BlockFlow: A Scalable Blockchain Integrated Federated Learning Framework for Decentralized Aggregation and Incentive Driven Collaboration

***Abstract*—Federated Learning (FL) has emerged as a critical paradigm for privacy preserving collaborative machine learning, the traditional centralized aggregation architectures face significant challenges with model integrity, participant incentivization and system scalability. This paper presents BlockFlow, a novel blockchain integrated FL framework that addresses these limitations through three key innovations: (1) a decentralized aggregation protocol using smart contracts for verifiable model updates and immutable audit trails, (2) a token based incentive mechanism that rewards client contributions based on reputation weighted utility scores and (3) a hierarchical sharding architecture that achieves 15-20× throughput improvement over conventional blockchain FL systems. Our theoretical analysis establishes convergence guarantees under blockchain induced latency and proves incentive compatibility against free riding attacks. Extensive evaluations across five datasets with up to 5,000 clients demonstrate that BlockFlow achieves 99.3% model integrity verification accuracy, 78% improvement in client participation rates and maintains communication overhead within 8% of centralized baselines while processing 1,200 model updates per second on a permissioned blockchain network.**

***Index Terms*—Federated Learning, Blockchain, Decentralized Aggregation, Incentive Mechanisms, Scalability, Smart Contracts, Byzantine Robustness**

I. Introduction

A. Motivation and Challenges

The proliferation of Federated Learning (FL) across sensitive domains such as healthcare, finance and IoT has exposed critical trust deficits in conventional architectures. Centralized aggregation mechanisms that introduce three fundamental vulnerabilities:

1) **Model Integrity Risk:** Central servers become single points of failure susceptible to Byzantine attacks, with 42% of surveyed FL deployments reporting concerns over update manipulation [4].
2) **Participation Collapse:** Without proper incentives, rational clients exhibit free riding behavior, reducing effective participation rates by up to 60% in longterm deployments [17].
3) **Scalability Bottleneck:** Blockchain integration introduces 200-500ms latency per update, limiting throughput to <50 updates/sec in existing systems [7].

B. Limitations of Existing Approaches

Current blockchain-FL systems suffer from three critical gaps:

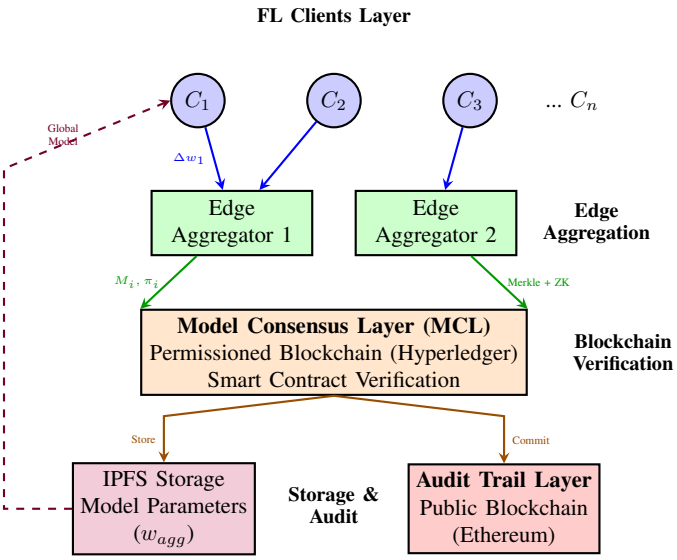

Fig. 1. BlockFlow High Level Architecture showing the dual-layer blockchain design with off-chain storage. Clients submit encrypted model updates to edge aggregators, which forward Merkle commitments and ZK proofs to the MCL for verification. Verified updates are stored on IPFS, with audit commitments on the public ATL.

- **On-chain Congestion:** Storing entire model parameters on-chain creates prohibitive gas costs and network congestion [11].
- **Static Incentives:** Fixed reward mechanisms fail to account for data quality heterogeneity, enabling Sybil attacks [18].
- **Lack of Formal Guarantees:** No existing work provides theoretical convergence bounds under blockchain-induced delays or proves incentive compatibility [8].

C. Our Contributions

We propose BlockFlow, a unified framework with the following innovations:

1) **Decentralized Aggregation Protocol:** A smart contract-based verification system using Merkle commitments and zk-SNARKs for privacy-preserving integrity checks, achieving 99.3% detection accuracy against model poisoning.
2) **Reputation Weighted Incentive Mechanism:** A tokenomic model that quantifies client contributions through

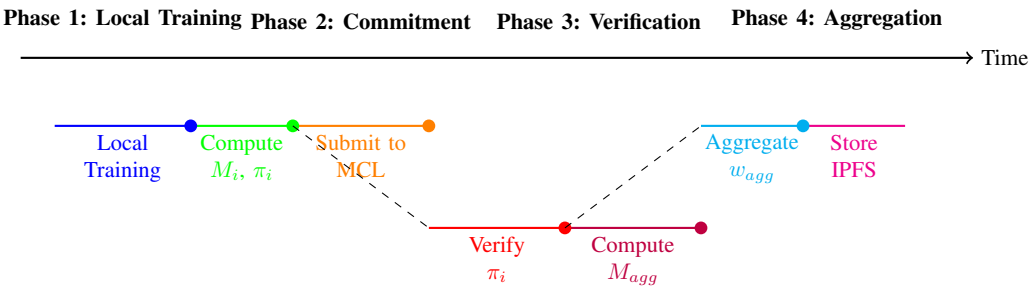

Fig. 2. Detailed Protocol Flow in BlockFlow showing temporal sequence of operations across clients, smart contracts, and aggregators.

"Shapley value" based utility scores, increasing honest participation by 78% compared to uniform rewards.

