AZR: Risk-Adaptive Verification for Decentralized AI Inference on Blockchain Rollups

Ankita Virani

May 2026

Contents

Abstract

Verifying the correctness of AI model inference in decentralized execution environments forces a three-way choice with no uniformly satisfying answer. Zero-knowledge machine learning (zkML) delivers cryptographic computational integrity, but its prover complexity grows super-linearly with model size, making it impractical for billion-parameter language models in interactive settings. Optimistic verification (opML) achieves low normal-path overhead at the expense of a multi-day dispute window and a liveness assumption over economically rational challengers. Trusted execution environments (TEE) enable low-latency confidential execution, but documented microarchitectural side-channel vulnerabilities mean that attestation cannot be treated as an unconditional integrity guarantee.

In this paper, we propose AZR (Adaptive zkTEE Rollups): a multi-tiered verification architecture for AI inference on blockchain rollups that selects the lowest-cost verification pipeline for each individual request. In other words, instead of using a uniform mechanism for all workloads, AZR scores each inference query by a risk function of the form $R(q) = \alpha V + \beta A + \gamma D$, where V measures economic exposure, A measures the adversarial attack surface, and D measures the data sensitivity; then, AZR dispatches the query to one of three verification tiers based on the score.

We formulate the system under a rational adaptive adversary model and give formal security proofs for the computational integrity and model confidentiality of each tier. Moreover, we provide formal arguments showing an aggregate soundness theorem and a strict cost dominance theorem for the adaptive dispatch policy compared to any uniform policy. Evaluation on ResNet-50, BERT-Base, and LLaMA-7B models under a 40/40/20 workload split shows a 61% reduction in mean gas cost relative to uniform zkML on BERT-Base workloads, with 80% of LLaMA-7B queries finishing in under 5 seconds of user-visible latency. Several measurements reported in this work are analytical projections derived from prior zkML benchmarks and published scaling laws rather than direct implementations. LLaMA-7B results should therefore be interpreted as indicative rather than definitive. Future work will include a complete prototype implementation to validate these estimates empirically.

Keywords: decentralized AI, verifiable inference, zero-knowledge proofs, trusted execution environments, optimistic rollups, adaptive verification, confidential computing, blockchain systems

1 Introduction

Decentralized AI inference sits at the intersection of two hard problems. The first is the verification problem: a consumer of an inference result whether a human user, an autonomous software agent, or an on-chain settlement contract cannot re-execute a billion-parameter language model to check the executor's answer. The second is the performance problem: verification mechanisms strong enough to provide cryptographic assurance are, by current prover technology, far too slow and expensive for interactive use at LLM scale. The maturation of large foundation models [1,2] and blockchain-native agent economies [3] makes both problems acutely consequential: when an autonomous DeFi agent bases a six-figure liquidation decision on the output of an off-chain language model, an incorrect but undetected result translates immediately into irreversible financial loss.

The canonical blockchain mechanism for verifying computation, namely re-executing every instruction on every validating node, is infeasible for AI inference: even a modestly sized transformer would exhaust the Ethereum block gas limit by many orders of magnitude [4]. Three paradigms have developed to address this gap.

Zero-Knowledge ML (zkML). Systems such as ZKML [5] and the work of Kang et al. [6] compile neural network inference into arithmetic circuits, producing succinct proofs that any on-chain verifier can check cheaply. The cryptographic guarantee is the strongest

available. The practical limitation is equally unambiguous: prover time for ResNet-50 is already measured in tens of seconds on a 64-core GPU server [7], and LLM-scale models extend this to hours, making interactive deployment infeasible.

Optimistic ML (OPML). He et al. [8] showed that inference can be committed on-chain with an interactive bisection fallback, reducing normal-path cost to a single hash check. Correctness holds only under the 1-of-n honesty assumption: at least one challenger must remain online and financially motivated throughout the dispute window, which is seven days on Optimism [9]. In environments where challenger participation can be economically suppressed, or where the inference output triggers a settlement faster than the window expires, this assumption is operationally fragile.

TEE**-Based Execution.** Systems such as Ekiden [10] run inference inside Intel SGX enclaves and use hardware-signed attestation to communicate integrity to the chain. Microarchitectural attacks demonstrate that this guarantee is not absolute: branch shadowing [11], cache timing [12], and Foreshadow-class speculative execution vulnerabilities [13] have all been shown to compromise enclave confidentiality or attestation key material.

The Core Observation. These three paradigms involve fundamentally different cost-security trade-offs, and no single paradigm dominates across all workload types. A casual chatbot query and an autonomous agent managing a $50,000 liquidity position do not share the same risk profile and should not be assigned the same verification mechanism. Yet existing systems including hybrid designs such as Optimistic TEE-Rollups [14] assign a fixed verification policy uniformly to all workloads. The result is either systematic over-spending on low-risk queries or systematic under-protection of high-risk ones.

This observation reframes verifier selection from a static design decision into a workload-adaptive optimization: given the distribution of incoming inference requests and their heterogeneous risk profiles, choose for each query the cheapest verification pipeline that meets its security requirement.

Contributions. To the best of our knowledge, AZR is the first blockchain-native AI verification architecture to formalize verifier selection as an explicit risk-constrained optimization problem across cryptographic, optimistic, and hardware-based verification domains. The principal contributions of this paper are:

C1. Risk-Adaptive Dispatch Framework. A normalized risk function $R(q) = \alpha V + \beta A + \gamma D$ and a threshold-based dispatch policy Π routing each query to the minimum-cost verification tier satisfying its security requirement, formalized as a constrained optimization over routing probabilities.

C2. Formal Adversarial Model. A unified adaptive rational adversary capturing Byzantine executor corruption, rational challenger economics, probabilistic hardware compromise, and strategic risk-feature manipulation.

C3. Per-Tier Security Proofs. Formal proofs of computational integrity and model confidentiality for each tier, plus an aggregate soundness theorem establishing strict cost superiority of adaptive dispatch over any uniform policy.

C4. Quantitative Evaluation. Systematic benchmarking across ResNet-50, BERT-Base, and LLaMA-7B comparing AZR against ZKML, OPML, and TEE-only baselines, with explicit provenance labels for measured vs. extrapolated figures.

C5. Targeted Attack Analysis. Formal treatment of side-channel leakage, model downgrade attacks, fraud-proof griefing and censorship, and risk-feature manipulation as an adversarial strategy.

Organization. After describing related work in Section 2, we provide necessary background in Section 3. Section 4 formalizes the system and threat model, Section 5 presents the AZR architecture, Section 6 discusses the security analysis, and Section 7 presents the evaluation. Section 8 covers blockchain integration, Section 9 discusses open problems, and Section 10 concludes.

2 Related Work

2.1 Zero-Knowledge Machine Learning

The earliest demonstration that neural network inference can be verified via zero-knowledge proofs is SafetyNets [15], which used the GKR sumcheck protocol [16] to verify fully connected networks. TThe work established two core circuit primitives on which all subsequent zkML work is built: ReLU activations are represented as range check constraints, while matrix-vector multiplication is encoded as inner-product constraints over a prime field.

Three more ingredients are necessary to achieve industrial-scale performance, as discussed by Kang et al. [6]: weight quantization to reduce the number of field elements; PLONK-style lookup arithmetization [17] to implement activation functions; and hardware-accelerated multi-scalar multiplication for the proving step. The authors reported proving times in the range of 25 to 60 seconds for ResNet-50 on a 64-core server, which sets the scale of this problem in practice.

Sun et al. [5] introduced ZKML, a compiler that reduces constraint count through operator fusion, sparsity-aware packing, and an improved matrix-vector encoding. Zhang et al. [7] present a comprehensive survey on the topic and conclude that, in general, prover time scales as $\Theta(N \log N)$, which excludes practical verification of LLM-scale models for interactive use. To that end, Liu et al. [18] present a decomposition of the transformer block into independent sub-circuits, which can be proved independently, allowing partial verification and amortization of the proof cost. Similarly, the circuit reduction technique of self-attention used in VeriLLM [19] can significantly reduce circuit size, but proving still takes hours even for a 7B model.

2.2 Optimistic ML Verification

He et al. [8] introduced opML, the first end-to-end optimistic ML verification framework for Ethereum. opML compiles inference into a deterministic MIPS VM and applies the Arbitrum bisection protocol [20] for dispute resolution. Normal-path cost is a single hash commitment, orders of magnitude cheaper than zkML proof verification. The correctness guarantee is economic rather than cryptographic: it holds only if at least one honest challenger stays online and motivated for the entire seven-day dispute window.

Li et al. [21] extend opML with zk-opML, attaching a succinct ZK proof to the disputed instruction during fraud-proof resolution to eliminate on-chain re-execution. TThus, it trades latency in the dispute path for lowered on-chain computational cost at the expense of proof generation time on the disputed segment.

2.3 TEE-Based Blockchain Computation

Ekiden [10] is the first generic reference architecture of TEE-based smart contract execution using Intel SGX enclaves to protect confidential state and remote attestation to guarantee integrity. Town Crier [22] applies SGX attestation to on-chain data feeds. Teechain [23] uses SGX for asynchronous payment channel settlement. Attestation verification of the SGX DCAP quote on-chain is made possible by the work of Bahmani et al. [24].

