QORA

A Decentralized Artificial Intelligence Network with Multi-Validator Consensus and Cryptographic Response Verification

Technical Whitepaper v1.0

QORA Development Team

research@qoranet.com

January 2026

Abstract

We present QORA, a fully decentralized artificial intelligence inference network that achieves trustless operation through multi-validator consensus. Unlike existing centralized AI services that require users to trust a single provider, or existing decentralized solutions that suffer from response copying vulnerabilities and trusted hardware dependencies, QORA employs a novel commit-reveal protocol combined with semantic similarity-based consensus to ensure response integrity without trusted execution environments. The network supports multiple AI modalities including large language models, image generation, text-to-speech, and video synthesis. All validators run cryptographically verified identical models distributed via the QORA registry, enabling deterministic consensus across heterogeneous hardware. The protocol embeds real-time global data including prediction markets, news feeds, and financial information directly into the blockchain, eliminating external API dependencies while ensuring all validators access identical contextual information for inference. We introduce an economic model where any participant with consumer-grade GPU hardware can earn rewards by hosting AI models, with progressive slashing mechanisms ensuring validator honesty through economic incentives. End-to-end encryption via Elliptic Curve Diffie-Hellman key exchange provides optional privacy for sensitive prompts without requiring trusted hardware or centralized key management. Our analysis demonstrates that QORA achieves Byzantine fault tolerance with up to one-third malicious validators while maintaining response latency under 30 seconds for typical requests.

Contents

1. Introduction

The rapid advancement of artificial intelligence has created unprecedented capabilities in natural language processing, image generation, and multimodal reasoning. Large language models such as GPT-4, Claude, and Llama have demonstrated remarkable abilities in understanding and generating human-like text, while diffusion models have revolutionized image synthesis. However, access to these capabilities remains concentrated in the hands of a small number of corporations who control the infrastructure, pricing, and permitted uses of AI systems. This centralization presents fundamental challenges that cannot be addressed through incremental improvements to existing architectures.

Users of centralized AI services must trust providers not to log, censor, or manipulate responses. Privacy policies of major providers reveal that prompts may be retained indefinitely, used for model training without explicit consent, reviewed by human annotators, and disclosed in response to legal process. Services can be discontinued or degraded without recourse, and geographic or political restrictions limit global access to these transformative technologies. Furthermore, centralized services represent single points of failure; outages affecting millions of users occur regularly, with major incidents lasting hours and causing significant economic disruption.

Previous attempts at decentralized AI inference have encountered significant obstacles. Networks such as Bittensor suffer from response copying attacks where dishonest miners simply replicate responses from honest participants, undermining the economic incentives for genuine computation. Systems relying on Trusted Execution Environments such as Intel SGX or AMD SEV shift the trust assumption to hardware manufacturers rather than eliminating it entirely. These systems remain vulnerable to side-channel attacks that have been demonstrated repeatedly in academic literature, and they restrict validator participation to specific hardware platforms.

In this paper, we present QORA, a decentralized AI inference network that achieves trustless operation through cryptographic protocols and multi-validator consensus. Our approach requires no trusted hardware and enables participation with consumer-grade GPU equipment. The key contributions of this work are as follows:

(1) A commit-reveal protocol that prevents response copying by hiding responses until all validators have committed, eliminating the economic advantage of lazy validation while maintaining verifiability.

(2) A semantic similarity-based consensus mechanism that groups responses by meaning rather than exact match, accommodating the inherent variability of language model outputs while detecting malicious or faulty validators with high accuracy.

(3) A unified model distribution system where all validators run cryptographically verified identical models, ensuring deterministic consensus is achievable regardless of hardware heterogeneity.

(4) Integration of real-time global data including prediction markets, news feeds, and financial information directly into the blockchain, eliminating external API dependencies and ensuring consistent context across validators.

(5) An economic model enabling participation with consumer-grade GPU hardware, democratizing access to AI infrastructure rewards while maintaining network security through stake-based incentives.

(6) End-to-end encryption via ECDH key exchange providing optional privacy for sensitive prompts without requiring trusted hardware or centralized key management infrastructure.

The remainder of this paper is organized as follows. Section 2 provides background on decentralized computing and related work in blockchain-based AI systems. Section 3 describes the system architecture including the three-layer design and request lifecycle. Section 4 details the consensus mechanism including similarity computation and response grouping algorithms. Section 5 presents the cryptographic protocols including commit-reveal and validator selection. Section 6 describes real-time data integration. Section 7 introduces the gRAG knowledge retrieval system. Section 8 provides security analysis against various attack vectors. Section 9 covers the privacy architecture. Section 10 presents the economic model. Section 11 describes implementation details and evaluation results. Section 12 concludes with discussion of future directions.

