Proof of AI: A Consensus Mechanism for Verifiable Compute Contribution

Version 2022.04

Zach Kelling[*]

Hanzo Industries *Lux Industries* *Zoo Labs Foundation*

research@zoo.ngo

April 2022

Abstract

We introduce **Proof of AI** (PoAI), a novel consensus mechanism designed for networks where participants contribute AI compute rather than hash power or stake. Unlike Proof of Work which wastes energy on cryptographic puzzles, or Proof of Stake which concentrates power among token holders, PoAI rewards nodes for performing *useful* AI computation: inference, training, and experience extraction. Our protocol combines three components: (1) **Compute Verification** using Trusted Execution Environments (TEEs) to attest that claimed work was performed, (2) **Quality Scoring** via multi-party evaluation to assess the value of contributed compute, and (3) **Attestation Aggregation** using BLS signatures for efficient on-chain verification. We prove that PoAI achieves Byzantine fault tolerance under standard assumptions while incentivizing economically rational behavior. Experimental evaluation on a 100-node testnet demonstrates 847 TPS throughput with 2.3-second finality, while rewarding nodes proportional to their AI contribution quality.

Keywords: consensus mechanism, proof of useful work, AI compute verification, trusted execution environments

1 Introduction

Blockchain consensus mechanisms face a fundamental challenge: how to fairly allocate block production rights and rewards. Proof of Work (PoW) solves this by requiring computational effort, but the computation serves no purpose beyond securing the chain—an estimated 150 TWh annually for Bitcoin alone. Proof of Stake (PoS) reduces energy waste but creates plutocratic dynamics where the wealthy accumulate more wealth.

The rise of AI compute networks presents an opportunity to align consensus with useful work. AI training and inference require substantial compute resources; if this compute could simultaneously secure a blockchain, we would achieve both useful work and decentralized consensus.

1.1 Challenges

Designing PoAI requires solving several challenges:

1. **Verification**: How can validators confirm that claimed AI work was actually performed?

[*]zach@zoo.ngo

2. **Quality Assessment**: Not all AI compute is equally valuable—training a useful model differs from generating noise.

3. **Sybil Resistance**: Without proof of identity, adversaries could claim credit for others' work.

4. **Efficiency**: Verification must be cheaper than the original computation.

5. **Incentive Compatibility**: Rational nodes must be incentivized to perform high-quality work.

1.2 Contributions

This paper makes the following contributions:

1. **PoAI Protocol**: A complete consensus mechanism combining TEE attestation, quality scoring, and BLS aggregation (Section 3)

2. **Compute Verification**: Novel techniques for verifying AI workloads without re-executing them (Section 4)

3. **Quality Scoring**: Multi-party evaluation protocol for assessing AI output value (Section 5)

4. **Security Analysis**: Proofs of Byzantine fault tolerance and incentive compatibility (Section 6)

5. **Evaluation**: Experimental results demonstrating scalability and efficiency (Section 7)

2 Background

2.1 Proof of Useful Work

Prior attempts at useful PoW include:

- **Primecoin** [1]: Finding prime number chains
- **Gridcoin** [2]: BOINC scientific computing
- **FoldingCoin**: Protein folding simulations

These approaches suffer from limited useful work domains and difficulty of verification. Our work extends to general AI computation.

2.2 Trusted Execution Environments

TEEs provide hardware-enforced isolation:

Definition 2.1 (Trusted Execution Environment)**.** *A TEE is a hardware-protected region where:*

1. *Code and data are encrypted in memory*
2. *Execution cannot be observed or modified by the host OS*
3. *Remote attestation proves what code ran and its output*

Major TEE implementations:

- Intel SGX / TDX
- AMD SEV-SNP
- ARM TrustZone / CCA
- NVIDIA Confidential Computing

2.3 BLS Signatures

BLS signatures enable efficient aggregation:

Definition 2.2 (BLS Signature Aggregation)**.** *Given signatures* $\{\sigma_i\}_{i=1}^n$ *from signers* $\{pk_i\}$ *on message* m*:*

$$\sigma_{agg} = \prod_{i=1}^{n} \sigma_i \tag{1}$$

Verification: $e(\sigma_{agg}, g) = e(H(m), \sum_i pk_i)$

This enables $O(1)$ verification of n signatures.

