accepted votes to overturn the honest vote margin.

C.3 Consensus Recovery

We now combine the honest completeness guarantee from § C.1 with the budgeted attack-success bound from § C.2.

Let

$$H_y := |\{i \in \mathcal{H} : y_i = y\}|$$

be the number of honest submissions supporting answer y. Assume that the honest plurality answer

$$y_{\mathrm{hon}} = \arg\max_{y \in \mathcal{Y}} H_y$$

is unique, and define its vote margin as

$$\Delta := H_{y_{\mathrm{hon}}} - \max_{y \neq y_{\mathrm{hon}}} H_y.$$

Let

$$R_H := \sum_{i \in \mathcal{H}} (1 - V_\theta(x, y_i, r_i))$$

be the number of honest submissions rejected by PoT. The actual number of malicious votes entering the aggregation is at most $S_{\mathcal{M}}$ and is also bounded by $|\mathcal{M}|$. We use $S_{\mathcal{M}}$ in the analysis to obtain a conservative upper bound, and assume that all successful malicious submissions support the same competing answer.

Lemma C.4 (Deterministic consensus recovery)**.** *If*

$$R_H + S_{\mathcal{M}} < \Delta,$$

then majority voting over the verified submissions returns y_{hon}.

Proof. After verification, y_{hon} retains at least

$$H_{y_{\mathrm{hon}}} - R_H$$

honest votes. Any competing answer $y \neq y_{\mathrm{hon}}$ receives at most

$$H_y + S_{\mathcal{M}}$$

votes. Hence

$$(H_{y_{\mathrm{hon}}} - R_H) - (H_y + S_{\mathcal{M}}) \ge \Delta - R_H - S_{\mathcal{M}} > 0.$$

Therefore, y_{hon} remains the unique plurality answer. □

For a fixed trace length T, Theorem C.2 gives the honest rejection bound

$$\delta_H := \exp\left(-\frac{T\gamma^2}{2B^2}\right).$$

Since honest traces are independently sampled,

$$\Pr(R_H \ge a) \le \sum_{j=a}^{|\mathcal{H}|} \binom{|\mathcal{H}|}{j} \delta_H^j (1 - \delta_H)^{|\mathcal{H}|-j}$$

for every integer $a \ge 1$.

Combining the honest-rejection tail bound above with Theorem C.3 yields the following consensus recovery guarantee.

Theorem C.5 (Budget-dependent consensus recovery)**.** *For every integer* $a \in \{1, \dots, \Delta\}$,

$$\Pr\left(\widehat{y} \neq y_{\text{hon}}\right) \leq \sum_{j=a}^{|\mathcal{H}|} \binom{|\mathcal{H}|}{j} \delta_H^j (1-\delta_H)^{|\mathcal{H}|-j} + \sum_{j=\Delta-a+1}^{|\mathcal{M}|} \binom{|\mathcal{M}|}{j} q_{\mathcal{A}}^j (1-q_{\mathcal{A}})^{|\mathcal{M}|-j}.$$

Proof. By Lemma C.4, consensus failure requires

$$R_H + S_{\mathcal{M}} \geq \Delta.$$

Fix any $a \in \{1, \dots, \Delta\}$. If $R_H < a$, then $R_H \leq a-1$, so consensus failure requires

$$S_{\mathcal{M}} \geq \Delta - R_H \geq \Delta - a + 1.$$

Therefore,

$$\{R_H + S_{\mathcal{M}} \geq \Delta\} \subseteq \{R_H \geq a\} \cup \{S_{\mathcal{M}} \geq \Delta - a + 1\}.$$

Applying the union bound together with the honest-rejection tail bound and Theorem C.3 proves the result. □

The tightest bound is obtained by minimizing the right-hand side over $a \in \{1, \dots, \Delta\}$. The result explicitly combines the honest rejection probability δ_H, the per-attempt attack success probability $q_{\mathcal{A}}$, and the honest vote margin Δ.