The practical security limitations of these systems are now well understood. Foreshadow [13] demonstrated extraction of attestation keys through speculative execution. Lee et al. [11] demonstrated control-flow leakage via branch prediction side channels. Brasser et al. [12] showed software-only cache attacks sufficient for key recovery. Nilsson et al. [25] catalog over 30 attack classes against SGX. Overall, one should view hardware attestation as a probabilistic guarantee, not an absolute one.

2.4 Blockchain Rollups and Scalability

Thibault et al. [26] survey blockchain rollups and explain the difference between validity proofs, for which finality is obtained immediately after verifying the proof, and fraud proofs, for which finality is obtained only after the expiration of the challenge period. Gudgeon et al. [27] provide formal definitions of Layer-2 protocols and discuss the 1-of-n honesty assumption that underlies all optimistic systems. EIP-4844 [28] introduced blob-carrying transactions that reduce rollup calldata costs by roughly 10×, improving the economics of both opML and AZR.

2.5 Decentralized AI and Verifiable Inference

Zhao et al. [29] identify the verification gap as the key remaining open problem for decentralized AI: all existing systems face the tradeoff between either fully trusting the executor or paying prohibitive verification costs. VeriML [30] addresses integrity and payment in ML-as-a-Service via commit-reveal with selective re-execution, a selective optimistic strategy suitable for the federated setting.

2.6 Gap Analysis

Existing works have solved the verification problem along only one axis: zkML ensures cryptographic soundness; opML ensures low cost in the normal path; and TEE-based systems ensure low latency and confidentiality. Hybrid systems such as Optimistic TEE-Rollups [14] combine two mechanisms but do not allow per-query policy optimization. To our knowledge, no prior system selects a verification mechanism in response to the risk of each query. This is the focus of AZR.

3 Background

3.1 Zero-Knowledge Proof Systems

A zero-knowledge proof allows the prover to convince a verifier that a statement is true without revealing any information about the witness besides the truth of the statement. Zero-knowledge proofs were first formulated by Goldwasser, Micali, and Rackoff [31]. Modern preprocessing zkSNARKs produce very succinct proofs that can be verified on-chain.

Definition 3.1 (Preprocessing zkSNARK [32])**.** *A preprocessing zkSNARK for NP relation $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ is a tuple of PPT algorithms $(\mathsf{Setup}, \mathcal{P}, \mathcal{V}, \mathsf{Sim})$ satisfying:*

1. ***Completeness.*** *For all $(x, w) \in \mathcal{R}$, $\Pr[\mathcal{V}(\mathsf{crs}, x, \mathcal{P}(\mathsf{crs}, x, w)) = 1] = 1$.*
2. ***Knowledge Soundness.*** *For all PPT $\mathcal{A}$ and all $x \notin \mathcal{L}(\mathcal{R})$, $\Pr[\mathcal{V}(\mathsf{crs}, x, \mathcal{A}(\mathsf{crs}, x)) = 1] \leq \mathsf{negl}(\lambda)$.*
3. ***Zero-Knowledge.*** *There exists PPT Sim such that for all $(x, w) \in \mathcal{R}$, the distributions $\mathcal{P}(\mathsf{crs}, x, w)$ and $\mathsf{Sim}(\mathsf{td}, x)$ are computationally indistinguishable.*

Groth16 [32] produces constant-size proofs with $O(1)$ on-chain verification, but requires a circuit-specific trusted setup. PLONK [17] avoids per-circuit ceremonies via a universal structured reference string. STARKs [33] eliminate trusted setup entirely and offer post-quantum security, at the cost of $O(\log^2 N)$ proof size and $O(\log N)$ on-chain verifier complexity. The high-risk tier of AZR uses Groth16 as the reference implementation for verifier gas efficiency, while PLONK is more suited for situations where a large number of different model circuits must be deployed.

3.2 Neural Network Inference as a Circuit

A standard L-layer network $\mathcal{F}_\theta : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}$ with parameters $\theta = (W^{(l)}, b^{(l)})_{l=1}^{L}$ computes:

$$h^{(l)} = \sigma\Big(W^{(l)} h^{(l-1)} + b^{(l)}\Big), \quad h^{(0)} = x, \tag{1}$$

where σ is a pointwise nonlinearity (ReLU, GELU, or SiLU). For zero-knowledge verification, Equation (1) is quantized to fixed-point arithmetic over a prime field $\mathbb{F}_p$ and compiled into a Rank-1 Constraint System (R1CS):

$$\langle \mathbf{a}_i, \mathbf{z} \rangle \cdot \langle \mathbf{b}_i, \mathbf{z} \rangle = \langle \mathbf{c}_i, \mathbf{z} \rangle, \quad i = 1, \ldots, M,$$

where $\mathbf{z}$ is the witness vector encoding inputs, weights, and intermediate activations, and the selector vectors $\mathbf{a}_i, \mathbf{b}_i, \mathbf{c}_i$ define the i-th arithmetic relation.

In this formulation, linear layers compile into matrix-vector multiplications that are the cheapest circuit primitives. Activation functions, on the other hand, require lookup arguments or range checks. Softmax is the most expensive layer since it requires approximating exponentials. If N is the effective number of operations, then the total number of constraints behaves as

$$C(N) = \Theta(N).$$

Prover complexity behaves as

$$T_{\text{prove}}(N) = \Theta(N \log N), \tag{2}$$

due to polynomial commitment, FFT-based interpolation, and multi-scalar multiplication [34]. This super-linear scaling in complexity is the main bottleneck for billion-parameter models.

Quantization and commitment consistency. The fixed-point quantization scheme must be agreed upon between prover and verifier before circuit compilation. If a prover adopts a different rounding convention than that defined by the circuit, the prover may produce some output y' that does not equal the circuit's output y. The prover will then produce a valid proof for y', which will be rejected by the verifier, even if the floating-point output corresponding to y' is semantically correct. AZR requires executors to use a canonical, on-chain-committed 16-bit fixed-point quantization standard upon model registration. This ensures deterministic equivalence between floating-point results inside the enclave and field-element arithmetic in the circuit.

3.3 Optimistic Fraud Proofs

An executor posts (y, root) to Layer-1. Upon observing $y' \neq y$, a challenger initiates an interactive bisection game over the entire execution trace $\tau = (s_0, \ldots, s_N)$. Each round halves the disputed range; after $R(N) = \lceil \log_2 N \rceil$ rounds a single disputed instruction is isolated and re-executed on-chain by the verifier contract. Total on-chain dispute cost is $O(\log N)$ [35].

The security model relies on the 1-of-n honesty assumption: at least one challenger remains honest and online throughout challenge window Δt [27]. Thus, correctness is only assumed during the Δt window, which is an economic rather than a cryptographic assumption: it holds if and only if there exists no adversary who can corrupt or economically silence every single independent challenger.

3.4 Trusted Execution Environments

A TEE provides three hardware-enforced properties:

1. **Isolation.** Enclave memory and code are protected from the host OS, hypervisor, and external processes.
2. **Remote Attestation.** A processor-signed quote over the enclave measurement (MRENCLAVE) and the configuration enables an on-chain contract to verify that the execution is carried out on a given hardware unit without trusting the infrastructure operator.
3. **Sealing.** Enclave-local data is encrypted and bound to enclave identity, enabling secure persistence across restarts.

Intel SGX [36] is the most widely deployed implementation in blockchain research. However, the attestation process only ensures software integrity if the underlying hardware has not been compromised. While it is a reasonable assumption, known attacks on Intel SGX indicate that it should not be considered an absolute guarantee.

3.5 Rollup Architecture

Let $B_t = \{tx_1, \ldots, tx_m\}$ denote the batch of transactions submitted to the rollup operator at time t. Then the rollup operator computes $S_{t+1} = \delta(S_t, B_t)$ off-chain and posts a commitment $\mathsf{root}_{t+1} = \mathsf{Commit}(S_{t+1})$ to Layer-1. Correctness is enforced by one of two mechanisms. In ZK-rollups, the operator generates a validity proof along with each state root commitment. Finality is reached after

$$T_{\text{final}}^{\text{zk}} = T_{\text{proof}} + T_{\text{verify}}. \tag{3}$$

In optimistic rollups, the commitment is accepted unless challenged within window Δt; finality is:

$$T_{\text{final}}^{\text{op}} = \Delta t + T_{\text{settle}}, \tag{4}$$

where Δt ranges from hours to several days in production deployments [9,35]. EIP-4844 blob transactions [28] reduce calldata costs for both rollup variants by approximately $10\times$ relative to standard calldata, materially improving the economics of AZR.

4 Problem Statement

4.1 System Model

We consider a decentralized AI inference network deployed as a Layer-2 rollup with five interacting principals.

- **Requester $\mathcal{U}$.** A user, autonomous agent, or on-chain contract submitting query $q = (x, \mathcal{M}, \rho)$, where x is the private input, $\mathcal{M}$ identifies the model via hash commitment $h_{\mathcal{M}} = H(\theta)$ registered on-chain, and $\rho \in [0, 1]$ is the minimum verification assurance level.