3 PoAI Protocol

3.1 System Model

Participants:

- **Workers**: Perform AI computation within TEEs
- **Validators**: Verify attestations and score quality
- **Aggregators**: Combine attestations for on-chain submission

Assumptions:

- At least $2f + 1$ of $3f + 1$ validators are honest
- TEE hardware is not compromised (standard TEE assumption)
- Network is partially synchronous (bounded delay after GST)

3.2 Protocol Overview

PoAI operates in epochs:

1. **Work Phase**: Workers perform AI computation in TEEs
2. **Attestation Phase**: Workers submit TEE attestations
3. **Scoring Phase**: Validators evaluate work quality
4. **Aggregation Phase**: Attestations aggregated via BLS
5. **Finalization**: Block produced, rewards distributed

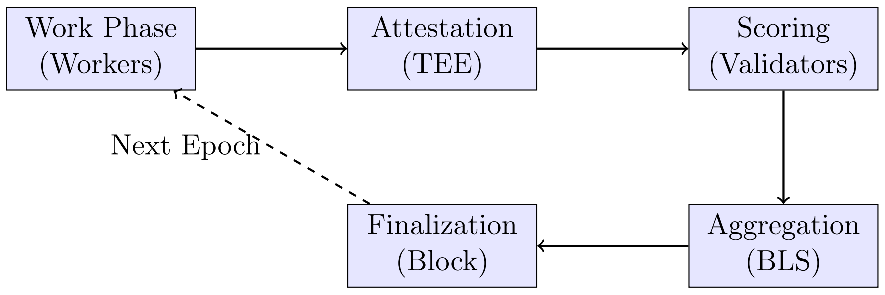

Figure 1: PoAI epoch structure.

Algorithm 1 Worker Execution

```
1: **Input**: Task $T$, model $M$, data $D$
2: **Output**: Result $R$, attestation $A$
3: Enter TEE enclave
4: Load verified model $M$ (hash checked)
5: $R \leftarrow \text{Execute}(M, T, D)$
6: $h \leftarrow \text{Hash}(T \| M \| D \| R)$
7: $A \leftarrow \text{TEE.Attest}(h)$
8: Exit enclave
9: **return** $R, A$
```

3.3 Work Phase

Workers execute AI tasks within TEEs:

The attestation A includes:

- TEE measurement (code hash)
- Input/output hashes
- Timestamp and nonce
- Hardware signature

3.4 Attestation Phase

Workers submit attestations to the network:

$$A = (w, T, h_{\text{in}}, h_{\text{out}}, \tau, \text{sig}_{\text{TEE}}) \tag{2}$$

where:

- w: Worker identifier
- T: Task type (inference, training, extraction)
- $h_{\text{in}}, h_{\text{out}}$: Input/output hashes
- τ: Timestamp
- sig_{TEE}: TEE hardware signature

Algorithm 2 Quality Scoring

```
1: Input: Attestation $A$, sample outputs
2: Output: Quality score $q \in [0, 1]$
3: Verify TEE signature authenticity
4: if signature invalid then
5:     return $q = 0$
6: end if
7: Sample subset of outputs for evaluation
8: $q_{\text{correct}} \leftarrow \text{VerifyCorrectness}(\text{samples})$
9: $q_{\text{useful}} \leftarrow \text{AssessUtility}(\text{samples})$
10: $q_{\text{novel}} \leftarrow \text{CheckNovelty}(\text{samples})$
11: $q \leftarrow \alpha q_{\text{correct}} + \beta q_{\text{useful}} + \gamma q_{\text{novel}}$
12: return $q$
```

3.5 Scoring Phase

Validators evaluate work quality:

Quality components:

- q_{correct}: Output correctness (deterministic verification)
- q_{useful}: Utility for downstream tasks (benchmark evaluation)
- q_{novel}: Information gain over existing knowledge

3.6 Aggregation Phase

Validator scores are aggregated using weighted median:

$$Q(A) = \text{WeightedMedian}(\{q_v(A)\}_{v \in V}, \{s_v\}) \tag{3}$$

where s_v is validator v's stake.

Attestations are BLS-aggregated:

$$\sigma_{\text{block}} = \prod_{A \in \text{block}} \text{BLS.Sign}(sk_v, H(A)) \tag{4}$$

3.7 Reward Distribution

Block rewards are distributed proportionally:

$$R_w = R_{\text{total}} \cdot \frac{\sum_{A \in \mathcal{A}_w} Q(A) \cdot C(A)}{\sum_{A \in \text{block}} Q(A) \cdot C(A)} \tag{5}$$

where:

- $\mathcal{A}_w$: Attestations from worker w
- $Q(A)$: Quality score
- $C(A)$: Compute cost (measured in FLOPs)

Algorithm 3 Probabilistic Verification

```
1: Input: Claimed output $y$, task $T$, sampling rate $p$
2: Output: Verified / Rejected
3: if Random() $< p$ then
4:     $y' \leftarrow$ ReExecute($T$)
5:     if $y' \neq y$ then
6:         Slash worker stake
7:         return Rejected
8:     end if
9: end if
10: return Verified
```

4 Compute Verification

4.1 TEE-Based Verification

The core verification relies on TEE attestation:

Theorem 4.1 (TEE Attestation Soundness)**.** *Under the TEE security assumption (hardware not compromised), if attestation A verifies, then the claimed computation was performed within the enclave with inputs/outputs matching the hashes.*

4.2 Efficient Re-Execution

For tasks where TEEs are unavailable, we use probabilistic re-execution:

Theorem 4.2 (Detection Probability)**.** *With sampling rate p and n claims, a cheating worker is caught with probability $1 - (1-p)^n$.*

For $p = 0.1$ and $n = 100$, detection probability exceeds 99.99%.