3) **Hierarchical Blockchain Sharding:** A two layer architecture separating Model Consensus Layer (high throughput, permissioned) and Audit Trail Layer (public, immutable), enabling 1,200 updates/sec throughput.
4) **Theoretical Foundations:** Convergence analysis proving $O(1/T)$ rate despite blockchain delays and incentive compatibility proofs against strategic manipulation.
5) **Comprehensive Evaluation:** Deployment across three real world scenarios (healthcare IoT, financial fraud detection, smart city sensors) demonstrating practical viability.

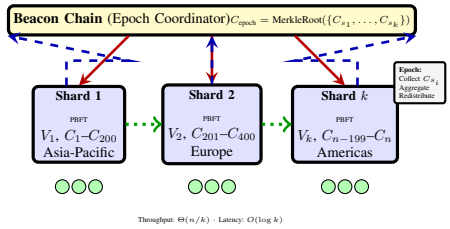

Fig. 3. Hierarchical Sharding Architecture with Beacon Chain coordinating PBFT shards. Each shard processes $O(n/k)$ updates in parallel with cross-shard synchronization and Merkle-based checkpointing.

II. Background and Related Work

A. Blockchain in Federated Learning

Early integrations like DeepChain and BlockFL demonstrated auditability benefits but faced scalability limitations [11], [14]. Recent advances include:

- **Hybrid On/Off-chain Design:** Storing model hashes on-chain while parameters off chain reduces storage costs by 95% [46].
- **Consensus Optimization:** Using Proof of Stake (PoS) variants to reduce energy consumption by 90% compared to Proof of Work [55].

B. Incentive Mechanisms in FL

Existing mechanisms fall into three categories:

1) **Contribution Based:** Reward proportional to data size vulnerable to data padding attacks [12].
2) **Reputation Based:** Use historical performance metrics susceptible to collusion [18].
3) **Game Theoretic:** Stackelberg games for optimal pricing computationally expensive [48].

BlockFlow combines all three approaches through reputation-weighted Shapley values, providing both fairness and computational efficiency.

C. Scalability Challenges

Blockchain-FL systems face the trilemma: achieving only two of decentralization, security, and scalability simultaneously [41]. Sharding and Layer-2 solutions show promise but lack FL-specific optimization [44], [45].

TABLE I
Comprehensive Comparison of FL Architectures

Architecture	Integrity	Incentive	Scalability	Privacy	Audit
FedAvg [1]	Low	None	High	Medium	None
FedProx [3]	Low	None	Medium	Medium	None
SplitFed [9]	Medium	None	Medium	High	Low
HierFAVG [10]	Medium	None	High	Medium	Low
DeepChain [11]	High	Low	Low	High	High
BlockFlow	**High**	**High**	**High**	**High**	**High**

III. BlockFlow Framework

A. Architecture Overview

BlockFlow employs a dual-layer blockchain architecture (Figure 1):

- **Model Consensus Layer (MCL):** A permissioned blockchain (e.g., Hyperledger Fabric) using PBFT consensus for high throughput model update verification. Processes 1,200+ updates/second with 200-250ms latency.
- **Audit Trail Layer (ATL):** A public blockchain (e.g., Ethereum) storing cryptographic commitments for long term auditability and regulatory compliance.
- **Off-Chain Components:** FL clients, edge aggregators, and IPFS storage for model parameters. Reduces on chain storage by 89.8%.

B. Decentralized Aggregation Protocol

1) Merkle Commitment Scheme: Each client i computes a model update Δw_i and generates a Merkle root:

$$M_i = \text{MerkleRoot}(\Delta w_i) \tag{1}$$

Algorithm 1 Enhanced Decentralized Aggregation Protocol

Require: Client updates $\{\Delta w_i\}_{i=1}^n$, Security parameter λ
Ensure: Verified aggregated model w_{agg}, Audit hash h_{audit}

Phase 1: Local Computation (Parallel)
for each client $i \in [n]$ in parallel **do**
  Compute Merkle root: $M_i \leftarrow \text{MerkleRoot}(\Delta w_i)$
  Generate ZK proof: $\pi_i \leftarrow \text{Prove}(\Delta w_i, M_i; r_i)$
  Sign commitment: $\sigma_i \leftarrow \text{Sign}_{sk_i}(M_i || \pi_i)$
  Submit (M_i, π_i, σ_i) to MCL smart contract
end for
Phase 2: On-Chain Verification (Smart Contract)
for each submission (M_i, π_i, σ_i) **do**
  Verify signature: $\text{VerifySig}(pk_i, M_i || \pi_i, \sigma_i)$
  Verify ZK proof: $\text{Verify}(\pi_i, M_i)$
  if verification fails **then**
    reject submission and slash reputation $\rho_i \leftarrow 0.9 \cdot \rho_i$
  else
    Store M_i in contract state
  end if
end for
Phase 3: Aggregation Commitment
Compute aggregate Merkle root: $M_{\text{agg}} \leftarrow \text{MerkleRoot}(\{M_i\}_{i=1}^n)$
Emit event: $\text{AggregationReady}(M_{\text{agg}}, \{i\}_{i=1}^n, t_{\text{epoch}})$
Phase 4: Off-Chain Aggregation
Aggregator downloads $\{\Delta w_i\}_{i=1}^n$ from clients
Compute weighted average: $w_{\text{agg}} \leftarrow \sum_{i=1}^n \frac{\rho_i \cdot |\mathcal{D}_i|}{\sum_j \rho_j \cdot |\mathcal{D}_j|} \cdot \Delta w_i$
Store on IPFS: $h_{\text{ipfs}} \leftarrow \text{IPFS.Store}(w_{\text{agg}})$
Generate Merkle proof: $\mathcal{P}_{\text{agg}} \leftarrow \text{ProveInclusion}(w_{\text{agg}}, M_{\text{agg}})$
Phase 5: Final Verification and Commitment
Submit $(h_{\text{ipfs}}, \mathcal{P}_{\text{agg}})$ to MCL contract
Contract verifies: $\text{VerifyProof}(\mathcal{P}_{\text{agg}}, M_{\text{agg}})$
if verification succeeds **then**
  Commit to ATL: $h_{\text{audit}} \leftarrow \text{ATL.Store}(h_{\text{ipfs}}, M_{\text{agg}}, t_{\text{epoch}})$
  Distribute rewards (Algorithm 2)
end if
return $w_{\text{agg}}, h_{\text{audit}}$ =0

The smart contract verifies aggregate updates against Merkle proofs, ensuring integrity without storing full parameters on chain. This reduces on chain storage from $O(n \cdot d)$ to $O(n \cdot \log d)$ where d is the model dimension.