C.4 Efficiency

Let $C_{\text{pf}}(L)$ denote the cost of a single teacher-forced prefill pass on an input of length L, and let $C_{\text{dec}}(L)$ denote the cost of one autoregressive decode step at context length L.

For any fixed query x and candidate CoT trace r of length T, computing the verification rule $V_\theta(x, g(r), r)$ requires one prefill pass over the concatenated sequence (x, r) together with linear-time post-processing of the returned logits:

$$\text{Cost}_{\text{verify}} = C_{\text{pf}}(|x| + T) + O(T|\mathcal{V}|).$$

In contrast, reproducing a fresh trace of length T from the same query requires one initial prefill pass followed by T decode steps:

$$\text{Cost}_{\text{gen}} = C_{\text{pf}}(|x|) + \sum_{t=1}^{T} C_{\text{dec}}(|x| + t - 1).$$

Thus, verification replaces T sequential decode rounds with a single teacher-forced pass and lightweight post-processing. We report on the empirical comparison of the latency in Figure 8.

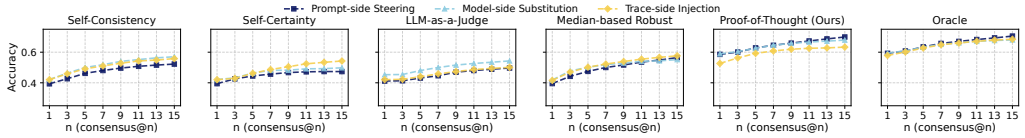
D Implementation

We implement our system in Python 3.12 using PyTorch 2.9.1+cu128 and FlashAttention 2.8.3. Model inference and TTS execution are built on top of vLLM[1]. To support PoT verification, we modify vLLM to expose token-level prefill log-probabilities that more closely match decoding-time log-probabilities. In the original vLLM implementation, prefill returns only prompt log-probabilities for unsampled tokens, which can differ substantially from the log-probabilities observed during stochastic decoding. To reduce this mismatch, we extend both the sampler and the GPU model runner to make log-probability computation more consistent between prefill and decode (`vllm/v1/sample/sampler.py` and `vllm/v1/worker/gpu_model_runner.py`). We also observe that vLLM includes several implicit serving-time optimizations, such as chunked prefill, which prevent prefill from exactly reproducing the log-probabilities seen during decoding. We intentionally retain these optimizations for two reasons. First, efficiency is a first-class design goal, and we do not want PoT consensus to come at the cost of significantly degraded serving performance. Second, in our

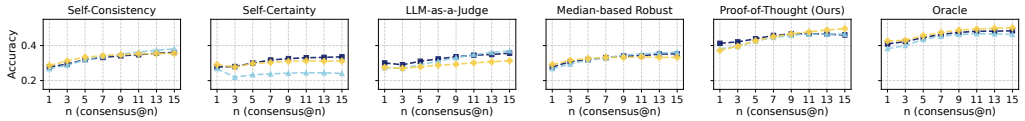
[1]Commit ID `adcf682fc7d1835d037da331922751e880c8bc25`

Table 3: Model configurations & decoding hyperparameters.

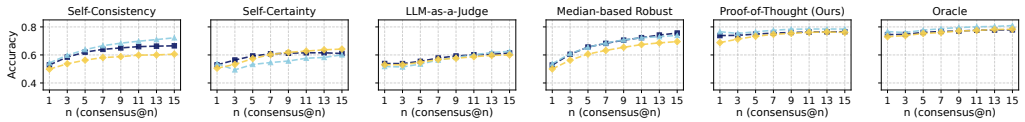
Model	Qwen3-4B	Qwen3-14B	Qwen3-32B	DeepSeek-R1-8B
Precision	FP8	FP8	FP8	BF16
Temperature	0.6	0.6	0.6	0.6
Top-p	0.95	0.95	0.95	0.95
Top-k	20	20	20	-1
Min-p	0	0	0	0
Max output length	38,912	38,912	38,912	32,768



(a) Qwen3-4B



(b) DeepSeek-R1-8B



(c) Qwen3-14B

Figure 11: Accuracy of distributed TTS on BRUMO under 30% different types of malicious participants across different models.

evaluation, both honest and malicious traces are subject to the same approximation error, so the resulting oddity scores remain comparable in practice.

