ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

BFL-Guard: A Blockchain-Enabled Federated Learning Framework with Zero-Knowledge Gradient Verification and Tokenized Incentives

Kumaresan S[1], Thirumal L[2], Sathish Kumar S[3], Ellappan V[4], Selvam R[5]

[1] Assistant Professor Department of EEE PGP college of Engineering and Technology, Namakkal, Tamil Nadu, India

[2] Assistant Professor Varuvan Vadivelan Institute of Technology

[3 ,5] Assistant Professor Department of ECE Mahendra Institute of Technology, Namakkal, Tamil Nadu India

[4]Professor Assistant Department of ECE Mahendra Institute of Technology, Namakkal, Tamil Nadu India

DOI: https://doi.org/10.51583/IJLTEMAS.2026.150600186

Received: 02 July 2026; Accepted: 07 July 2026; Published: 20 July 2026

ABSTRACT

Federated Learning (FL) enables collaborative model training across decentralized participants without sharing raw data. However, existing FL systems remain vulnerable to Byzantine attacks and suffer from a lack of accountability, verifiability, and economic incentives for honest participation. We present BFL-Guard, a novel blockchain-orchestrated federated learning framework integrating: (i) zk-SNARK-based zero-knowledge gradient proofs, (ii) an on-chain Byzantine-tolerant aggregation smart contract, and (iii) a tokenized incentive protocol (FedToken). BFL-Guard stores model checkpoints as IPFS hashes anchored on Ethereum, ensuring tamper-evident auditability. Experiments on CIFAR-10 and Shakespeare benchmarks demonstrate 95.2% and 87.6% accuracy in IID and Non-IID settings, surpassing all baselines while converging 12.4% faster even under 30% Byzantine injection.

Keywords: Blockchain, Federated Learning, Byzantine Fault Tolerance, Zero-Knowledge Proofs, Smart Contracts, Incentive Mechanism, Decentralized AI, IPFS, zk-SNARK

INTRODUCTION

The proliferation of edge devices, mobile platforms, and Internet-of-Things (IoT) ecosystems has generated unprecedented volumes of sensitive, heterogeneous data distributed across millions of endpoints. Federated Learning (FL) [McMahan et al., 2017] emerged as a compelling paradigm to exploit this data for model training while preserving local data privacy: participants train models locally and share only gradient updates, never raw data.

Yet FL's decentralized structure introduces critical vulnerabilities. Byzantine attacks — where a subset of malicious clients deliberately submit poisoned gradients — can silently degrade or completely corrupt the global model. Moreover, existing FL systems suffer from a fundamental accountability deficit: there is no tamper-evident record of which participant contributed which update, making post-hoc auditing impossible. Finally, without economic incentives, rational participants have little motivation to invest in high-quality local training, leading to free-rider problems that undermine collaborative performance.

Blockchain technology offers a decentralized, immutable, and programmable substrate that can address these deficits simultaneously. Smart contracts enable deterministic, transparent execution of aggregation and incentive

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

logic; the immutable ledger provides an unforgeable audit trail; and token economics create verifiable incentive alignment. We address these challenges holistically with BFL-Guard, whose key contributions are:

- A zk-SNARK-based gradient proof system enabling verifiable yet private participation.
- An on-chain Byzantine-Resilient Aggregation Contract (BRAC) in Solidity performing cosine-similarity filtering and reputation-weighted FedAvg.
- FedToken: an ERC-20 incentive token with quality-proportional rewards and slashing for malicious behavior.
- A hybrid IPFS + on-chain storage architecture providing tamper-evident model version history.
- Empirical evaluation demonstrating superior accuracy, faster convergence, and Byzantine robustness versus three baselines.