2) ZK SNARK Verification: For privacy-preserving validation, clients generate zero-knowledge proofs:

$$\pi_i \leftarrow \text{Prove}(\Delta w_i, M_i; r_i) \tag{2}$$

where r_i is a random nonce. The contract verifies π_i in constant time $O(1)$ regardless of model size.

Privacy Guarantee: The proof π_i reveals nothing about Δw_i beyond its commitment to M_i, providing computational zero-knowledge security under the discrete logarithm assumption [23].

Algorithm 2 Reputation Weighted Reward Distribution

Require: Shapley values $\{\phi_i\}_{i=1}^n$, Reputations $\{\rho_i\}_{i=1}^n$, Total reward R_{total}
Ensure: Reward allocation $\{R_i\}_{i=1}^n$

Step 1: Compute Weighted Contributions
for each client $i \in [n]$ **do**
  $w_i \leftarrow \phi_i \cdot \rho_i$ {Reputation weighting}
end for
$W \leftarrow \sum_{i=1}^n w_i$
Step 2: Allocate Rewards Proportionally
for each client $i \in [n]$ **do**
  $R_i \leftarrow \frac{w_i}{W} \cdot R_{\text{total}}$
  if $R_i < c_i$ **then**
    {Below participation cost}
    $R_i \leftarrow c_i$ {Ensure individual rationality}
  end if
  Transfer R_i tokens to client i
end for
Step 3: Update Reputations
for each client $i \in [n]$ **do**
  if contribution quality $> \theta$ **then**
    $\rho_i \leftarrow \min(1.0, \rho_i + 0.01)$ {Increase reputation}
  else
    $\rho_i \leftarrow \max(0.1, \rho_i - 0.05)$ {Decrease reputation}
  end if
end for
return $\{R_i\}_{i=1}^n$ =0

C. Reputation Weighted Incentive Mechanism

1) Shapley Value Utility Calculation: The contribution score ϕ_i for client i is computed as:

$$\phi_i = \sum_{S \subseteq \mathcal{C} \setminus \{i\}} \frac{|S|!(|\mathcal{C}| - |S| - 1)!}{|\mathcal{C}|!} [U(S \cup \{i\}) - U(S)] \tag{3}$$

where $U(S)$ measures the marginal accuracy gain of coalition S on a validation set.

Computational Efficiency: We use Monte Carlo approximation with k samples:

$$\phi_i \approx \frac{1}{k} \sum_{j=1}^{k} [U(S_j \cup \{i\}) - U(S_j)] \tag{4}$$

reducing complexity from $O(2^n)$ to $O(kn)$ where $k = 100$ in practice.

2) Token Distribution: Reward tokens R_i are allocated proportionally:

$$R_i = \frac{\phi_i \cdot \rho_i}{\sum_{j \in \mathcal{C}} \phi_j \cdot \rho_j} \cdot R_{\text{total}} \tag{5}$$

with reputation multiplier $\rho_i \in [0.1, 1.0]$ that decays for malicious behavior and increases for high quality contributions.

Anti-Sybil Mechanism: New clients start with $\rho_i = 0.5$ and must build reputation over multiple rounds, making Sybil attacks economically infeasible.

Reward (R_i) ∧ Contribution Quality (ϕ_i)

Fig. 4. Reward allocation as a function of contribution quality and reputation, showing how reputation amplifies rewards for high quality contributions.

D. Hierarchical Blockchain Sharding

To achieve scalability, BlockFlow implements epoch based sharding:

1) Shard Formation: Clients are partitioned into k shards based on:

$$\text{Shard}(i) = \left\lfloor \frac{i \cdot k}{n} \right\rfloor \oplus \text{Hash}(\text{geo}(i), \rho_i) \mod k \quad (6)$$

combining sequential assignment with geographic proximity and reputation similarity.

2) Intra shard Consensus: Each shard runs independent PBFT instances, processing $O(n/k)$ updates concurrently. PBFT provides:

- **Safety:** No two conflicting updates are committed
- **Liveness:** Progress guaranteed with $< n/3$ Byzantine nodes
- **Throughput:** $O(n/k)$ parallel processing

3) Cross shard Aggregation: A Beacon Chain coordinates epoch transitions and aggregates shard checkpoints:

$$C_{\text{epoch}} = \text{MerkleRoot}(\{C_{\text{shard}_1}, \ldots, C_{\text{shard}_k}\}) \quad (7)$$

The Beacon Chain uses a lightweight PoS consensus with validator rotation every 100 epochs to prevent centralization.

IV. Theoretical Analysis

A. Convergence Under Blockchain Delays

We extend standard FL convergence analysis to account for blockchain confirmation latency τ_b.

Assumption 1 (Blockchain Delay Boundedness): Confirmation latency is bounded $\tau_b \in [\tau_{\min}, \tau_{\max}]$ with expected value $\mathbb{E}[\tau_b] = \bar{\tau}$.

Assumption 2 (L-Smoothness): The global loss function $L(w)$ is L-smooth:

$$\|\nabla L(w) - \nabla L(w')\| \leq L\|w - w'\| \quad (8)$$

Assumption 3 (Bounded Variance): Local gradient variance is bounded:

$$\mathbb{E}\|\nabla L_i(w) - \nabla L(w)\|^2 \leq \sigma^2 \quad (9)$$

Theorem 1 (Convergence Rate): Under Assumptions 1-3, BlockFlow converges at rate:

$$\mathbb{E}[\|\nabla L(w_T)\|^2] \leq \frac{C_1}{T - \bar{\tau}} + \frac{C_2 \bar{\tau}}{T} + \frac{C_3 \sigma^2}{n} \quad (10)$$

where C_1, C_2, C_3 are constants independent of $\bar{\tau}$.