To support majority voting over free-form outputs such as code in LiveCodeBench, we encode each answer using `BAAI/bge-base-en-v1.5` and compute pairwise cosine similarities. If the similarity between two answers exceeds 0.7, they provide mutual voting support to each other. The voting score of an answer is the sum of its similarities to all answers above this threshold, and we select the answer with the highest score. To support median-based aggregation in LiveCodeBench, we impose a lexicographic ordering on the extracted code strings and select the median candidate.

Using the embedding, we can also apply a diversity constraint as discussed in § 4.4: for each candidate answer, if the similarity between any two traces exceeds 0.97, only one is retained before aggregation.

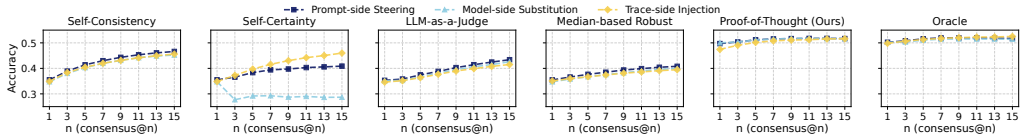
E Additional Experimental Results

E.1 Diverse Workloads

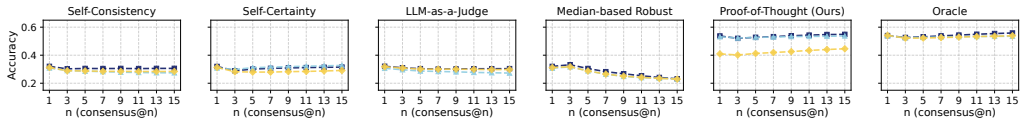
The effectiveness of PoT consensus carries over to BRUMO25, GPQA, and LCB, as shown in Figure 11, Figure 12, and Figure 13. These results confirm that PoT generalizes beyond a single benchmark, remaining effective across diverse reasoning workloads spanning mathematical competition problems, graduate-level STEM reasoning, and programming tasks.

E.2 Best-of-N Selection

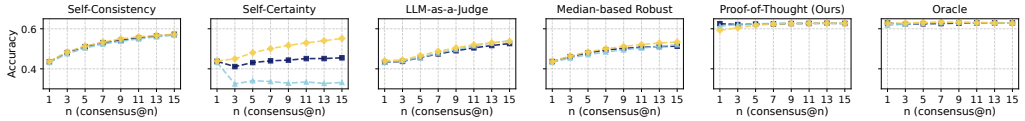
As shown in Figure 14, we evaluate the compatibility of our PoT consensus with Best-of-N selection. We apply PoT verification before self-certainty-based selection. Across all three attack surfaces,



(a) Qwen3-4B

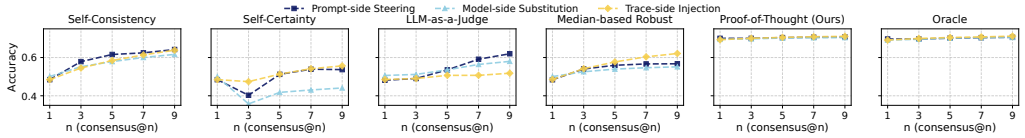


(b) DeepSeek-R1-8B

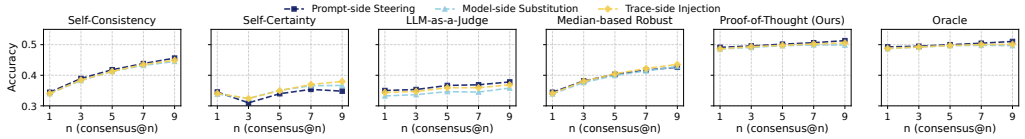


(c) Qwen3-14B

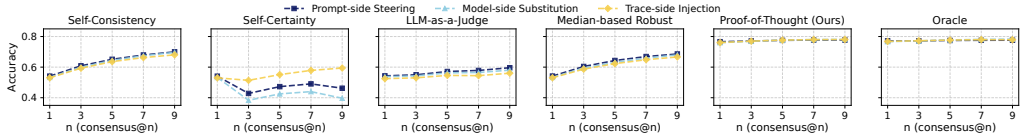
Figure 12: Accuracy of distributed TTS on GPQA under 30% different types of malicious participants.