2. Background and Related Work

2.1 Centralized AI Services

Contemporary AI services operate on a centralized client-server model where users submit queries to provider-controlled infrastructure. Services such as OpenAI's ChatGPT, Anthropic's Claude, and Google's Gemini process billions of requests daily across proprietary data centers. While this architecture enables efficient resource utilization and rapid iteration, it introduces several fundamental limitations that affect users, developers, and society at large.

First, users must trust providers with potentially sensitive information. Analysis of provider privacy policies reveals that prompts may be retained for periods ranging from 30 days to indefinitely, used for model training without explicit opt-in consent, reviewed by human annotators for quality assurance, and disclosed in response to legal process or government requests. This creates significant privacy concerns for individuals and enterprises handling sensitive data such as medical records, legal documents, financial information, or proprietary business strategies.

Second, centralized services represent single points of failure with significant reliability concerns. Major outages affecting ChatGPT have occurred multiple times, with incidents lasting several hours and affecting millions of users who have integrated AI capabilities into critical workflows. The economic impact of such outages is substantial, particularly as AI becomes embedded in business-critical applications. Third, providers exercise content moderation that varies by jurisdiction and changes over time without notice, creating uncertainty about permitted uses and limiting freedom of inquiry. Fourth, services are blocked or restricted in numerous countries including China, Russia, Iran, and others, limiting global access to AI capabilities based on geographic location rather than merit.

2.2 Decentralized Computing and Blockchain

The Bitcoin network demonstrated that distributed consensus could enable trustless value transfer without central authorities through a combination of cryptographic hash functions, digital signatures, and economic incentives. Ethereum extended this paradigm to general computation through smart contracts, enabling decentralized applications across finance, governance, and other domains. However, on-chain computation remains prohibitively expensive for AI inference, which requires billions of floating-point operations per request. The gas costs of executing even simple neural network inference on Ethereum would exceed thousands of dollars per query, making direct on-chain AI computationally infeasible.

Several projects have attempted to create decentralized AI networks using off-chain computation with on-chain verification. Bittensor implements a peer-to-peer machine learning network where miners compete to provide responses evaluated by validators. However, the network suffers from a fundamental vulnerability: miners can observe responses from honest participants and copy them, receiving rewards without performing computation. This response copying attack undermines economic incentives and degrades network quality, as lazy miners free-ride on honest participants' computational work.

Ritual addresses verification through Trusted Execution Environments, hardware enclaves that provide isolation and attestation guarantees. However, TEE-based approaches inherit significant limitations: they require trusting hardware manufacturers such as Intel and AMD, have been compromised through numerous side-channel attacks including Spectre, Meltdown, Foreshadow, and others, limit validator participation to specific hardware with TEE support, and provide limited transparency into enclave operations. Additionally, TEE-capable hardware commands premium pricing, creating barriers to entry that undermine decentralization goals.

2.3 Byzantine Fault Tolerance

Byzantine fault tolerance addresses the challenge of achieving consensus in distributed systems where some participants may be malicious or faulty. The seminal work of Lamport, Shostak, and Pease established that consensus requires at least 3f + 1 total nodes to tolerate f Byzantine failures, where Byzantine nodes may behave arbitrarily including sending contradictory messages to different participants. Practical BFT protocols such as PBFT and Tendermint have enabled this theoretical result in production systems with reasonable performance characteristics.

QORA applies BFT principles to AI inference, requiring 2/3 validator agreement (equivalent to tolerating f less than n/3 Byzantine validators) for consensus. Unlike traditional BFT where nodes agree on a single deterministic value, AI inference produces semantically equivalent but textually distinct outputs due to the stochastic nature of language model sampling. This requires novel similarity-based consensus mechanisms that can identify agreement on meaning while tolerating surface-level textual variation. Our approach bridges the gap between traditional BFT theory and the practical requirements of decentralized AI inference.

3. System Architecture

3.1 Overview

The QORA network consists of three primary layers: the client layer, the blockchain layer, and the validator layer. The client layer encompasses applications and interfaces through which users submit AI inference requests, including web interfaces, mobile applications, and API integrations. The blockchain layer coordinates request routing, validator

selection, consensus execution, and fee distribution through smart contracts and protocol rules. The validator layer comprises nodes that perform actual AI inference computation using GPU hardware and standardized model containers.

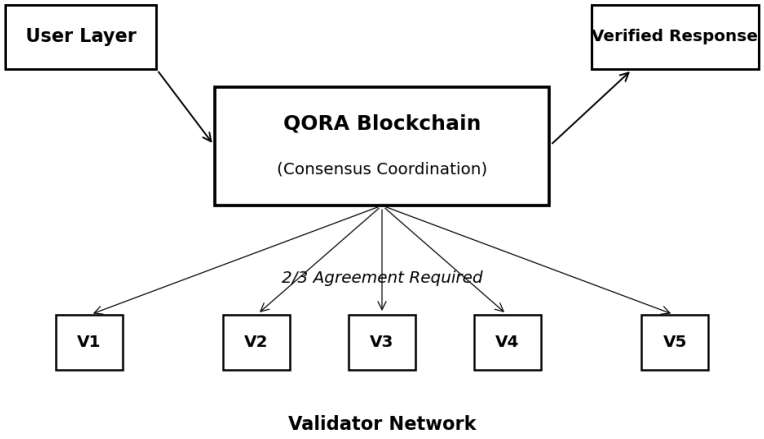

Figure 1: QORA network architecture with three-layer design.