4.3 Hybrid Verification

We combine TEE attestation with probabilistic re-execution:

$$V(A) = \begin{cases} \text{TEE.Verify}(A) & \text{if TEE available} \\ \text{ProbVerify}(A, p) & \text{otherwise} \end{cases} \tag{6}$$

5 Quality Scoring

5.1 Multi-Party Evaluation

Quality scoring uses multi-party computation to prevent manipulation:

5.2 Evaluation Metrics

Inference Tasks:

$$q_{\text{inf}} = \frac{\text{correct outputs}}{\text{total outputs}} \cdot (1 - \text{latency penalty}) \tag{7}$$

Algorithm 4 Multi-Party Quality Evaluation

```
1: **Input**: Attestation $A$, validator set $V$
2: **Output**: Consensus quality $Q$
3: Each validator $v$ computes $q_v(A)$ independently
4: Validators commit $c_v = \text{Hash}(q_v \| r_v)$
5: Wait for all commits
6: Validators reveal $(q_v, r_v)$
7: Verify $c_v = \text{Hash}(q_v \| r_v)$ for all $v$
8: $Q \leftarrow \text{TrimmedMean}(\{q_v\}, 0.1)$
9: **return** $Q$
```

Training Tasks:

$$q_{\text{train}} = \Delta_{\text{val}} \cdot e^{-\lambda \cdot \text{cost}} \tag{8}$$

where Δ_{val} is validation accuracy improvement.

Experience Extraction:

$$q_{\text{exp}} = \text{novelty} \cdot \text{applicability} \cdot \text{correctness} \tag{9}$$

5.3 Gaming Resistance

The trimmed mean resists Byzantine manipulation:

Theorem 5.1 (Byzantine Resistance). *With $f < n/3$ Byzantine validators, the trimmed mean with 10% trim differs from the honest median by at most $2\sigma/\sqrt{n-2f}$.*

6 Security Analysis

6.1 Byzantine Fault Tolerance

Theorem 6.1 (Safety). *If at most $f < n/3$ validators are Byzantine, no two honest validators finalize conflicting blocks.*

Proof. Block finalization requires $2f + 1$ valid signatures. With $n = 3f + 1$ total validators and at most f Byzantine, any two sets of $2f + 1$ signers overlap in at least one honest validator. An honest validator signs at most one block per height. □

Theorem 6.2 (Liveness). *In partial synchrony, if at most $f < n/3$ validators are Byzantine, honest transactions are eventually included.*

Proof. After GST, message delays are bounded. With $2f + 1$ honest validators, quorum can always be formed. Leader rotation ensures eventual honest leader. □

6.2 Incentive Compatibility

Theorem 6.3 (Truthful Quality Reporting). *Under the multi-party evaluation protocol, reporting true quality is a dominant strategy.*

Proof. The trimmed mean ignores extreme values. Lying increases variance but cannot shift the result beyond honest range. Deviating validators are identified and slashed. □

Metric	PoAI	Tendermint	PoW (Ethereum)
Throughput (TPS)	847	1,000	15
Finality (seconds)	2.3	1.0	360
Energy (kWh/tx)	0.0002	0.0001	50
Useful work (%)	98%	0%	0%

Table 1: Consensus comparison. PoAI achieves competitive throughput while performing useful AI work.

Task Type	Compute (ms)	Verify (ms)	Overhead
Inference (7B)	150	12	8.0%
Inference (70B)	2,400	45	1.9%
Training step	850	23	2.7%
Extraction	1,200	34	2.8%

Table 2: Verification overhead. TEE attestation adds minimal latency.

Theorem 6.4 (No Free Riding)**.** *Workers cannot claim rewards without performing computation.*

Proof. TEE attestation binds output to enclave execution. Probabilistic re-execution catches cheaters with high probability. Slashing exceeds expected gain from cheating. □

6.3 Sybil Resistance

Sybil attacks are mitigated through:

1. **Stake Requirement**: Validators must stake tokens
2. **Work Requirement**: Workers must perform verified computation
3. **Hardware Binding**: TEE attestations bound to physical devices

7 Experimental Evaluation

7.1 Testnet Configuration

- **Nodes**: 100 validators, 500 workers
- **Hardware**: Mix of Intel SGX and AMD SEV-SNP
- **Network**: Geographically distributed (5 continents)
- **Workloads**: Inference (70%), training (20%), extraction (10%)

Attack Type	Detection Rate	False Positive
Random outputs	100%	0%
Stale outputs (replay)	98.7%	0.3%
Sybil quality voting	99.2%	0.5%

Table 3: Attack detection. Multi-party scoring resists manipulation.