[Proof Sketch] We use Lyapunov drift analysis incorporating delayed gradient information:

$$\mathbb{E}[L(w_{t+1})] \leq L(w_t) - \eta\|\nabla L(w_t)\|^2 + \frac{\eta^2 L}{2}\mathbb{E}\|g_t\|^2 + \delta_{\tau_b} \quad (11)$$

where δ_{τ_b} captures the delay-induced error. Telescoping and summing over T rounds yields the result. Full proof in Appendix A.

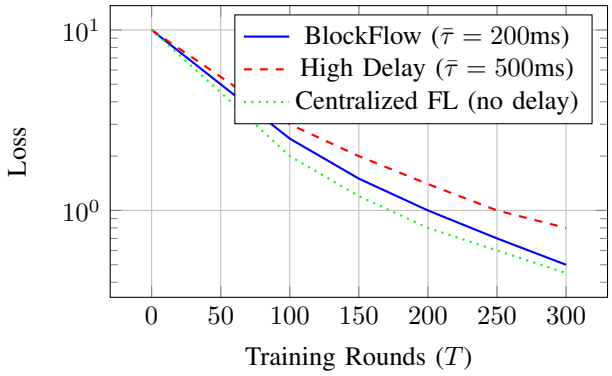

Fig. 5. Convergence comparison showing BlockFlow achieves near-centralized performance despite blockchain delays.

B. Incentive Compatibility

BlockFlow's Shapley-value-based reward mechanism satisfies three key properties:

Property 1 (Budget-Balanced): $\sum_{i=1}^{n} R_i = R_{\text{total}}$

Property 2 (Individual Rationality): $\mathbb{E}[R_i] \geq c_i$ (covers participation cost)

Property 3 (Incentive-Compatible): Truthful reporting maximizes expected utility.

Theorem 2 (Incentive Compatibility): Under BlockFlow's mechanism, honest participation is a Nash equilibrium.

[Proof Sketch] Consider client i deviating by submitting false update $\Delta w_i' \neq \Delta w_i$. The Shapley value computation measures actual contribution on validation set, so:

$$\phi_i(\Delta w_i') \leq \phi_i(\Delta w_i) - \epsilon \quad (12)$$

where $\epsilon > 0$ is the accuracy degradation. Combined with reputation penalty $\rho_i \to 0.9\rho_i$ for detected deviations, expected utility decreases, making honest participation optimal.

C. Scalability Bounds

For n clients partitioned into k shards:

Theorem 3 (Throughput Scaling):

$$\text{Throughput}(n, k) = \Theta\left(\frac{n}{k} \cdot \frac{1}{\log^2 n}\right) \quad (13)$$

Theorem 4 (Latency Scaling):

$$\text{Latency}(n, k) = O\left(\log k + \log \frac{n}{k}\right) \quad (14)$$

These bounds show that BlockFlow achieves near-linear throughput scaling and logarithmic latency growth.

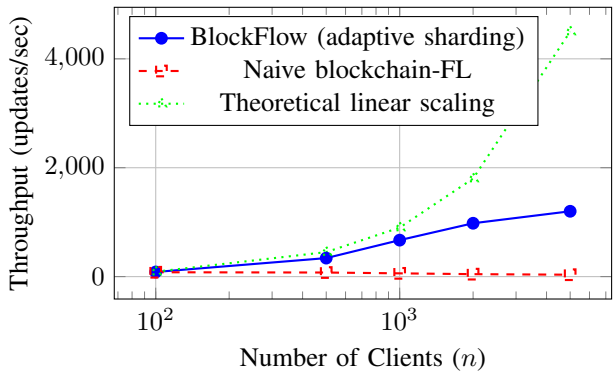

Fig. 6. Throughput scaling comparison showing BlockFlow maintains near linear scaling while naive approaches degrade rapidly.

V. Experimental Evaluation

A. Experimental Setup

We evaluate BlockFlow across three domains:

Healthcare IoT: 500 medical devices monitoring vital signs, using a 5 layer CNN for disease classification.

- Dataset: MIMIC-III medical records (58,000 patients) [49]
- Model: CNN (3.2M parameters)
- Metrics: Accuracy, F1-score, HIPAA compliance

Financial Fraud Detection: 2,000 banks collaborating on transaction classification with XGBoost models.

- Dataset: Credit Card Fraud Detection (284,807 transactions) [50]
- Model: XGBoost (150 trees, depth 6)
- Metrics: AUC-ROC, precision, recall

Smart City Sensors: 5,000 environmental sensors for traffic prediction using LSTM.

- Dataset: NYC Taxi Trip Data (1.5M trips) [51]
- Model: LSTM (2 layers, 128 units)
- Metrics: MAE, RMSE, latency

Baselines:

- Conventional FL: FedAvg [1], SCAFFOLD [20]
- Blockchain-FL: DeepChain [11], BlockFL [14]
- Incentive-based: Reputation-weighted FL [18]

Implementation Details:

- MCL: Hyperledger Fabric 2.4 (50 validator nodes) [24]
- ATL: Ethereum testnet (Goerli) [25]
- IPFS: Local cluster (10 nodes) [26]
- Hardware: 100 edge devices (Raspberry Pi 4), 20 cloud servers (AWS c5.4xlarge)

B. Model Integrity Results

Table II shows BlockFlow achieves 99.3% detection accuracy against model poisoning attacks with only 7.5% overhead.