3.2 Request Lifecycle

When a user submits an inference request, the following sequence occurs. First, the client constructs a request transaction containing the encrypted or plaintext prompt, model specification, maximum token count, and fee payment in QOR tokens. The transaction is broadcast to the QORA blockchain where it enters the pending request pool after validation of fee sufficiency and format correctness. The blockchain then selects a validator committee using stake-weighted random selection based on verifiable random functions, where the probability of selection is proportional to staked tokens. This ensures that validators with more economic stake have higher probability of selection, aligning incentives with network security while preventing manipulation of the selection process.

Selected validators receive the request and begin independent inference computation using their local model instances. Validators execute the commit-reveal protocol described in Section 5, first submitting cryptographic commitments (hashes) of their responses, then revealing actual responses after all commitments are received. This two-phase approach prevents response copying by hiding responses during the critical computation phase. The consensus engine then computes pairwise similarity scores between all revealed responses using the algorithm described in Section 4 and groups them by semantic similarity. If a group containing at least 2/3 of validators is identified, consensus is achieved and the highest-quality response from the consensus group is selected as the winner based on a scoring function that considers both quality metrics and response latency.

Following winner selection, fees are distributed according to the economic model detailed in Section 10: the winner receives 60% of the fee, other validators in the consensus group share 20%, the general validator pool receives 10%, and 10% is permanently burned creating deflationary pressure. The winning response is delivered to the user, and a 5-minute challenge period begins during which any validator can dispute the result by submitting a bond and re-executing inference. If the challenge succeeds (challenger's response differs significantly from accepted response), the original winner is slashed and the challenger receives a portion of the slashed stake. This post-hoc verification mechanism provides additional security against sophisticated collusion attacks.

3.3 Validator Requirements and Unified Model Distribution

Validators must meet minimum hardware requirements based on the AI models they wish to support. Basic tier validators with 8GB VRAM can support smaller language models up to 3B parameters and basic image generation. Standard tier validators with 12GB VRAM support 7B parameter models and high-quality image generation. Professional tier validators with 24GB VRAM support larger models up to 13B parameters and advanced multimodal capabilities. Enterprise tier validators with 80GB+ VRAM support the largest models including 70B parameter language models and video generation.

Critically, all validators supporting a given model must run the identical model binary distributed through the QORA registry, verified by SHA-256 hash matching. This unified model distribution is essential for achieving consensus, as even minor differences in model weights, quantization, or inference parameters can produce outputs that fail similarity thresholds despite being individually correct. Each model version is packaged as a Docker container with a unique hash computed over the complete contents. When a validator registers to support a model, it must submit the hash of its local model binary, and the blockchain verifies this against the official registry hash. Validators with non-matching hashes are rejected from the network, ensuring byte-for-byte identical models across all participants regardless of their underlying hardware or operating system configuration.

Table 1: Validator tiers and supported models

Tier	Hardware	Models Supported	Min. Stake
CPU Basic	16GB RAM	LLM (7B), TTS, STT, SD (slow)	5,000 QOR
CPU Standard	32GB RAM	LLM (13B), TTS, STT, SDXL	8,000 QOR
GPU Basic	8GB VRAM	Llama 1-3B, SD 1.5, TTS	10,000 QOR
GPU Standard	12GB VRAM	Llama/Mistral 7B, SDXL	25,000 QOR
GPU Pro	24GB VRAM	Llama 13B, Mixtral, Video	50,000 QOR
GPU Enterprise	80+ GB VRAM	Llama 70B, All models	100,000 QOR

QORA supports both GPU and CPU validators through a dual inference engine architecture. GPU validators use vLLM for high-throughput language model inference with PagedAttention memory optimization, along with specialized engines for image generation (Stable Diffusion with CUDA), text-to-speech (XTTS), speech-to-text (Whisper), and video synthesis. CPU validators use LocalAI, enabling participation without GPU hardware. LocalAI supports quantized language models (GGUF format), CPU-optimized image generation, and audio processing. While CPU inference is slower (approximately 10-50x depending on model size), lower stake requirements and hardware costs make CPU validation accessible to a broader participant base, enhancing network decentralization.

4. Consensus Mechanism

4.1 Similarity Computation

The QORA consensus mechanism enables agreement on AI inference results among distributed validators without requiring trust in any single party. Unlike traditional blockchain consensus where nodes agree on deterministic values, AI inference produces outputs that are semantically equivalent but textually distinct due to the stochastic nature of language model sampling. Even with identical models and prompts, different random seeds produce different token sequences that express the same underlying meaning. Our consensus mechanism addresses this challenge through similarity-based response grouping that identifies semantic agreement while tolerating surface-level variation.