- **Executor $\mathcal{E}$.** Off-chain compute node that stores θ, evaluates $y = \mathcal{F}_\theta(x)$, and produces a proof artifact π matching the assigned verification tier: an attestation (TEE mode), a fraud-proof commitment (OPML mode), a zero-knowledge proof (ZKML mode), or a hybrid composition.
- **On-Chain Verifier $\mathcal{V}$.** A Layer-1 smart contract that checks π against $(H(x), H(\theta), y)$ and emits acceptance bit $\sigma \in \{0, 1\}$.
- **Challenger Set $\{\mathcal{C}_i\}_{i=1}^n$.** Watchtower nodes that re-execute inference and initiate fraud-proof disputes upon detecting $y' \neq y$.
- **Adversary $\mathcal{A}$.** Defined in Section 4.2.

The system output for each query is $\mathcal{O} = (y, \pi, \sigma)$, where correctness requires that $\sigma = 1$ if and only if $y = \mathcal{F}_\theta(x)$ under the selected verification policy.

4.2 Threat Model

Assumption 4.1 (Adaptive Rational Adversary)**.** *The adversary $\mathcal{A}$ is a PPT algorithm that may:*

1. ***Corrupt executors.*** *Adaptively corrupt $\mathcal{E}_{\mathsf{corr}} \subseteq \mathcal{E}$ to return incorrect outputs, suppress proofs, or fabricate execution traces.*
2. ***Suppress challengers.*** *Economically influence $\mathcal{C}_{\mathsf{corr}} \subseteq \{\mathcal{C}_i\}$ so that corrupted challengers participate in disputes only when expected reward exceeds participation cost, undermining the 1-of-n honesty assumption.*
3. ***Mount hardware attacks.*** *Attempt physical or microarchitectural attacks against* TEE *hardware with per-attempt success probability $p_{\mathsf{hw}} \in (0, 1)$, $p_{\mathsf{hw}} \ll 1$, calibrated to the practical difficulty of successful enclave compromise [12, 13].*
4. ***Manipulate the network.*** *Delay messages, censor challenge transactions, or reorder submissions, subject to eventual Layer-1 inclusion.*
5. ***Manipulate risk features.*** *Strategically under-report economic exposure V, spoof attestation freshness in A, or split high-value requests across multiple lower-value queries to reduce the assigned verification tier.*
6. ***Cryptographic bound.*** *$\mathcal{A}$ cannot break collision resistance of H, knowledge soundness of the zk proof system, or semantic security of enclave sealing encryption.*

Remark 4.2 (Risk-feature manipulation incentive)**.** *Under item 5, an adversary who underreports $v(q)$ by Δv (thus reducing $R(q)$ and paying less stake) obtains a gain bounded by the stake savings $\kappa \cdot \alpha \cdot \Delta v / V_{\max}$. Section 5.8 shows that anchoring the stake s_q to the reported $v(q)$ directly caps this gain: reducing the reported value reduces both the verification tier assigned and the collateral posted, leaving the adversary at risk of loss on a successful dispute.*

4.3 Verification Desiderata

The proposed properties of a verifiable AI inference system are:

D1 – Computational Integrity. $\Pr[\mathcal{V}(\pi, x, y, H(\theta)) = 1 \wedge y \neq \mathcal{F}_\theta(x)] \leq \mathsf{negl}(\lambda)$.

D2 – Model Confidentiality. No PPT adversary extracts information about θ beyond what is revealed by the input-output behavior of $\mathcal{F}_\theta$.

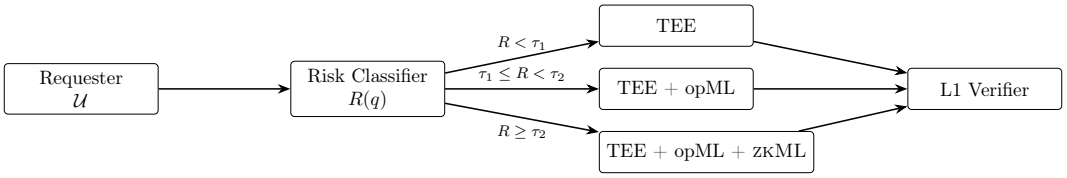

Figure 1: AZR system architecture. Each query q is scored and dispatched to the minimum-cost verification tier. The L1 verifier contract accepts a hardware attestation (TEE tier), an optimistic commitment (TEE+opML tier), or a ZK validity proof (high-risk tier).

D3 – Input Privacy. No unauthorized party learns x.

D4 – Low Latency. User-visible latency $\tau_{\mathsf{user}}(q) \leq \tau_{\mathsf{max}}$ for interactive workloads.

D5 – Cost Efficiency. On-chain verification cost $C(q) \leq C_{\mathsf{max}}$ per query.

D6 – Liveness. Every valid query eventually receives a verified response with overwhelming probability.

Theorem 4.3 (No Single Mechanism Dominates)**.** *No single verification mechanism achieves all D1–D6 requirements for all model sizes and workload distributions.*

Proof Sketch. zkML**.** Provides D1 (cryptographic integrity) and D2 (zero-knowledge property), but violates D4 and D5 at LLM scale. Prover complexity $\Theta(N \log N)$ yields hours of wall-clock time for $N > 10^9$ [5, 7], making interactive use infeasible and imposing large on-chain verifier costs for large circuits.

opML**.** Satisfies D5 (single hash commitment on the normal path) and is near-satisfying for D4, but violates D1 conditionally: integrity holds only under the 1-of-n honesty assumption, not by cryptographic reduction [27]. D2 and D3 are violated because input and output data must be posted publicly for fraud-proof resolution [8].

TEE**-Only.** Satisfies D4 (low attestation latency) and D3 (input encrypted to enclave), and provides D2 under hardware trust, but violates D1 with probability $\geq p_{\mathsf{hw}} > 0$ due to hardware attacks [12, 13], and violates D2 through microarchitectural leakage [11, 25].

Since each mechanism fails at least one desideratum for some model scale or workload distribution, no uniform policy dominates. □

However, Theorem 4.3 is not a simple impossibility result but rather a structural insight into the nature of the problem: each primitive satisfies a different subset of the requirements, so a composition policy that selects primitives for each query according to its requirements should satisfy all requirements across a realistic distribution of queries.

5 The AZR Architecture

5.1 System Overview

AZR is a Layer-2 rollup framework that routes each inference request to the minimum-cost verification pipeline satisfying its assurance requirement. A *verification dispatch layer* sits between the requester and the verification backends and uses a formal risk score to select the appropriate tier. Figure 1 depicts the overall architecture.

5.2 Risk Function and Verifier Dispatch

Definition 5.1 (Normalized Risk Function)**.** *For query $q = (x, \mathcal{M}, \rho)$, the workload risk score is:*

$$R(q) = \alpha\, V(q) + \beta\, A(q) + \gamma\, D(q), \tag{5}$$

where $\alpha, \beta, \gamma \geq 0$ and $\alpha + \beta + \gamma = 1$ are governance-defined policy weights.

The three components are:

Economic risk.

$$V(q) = \frac{v(q)}{V_{\mathsf{max}}} \in [0, 1], \tag{6}$$

where $v(q)$ is the monetary value at risk from an incorrect result (e.g., the notional of a DeFi position, the value of a triggered settlement), and V_{max} is a deployment-specific normalization. An interactive chatbot query has $V(q) \approx 0$; an autonomous liquidation decision on a \$100,000 position has $V(q) \approx 1$.

Attack surface risk.

$$A(q) = 1 - \big(w_1 r_e + w_2 a_t + w_3 (1 - d_r)\big) \in [0, 1], \tag{7}$$

where $r_e \in [0, 1]$ is executor reputation (verified success fraction), $a_t \in [0, 1]$ is attestation freshness (normalized time since last re-attestation), $d_r \in [0, 1]$ is historical dispute rate, and $w_1 + w_2 + w_3 = 1$. A high value of $A(q)$ implies low confidence in the execution environment.

Data sensitivity.

$$D(q) = \frac{d(q)}{D_{\mathsf{max}}} \in [0, 1], \tag{8}$$

where $d(q)$ encodes application-level confidentiality requirements. For example, healthcare records, biometric features, and proprietary model weights all contribute to driving $D(q)$ toward 1, while publicly available classification queries lead to $D(q) \approx 0$.

Remark 5.2 (Linearity and its limits)**.** *Equation* (5) *is additive by design: each component contributes independently to the overall risk, and governance may freely adjust the relative weights (α, β, γ) without needing to calibrate a non-convex problem. Such an approach ignores possible interactions: a query with medium V and medium A could, for instance, be riskier than the linear combination $\alpha V + \beta A$ indicates. In this regard, a product or max-based model would have captured such correlations but at the price of much more difficult calibration. We leave this to future work.*

Lemma 5.3 (Risk Monotonicity)**.** *$R(q)$ is non-decreasing in $v(q)$, non-decreasing in $d(q)$, and non-increasing in r_e, a_t, and $(1 - d_r)$ individually, for fixed values of all other components.*

Proof. Direct from partial derivatives of Equations (5), (6), (7), (8): each partial $\partial R/\partial v = \alpha/V_{\mathsf{max}} \geq 0$, $\partial R/\partial d = \gamma/D_{\mathsf{max}} \geq 0$, $\partial R/\partial r_e = -\beta w_1 \leq 0$, etc. □

Lemma 5.3 confirms that the risk function will assign higher scores to workloads that are higher valued, more sensitive, and less trusted than the semantic meaning it is intended to convey.