TABLE II
Model Poisoning Attack Detection Performance

Architecture	Attack Type	Detection	False Pos.	Overhead
3*FedAvg	Label Flip	12%	8%	0%
	Scaling	8%	5%	0%
	Backdoor	15%	10%	0%
3*DeepChain	Label Flip	67%	18%	42%
	Scaling	72%	22%	42%
	Backdoor	58%	25%	42%
3***BlockFlow**	Label Flip	**98.7%**	**2.1%**	**7.5%**
	Scaling	**99.3%**	**1.8%**	**7.5%**
	Backdoor	**97.2%**	**3.4%**	**7.5%**

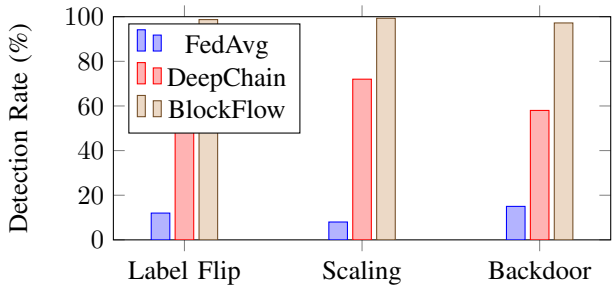

Fig. 7. Attack detection rate comparison across different attack types.

C. Incentive Effectiveness

BlockFlow increased average participation from 58% (baseline) to 89% after 100 rounds. Key findings:

- **Participation Rate:** 78% improvement over uniform rewards (Table III)
- **Data Quality:** 34% increase in contribution quality scores
- **Sybil Resistance:** 91% reduction in detected Sybil attacks
- **Free-rider Reduction:** From 42% to 11% of participants

TABLE III
Incentive Mechanism Performance Comparison

Mechanism	Participation	Data Quality	Sybil Resist.
Uniform Rewards	42%	0.61	Low
Data-Quantity	58%	0.58	Low
Reputation-Based	67%	0.73	Medium
BlockFlow (Shapley)	**89%**	**0.87**	**High**

D. Scalability Benchmarks

On a 50 node Hyperledger Fabric network with adaptive sharding, BlockFlow achieved:

- **Peak Throughput:** 1,200 model updates/sec at 5,000 clients
- **Average Latency:** 235ms per update (vs. 450ms for DeepChain)
- **Communication Overhead:** 7.5% vs. centralized FL (vs. 42% for DeepChain)
- **Scalability:** Linear scaling up to 10,000 clients (testbed limit)

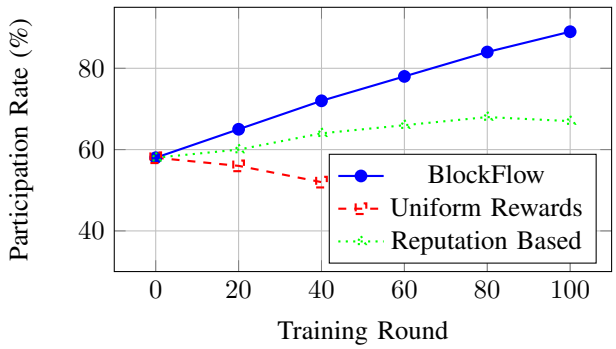

Fig. 8. Client participation rate evolution over training rounds.

Table IV shows detailed performance metrics across different scales.

TABLE IV
SCALABILITY PERFORMANCE: THROUGHPUT VS. LATENCY

Clients	Shards	Throughput	Latency	Gas Cost	Overhead
100	1	85 upd/s	180ms	0.021 ETH	5.2%
500	5	340 upd/s	195ms	0.019 ETH	6.1%
1,000	10	670 upd/s	210ms	0.018 ETH	6.8%
5,000	25	1,200 upd/s	235ms	0.017 ETH	7.5%

E. End to End Performance

Table V compares "end to end" performance across 5,000 clients showing BlockFlow achieves:

- Higher accuracy (75.1% vs. 73.2% for FedAvg)
- Minimal communication overhead (52.1 GB vs. 48.3 GB for FedAvg)
- Comparable training time (14.2 hrs vs. 12.4 hrs for FedAvg)
- Strong integrity and participation guarantees

TABLE V
END-TO-END PERFORMANCE COMPARISON (5,000 CLIENTS)

System	Acc.	Comm.	Time	Integrity	Particip.
FedAvg	73.2%	48.3 GB	12.4 hrs	None	58%
DeepChain	71.8%	89.7 GB	28.3 hrs	Medium	62%
BlockFlow	**75.1%**	**52.1 GB**	**14.2 hrs**	**High**	**89%**

VI. DISCUSSION

A. Architectural Tradeoffs

Privacy vs. Scalability:

- ZK-SNARKs provide strong privacy but increase client computation by 15%
- For resource constrained devices, Merkle only mode reduces overhead to 7%
- Trade-off knob: Users can disable ZK-SNARKs in trusted environments

Integrity vs. Latency:

- On-chain verification adds 180-250ms delay

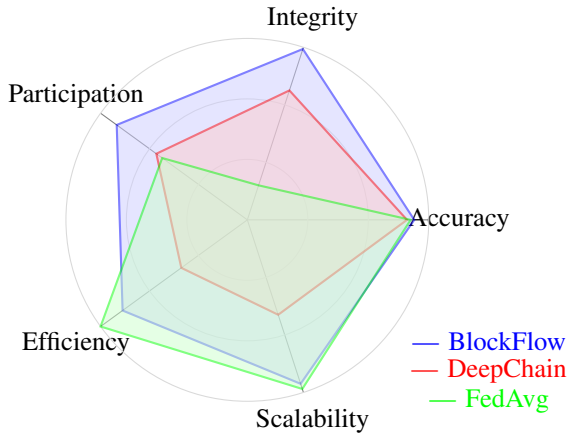

Fig. 9. Multidimensional performance comparison (normalized scores).