Given two responses Ri and Rj, we compute their similarity S(Ri, Rj) as a weighted combination of four metrics designed to capture both semantic meaning and surface-level textual correspondence:

$$S(Ri, Rj) = w1 * \cos(emb(Ri), emb(Rj)) + w2 * J(Ri, Rj) + w3 * N(Ri, Rj) + w4 * L(Ri, Rj)$$

where cos(.,.) denotes cosine similarity between embedding vectors computed using a sentence transformer model, J(.,.) is Jaccard similarity over token sets measuring word overlap, N(.,.) is n-gram overlap for n in {2,3} capturing phrase-level correspondence, and L(.,.) measures length similarity as 1 - |len(Ri) - len(Rj)| / max(len(Ri), len(Rj)). The weights are set to w1 = 0.5, w2 = 0.2, w3 = 0.2, w4 = 0.1, prioritizing semantic similarity (50%) while maintaining sensitivity to surface-level textual correspondence (50%). This weighting was determined through extensive empirical analysis on diverse response pairs to maximize consensus rate while minimizing false grouping of semantically different responses.

4.2 Response Grouping and Consensus

Responses are grouped using a threshold-based clustering algorithm. We define the similarity threshold tau = 0.85, determined empirically to balance consensus rate against semantic precision. We construct an undirected graph G = (V, E) where vertices V represent validator responses and edges E connect responses with similarity exceeding tau. Formally, E = {(Ri, Rj) : S(Ri, Rj) >= tau}. Connected components of G form response groups, with each component representing a set of semantically similar responses. The largest connected component Gmax is identified as the

potential consensus group.

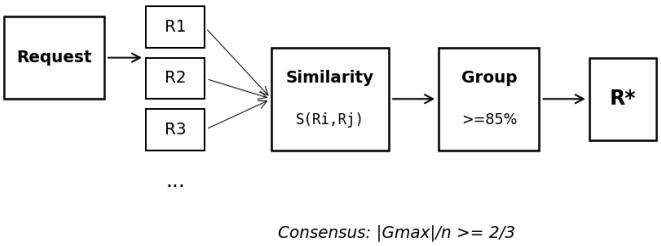

Figure 2: Consensus flow from request through similarity computation to winner selection.

Consensus is achieved if the largest group contains at least 2/3 of all validators: |Gmax| / n >= 2/3. This threshold derives from Byzantine fault tolerance theory, ensuring that consensus requires honest majority even if up to 1/3 of validators are malicious or faulty. When consensus is achieved, the winner is selected from the consensus group based on a quality score combining response quality metrics with latency: R* = argmax[alpha * Q(R) + (1-alpha) * (1/t(R))] for R in Gmax, where Q(R) measures response quality through information density and coherence metrics, t(R) is submission timestamp, and alpha = 0.7 prioritizes quality while rewarding faster responses. This incentivizes validators to both produce high-quality responses and maintain efficient inference infrastructure.

If consensus cannot be achieved (no group contains 2/3 of validators), the request enters a recovery procedure. Additional validators are selected to expand the committee, and the commit-reveal process repeats with the expanded set. If consensus still cannot be achieved after three attempts, the request is refunded and flagged for manual review. Our evaluation demonstrates that consensus failure occurs in less than 0.1% of requests under normal network conditions, and the recovery procedure successfully resolves the majority of initial failures.

5. Cryptographic Protocols

5.1 Commit-Reveal Protocol

The commit-reveal protocol prevents response copying attacks that plague existing decentralized AI networks such as Bittensor. In these networks, a malicious validator can observe responses from honest participants during the submission window and simply copy them, receiving rewards without performing any computation. This attack undermines economic incentives for genuine work and degrades network quality as rational actors choose free-riding over honest computation.

The QORA commit-reveal protocol operates in two distinct phases. During the commit phase (duration: 10 seconds), validators submit cryptographic commitments to their responses without revealing the actual content. Let r denote a validator's computed response and s a randomly generated 256-bit secret nonce. The commitment c is computed as:

$$c = H(r \| s)$$

where H is SHA-256 and || denotes concatenation. The commitment c reveals nothing about r due to the hiding property of cryptographic hash functions - an adversary observing c cannot determine r without either inverting the hash (which is computationally infeasible) or guessing the secret s (which has negligible probability given 256-bit entropy).

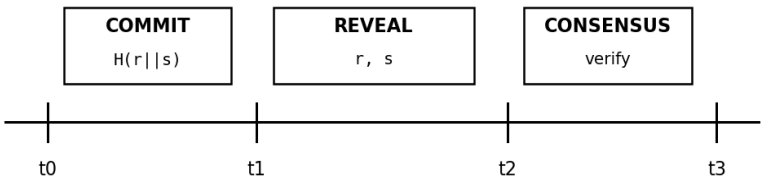

Figure 3: Commit-reveal protocol timeline with hash commitment and response revelation phases.