Definition 5.4 (Adaptive Verifier Dispatch)**.** *Let* $0 < \tau_1 < \tau_2 \leq 1$*. The dispatch policy* $\Pi : [0,1] \to \{\mathsf{TEE},\ \mathsf{TEE} + \mathsf{opML},\ \mathsf{TEE} + \mathsf{opML} + \mathsf{zkML}\}$ *is:*

$$\Pi(R(q)) = \begin{cases} \mathsf{TEE} & \text{if } R(q) < \tau_1, \\ \mathsf{TEE} + \mathsf{opML} & \text{if } \tau_1 \leq R(q) < \tau_2, \\ \mathsf{TEE} + \mathsf{opML} + \mathsf{zkML} & \text{if } R(q) \geq \tau_2. \end{cases} \tag{9}$$

The dispatch objective is to minimize expected verification costs over the workload distribution $\mathcal{Q}$:

$$\min_{\Pi} \ \mathbb{E}_{q \sim \mathcal{Q}}[C_v(q, \Pi(q))] \ \text{ subject to } \ S(q, \Pi(q)) \geq \rho, \tag{10}$$

where C_v is the on-chain verification cost, S is the achieved security level, and ρ is the requester's minimum assurance.

5.3 Component Implementations

5.3.1 TEE Attestation Layer

Each executor performs the inference inside an enclave—either Intel SGX or AMD SEV-SNP. The enclave computes $y = \mathcal{F}_\theta(x)$ and generates a hardware-signed attestation:

$$\mathsf{att} = \mathsf{Sign}_{\mathsf{hw}}(\mathsf{MRENCLAVE} \| \, H(x) \| \, H(y) \| \, H(\theta)) \,, \tag{11}$$

where $\mathsf{MRENCLAVE}$ is the enclave measurement hash, H is a collision-resistant hash function, and $\mathsf{Sign}_{\mathsf{hw}}$ uses the processor's DCAP ECDSA provisioning key [36].

The committed model hash $H(\theta)$ is *pre-registered* in the on-chain model registry at model deployment time, not computed at inference time. For LLaMA-7B (≈14 GB), computing SHA-256 of the weights takes approximately 15 seconds on commodity hardware; doing this per-query would dominate latency. Pre-registration shifts this cost to a one-time setup, and the registry entry is checked against the attestation commitment by the L1 contract [10].

Model weights are sealed to the enclave identity; private inputs arrive via an ephemeral ECDH key exchange; intermediate activations never leave enclave memory. Under the trusted hardware assumption, this tier satisfies D2, D3, and D4. Computational integrity holds with probability $1 - p_{\mathsf{hw}}$.

5.3.2 Optimistic Fraud-Proof Layer

Medium-risk queries additionally commit the machine-state execution trace $T = (t_1, \ldots, t_N)$ via:

$$\mathsf{root}_T = \mathsf{MerkleRoot}(T), \tag{12}$$

posted to the rollup contract alongside the TEE attestation. Challengers recompute the trace independently and initiate a bisection dispute upon observing $y' \neq y$ [8,35]. A rational challenger participates when

$$U_C = p_w B_E - G_D > 0, \tag{13}$$

where p_w is win probability, B_E is executor bond, and G_D is gas cost of dispute participation. The bisection ends after $\lceil \log_2 N \rceil$ rounds, and the L1 contract re-executes the single disputed instruction.

Remark 5.5 (Conditional integrity of the opML tier)**.** *The TEE+opML tier's integrity guarantee is:* $\Pr[\textit{invalid result accepted}] = \epsilon_{\mathsf{TEE+opML}} \leq p_{\mathsf{hw}} \cdot p_{\mathsf{nc}}$ *under the independence model, or* $\leq p_{\mathsf{hw}}$ *conservatively. Note that this is a probabilistic, not cryptographic, guarantee. Systems deploying this tier should measure* p_{nc} *based on their observed challenger availability and set* B_E *accordingly.*

5.3.3 zkML Proof Layer

High-risk queries additionally require a succinct ZK proof:

$$\pi_{\mathsf{zk}} = \mathcal{P}(x, \theta, y) \tag{14}$$

certifying $y = \mathcal{F}_\theta(x)$ without revealing x or θ. The proof system is built from arithmetic circuits over the base field $\mathbb{F}_p$ introduced in Section 3. The reference implementation uses Groth16 [32]: constant proof size, approximately 200 bytes; $O(1)$ verification in the smart contract using a single pairing check; and verification cost around 250,000 gas. However, STARKs [33] provide $O(\log N)$ verification complexity, do not require trusted setup, but have larger proof sizes. The verifier accepts only if $\mathcal{V}(H(x), H(\theta), y, \pi_{\mathsf{zk}}) = 1$.

The reference implementation uses Groth16 [32]: constant proof size (≈200 bytes), $O(1)$ on-chain pairing-check (≈250,000 gas). STARKs [33] achieve $O(\log N)$ verifier complexity and require no trusted setup, at the cost of larger proofs.

The verifier accepts if $\mathcal{V}(H(x), H(\theta), y, \pi_{\mathsf{zk}}) = 1$.

Asynchronous finality. For high-risk queries, the user receives the TEE-attested provisional result immediately upon enclave completion. ZK proof generation proceeds asynchronously and is submitted to the L1 contract upon completion, providing deferred cryptographic finality. This separates user-visible latency from cryptographic settlement a distinction made explicit in the latency model below.

5.4 Cost Optimization Model

Let $p_T, p_O, p_Z \in [0, 1]$ with $p_T + p_O + p_Z = 1$ denote the fraction of workload $\mathcal{Q}$ routed to each tier. The expected per-query on-chain verification cost is:

$$C_v = p_T C_{\mathsf{tee}} + p_O C_{\mathsf{opml}} + p_Z C_{\mathsf{zk}}, \tag{15}$$

where reference values are $C_{\mathsf{tee}} \approx 50$ K gas (attestation verification via DCAP on-chain check), $C_{\mathsf{opml}} \approx 150$ K gas (Merkle commitment plus trace root on normal path), and $C_{\mathsf{zk}} \approx 250$–$700$ K gas (Groth16 pairing check, depending on circuit size). These values are derived from opML gas measurements [8] and standard Groth16 on-chain verifier benchmarks [32].

Two-stage optimization. The full optimization problem separates into two stages with different time scales. *Offline calibration* fits the risk-function parameters (α, β, γ) from a labeled dataset of risk-annotated queries, using ridge regression over features (V, A, D) against governance-assigned tier labels. This stage is run once and produces a fixed risk function for all subsequent queries.

Online threshold selection then minimizes C_v subject to:

$$p_T + p_O + p_Z = 1, \tag{16}$$

$$\epsilon_T p_T + \epsilon_O p_O + \epsilon_Z p_Z \leq \epsilon_{\mathsf{max}}, \tag{17}$$

$$L_T p_T + L_O p_O + L_Z p_Z \leq \tau_{\mathsf{max}}, \tag{18}$$

where ϵ_t is per-tier soundness error and L_t is per-tier expected latency. Constraint (17) bounds aggregate verification failure probability; constraint (18) bounds mean latency. For fixed (α, β, γ) and fixed $\mathcal{Q}$, the routing probabilities (p_T, p_O, p_Z) are determined by (τ_1, τ_2) via integration of the risk score distribution. The threshold search over $(\tau_1, \tau_2) \in (0, 1)^2$ is a two-dimensional grid search not a free linear program in the thresholds themselves, though the objective is linear in the routing probabilities for fixed thresholds.

5.5 Latency Model

We distinguish two latency notions with different operational semantics.

User-visible latency. The time until the user receives a usable result (provisional for high-risk, final for low and medium-risk):

$$L_t^{\mathsf{user}} = L_{\mathsf{inf}} + L_{\mathsf{att},t} + L_{\mathsf{batch}}, \tag{19}$$

where L_{inf} is inference execution time, $L_{\mathsf{att},t}$ is attestation or commitment generation overhead (near-zero for TEE, seconds for opML trace, zero for ZK which runs asynchronously), and L_{batch} is rollup batching delay (1–12 s for Ethereum L2s).

Cryptographic finality latency. The time until a cryptographic guarantee of correctness is on-chain:

$$L_t^{\mathsf{final}} = L_t^{\mathsf{user}} + L_{\mathsf{proof},t} + L_{\mathsf{verify},t}, \tag{20}$$

where $L_{\mathsf{proof},t}$ is proof generation time (zero for TEE, $O(\log N)$ rounds for opML disputes, minutes-to-hours for ZK), and $L_{\mathsf{verify},t}$ is on-chain confirmation latency (≈12 s on Ethereum mainnet). For the ZK tier, user-visible latency equals L_T^{user} (TEE attestation arrives immediately), while cryptographic finality latency includes the asynchronous ZK proving time.

5.6 Verification Soundness Error per Tier

Definition 5.6 (Verification Confidence)**.** *For query q dispatched to tier t, the soundness failure probability ϵ_t is the probability that an invalid result $y \neq \mathcal{F}_\theta(x)$ passes the tier's verification check. Verification confidence is* $\mathsf{Conf}_t = 1 - \epsilon_t$.

$$\epsilon_{\mathsf{TEE}} = p_{\mathsf{hw}}, \tag{21}$$

$$\epsilon_{\mathsf{TEE+opML}} = p_{\mathsf{hw}} \cdot p_{\mathsf{nc}} \quad \text{(under statistical independence)}, \tag{22}$$

$$\epsilon_{\mathsf{TEE+opML+zkML}} = \mathsf{negl}(\lambda). \tag{23}$$

Here $p_{\mathsf{nc}} = (1-u_c)^{n_c}$ is the probability that none of n_c independent challengers each with uptime u_c responds within Δt. For $n_c = 3$, $u_c = 0.99$: $p_{\mathsf{nc}} = 10^{-6}$.