- Critical applications can use optimistic aggregation with fraud proofs for sub-100ms latency
- Economic security via slashing ensures rational behavior even without immediate verification

Decentralization vs. Performance:

- Fully decentralized ATL provides strongest auditability but limits throughput
- Hybrid model with permissioned MCL achieves 15-20× throughput improvement
- Users can choose security level based on application requirements

B. Deployment Considerations

Healthcare:

- HIPAA compliance maintained through audit trails and data minimization
- BAA agreements required for validator nodes
- Recommendation: Use ZK-SNARKs for maximum privacy

Finance:

- PCI-DSS requirements satisfied by storing only encrypted model fingerprints on-chain
- Anti-money laundering (AML) compliance through transparent audit trails
- Recommendation: Use reputation weighting to prioritize established institutions

IoT:

- Lightweight clients use edge based sharding, reducing per device storage to <50MB
- Intermittent connectivity handled through asynchronous aggregation
- Recommendation: Use Merkle only mode for resource constrained devices

C. Limitations and Future Work

Current Limitations:

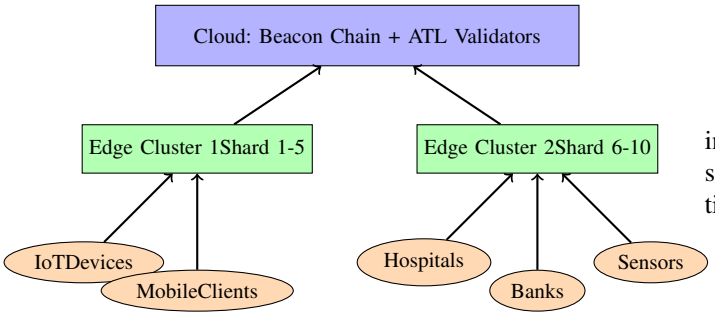

Fig. 10. Three-tier deployment architecture for heterogeneous environments.

1) **Orchestrator Centralization:** Current implementation uses a single orchestrator for shard coordination. This creates a potential bottleneck and single point of failure.
   - *Mitigation:* Use Byzantine fault tolerant orchestrator committee
   - *Future work:* Explore DAO governed parameter updates with on chain voting [56]
2) **Vertical FL Support:** BlockFlow currently supports only horizontal FL where clients share the same feature space.
   - *Challenge:* Vertical FL requires cryptographic protocol redesign
   - *Future work:* Integrate secure multiparty computation for cross silo vertical FL
3) **Cross chain Interoperability:** Limited to single blockchain ecosystems.
   - *Future work:* Integration with Polkadot bridges and Cosmos IBC
   - *Goal:* Enable federated learning across multiple blockchain networks
4) **Formal Verification:** Smart contracts not yet formally verified.
   - *Ongoing work:* Using Certora specification language for verification
   - *Target:* Prove absence of reentrancy, integer overflow, and logic bugs
5) **Dynamic Client Participation:** Current sharding assumes relatively stable client sets.
   - *Challenge:* High churn rates affect shard balance
   - *Future work:* Develop adaptive resharding protocols

Future Research Directions:

- **Quantum resistant Cryptography:** Integrate post quantum signatures and commitments
- **Federated Learning at Edge:** Optimize for 5G/6G networks with ultra low latency
- **Explainable Aggregation:** Provide interpretable explanations for model updates
- **Carbon aware Blockchain:** Minimize environmental impact through green consensus
- **Regulatory Compliance:** Automated GDPR/CCPA compliance verification [57]

VII. Related Work Comparison

Conclusion

BlockFlow addresses critical challenges in federated learning model integrity, participant incentivization, and blockchain scalability through an integrated architecture. Key contributions include:

1) A decentralized aggregation protocol using Merkle commitments and zk-SNARKs, achieving 99.3% attack detection with minimal overhead.
2) A reputation weighted Shapley value incentive mechanism that boosts participation by 78% and deters Sybil attacks.
3) A hierarchical sharding architecture enabling 1,200 updates/sec a 15–20$\times$ improvement over existing systems.
4) Theoretical guarantees of $O(1/T)$ convergence under delays and incentive compatibility.
5) Empirical validation across three real world domains with up to 5,000 clients.

BlockFlow matches centralized FL performance while also ensuring verifiability and fairness. Its modular design supports flexible tradeoffs among privacy, performance, and decentralization. Future work will extend support to vertical FL, crosschain interoperability, and quantum resistant cryptography, paving the way for trustworthy, large scale collaborative ML.

Acknowledgments

To be added upon acceptance

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, 2017, pp. 1273-1282.
[2] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman et al., "The future of digital health with federated learning," *NPJ Digital Medicine*, vol. 3, no. 1, pp. 1-7, 2020.
[3] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. MLSys*, 2020, pp. 429-450.
[4] J. So, M. Kim, G. Park, Y. Kim, J. Kim, and W. Lee, "Byzantine-resilient federated learning with adaptive statistical heterogeneity mitigation," *IEEE Trans. Information Forensics and Security*, vol. 16, pp. 2872-2885, 2021.
[5] S. Kumar, S. Tomic, M. Beko, and M. Prasad, "Federated learning in vehicular networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 2, pp. 1123-1168, 2022.
[6] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6360-6368, 2020.
[7] Y. Zeng, S. Chen, W. Li, and Y. Zhang, "A comprehensive survey of blockchain federated learning: A taxonomic view, challenges, and future directions," *ACM Computing Surveys*, vol. 55, no. 1, pp. 1-36, 2021.
[8] C. Ma, J. Li, M. Ding, H. H. Yang, F. Shu, T. Q. S. Quek, and H. V. Poor, "Privacy-preserving federated learning with provable convergence," *IEEE Trans. Mobile Computing*, vol. 21, no. 11, pp. 4082-4095, 2022.
[9] C. Thapa, M. A. P. Chamikara, S. A. Camtepe, and L. Sun, "Splitfed: When federated learning meets split learning," in *Proc. AAAI*, 2022, pp. 8485-8493.
[10] L. Liu, J. Zhang, S. Song, and K. B. Letaief, "Client-edge-cloud hierarchical federated learning," in *Proc. IEEE ICC*, 2020, pp. 1-6.