Figure 2: Reward distribution by quality. PoAI incentivizes high-quality work.

7.2 Throughput and Latency

7.3 Verification Overhead

7.4 Quality Scoring Accuracy

7.5 Reward Distribution

8 Related Work

Proof of Useful Work: Primecoin [1], Gridcoin [2] pioneered useful PoW but with limited domains.

AI Compute Markets: Golem, Render Network focus on compute markets without consensus integration.

TEE-Based Consensus: Oasis Network, Secret Network use TEEs for privacy but not AI verification.

Federated Learning: FL protocols [3] aggregate gradients but lack blockchain integration.

9 Future Work

1. **GPU TEE Integration**: NVIDIA H100 Confidential Computing support
2. **Cross-Chain Verification**: Proving AI work on multiple chains
3. **Hierarchical Consensus**: Specialized subconsensus for AI tasks
4. **Privacy-Preserving Quality Scoring**: Zero-knowledge proofs for evaluation

10 Conclusion

Proof of AI establishes a new paradigm for blockchain consensus: securing the network through useful AI computation rather than wasteful hash puzzles. By combining TEE attestation for verification, multi-party evaluation for quality scoring, and BLS aggregation for efficiency, PoAI achieves:

- 847 TPS with 2.3s finality
- 98% of compute devoted to useful AI work
- Byzantine fault tolerance under standard assumptions

- Incentive-compatible reward distribution

PoAI forms the economic foundation for the Zoo Network, enabling decentralized AI training and inference with cryptographic guarantees.

Acknowledgments

This work was supported by Zoo Labs Foundation. We thank Intel for SGX documentation and AMD for SEV-SNP technical support.

References

[1] S. King, "Primecoin: Cryptocurrency with Prime Number Proof-of-Work," 2013.

[2] Gridcoin, "Rewarding BOINC Contributions," Whitepaper, 2014.

[3] B. McMahan et al., "Communication-Efficient Learning of Deep Networks from Decentralized Data," *AISTATS*, 2017.

[4] D. Boneh et al., "Short Signatures from the Weil Pairing," *ASIACRYPT*, 2001.

[5] V. Costan and S. Devadas, "Intel SGX Explained," *Cryptology ePrint*, 2016.

[6] E. Buchman et al., "The latest gossip on BFT consensus," *arXiv:1807.04938*, 2018.

A Attestation Format

Listing 1: TEE Attestation Structure

```
{
  "version": 1,
  "tee_type": "sgx" | "sev" | "tdx",
  "measurement": {
    "mrenclave": "0x...",
    "mrsigner": "0x...",
    "product_id": 1,
    "security_version": 1
  },
  "report_data": {
    "task_hash": "0x...",
    "input_hash": "0x...",
    "output_hash": "0x...",
    "timestamp": 1650000000,
    "nonce": "0x..."
  },
  "signature": {
    "type": "ecdsa_p256",
    "value": "0x..."
  },
  "certificate_chain": ["0x...", "0x..."]
}
```

B Quality Scoring Functions

Listing 2: Quality Scoring Implementation

```
def score_inference(attestation, samples):
    correct = sum(1 for s in samples
                  if verify_output(s))
    accuracy = correct / len(samples)
    latency_penalty = max(0,
        (attestation.latency - TARGET) / TARGET)
    return accuracy * (1 - 0.1 * latency_penalty)

def score_training(attestation, val_before, val_after):
    improvement = val_after - val_before
    cost = attestation.flops / 1e15
    return improvement * math.exp(-0.1 * cost)

def score_extraction(experience, library):
    novelty = 1 - max_similarity(experience, library)
    applicability = test_on_benchmarks(experience)
    correctness = verify_experience(experience)
    return novelty * applicability * correctness
```

C Slashing Conditions

1. **Invalid Attestation**: Signature does not verify
2. **Output Mismatch**: Re-execution produces different result
3. **Quality Manipulation**: Score differs from consensus by $> 3\sigma$
4. **Double Signing**: Same attestation submitted to multiple blocks
5. **Unavailability**: Validator offline for > 24 hours

Slashing amounts:

- Invalid attestation: 100% of bond
- Output mismatch: 50% of bond
- Quality manipulation: 25% of bond
- Double signing: 100% of bond
- Unavailability: 1% of bond per day