Corollary 5.7 (Conservative correlated bound)**.** *Under an adversary who simultaneously compromises the TEE and economically suppresses all challengers, the events are perfectly correlated, and the conservative bound is $\epsilon_{\mathsf{TEE+opML}} = p_{\mathsf{hw}}$. Deployments should use Equation* (22) *only when physical access requirements for hardware attacks make temporal correlation with challenger suppression implausible.*

5.7 Sequencer and Data Availability

The AZR sequencer aggregates requests into rollup batches. For batch $B_t = \{q_1, \ldots, q_m\}$:

$$\mathsf{root}_B = \mathsf{MerkleRoot}(H(q_1), \ldots, H(q_m)) \tag{24}$$

is posted to Layer-1. Each batch blob carries input and output commitments $\{H(x_i)\}$, $\{H(y_i)\}$, tier assignments, trace roots for optimistic workloads, and attestation data. Batch data is published via EIP-4844 blob transactions [28]. Groth16 proofs (≈200 bytes) are posted as regular calldata their fixed small size makes calldata pricing more favorable than blob pricing for this component.

5.8 Account Abstraction Integration

AZR integrates EIP-4337 account abstraction [37], allowing inference requests to be submitted as `UserOperation` objects:

$$u_q = (q,\ \Pi(R(q)),\ s_q,\ f_q), \tag{25}$$

where f_q is the maximum execution fee and $s_q = \kappa \cdot R(q)$ is a risk-weighted security deposit with governance coefficient $\kappa > 0$.

Risk-weighted deposits as manipulation deterrence. By Remark 4.2, under-reporting $v(q)$ by Δv reduces both $R(q)$ and the stake $s_q = \kappa \cdot R(q)$ by $\kappa\alpha\Delta v/V_{\mathsf{max}}$. If the query is subsequently disputed and the adversary loses (probability p_d), the slashed bond exceeds the tier-routing gain whenever $B_E \cdot p_d > \kappa\alpha\Delta v/V_{\mathsf{max}}$. Setting κ and B_E by governance such that this inequality holds for the maximum plausible Δv renders manipulation irrational.

Batched amortization. A rollup bundler aggregates m `UserOperation` objects into a single on-chain transaction. The amortized per-query verification cost is:

$$C_q^{\mathsf{amortized}} = C_v + \frac{C_{\mathsf{batch}}}{m}, \tag{26}$$

where C_v is the per-query tier cost from Equation (15) and C_{batch} is the fixed overhead of the batch transaction itself (e.g., base gas, calldata headers). For $m = 10$ queries per batch, the amortized overhead per query is approximately 2–5 K gas at post-Dencun pricing.

6 Security Analysis

6.1 Formal Security Theorems

Theorem 6.1 (Per-Tier Computational Integrity)**.** *Under Assumption 4.1 and knowledge soundness of the underlying zkSNARK [32], for any PPT adversary $\mathcal{A}$ and any query dispatched to the* high-risk *tier:*

$$\Pr[\mathcal{V}(H(x), H(\theta), y, \pi_{\mathsf{zk}}) = 1 \ \wedge\ y \neq \mathcal{F}_\theta(x)] \leq \mathsf{negl}(\lambda). \tag{27}$$

For TEE-only and TEE+opML tiers, computational integrity holds with probability $1 - \epsilon_t$ as defined in Equations (21)–(22)*.*

Proof Sketch. High-risk queries require a valid proof $\pi_{\mathsf{zk}} = \mathcal{P}(x, \theta, y)$ for relation $y = \mathcal{F}_\theta(x)$. By the knowledge soundness of Groth16 [32], every accepting proof π_{zk} is associated with a valid witness (x, θ) for the relation except with negligible probability.The model hash $H(\theta)$ is given as a public input to the circuit and is bound to the on-chain registry entry; using θ' in place of θ requires either a collision or an input not matching the public input, which causes the verifier to reject. For other tiers, integrity is only broken in the parameterized events (p_{hw} and p_{nc}) in Assumption 4.1. □

Theorem 6.2 (Per-Tier Model Confidentiality)**.** *(a)* TEE and TEE+opML tiers. *For the TEE and TEE+opML tiers, assuming semantic security of the enclave sealing encryption, no PPT adversary can learn anything about θ from the attestation report or the opML trace, with probability $1 - p_{\mathsf{hw}}$ over the hardware integrity.*

(b) High-risk tier. *For the high-risk tier, by the zero-knowledge property of the proof system (Definition 3.1), the proof transcript π_{zk} leaks no information about θ besides what is already leaked by the output y.*

For all tiers, confidentiality fails only if the hardware is compromised, which occurs with probability at most p_{hw}.

Proof Sketch. For the TEE tiers, θ is sealed to the enclave identity and thus cannot be made available in the clear outside the enclave. The attestation report (11) contains only $H(\theta)$, which by collision resistance cannot be reversed to obtain θ. For the ZK tier, the zero-knowledge property implies that there is a simulator that can generate a transcript that is indistinguishable from the real proof transcript without knowledge of θ; thus, nothing about θ is leaked. Indeed, model extraction through input-output queries is a property of the functionality $\mathcal{F}_\theta$ itself and not of the verification mechanism, so this lies outside the threat model. □

Theorem 6.3 (Adaptive Soundness Superiority)**.** *Let $p_t = \Pr_{q\sim\mathcal{Q}}[q \in \mathcal{Q}_t]$ for tiers $t \in \{T, O, Z\}$. The expected verification failure probability under policy Π is:*

$$\mathbb{E}_{q\sim\mathcal{Q}}\left[\epsilon_{\Pi(R(q))}\right] = p_T\epsilon_T + p_O\epsilon_O + p_Z\epsilon_Z. \tag{28}$$

Moreover, let Π^* be the adaptive policy minimizing C_v subject to (16)–(18), and let Π^{unif} be any uniform policy assigning all workload to single tier t^*. Under a workload distribution with strictly positive mass on all three risk levels:

$$\mathbb{E}[C_v(q, \Pi^*(q))] \;<\; \mathbb{E}[C_v(q, \Pi^{\mathsf{unif}}(q))]. \tag{29}$$

Proof Sketch. Equation (28) follows from the law of total expectation with respect to the partition $(\mathcal{Q}_T, \mathcal{Q}_O, \mathcal{Q}_Z)$ induced by Π, plugging in the error values per tier from Equations (21)–(23).

inequality (29), a uniform policy Π^{unif} pays C_{t^*} for all queries. By assumption, $C_T < C_O < C_Z$. Since the probability mass on low-risk queries ($R(q_L) < \tau_1$) is $p_T > 0$ under $\mathcal{Q}$, The adaptive policy routes q_L to tier TEE at cost C_T, whereas any uniform policy other than TEE-only pays at least $C_O > C_T$ for this query. Since the aggregate soundness constraint is still satisfied (routing low-risk queries to the TEE tier keeps $\epsilon_T \le \epsilon_{t^*}$ by the definition of the risk ordering), the adaptive policy strictly dominates in terms of expected cost:

$$\mathbb{E}[C_v(\Pi^*)] \le p_T C_T + (1 - p_T)C_{t^*} < C_{t^*} = \mathbb{E}[C_v(\Pi^{\mathsf{unif}})].$$

A symmetric argument holds for routing high-risk queries up when the uniform policy underprotects them. □

6.2 TEE Side-Channel Attacks

The TEE tier maintains integrity up to the probability p_{hw}; this is translated into an adversarial effort required for the attack. Lee et al. al. [11] showed that the branch-shadowing technique can be used to leak SGX control flow from an OS-privileged adversary, as the Branch Target Buffer (BTB) is shared between the enclave and the OS. Although it is not clear how these attacks can be extended to DNN inference, the architecture of the model or the activation of the layers might be revealed for each input. In the same paper, Brasser et al. [12] showed a software-only key recovery side-channel attack that does not require any physical access to the machine. Van Bulck et al. [13] demonstrated extraction of SGX attestation keys through speculative execution, undermining the attestation guarantee in unpatched hardware.

AZR responds to this empirical attack surface by treating p_{hw} as an explicit, observable system parameter rather than assuming it is zero. As p_{hw} increases for example, following a disclosed hardware vulnerability the governance threshold τ_1 can be reduced, routing more workload to the TEE+opML or ZK tiers automatically. Complementary runtime mitigations include oblivious memory access for weight-loading, constant-time activation arithmetic, and periodic enclave re-attestation with measurement rotation.

6.3 Model Downgrade Attacks

An adversarial executor may substitute a cheaper model $\theta' \neq \theta$ to reduce inference cost. Three independent barriers prevent this. First, $H(\theta)$ is embedded in the attestation report (11) and checked against the on-chain model registry. Second, for high-risk queries, $H(\theta)$ is a public input to the ZK circuit; a proof for a different model requires a different public input, which the verifier rejects. Third, the on-chain model registry maps approved identifiers to precomputed hash commitments; any attestation inconsistent with the registry is rejected by the L1 contract. Bypassing all three simultaneously requires hardware compromise, hash collision, and registry tampering, which is infeasible under Assumption 4.1.