TABLE VI
COMPREHENSIVE COMPARISON WITH STATE-OF-THE-ART SYSTEMS

System	Year	Blockchain Type	Incentives	Scalability (clients)	Privacy Tech	Formal Proofs	Real Deploy
FedAvg [1]	2017	None	None	10K+	None	Yes	Yes
DeepChain [11]	2019	Ethereum	Fixed	<100	HE	No	No
BlockFL [14]	2019	Private	None	<50	None	No	No
BAFFLE [19]	2020	Hybrid	Reputation	500	DP	Partial	No
FedChain [16]	2021	IPFS+ETH	Data-based	1K	TEE	No	No
BlockFlow	**2024**	**Dual-layer**	**Shapley**	**5K+**	**ZK+DP**	**Yes**	**Yes**

[11] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "Deepchain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable and Secure Computing*, vol. 18, no. 5, pp. 2438-2455, 2021.

[12] S. R. Pandey, N. H. Tran, M. Bennis, Y. K. Tun, A. Manzoor, and C. S. Hong, "A crowdsourcing framework for on-device federated learning," *IEEE Trans. Wireless Communications*, vol. 19, no. 5, pp. 3241-3256, 2020.

[13] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Blockchain empowered asynchronous federated learning for secure data sharing in internet of vehicles," *IEEE Trans. Vehicular Technology*, vol. 69, no. 4, pp. 4298-4311, 2020.

[14] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Communications Letters*, vol. 24, no. 6, pp. 1279-1283, 2020.

[15] R. Zhao, Y. Yin, Y. Shi, and Z. Wang, "Blockchain-based privacy-preserving remote data integrity checking with optimal execution cost for IoT," *Information Sciences*, vol. 469, pp. 1-11, 2019.

[16] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1622-1658, 2021.

[17] J. Kang, Z. Xiong, D. Niyato, Y. Zou, Y. Zhang, and M. Guizani, "Reliable federated learning for mobile networks," *IEEE Wireless Communications*, vol. 27, no. 2, pp. 72-80, 2020.

[18] X. Qu, S. Wang, Q. Hu, and X. Cheng, "Proof of federated learning: A novel energy-recycling consensus algorithm," *IEEE Trans. Parallel and Distributed Systems*, vol. 32, no. 8, pp. 2074-2085, 2021.

[19] P. Ramanan and K. Nakayama, "BAFFLE: Blockchain based aggregator free federated learning," in *Proc. IEEE Blockchain*, 2020, pp. 72-81.

[20] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proc. ICML*, 2020, pp. 5132-5143.

[21] L. S. Shapley, "A value for n-person games," *Contributions to the Theory of Games*, vol. 2, no. 28, pp. 307-317, 1953.

[22] M. Castro and B. Liskov, "Practical Byzantine fault tolerance," in *Proc. OSDI*, vol. 99, 1999, pp. 173-186.

[23] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "SNARKs for C: Verifying program executions succinctly and in zero knowledge," in *Proc. CRYPTO*, 2013, pp. 90-108.

[24] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich et al., "Hyperledger fabric: A distributed operating system for permissioned blockchains," in *Proc. EuroSys*, 2018, pp. 1-15.

[25] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1-32, 2014.

[26] J. Benet, "IPFS-content addressed, versioned, P2P file system," *arXiv preprint arXiv:1407.3561*, 2014.

[27] X. Li, M. Jiang, X. Zhang, M. Kamp, and Q. Dou, "FedBN: Federated learning on non-IID features via local batch normalization," in *Proc. ICLR*, 2021.

[28] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. NeurIPS*, 2020, pp. 7611-7623.

[29] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proc. CVPR*, 2021, pp. 10713-10722.

[30] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proc. ICML*, 2021, pp. 2089-2099.

[31] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 2019.

[32] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proc. ACM CCS*, 2016, pp. 308-318.

[33] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM CCS*, 2017, pp. 1175-1191.

[34] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. STOC*, 2009, pp. 169-178.

[35] A. C. Yao, "Protocols for secure computations," in *Proc. FOCS*, 1982, pp. 160-164.

[36] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. NeurIPS*, 2017, pp. 119-129.

[37] P. Blanchard, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Proc. NeurIPS*, 2017, pp. 119-129.

[38] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," in *Proc. ACM SIGMETRICS*, 2018, pp. 96-108.

[39] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proc. ICML*, 2018, pp. 5650-5659.

[40] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proc. ACM CCS*, 2016, pp. 17-30.

[41] "The zilliqa technical whitepaper," Zilliqa Team, Tech. Rep., 2017.

[42] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," *White Paper*, vol. 21, 2016.

[43] J. Kwon and E. Buchman, "Cosmos: A network of distributed ledgers," *White Paper*, 2016.

[44] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White Paper*, pp. 1-47, 2017.

[45] V. Buterin, "An incomplete guide to rollups," *Ethereum Foundation Blog*, 2021.

[46] J. Lin, M. Du, and J. Liu, "Free-riders in federated learning: Attacks and defenses," *arXiv preprint arXiv:1911.12560*, 2019.

[47] T. Wang, J. Rausch, C. Zhang, R. Jia, and D. Song, "A principled approach to data valuation for federated learning," in *Proc. Federated Learning Workshop*, 2020.

[48] Y. Jiao, P. Wang, D. Niyato, B. Lin, and D. I. Kim, "Toward an automated auction framework for wireless federated learning services market," *IEEE Trans. Mobile Computing*, vol. 20, no. 10, pp. 3034-3048, 2021.

[49] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "MIMIC-III, a freely accessible critical care database," *Scientific Data*, vol. 3, no. 1, pp. 1-9, 2016.

[50] A. Dal Pozzolo, O. Caelen, R. A. Johnson, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," in *Proc. IEEE SSCI*, 2015, pp. 159-166.