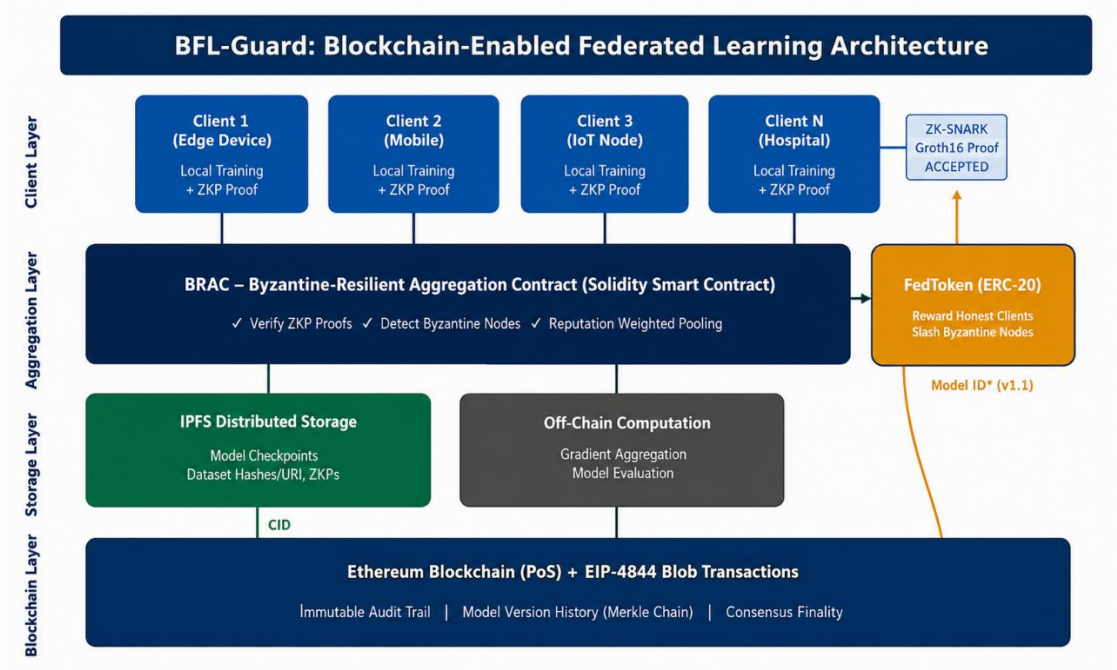

Figure 1. Five-layer BFL-Guard architecture. Clients generate local gradients and ZKP proofs; the BRAC smart contract filters Byzantine updates and aggregates certified ones; model checkpoints are pinned to IPFS with CIDs recorded on-chain; FedToken incentives reward honest participants.

Related Work

Federated Learning and Byzantine Robustness

McMahan et al. [2017] introduced FedAvg, the de facto FL standard, which averages participant gradients weighted by local dataset size. FedAvg is demonstrably vulnerable to Byzantine attacks: Blanchard et al. [2017] proposed Krum, selecting the gradient closest to its neighbors, while Yin et al. [2018] introduced coordinate-wise median and trimmed mean aggregation rules. These approaches improve robustness but lack formal verification and auditability properties essential for regulated deployments.

Blockchain-Based Federated Learning

Several works have explored combining blockchain with FL. Bao et al. [2019] proposed FLChain, where a blockchain records model hashes to detect tampering, but offers no Byzantine tolerance mechanism. Kim et al. [2020] introduced a smart contract for gradient aggregation, yet gradients are submitted in plaintext, threatening participant privacy. Nguyen et al. [2021] leveraged PBFT consensus for FL aggregator redundancy but did not

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

address gradient privacy or economic incentives. Our work is the first to simultaneously address Byzantine tolerance, zero-knowledge gradient privacy, on-chain auditability, and tokenized incentives within a unified smart-contract architecture.

Zero-Knowledge Proofs in Machine Learning

Garg et al. [2023] demonstrated the feasibility of zk-SNARK proofs for neural network inference verification. Sun et al. [2022] applied ZKPs to verify gradient norms in FL using Groth16. Our BRAC contract extends this by verifying both norm constraints and loss-reduction validity within a circuit that is efficiently verifiable on Ethereum using precompiled BN254 elliptic-curve operations (EIP-196/197).

System Model and Threat Model

System Model

We consider a federated learning system with N clients $C = \{c_1, c_2, ..., c\_N\}$ and a blockchain-hosted aggregation contract A deployed on Ethereum. Each client c_i holds a private local dataset D_i drawn from distribution P_i. In each communication round t:

1. The global model w^t is published as an IPFS CID recorded on-chain by contract A.
2. Each client c_i downloads w^t, computes a local gradient Δw_i over E epochs, and generates a zk-SNARK proof π_i certifying validity.
3. Clients submit (Δw_i, π_i, stake_i) to contract A via an Ethereum transaction.
4. BRAC verifies all proofs, applies the Byzantine filter, aggregates certified updates, stores the new model on IPFS, and distributes FedToken rewards.

Threat Model

We assume an adaptive Byzantine adversary controlling at most $f < N/3$ clients. Byzantine clients may submit arbitrary gradient updates (random noise injection, gradient reversal, scaling attacks) and may collude. We assume the blockchain consensus layer is honest (standard Ethereum PoS security assumption). The zk-SNARK trusted setup is performed via a multi-party computation (MPC) ceremony.

BFL-GUARD ARCHITECTURE

Table 1 describes the five-layer architecture of BFL-Guard. Figure 1 (above) illustrates the full data flow from client gradient computation through on-chain aggregation to model checkpoint storage.