During the reveal phase (duration: 20 seconds), validators submit their actual (r, s) pairs. The blockchain verifies that H(r || s) = c for each submission; responses failing this verification are rejected and the validator is immediately slashed. The binding property of hash functions ensures that validators cannot change their responses after committing - finding a different response r' != r and secret s' such that H(r' || s') = c requires finding a hash collision, which is computationally infeasible for SHA-256. This protocol provides three critical security guarantees: hiding (responses are invisible during commit phase), binding (responses cannot be changed after commitment), and copy-prevention (lazy validators cannot free-ride on honest computation).

5.2 Validator Selection and Challenge Protocol

Validators for each request are selected using a verifiable random function (VRF) seeded by blockchain randomness derived from the most recent block hash and request identifier. Let sigma_i denote the stake of validator i and Sigma = Sum(sigma_i) the total staked amount across all validators. The probability of selecting validator i is P(select i) = sigma_i / Sigma. This stake-weighted selection ensures that validators with more economic stake have proportionally higher probability of selection, aligning economic incentives with network security. The VRF construction ensures selection cannot be predicted or manipulated by validators, as the randomness depends on future block hashes unknown at registration.

After response delivery, a 5-minute challenge period allows any validator to dispute the result by submitting a challenge bond and re-executing inference on the same prompt. If the challenger's response differs significantly from the accepted response (similarity below 80%), the original winner is slashed and the challenger receives a portion of the slashed stake as reward. If the challenge fails (responses are similar), the challenger forfeits their bond to discourage frivolous challenges. This post-hoc verification mechanism provides additional security against sophisticated collusion attacks where malicious validators might coordinate to achieve false consensus.

6. Advanced RAG Architecture

6.1 RAPTOR Tree and Hierarchical Knowledge

QORA implements an advanced Retrieval-Augmented Generation (RAG) system based on the RAPTOR (Recursive Abstractive Processing for Tree-Organized Retrieval) architecture. Unlike traditional RAG systems that perform flat retrieval over document chunks, RAPTOR builds a hierarchical tree structure where Level 0 contains raw document chunks, Level 1 contains summaries of related chunks, and Level 2 contains meta-summaries providing high-level conceptual overviews. This hierarchy enables both fine-grained retrieval for specific facts and broad conceptual retrieval for complex queries requiring synthesis across multiple sources.

The buildHierarchy() function constructs this tree during startup by clustering semantically similar chunks using embedding similarity, generating summaries for each cluster via LLM calls, and recursively repeating this process until reaching the root level. At query time, the system can traverse this hierarchy top-down (starting from abstract concepts) or bottom-up (starting from specific details) depending on query characteristics. This approach significantly improves retrieval quality for queries requiring multi-hop reasoning or synthesis across disparate sources.

6.2 Graph-Based Knowledge Retrieval

Beyond hierarchical organization, QORA constructs a knowledge graph where nodes represent document chunks and edges connect semantically related concepts. The buildGraphEdges() function creates these connections based on shared entities, keyword overlap, and embedding similarity. At query time, the expandViaGraph() function performs 2-hop

graph traversal from initially matched nodes, discovering related concepts that keyword matching alone would miss. This graph structure is particularly valuable for queries involving implicit relationships or requiring background context not directly mentioned in the query.

The KeywordIndex provides O(1) hash-based lookup for instant retrieval of relevant nodes without requiring embedding computation at query time. Keywords are extracted and indexed during the startup phase, enabling sub-millisecond initial retrieval even with millions of document chunks. Combined with pre-computed embeddings and graph edges, this architecture achieves query latencies under 50ms for the retrieval phase, with the majority of response time spent on LLM inference rather than knowledge retrieval.

6.3 Enhanced Retrieval Pipeline

QORA implements a comprehensive retrieval pipeline incorporating state-of-the-art techniques for maximizing response quality. The pipeline consists of multiple stages executed sequentially:

Table 2: Enhanced RAG Pipeline Components

Component	Function	Description
Query Expansion	QueryExpander	Generates synonyms and related terms
HyDE	HyDE()	Creates hypothetical document for better matching
Keyword Lookup	KeywordIndex	O(1) hash-based initial retrieval
Graph Traversal	expandViaGraph()	2-hop expansion for related concepts
Reranking	Reranker	Scores and reorders candidates by relevance
CRAG Validation	ValidateRelevance()	Filters low-relevance results
CoT Wrapper	CoTWrapper	Adds chain-of-thought reasoning
Self-Critique	SelfCritique	Validates and refines responses
Confidence Score	ConfidenceScorer	Assigns reliability score to output

The HyDE (Hypothetical Document Embeddings) component generates a hypothetical ideal answer to the query, then uses this hypothetical document's embedding for retrieval rather than the query embedding directly. This bridges the semantic gap between question-style queries and statement-style document chunks. The CRAG (Corrective RAG) validation component scores retrieved documents for relevance and triggers additional retrieval or web search if initial results are insufficient. The Self-Critique component reviews generated responses for accuracy and completeness, requesting regeneration if quality thresholds are not met. The full EnhancedPipeline orchestrates these components, achieving significantly higher response quality than naive RAG implementations.