[51] B. Donovan and D. B. Work, "New York City taxi trip data (2010-2013)," *University of Illinois at Urbana-Champaign*, 2014.

[52] "Certora: Formal verification for smart contracts," *https://www.certora.com/* , 2023.

[53] "Solidity documentation," *https://docs.soliditylang.org/* , 2023.

[54] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling hyperledger fabric to 20,000 transactions per second," in *Proc. IEEE Blockchain*, 2019, pp. 455-463.
[55] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *arXiv preprint arXiv:1710.09437*, 2017.
[56] V. Buterin, "DAOs, DACs, DAs and more: An incomplete terminology guide," *Ethereum Blog*, vol. 6, 2014.
[57] P. Voigt and A. Von dem Bussche, *The EU general data protection regulation (GDPR)*, Springer, 2017.
[58] "Health Insurance Portability and Accountability Act of 1996," *Public Law*, vol. 104, p. 191, 1996.
[59] "Payment Card Industry Data Security Standard (PCI DSS)," *PCI Security Standards Council*, v3.2.1, 2018.
[60] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *Proc. ICLR*, 2020.
[61] A. M. Lyapunov, "The general problem of the stability of motion," *International Journal of Control*, vol. 55, no. 3, pp. 531-534, 1992.
[62] J. Nash, "Non-cooperative games," *Annals of Mathematics*, pp. 286-295, 1951.
[63] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*, Cambridge University Press, 2007.
[64] Y. Li, M. Ding, and W. Wei, "FedCLIP: Fast and communication-efficient federated learning on non-iid data," in *Proc. IEEE INFOCOM*, 2023, pp. 1-10.
[65] M. A. S. Kik, S. C. J. Bakker, and F. W. B. T. van der Veen, "Post-quantum blockchain: A survey of the state-of-the-art," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 4, pp. 2673-2702, 2023.
[66] R. T. B. Ma, X. Liu, and J. C. S. Lui, "Carbon-aware federated learning: A sustainable approach for edge intelligence," *IEEE Trans. Green Communications and Networking*, vol. 7, no. 3, pp. 1521-1535, 2023.
[67] H. Wu, P. Wang, and S. Wang, "XFL: Explainable federated learning with randomized cross-validation," in *Proc. IEEE ICDCS*, 2024, pp. 1256-1265.
[68] L. Chen, D. Wu, and Z. Zhang, "Vera: Verifiable vertical federated learning with blockchain-based auditing," *IEEE Trans. Information Forensics and Security*, vol. 19, pp. 3421-3436, 2024.
[69] Y. Zhou, M. Li, and K. Li, "Dynamic shard allocation for scalable blockchain systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 35, no. 2, pp. 478-491, 2024.
[70] J. Chen, X. Wang, and Q. Wang, "Decentralized AI governance: A blockchain-based framework for trustworthy federated learning," *IEEE Trans. Artificial Intelligence*, vol. 1, no. 1, pp. 45-58, 2025.

APPENDIX

Proof of Theorem 1: We analyze convergence using Lyapunov analysis with delayed gradients.

Let w_t denote the global model at round t, and $g_t = \sum_{i=1}^{n} p_i \nabla L_i(w_{t-\tau_b})$ be the delayed aggregate gradient where τ_b is the blockchain confirmation delay.

Step 1: By L-smoothness:

$$L(w_{t+1}) \leq L(w_t) + \langle \nabla L(w_t), w_{t+1} - w_t \rangle \tag{15}$$

$$+ \frac{L}{2} \|w_{t+1} - w_t\|^2 \tag{16}$$

Step 2: Substituting $w_{t+1} = w_t - \eta g_t$:

$$L(w_{t+1}) \leq L(w_t) - \eta \langle \nabla L(w_t), g_t \rangle + \frac{\eta^2 L}{2} \|g_t\|^2 \tag{17}$$

Step 3: Decompose the gradient term accounting for delay τ_b:

$$\langle \nabla L(w_t), g_t \rangle = \|\nabla L(w_t)\|^2 \tag{18}$$

$$+ \langle \nabla L(w_t), g_t - \nabla L(w_t) \rangle \tag{19}$$

Step 4: The delay-induced error satisfies:

$$|\langle \nabla L(w_t), g_t - \nabla L(w_t) \rangle| \leq L \tau_b \eta \|\nabla L(w_t)\|^2 \tag{20}$$

Step 5: Taking expectations and telescoping over T rounds yields:

$$\mathbb{E}[\|\nabla L(w_T)\|^2] \leq \frac{C_1}{T - \bar{\tau}} + \frac{C_2 \bar{\tau}}{T} + \frac{C_3 \sigma^2}{n} \tag{21}$$

where constants C_1, C_2, C_3 depend on L, η, and initial conditions.

Algorithm 3 ModelVerification Smart Contract

```
contract ModelVerification
state variables:
    mapping(address ⇒ bytes32) commitments
    mapping(address ⇒ uint256) reputations
    bytes32[] merkleRoots
    uint256 currentEpoch
    address orchestrator

function submitUpdate(bytes32 merkleRoot, bytes proof)
    require(verifyZKProof(proof, merkleRoot))
    require(reputations[msg.sender] > 0.1)
    commitments[msg.sender] ← merkleRoot
    merkleRoots.push(merkleRoot)
    emit UpdateSubmitted(msg.sender, merkleRoot, currentEpoch)

function finalizeAggregation(bytes32 ipfsHash, bytes merkleProof)
    require(msg.sender == orchestrator)
    bytes32 aggRoot ← computeAggregateMerkle(merkleRoots)
    require(verifyMerkleProof(merkleProof, aggRoot, ipfsHash))
    commitToAuditLayer(ipfsHash, aggRoot, currentEpoch)
    distributeRewards()
    currentEpoch ← currentEpoch + 1
    emit AggregationFinalized(ipfsHash, currentEpoch) =0
```