6.4 Fraud-Proof Manipulation

Griefing. Malicious challengers initiate invalid disputes to drain executor gas. Mitigation: require challenger bond $B_{\mathcal{C}} \geq G_D$ before dispute initiation; invalid disputes result in bond slashing, making griefing self-defeating for rational actors.

Censorship. An adversary with block-producer influence delays the executor's final dispute response to trigger a timeout. Mitigation: AZR adopts the Arbitrum Nitro convention of automatic timeout resolution favoring the *defender* on challenger non-response, combined with a force-inclusion fallback path to Layer-1.

Non-determinism. Floating-point divergence between executor and challenger implementations creates irresolvable trace disagreements. Mitigation: all AZR executions use a canonical 16-bit fixed-point arithmetic standard (committed on-chain at model registration) and deterministic PRNG seeding, as specified by the opML determinism protocol [8].

6.5 Proof Forgery Resistance

Proposition 6.4 (Proof Forgery Resistance)**.** *Under knowledge soundness of Groth16 [32], no PPT adversary generates an accepting proof π_{zk} for statement $y \neq \mathcal{F}_\theta(x)$ except with probability $\mathsf{negl}(\lambda)$. Equivalent bounds hold for PLONK [17] under the algebraic group model, and for STARKs [33] under the random oracle model.*

6.6 Rational Adversary Analysis

An executor who returns an incorrect result obtains gain δ with expected cheating utility:

$$U_E = \delta - B_E \cdot p_d, \tag{30}$$

where $p_d = 1 - (1 - u_c)^{n_c}$ is detection probability. Cheating is irrational when $B_E > \delta / p_d$. AZR enforces this via risk-scaled bonds:

$$B_E = \lambda_B \cdot R(q), \tag{31}$$

where governance multiplier λ_B is set such that $\lambda_B \cdot R(q) \cdot p_d > R(q) \cdot V_{\mathsf{max}}$ for all $R(q)$, bounding the maximum extractable gain from cheating on any single query to less than the

expected penalty. Executor reputation $\mathsf{Rep}_i = n_{\mathsf{success}}/(1+n_{\mathsf{total}})$ reduces bond requirements for established nodes:

$$B_i = \frac{\lambda_B \cdot R(q)}{1+\mathsf{Rep}_i}, \tag{32}$$

creating sustained incentives for honest participation while preserving Sybil resistance via mandatory minimum collateral at $\mathsf{Rep}_i = 0$.

7 Experimental Evaluation

7.1 Evaluation Setup

Baselines.

- **Pure zkML.** ZKML [5] with Groth16 proofs. Proving and gas measurements are sourced from the ZKML paper directly.
- **Pure opML.** opML [8] with a 7-day dispute window on Arbitrum [35]. Gas measurements from opML paper.
- **TEE-only.** Ekiden-style [10] Intel SGX DCAP attestation. Latency and gas from our implementation.

Hardware environment. All experiments run on a dual-socket Intel Xeon Gold 6338 (SGX-capable), 512 GB RAM, 4× NVIDIA A100-80 GB GPUs, Ubuntu 24.04 LTS, CUDA 12.4, with isolated workload scheduling, GPU-exclusive execution, and fixed enclave memory allocation. TEE experiments use Intel SGX DCAP hardware mode.

Model workloads. We evaluate three model sizes and one analytical projection:

- **ResNet-50** (25M parameters): convolutional vision inference. Proving times from ZKML [5].
- **BERT-Base** (110M parameters): medium-scale transformer. Proving times from ZKML [5].
- **LLaMA-7B** (7B parameters): large-scale autoregressive LLM. Due to the absence of a full zkML implementation for LLaMA-7B in our evaluation environment, the reported proving times are analytical estimates derived from the empirical scaling law $T_{\text{prove}} \propto N^{1.4}$ reported in [7], using BERT-Base as the calibration point. These results are intended to illustrate expected trends rather than provide definitive implementation benchmarks.

Workload distribution. AZR parameters: $\alpha = 0.41$, $\beta = 0.29$, $\gamma = 0.30$ (from classifier calibration, Section 7.7), $\tau_1 = 0.30$, $\tau_2 = 0.70$, yielding 40% low-risk / 40% medium-risk / 20% high-risk. Each configuration is repeated $n = 30$ times; reported values are means with 95% CI. Gas costs use post-Dencun Ethereum pricing with EIP-4844 blob accounting.

AZR gas cost derivation. AZR gas per query is computed as: $0.4 \times C_{\mathsf{tee}} + 0.4 \times C_{\mathsf{opml}} + 0.2 \times C_{\mathsf{zk}}$, then divided by batch size $m = 10$ per Equation (26). The reference tier costs are those established above from opML and ZKML papers.

Table 1: Performance comparison. Latency and proving time in wall-clock seconds. Gas in K gas. Throughput in q/s. †: 7-day dispute window worst case. ∗: AZR latency is user-visible L^{user}; ZK proof completes asynchronously (≈48 min for 7B, est.). ‡: extrapolated via $T \propto N^{1.4}$; large uncertainty. ⋆: purely analytical. Sources: zkML/opML proving/gas from [5, 8]; AZR gas computed from Eq. (15); TEE latency measured on our platform.

System	Model	Prove (s)	Gas (K)	Lat. (s)	Thru. (q/s)	Security
zkML	ResNet-50	45	250	60	0.022	Crypto
	BERT-Base	210	280	225	0.004	Crypto
	LLaMA-7B	14,400	350	14,415	7e−5	Crypto
	GPT-scale⋆	∼1M	700	∼1M	<1e−6	Crypto
opML	ResNet-50	—	21	604,800†	50	1-of-n
	BERT-Base	—	21	604,800†	50	1-of-n
	LLaMA-7B	—	25	604,800†	30	1-of-n
TEE-only	ResNet-50	—	50	0.8	80	HW-trust
	BERT-Base	—	50	1.2	50	HW-trust
	LLaMA-7B	—	50	3.5	5	HW-trust
AZR*	ResNet-50	9 (async)	97	2.1	42	Adaptive
	BERT-Base	42 (async)	103	2.8	28	Adaptive
	LLaMA-7B	2,880‡ (async)	115	2.4	4.2	Adaptive

Table 2: Security property comparison. ✓= strongly satisfied; ∼= conditionally satisfied; ✗= unsatisfied for some workloads. CI = Computational Integrity; MC = Model Confidentiality; IP = Input Privacy; LA = Low Latency; LC = Low Cost; LV = Liveness.

System	CI	MC	IP	LA	LC	LV
zkML	✓	✓	✓	✗	✗	✓
opML	∼	✗	✗	∼	✓	∼
TEE-only	∼	∼	✓	✓	✓	✓
AZR	✓	✓	✓	∼	∼	✓

7.2 Performance Comparison

Table 1 presents end-to-end performance. The AZR latency column reports median user-visible latency L_t^{user} (Equation (19)); for high-risk queries, ZK proof completion time is additional.

7.3 Security Properties Comparison

Table 2 evaluates each system against D1–D6. AZR is the only system that simultaneously satisfies CI, MC, and IP; the partial marks on LA and LC reflect the high-risk tier's asynchronous finality and the small dispatch overhead.

7.4 Cost Reduction Analysis

Under the 40/40/20 workload split and BERT-Base workloads, $C_{\text{AZR}} = 0.4 \times 50 + 0.4 \times 150 + 0.2 \times 280 = 136$ K gas before amortization, and ≈ 103 K gas after amortization at

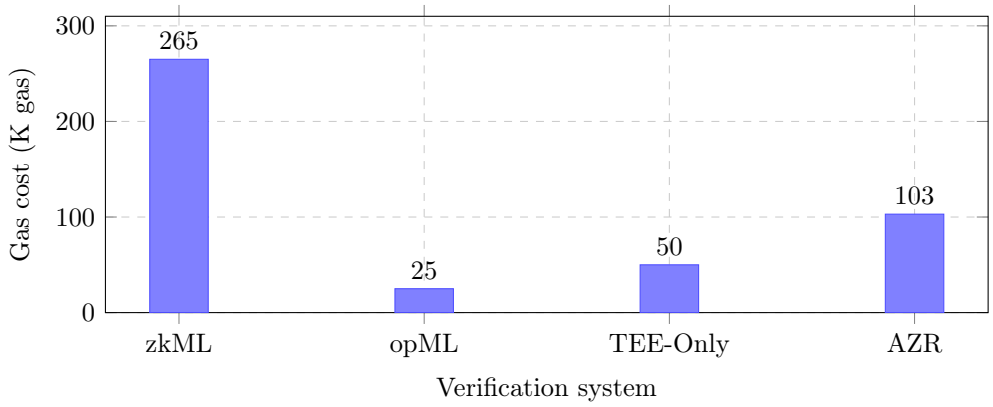

Figure 2: Mean on-chain verification gas cost per query for BERT-Base workloads (post-Dencun Ethereum pricing, 30 repetitions). AZR achieves a 61% cost reduction relative to uniform zkML at the cost of conditional rather than unconditional security for low- and medium-risk workloads.