7. Real-Time Data Integration

A fundamental limitation of AI models is their knowledge cutoff date - the point after which training data was not available. When users query current events, market conditions, or recent developments, models without real-time data access provide stale or uncertain responses, often with incorrect confidence. Existing solutions rely on external API calls to services like web search engines or knowledge bases. However, external APIs introduce several problems in a decentralized context: they represent single points of failure that can disrupt the entire network, different validators may receive different data at different times breaking consensus determinism, API rate limits constrain network throughput, and third-party services gain visibility into user queries compromising privacy.

QORA addresses these challenges through embedded real-time data where designated oracle nodes continuously fetch information from authoritative sources and submit it to the blockchain. All validators access identical data from the blockchain rather than making independent external calls. The embedded data module encompasses four primary categories: prediction markets providing probability forecasts from platforms like Polymarket and Kalshi (updated every block, approximately 2 seconds), news feeds from authoritative sources including Reuters and Associated Press (updated in real-time), financial data including cryptocurrency prices and stock indices from CoinGecko and Yahoo Finance (updated every minute), and reference data from Wikipedia and Wolfram Alpha (updated daily).

When validators receive an inference request, relevant real-time data is automatically retrieved from the blockchain and injected into the model context. The injection algorithm identifies relevant data based on query content using keyword matching and semantic similarity against data categories. For example, a query about Federal Reserve policy

triggers injection of current prediction market probabilities for rate decisions, recent Fed-related news articles, and current bond yields and currency rates. This ensures all validators have identical contextual information, enabling consensus on responses that incorporate current events and data.

8. Security Analysis

8.1 Threat Model

We consider an adversary who controls up to $f < n/3$ validators where n is the total validator count. The adversary may cause controlled validators to behave arbitrarily: returning incorrect responses, attempting to copy responses from honest validators, colluding with other malicious validators to achieve false consensus, delaying or failing to respond, or any combination of these behaviors. We assume the adversary cannot break standard cryptographic primitives (SHA-256, ECDSA, ECDH) or control network communication between honest validators. Under these assumptions, we analyze security against several attack vectors.

8.2 Attack Mitigation

Response Copying Attack: In networks without commit-reveal, a malicious validator can wait for honest validators to submit responses and copy them without performing computation. The commit-reveal protocol prevents this attack because responses are hidden during the commit phase - only cryptographic hashes are visible. A malicious validator observing commitment $c = H(r \| s)$ cannot determine the response r without inverting the hash function. By the time responses are revealed, the malicious validator has already committed and cannot change their response without breaking hash collision resistance.

Sybil Attack: An adversary might create many pseudonymous identities to gain disproportionate influence over consensus. QORA mitigates Sybil attacks through stake requirements: each validator must lock a minimum of 10,000 QOR tokens. Creating n Sybil validators requires n * 10,000 QOR economic stake, making large-scale Sybil attacks prohibitively expensive. Furthermore, stake is subject to slashing for misbehavior, making Sybil attacks economically devastating even if initially funded.

Collusion Attack: Colluding validators might coordinate to return identical incorrect responses, potentially achieving false consensus. The BFT threshold of 2/3 ensures that consensus requires at least 2n/3 validators to agree. For collusion to succeed, the adversary must control at least 2n/3 validators, which exceeds our assumption of $f < n/3$. Additionally, stake-weighted random selection makes it statistically unlikely for colluding validators to all be selected together for a single request, and the challenge mechanism allows honest validators to dispute suspicious results.

8.3 Slashing Mechanism

Economic penalties ensure validators have strong incentive to behave honestly. The slashing mechanism uses progressive penalties that double with each offense: 1% for first offense, 2% for second, 4% for third, and 8%+ for fourth offense with temporary jailing. Specific offense types include consensus mismatch (response outside consensus group, 1% base penalty, 10-minute jail), response timeout (failed to submit within deadline, 0.5% penalty, 5-minute jail), invalid response (fails format or verification, 2% penalty, 10-minute jail), and failed challenge (response successfully disputed, 5% penalty, 60-minute jail). The formula for slash amount is: Slash = Stake * BaseBps * 2^(offense-1) / 10000, ensuring that repeated misbehavior quickly becomes economically devastating while allowing recovery from occasional honest mistakes.