(a) Qwen3-4B



(b) DeepSeek-R1-8B



(c) Qwen3-14B

Figure 13: Accuracy of distributed TTS on LiveCodeBench v6 under 30% different types of malicious participants across different models.

PoT consistently and substantially outperforms Self-Certainty for Best-of-N, demonstrating that pre-aggregation filtering is critical for reliable Best-of-N selection in adversarial settings. These results show that PoT is not limited to majority-voting-based aggregation but can be naturally incorporated into Best-of-N pipelines as a general-purpose verification front-end.

F AI USE STATEMENT

In this work, the authors used generative AI tools to improve the clarity and readability of the manuscript, assist in identifying and surveying potentially relevant literature, and generate parts of the software implementation. The authors independently consulted the original sources and verified all cited work. All AI-assisted text and code were manually reviewed for accuracy and correctness before being incorporated into this work. Generative AI tools were not used to formulate the research questions, develop the proposed methodology, derive mathematical claims or proofs, design the

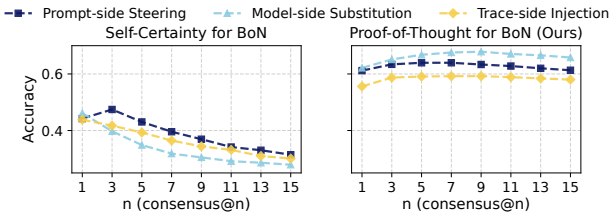


Figure 14: Accuracy of Best-of-N on Qwen3-4B and AIME25 under 30% malicious participants.

experiments, or interpret the results. The authors take full responsibility for the final content of this paper, including all text and code produced with the assistance of generative AI.

G Reproducibility Statement

To support reproducibility, we enumerate all artifacts associated with the core contributions of this paper and describe how reviewers can access them during double-blind review in the following.

Code. The full implementation is available at the following anonymous repository:

`https://anonymous.4open.science/r/RDTTS-44F5`

The repository includes:

- `simulation.py` — generates multi-sample chain-of-thought traces for a given model and benchmark via vLLM.
- `eval_consensus.py` / `eval_consensus_bon.py` — evaluates consensus robustness under adversarial injection at a fixed or varying malicious ratio.
- `eval_answer_token_prob.py` — scores answer tokens via prompt log-probabilities, used by the PoT verifier.
- `consensus/` — implements all aggregation methods (self-consistency, median, PoT, LLM-as-a-judge, and Best-of-N), as well as adversarial data preparation logic.
- `eval/` — dataset loading utilities for all four benchmarks.
- `vllm/` — two patched vLLM source files that expose the prompt log-probability hooks required by the verifier.

Instructions for environment setup and reproducing all experiments are provided in `README.md`.

Datasets. All benchmarks are publicly available and loaded automatically via the Hugging Face `datasets` library at runtime; no manual download is required.

- **AIME 2025** — `MathArena/aime_2025`
- **BRUMO 2025** — `MathArena/brumo_2025`
- **GPQA Diamond** — `Idavidrein/gpqa`
- **LiveCodeBench** — `sam-paech/livecodebench-code_generation_lite`

Models. All models are publicly available on the Hugging Face Hub and are downloaded automatically by vLLM at runtime.

- `deepseek-ai/DeepSeek-R1-Distill-Llama-8B`
- `Qwen/Qwen3-4B-FP8`, `Qwen/Qwen3-14B-FP8`, and `Qwen/Qwen3-32B-FP8`