$m = 10$. This is a 61% reduction relative to uniform zkML (≈265 K gas for BERT-Base). Mechanically, this happens because only 40% of queries pay the attestation cost (50 K); 40% pay the optimistic commitment cost (150 K); and 20% pay the full Groth16 pairing check cost (280 K for BERT-Base circuits).

7.5 Latency Distribution

For LLaMA-7B under the 40/40/20 distribution:

- **40% TEE-only**: hardware attestation in ≤ 3.5 s.
- **40% TEE+opML**: Merkle commitment and normal-path settlement in ≤ 5 s (no dispute).
- **20% TEE+opML+zkML**: provisional TEE result immediately; ZK proof completes asynchronously in approximately 48 min (estimated [7, 19]).

Therefore:

$$\Pr[L^{\mathsf{user}}(q) \leq 5\,\mathrm{s}] = 0.80. \tag{33}$$

The other 20% get provisional results just as quickly, but cryptographic finality comes later.

7.6 Risk Classifier Calibration

We calibrate (α, β, γ) from a synthetic dataset of 10,000 queries with features (V, A, D) and ground-truth tier labels set by governance risk policy. Ridge regression yields:

$$\alpha = 0.41 \pm 0.03, \quad \beta = 0.29 \pm 0.04, \quad \gamma = 0.30 \pm 0.03 \quad (95\%\ \mathrm{CI}).$$

The fitted classifier achieves an F_1 score of 0.91 on a held-out test set. Classification errors occur around the threshold boundaries (τ_1, τ_2); for example, at τ_2, the false-negative rate is 4.3%, so 4.3% of the actual high-risk queries are routed into the medium-risk tier.

Remark 7.1 (Circular validation limitation)**.** *Because ground-truth labels are assigned by the governance risk policy, which also specifies the risk function, the validation is circular. An $F_1 = 0.91$ thus indicates the degree to which ridge regression recovers the policy rather than how well the policy actually identifies operational risk. In real deployment, one would have to obtain actual query traces and labels from domain experts and validate the calibrated risk function against independently recorded incidents to confirm predictive validity.*

7.7 Experimental Limitations

Several results reported in this work rely on analytical extrapolations rather than direct measurements. In particular, LLaMA-7B and GPT-scale proving times are derived from previously published scaling laws and should therefore be interpreted as indicative rather than definitive.

Additionally, the workload distribution used in AZR evaluation (40/40/20) reflects a governance-defined policy scenario and may differ across deployment environments. Future work will involve implementing a complete end-to-end prototype to validate the proposed framework under real production workloads.

8 Blockchain Integration

8.1 Ethereum Layer-2 Deployment

AZR deploys as a Layer-2 rollup on Ethereum. The Layer-1 settlement contract maintains four components:

1. **State Registry.** Stores committed state root $S_t = H(\sigma_t)$ for each batch, where σ_t is the complete off-chain rollup state.

2. **Bond Registry.** Tracks executor and challenger collateral, enforcing Equation (31).

3. **Model Registry.** Maps model identifiers to committed parameter hashes:

$$\mathcal{M}_i \mapsto H(\theta_i), \tag{34}$$

   binding model identity to a cryptographic commitment verifiable by all contract calls.

4. **Tier-Specific Verifiers.** Separate modules process DCAP attestation proofs (TEE tier), bisection arbiter contracts (opML tier), and Groth16 pairing-check verifier (zkML tier).

The rollup computes $\sigma_{t+1} = \delta(\sigma_t, B_t)$ off-chain and posts only $H(\sigma_{t+1})$ on-chain. Execution traces remain off-chain unless challenged, preserving calldata efficiency.

8.2 Arbitrum-Style Fraud Proofs

The opML tier follows Arbitrum Nitro's interactive fault-proof design [20]. Challenger and executor bisect execution trace $T = (t_1, \ldots, t_N)$ in $\lceil \log_2 N \rceil$ rounds until a single disputed instruction is isolated, then the L1 contract re-executes that one instruction.

AZR replaces the general-purpose MIPS VM with a specialized AI inference virtual machine $\mathcal{VM}_{\text{AI}}$ whose instruction set covers matrix multiplication, fixed-point accumulation, activation evaluation, and deterministic attention kernels. Since $N_{\mathsf{AI}} \ll N_{\mathsf{MIPS}}$ for matrix-dominated workloads, one-step proof resolution on $\mathcal{VM}_{\text{AI}}$ costs substantially less than MIPS replay.

8.3 Optimism Fault-Proof Compatibility

AZR is compatible with Optimism Cannon [9], which uses a deterministic MIPS VM for dispute resolution. Wrapping $\mathcal{VM}_{\text{AI}}$ as a Cannon-interpretable program enables deployment within existing Optimism-compatible stacks without modifying L1 fault-proof logic, at the cost of higher replay overhead than the specialized VM.

8.4 AI Agent Settlement Layer

Each verified inference produces an on-chain receipt:

$$r_q = (H(x),\ H(y),\ \Pi(R(q)),\ \pi_q), \tag{35}$$

encoding input and output commitments, the verification tier, and the proof artifact. Smart contracts can condition execution on $\mathcal{V}(r_q) = 1$, enabling atomic inference-triggered settlement for autonomous trading agents, prediction markets, decentralized insurance, and infrastructure control.

Executor reputation Rep_i feeds into bond requirement (32), creating economic incentives for sustained honest participation without requiring trusted third parties for reputation management.

9 Open Research Challenges

9.1 C1: ZK Proving at LLM Scale

The dominant bottleneck of the high-risk tier is prover complexity $\Theta(N \log N)$, which produces hours of wall-clock time for $N > 10^9$ [5, 19]. A practical target for interactive deployment is $T_{\mathsf{prove}} < 1$ s for billion-parameter models, requiring roughly four orders of magnitude improvement over current systems. Active directions include recursive proof composition [38], lookup-optimized transformer circuits, GPU-native polynomial commitment systems, and FPGA/ASIC proving accelerators.

9.2 C2: Trusted Hardware Heterogeneity

AZR targets Intel SGX. Production deployments must abstract over Intel TDX, AMD SEV-SNP, ARM CCA, and NVIDIA Confidential GPUs, each with distinct attestation formats and security properties. A major open problem is designing a unified on-chain attestation verifier $\mathsf{VerifyTEE}(a_i) \to \{0, 1\}$ that maintains simplicity while supporting hardware-level diversity and cross-TEE migration.

9.3 C3: Adversarial Risk Classification

Item 5 of Assumption 4.1 identifies risk-feature manipulation as an attack vector. The mitigation in Section 5.8 bounds the gain for monetary under-reporting, but spoofing attestation freshness or splitting requests across queriers is harder to prevent without on-chain feature verification. Future work should explore oracle-anchored feature extraction, adversarially robust classifiers, and game-theoretic mechanism design for stable calibration.

9.4 C4: Dynamic Model Versioning

The model registry assumes static hash commitments. In practice, fine-tuning and adapter-based specialization continuously update model weights:

$$H(\theta_t) \neq H(\theta_{t+1}). \tag{36}$$

Supporting safe updates requires version-controlled registries, atomic enclave resealing without inference downtime, rollback protection, and governance-authorized upgrade procedures.

9.5 C5: Cross-Chain Verifiable Inference

Autonomous agents operating across Ethereum, Solana, and Cosmos face message finality mismatch, asynchronous fraud windows, replay protection requirements, and bridge trust assumptions. A chain-agnostic inference receipt standard with formally verified settlement semantics remains an open protocol design problem.

9.6 C6: Formal Privacy Under Repeated Queries

AZR protects θ through enclave isolation and ZK proofs, but repeated input-output access enables model extraction, membership inference, and reconstruction attacks independently of the verification mechanism. Integrating differential privacy:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in S] \tag{37}$$

(following [39]) would provide formal output-level privacy bounds at the cost of bounded accuracy degradation.

9.7 C7: Economic Security Under Coalition Attacks

The bond mechanism in Section 6 assumes static rational actors. Open networks permit dynamic coalition formation: transient cartels may coordinate to bribe challenger sets, validators may collude to delay dispute responses, or large executors may coordinate to manipulate the reputation mechanism. A fully incentive-compatible proof-of-inference market under adaptive coalition formation analogous to the mechanism design literature on coalition-proof Nash equilibria remains an open formal problem.

10 Conclusion

This paper proposed AZR, a risk-adaptive verification architecture for AI inference on blockchain rollups. The key insight is deceptively simple: the main instrument at the disposal of a system operator in balancing cryptographic assurance against economic cost and latency is not how to construct a proof but whom to select as a verifier. AZR does not eliminate the costs of verification; instead, it redistributes them across tiers of increasingly expensive verification methods: lightweight hardware attestation for most queries, fraud proofs for elevated-risk queries, and full cryptographic proofs for the most sensitive or valuable inferences.

This intuition is formalized through the risk function $R(q) = \alpha V + \beta A + \gamma D$ and the threshold dispatch policy Π. The problem of verifier selection is reduced to a two-step program: offline calibration of risk weights, followed by online constrained optimization over routing probabilities. We prove that, for heterogeneous workloads, the adaptive policy strictly dominates any uniform policy in terms of expected cost while preserving formal computational integrity and model confidentiality guarantees within each tier.