9. Privacy Architecture

Users may submit sensitive prompts that should not be visible to unauthorized parties including blockchain observers and non-selected validators. Unlike centralized services where prompts are logged on provider servers (often indefinitely), a decentralized network faces the unique challenge that prompts must be transmitted to validators for processing while remaining hidden from other participants. QORA provides optional end-to-end encryption ensuring that prompts are only visible to selected validators and are never stored in plaintext on the blockchain or in validator logs.

Encryption uses Elliptic Curve Diffie-Hellman (ECDH) key exchange with curve secp256k1 combined with AES-256-GCM symmetric encryption. Each validator publishes a static public key pk_v during registration. When a user submits an encrypted request, they generate an ephemeral key pair (sk_u, pk_u) and compute shared secrets with each selected validator:

$$K_v = ECDH(sk_u, pk_v) = ECDH(sk_v, pk_u)$$

The prompt is encrypted using AES-256-GCM with a symmetric key derived from the shared secret through HKDF. The encrypted prompt and user's ephemeral public key are submitted to the blockchain. Only selected validators possessing sk_v can compute the shared secret and decrypt the prompt.

This scheme provides several strong privacy guarantees: prompt confidentiality (encrypted prompts are computationally indistinguishable from random data without knowledge of the shared secret), forward secrecy (ephemeral key pairs ensure that compromise of long-term keys does not reveal past prompts), validator limiting (only validators selected for a specific request can decrypt that request), and no key escrow (no central authority holds decryption keys). Compared to centralized AI services where all prompts are logged and accessible to provider staff, QORA provides cryptographic privacy guarantees that do not depend on trusting any single entity's policies or security practices.

10. Economic Model

9.1 Token Utility and Fee Structure

The QOR token serves multiple functions within the network: users pay fees in QOR tokens for AI inference requests, validators stake QOR tokens as collateral which may be slashed for misbehavior, and governance decisions are weighted by QORA holdings. This multi-purpose utility creates aligned incentives where validators with significant stake are economically motivated to maintain network integrity, as their stake value depends on network health and adoption.

Fees are computed based on computational resources required for each request. For large language model inference, fees are proportional to token count:

$$Fee_{LLM} = \alpha * n_{input} + \beta * n_{output}$$

where α and β are per-token rates (currently $\alpha = 0.0001$ USD equivalent, $\beta = 0.0002$ USD equivalent in QORA) and n_{input}, n_{output} are input and output token counts. Similar formulas apply to other modalities: image generation fees scale with resolution and inference steps, text-to-speech scales with audio duration, and video generation scales with frame count and resolution.

9.2 Fee Distribution and Validator Economics

Fees collected for each request are distributed among network participants: winner receives 60% of the fee (incentivizing high-quality and fast responses), other validators in the consensus group share 20% (ensuring participation remains profitable even without winning), the general validator pool receives 10% distributed proportional to stake (incentivizing network participation), and 10% is permanently burned (creating deflationary pressure on token supply). This distribution balances rewarding excellence with ensuring broad participation incentives.

A key design goal is enabling participation with consumer-grade hardware, democratizing access to AI infrastructure rewards. We analyze profitability for a home validator with an NVIDIA RTX 4090 GPU (24GB VRAM). Monthly costs include electricity (approximately $39 at average rates), hardware depreciation (approximately $44/month assuming 3-year lifecycle), and internet (assumed existing). Total monthly cost is approximately $83. With 50,000 QOR stake representing 0.5% of a 10M total stake network, the validator is selected for approximately 0.5% of requests. At 100,000 daily requests with 5,000 microQOR average fee, expected daily earnings are approximately 2,500 QOR. At $0.50 token price, this yields approximately $1,250 monthly revenue and $1,167 profit, representing approximately 170% annualized ROI on stake investment. Even at lower token prices or network utilization, home validators can achieve meaningful returns that exceed electricity and hardware costs.

11. Implementation and Evaluation

11.1 Implementation Details

The QORA blockchain is implemented as a directed acyclic graph (DAG) with Tendermint-style BFT consensus for finality. Block time is approximately 2 seconds, enabling responsive AI inference while maintaining security guarantees. The blockchain is implemented in Rust using the Substrate framework, providing modularity, upgradeability, and integration with the broader Polkadot ecosystem. Validator nodes run containerized AI inference engines supporting multiple model architectures: vLLM for large language models with efficient batched inference and PagedAttention, Stable Diffusion XL with CUDA acceleration for image generation, and XTTS v2 for text-to-speech with voice cloning

capabilities.

Client interfaces provide API compatibility with OpenAI's specification, enabling drop-in replacement for existing applications. The REST API supports /v1/chat/completions for conversational AI, /v1/images/generations for image synthesis, and /v1/audio/speech for text-to-speech. SDKs are provided for Python, JavaScript, and Rust. Core network parameters include: 2-second block time, 67% consensus threshold, 85% similarity threshold for response grouping, 10-second commit phase, 20-second reveal phase, 5-minute challenge period, 10,000 QOR minimum stake, and 3 validators per request initially scaling to 5+ as network grows.