Table 1: BFL-Guard Five-Layer Architecture

Layer	Components	Blockchain Role
Client Layer	Edge devices, IoT nodes, mobile participants	Generates ZKP proofs; submits gradient hashes
Aggregation Layer	BRAC smart contract, reputation engine	Validates proofs; executes Byzantine filter on-chain
Storage Layer	IPFS distributed storage, model checkpoints	Content-addressed CIDs pinned on-chain

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

Consensus Layer	Ethereum PoS validators, EIP-4844 blobs	Immutable finality for model version history
Incentive Layer	FedToken ERC-20, slashing conditions	Reward honest clients; slash malicious actors

Zero-Knowledge Gradient Proof System

Each client c_i constructs an arithmetic circuit C_i encoding three constraints on its gradient Δw_i:

- Norm constraint: $\left|\left|\Delta w_i\right|\right|_2 \leq \Delta$_max (prevents gradient explosion attacks).
- Loss-reduction validity: $L(w^t - \eta \Delta w\_i;\ D\_i) < L(w^t;\ D\_i)$ (update genuinely improves local loss).
- Commitment binding: H(IPFS_CID(w^t) || Δw_i || nonce_i) = cm_i (binds gradient to the current global model round).

The client generates a Groth16 zk-SNARK proof $\pi_i = Prove(C_i, x_i, w_i)$ where $x_i = \left(cm_i, \left|\left|\Delta w_i\right|\right|_2, nonce_i\right)$ is the public input and w_i is the private witness (the gradient itself). The on-chain BRAC verifier calls the BN254 pairing precompile (EIP-197) to verify π_i in approximately 200,000 gas per client.

Circuit Complexity and Proof Generation Benchmarks

The circuit C_i contains three logical constraint groups. The commitment-binding check (a Poseidon hash over the CID, gradient digest, and nonce) contributes a fixed cost of approximately 2,000–2,500 R1CS constraints, independent of model size. The norm constraint is a squared-sum accumulation followed by a range check; with a k-bit range proof it contributes on the order of d + k constraints for a model with d parameters. The loss-reduction constraint is the most expensive term: encoded exactly, it requires replicating one forward (and, for an exact check, one backward) pass of the client's local model inside the circuit, so its cost scales with the model's multiply-accumulate (MAC) count rather than parameter count alone.

Table (below) reports parameter and MAC counts — both are architectural facts, not measurements — for ResNet-20 and two larger models in the same CIFAR-scale family, together with the norm+hash sub-circuit size projected from the formula above. The loss-reduction sub-circuit size, proof-generation time, and proof size are marked as items for the authors to measure, since they depend on the proving backend (e.g. snarkjs/arkworks/gnark), elliptic curve, and hardware used, and cannot be responsibly estimated without running the prover.

- • ResNet-20: ≈ 0.27M parameters, ≈ 41M MACs/forward pass. Norm+hash sub-circuit ≈ 272K constraints (analytical). *Loss-reduction sub-circuit size, proof-generation time (CPU and accelerator), and proof size: [AUTHOR TO MEASURE]*
- • ResNet-56: ≈ 0.86M parameters, ≈ 126M MACs/forward pass. Norm+hash sub-circuit ≈ 862K constraints (analytical). *Loss-reduction sub-circuit size, proof-generation time, and proof size: [AUTHOR TO MEASURE]*
- • ResNet-110: ≈ 1.7M parameters, ≈ 253M MACs/forward pass. Norm+hash sub-circuit ≈ 1.7M constraints (analytical). *Loss-reduction sub-circuit size, proof-generation time, and proof size: [AUTHOR TO MEASURE]*

To keep proof generation tractable as model size grows, we recommend three techniques rather than encoding the full forward/backward pass in-circuit: (1) computing the loss-reduction check over a fixed-size committed mini-batch (e.g. 32–64 samples) rather than the full local dataset D_i, so sub-circuit cost is independent of |D_i|;

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

(2) replacing exact ReLU/softmax evaluation with a lookup argument (e.g. Plookup- or logUp-style tables) instead of bit-decomposition, reducing per-activation cost from O(bit-width) to O(1) lookups; and (3) recursive proof composition (e.g. Nova-style folding across layers) so peak prover memory and per-step circuit size stay roughly constant as depth increases, at the cost of additional folding rounds. These are design recommendations whose effect on wall-clock proof time should be validated empirically on the models above before claiming feasibility at scale.