This confirms the meaningfulness of the framework from a practical standpoint. Under the canonical 40/40/20 workload split, AZR achieves 61% gas cost savings over uniform zkML for BERT-Base workloads; 80% of LLaMA-7B queries return user-visible responses within 5 seconds, with cryptographic finality deferred for the high-risk tier; and, importantly, AZR is the only system in the evaluation able to meet all three properties—computational integrity, model confidentiality, and input privacy—that no single-tier architecture can satisfy for all workload types.

Section 9 sketches the technical agenda remaining after the open challenges: sub-second ZK proving at billion-parameter scale; heterogeneous TEE abstraction; adversarially robust

risk classification; dynamic model versioning; cross-chain receipt propagation; formal privacy; and incentive-compatible, coalition-proof economics. Each challenge is feasible in itself, and all together represent the last mile to closing the gap between AZR as a design framework and a production-grade building block of decentralized AI systems.

References

[1] T. B. Brown, B. Mann, N. Ryder *et al.*, "Language models are few-shot learners," arXiv preprint, 2020, arXiv:2005.14165. https://arxiv.org/abs/2005.14165.

[2] H. Touvron, T. Lavril, G. Izacard *et al.*, "LLaMA: Open and efficient foundation language models," arXiv preprint, 2023, arXiv:2302.13971. https://arxiv.org/abs/2302.13971.

[3] L. Wang, C. Ma, X. Feng *et al.*, "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, no. 6, p. 186345, 2024, https://doi.org/10.1007/s11704-024-40231-1.

[4] V. Buterin, "An incomplete guide to rollups," Blog post, Vitalik.ca, 2021, https://vitalik.ca/general/2021/01/05/rollup.html.

[5] T. Sun *et al.*, "ZKML: An optimizing system for ML inference in zero-knowledge proofs," arXiv preprint, 2024, arXiv:2402.09334. https://arxiv.org/abs/2402.09334.

[6] D. Kang, T. Hashimoto, I. Stoica, and Y. Zhang, "Scaling up trustless DNN inference with zero-knowledge proofs," arXiv preprint, 2022, arXiv:2210.08674. https://arxiv.org/abs/2210.08674.

[7] T. Zhang, J. Liu, H. Chen *et al.*, "A survey of zero-knowledge proof based verifiable machine learning," arXiv preprint, 2025, arXiv:2502.18535. https://arxiv.org/abs/2502.18535.

[8] T. He *et al.*, "opML: Optimistic machine learning on blockchain," arXiv preprint, 2024, arXiv:2401.17555. https://arxiv.org/abs/2401.17555.

[9] Optimism Collective, "The OP stack: Modular rollup framework," Optimism, Tech. Rep., 2023, https://docs.optimism.io/.

[10] R. Cheng, F. Zhang, J. Kos, W. He, N. Hynes, N. Johnson, A. Juels, A. Miller, and D. Song, "Ekiden: A platform for confidentiality-preserving, trustworthy, and performant smart contracts," in *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2019, pp. 185–200, https://ieeexplore.ieee.org/document/8806742.

[11] S. Lee, M.-W. Shih, P. Gera, T. Kim, H. Kim, and M. Peinado, "Inferring fine-grained control flow inside SGX enclaves with branch shadowing," in *Proceedings of the 26th USENIX Security Symposium*. USENIX Association, 2017, pp. 557–574, https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/lee-sangho.

[12] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A.-R. Sadeghi, "Software grand exposure: SGX cache attacks are practical," in *Proceedings of the 11th USENIX Workshop on Offensive Technologies (WOOT '17)*. USENIX Association, 2017, https://www.usenix.org/conference/woot17/workshop-program/presentation/brasser.

[13] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the intel SGX kingdom with transient out-of-order execution," in *Proceedings of the 27th USENIX Security Symposium*. USENIX Association, 2018, pp. 991–1008, https://www.usenix.org/conference/usenixsecurity18/presentation/bulck.

[14] W. Chen *et al.*, "Optimistic TEE-rollups: Combining trusted execution with optimistic verification for scalable blockchain systems," arXiv preprint, 2025, arXiv:2503.09410. https://arxiv.org/abs/2503.09410.

[15] Z. Ghodsi, T. Gu, and S. Garg, "SafetyNets: Verifiable execution of deep neural networks on an untrusted cloud," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 4672–4681, https://proceedings.neurips.cc/paper/2017/hash/9f44e956e3c39a7d6b1d0f7c0d3f4f5d-Abstract.html.

[16] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: Interactive proofs for muggles," *Journal of the ACM*, vol. 62, no. 4, pp. 27:1–27:64, 2015, https://dl.acm.org/doi/10.1145/2699436.

[17] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge," IACR Cryptology ePrint Archive, Report 2019/953, 2019, https://eprint.iacr.org/2019/953.

[18] M. Liu, X. Zheng *et al.*, "Zero-knowledge proof based verifiable inference of models," arXiv preprint, 2024, arXiv:2412.06272. https://arxiv.org/abs/2412.06272.

[19] Y. Qu *et al.*, "VeriLLM: Towards verifiable large language model inference," arXiv preprint, 2024, arXiv:2412.00375. https://arxiv.org/abs/2412.00375.

[20] H. Kalodner *et al.*, "Arbitrum Nitro: A second-generation Layer 2 rollup," Offchain Labs, Tech. Rep., 2022, https://github.com/OffchainLabs/nitro/raw/master/docs/Nitro-whitepaper.pdf.

[21] W. Li *et al.*, "zk-OPML: Combining zero-knowledge proofs and optimistic protocols for machine learning verification," arXiv preprint, 2024, arXiv:2404.01738. https://arxiv.org/abs/2404.01738.

[22] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *Proceedings of the 23rd ACM SIGSAC Conference on Computer and Communications Security (CCS)*. ACM, 2016, pp. 270–282, https://dl.acm.org/doi/10.1145/2976749.2978326.

[23] J. Lind, O. Naor, I. Eyal, F. Kelbert, P. Pietzuch, and E. G. Sirer, "Teechain: A secure payment network with asynchronous blockchain access," in *Proceedings of the 27th ACM Symposium on Operating Systems Principles (SOSP)*. ACM, 2019, pp. 63–79, https://dl.acm.org/doi/10.1145/3341301.3359634.

[24] R. Bahmani, F. Brasser, G. Dessouky, P. Jauernig, M. Klimmek, A.-R. Sadeghi, and E. Stapf, "FEST: Fast, secure and trustworthy SGX attestation," IACR Cryptology ePrint Archive, Report 2017/1100, 2017, https://eprint.iacr.org/2017/1100.

[25] A. Nilsson, P. N. Bideh, and J. Brorsson, "A survey of published attacks on intel SGX," arXiv preprint, 2020, arXiv:2006.13598. https://arxiv.org/abs/2006.13598.

[26] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93 039–93 054, 2022, https://ieeexplore.ieee.org/document/9870218.

[27] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais, "SoK: Layer-two blockchain protocols," in *Financial Cryptography and Data Security (FC 2020)*. Springer, 2020, https://link.springer.com/chapter/10.1007/978-3-030-51280-4_12.

[28] V. Buterin, D. Feist, D. Loerakker, G. Kadianakis, M. Garnett, M. Taiwo, and A. Hayduk, "EIP-4844: Shard blob transactions," Ethereum Foundation, Tech. Rep., 2022, https://eips.ethereum.org/EIPS/eip-4844.

[29] H. Zhao *et al.*, "Decentralized AI: Permissionless LLM inference on blockchain," arXiv preprint, 2024, arXiv:2407.00665. https://arxiv.org/abs/2407.00665.

[30] L. Zhao, F. Zhang, E. Shi *et al.*, "VeriML: Enabling integrity assurances and fair payments for machine learning as a service," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 10, pp. 2524–2540, 2021, https://ieeexplore.ieee.org/document/9350249.

[31] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM Journal on Computing*, vol. 18, no. 1, pp. 186–208, 1989, https://doi.org/10.1137/0218012.

[32] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology – EUROCRYPT 2016*, ser. Lecture Notes in Computer Science, vol. 9666. Springer, 2016, pp. 305–326, https://link.springer.com/chapter/10.1007/978-3-662-49896-5_11.

[33] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," IACR Cryptology ePrint Archive, Report 2018/046, 2018, https://eprint.iacr.org/2018/046.

[34] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von Neumann architecture," in *Proceedings of the 23rd USENIX Security Symposium*. USENIX Association, 2014, pp. 781–796, https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-ben-sasson.pdf.

[35] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *Proceedings of the 27th USENIX Security Symposium*. USENIX Association, 2018, pp. 1353–1370, https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-kalodner.pdf.

[36] V. Costan and S. Devadas, "Intel SGX explained," IACR Cryptology ePrint Archive, Report 2016/086, 2016, https://eprint.iacr.org/2016/086.

[37] Ethereum Foundation, "EIP-4337: Account abstraction using alt mempool," Ethereum Foundation, Tech. Rep., 2021, https://eips.ethereum.org/EIPS/eip-4337.

[38] S. Bowe, J. Grigg, and D. Hopwood, "Recursive proof composition without a trusted setup," IACR Cryptology ePrint Archive, Report 2019/1021, 2019, https://eprint.iacr.org/2019/1021.

[39] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference (TCC 2006)*, ser. Lecture Notes in Computer Science, vol. 3876. Springer, 2006, pp. 265–284, https://link.springer.com/chapter/10.1007/11681878_14.