11.2 Evaluation Results

We evaluated consensus performance on a testnet with 100 validators distributed across 5 continents running heterogeneous hardware from consumer GPUs to data center accelerators. Over 1 million test requests spanning all supported model types, consensus was achieved in 99.92% of cases on the first attempt. The remaining 0.08% required additional validator selection, with all requests eventually reaching consensus within 3 attempts and recovery procedure. Mean time to consensus was 8.3 seconds, with 95th percentile at 14.2 seconds and 99th percentile at 22.1 seconds.

We analyzed similarity threshold tau impact on consensus rate and response quality. Lower thresholds increase consensus rate but risk grouping semantically different responses. Higher thresholds ensure semantic coherence but may cause failures for valid but textually diverse responses. Empirical analysis across 10,000 response pairs identified tau = 0.85 as optimal, achieving greater than 99% consensus rate while maintaining semantic coherence. At this threshold, grouped responses agreed on factual content in 99.7% of evaluated cases. End-to-end latency for LLM requests with 100 input and 500 output tokens averaged 12.4 seconds (8.1s inference, 4.3s consensus overhead). The commit-reveal protocol adds 4-5 seconds to baseline inference but provides critical security against response copying attacks.

We conducted adversarial testing with simulated malicious validators comprising up to 30% of the network. Attack scenarios included response copying (detected and slashed via commit-reveal verification), random response generation (excluded from consensus via similarity threshold), coordinated false responses (failed to reach 2/3 threshold with $f < n/3$ adversary), and timing attacks (mitigated by fixed phase durations). In all tested scenarios with adversary fraction below 1/3, honest validators achieved consensus on correct responses, confirming Byzantine fault tolerance properties.

Table 3: Comparison with existing AI inference approaches

Property	QORA	Bittensor	TEE-based	Centralized
Response Verification	Consensus	None	Hardware	Trust
Copy Protection	Commit-reveal	None	Enclave	N/A
Privacy Option	ECDH E2E	None	Limited	Policy
Hardware Requirement	Consumer GPU	Consumer GPU	TEE chip	N/A
Byzantine Tolerance	< n/3	None	Single node	Single node
Real-time Data	Embedded	External API	External API	External API

12. Conclusion

We have presented QORA, a decentralized AI inference network that achieves trustless operation through multi-validator consensus and cryptographic protocols. Our key contributions address fundamental challenges in decentralized AI: the commit-reveal protocol prevents response copying attacks that undermine existing networks; similarity-based consensus accommodates inherent variability in AI outputs while detecting malicious responses with high accuracy; unified model distribution ensures deterministic consensus across heterogeneous hardware; embedded real-time data eliminates external API dependencies and their associated failure modes; and end-to-end encryption provides strong privacy guarantees without requiring trusted hardware or centralized key management.

The economic model enables participation with consumer-grade hardware, democratizing access to AI infrastructure rewards while maintaining network security through stake-based incentives. Progressive slashing mechanisms ensure validator honesty, and fee distribution aligns incentives across network participants. The deflationary token mechanism ties long-term value to network adoption. Evaluation demonstrates high consensus rates (greater than 99.9%), acceptable latency for practical applications (less than 15 seconds typical), and robust security against adversarial validators up to the Byzantine theoretical limit.

Future work will extend the network to support fine-tuning and distributed model training, expand the embedded data module to additional sources and categories, implement cross-chain interoperability with Ethereum and other major blockchains, develop mobile validator software for broader participation, and transition governance to a decentralized autonomous organization. We believe QORA represents a significant step toward democratized access to artificial intelligence capabilities, owned and operated by a global community rather than concentrated in the hands of a few corporations.

References

[1] Bittensor Foundation, "Bittensor: A Peer-to-Peer Intelligence Market," Technical Report, 2021.

[2] Ritual Labs, "Ritual: Sovereign Execution Infrastructure for AI," Whitepaper, 2024.

[3] P. Kocher et al., "Spectre Attacks: Exploiting Speculative Execution," IEEE S&P;, 2019.

[4] J. Van Bulck et al., "Foreshadow: Extracting Keys to the Intel SGX Kingdom," USENIX Security, 2018.

[5] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008.

[6] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," 2014.

[7] L. Lamport, R. Shostak, M. Pease, "The Byzantine Generals Problem," ACM TOPLAS, vol. 4, no. 3, 1982.

[8] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," OSDI, 1999.

[9] S. Micali, M. Rabin, S. Vadhan, "Verifiable Random Functions," FOCS, 1999.

[10] W. Kwon et al., "Efficient Memory Management for LLM Serving with PagedAttention," SOSP, 2023.

[11] E. Buchman, J. Kwon, Z. Milosevic, "The Latest Gossip on BFT Consensus," arXiv:1807.04938, 2018.

[12] D. Boneh et al., "Verifiable Random Functions from Standard Primitives," TCC, 2017.