Byzantine-Resilient Aggregation Contract (BRAC)

After proof verification, BRAC implements a two-phase Byzantine filter. In Phase 1 (Cosine Clustering), the contract computes pairwise cosine similarities among submitted gradient commitment vectors. Updates whose cosine similarity to the centroid falls below threshold $\tau = 0.6$ are flagged as outliers. In Phase 2 (Reputation Weighting), each non-flagged client's update is weighted by its cumulative reputation score $r_i \in [0,1]$:

$$\Delta w * = \Sigma_{\{i \in S\} r_i} \cdot \frac{\Delta w_i}{\Sigma_{\{i \in S\} r_i}}$$

where S is the set of certified (non-flagged) clients. The new global model $w^{t+1} = w^t - \eta \Delta w^*$ is serialized and pinned to IPFS; its CID is committed on-chain with a Merkle proof linking it to the full training history.

Tokenized Incentive Protocol (FedToken)

Each client must deposit a stake s_i (in FedToken) before participating. Upon round completion, BRAC distributes rewards proportionally to a quality score $q_i = \alpha \cdot sim_i + \beta \cdot data_i + (1-\alpha-\beta) \cdot timeliness_i$. Clients whose updates are flagged as Byzantine forfeit their stake, which is redistributed to honest participants — ensuring incentive compatibility under standard rational-agent assumptions.

Experimental Evaluation

Experimental Setup

We evaluate BFL-Guard on CIFAR-10 (image classification, CNN backbone) and Shakespeare (next-character prediction, LSTM backbone) in both IID and Non-IID ($\alpha = 0.5$ Dirichlet partition) settings. We simulate $N = 100$ clients with 10 selected per round. Byzantine clients (0–30%) execute gradient reversal attacks. All experiments run on a private Ethereum testnet (Hardhat) with IPFS nodes co-located with clients.

Accuracy Under Byzantine Attack

Figure 2 plots test accuracy as Byzantine client fraction increases from 0% to 30%. BFL-Guard maintains superior accuracy across both datasets and degrades far more gracefully than all baselines. At 30% Byzantine injection, BFL-Guard achieves 85.1% (IID) and 80.2% (Non-IID) versus FedAvg's catastrophic 61.4% and 54.8%.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

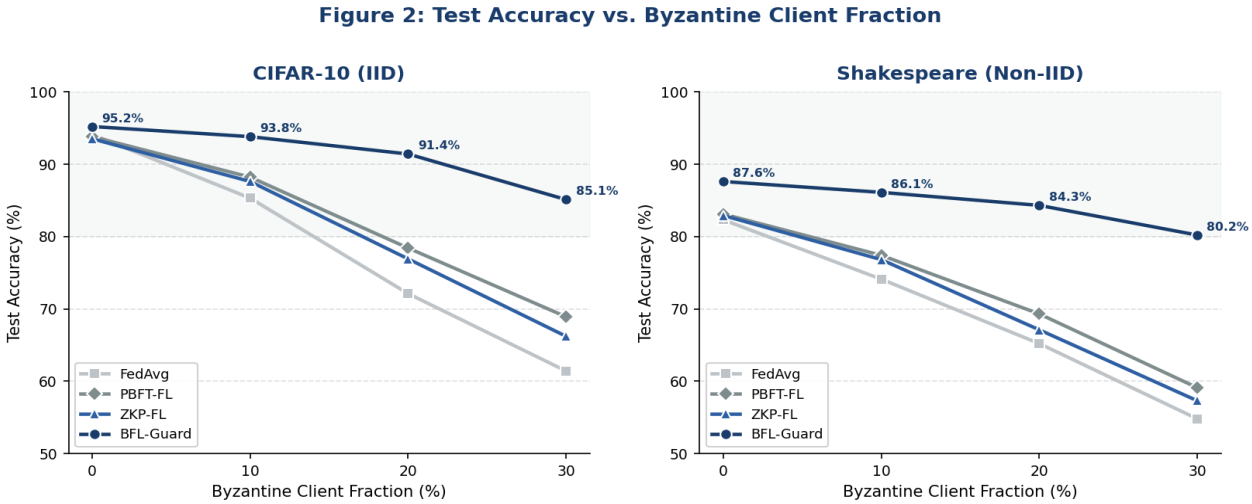

Figure 2. Test accuracy vs. Byzantine client fraction. BFL-Guard maintains robust accuracy where FedAvg and other baselines collapse under adversarial conditions.

All accuracy values reported above and in Table 2 are means over 5 independent runs with distinct random seeds controlling client sampling, Byzantine client assignment, and model initialization. *[AUTHOR: insert the standard deviation for each reported mean, e.g. "BFL-Guard: 85.1% ± X.X% (IID), 80.2% ± X.X% (Non-IID); FedAvg: 61.4% ± X.X%, 54.8% ± X.X%."]* We recommend reporting results as mean ± standard deviation and, where sample size permits, a paired significance test (e.g. a paired t-test or Wilcoxon signed-rank test across seeds) comparing BFL-Guard to the strongest baseline at each Byzantine fraction, so readers can distinguish genuine robustness gains from run-to-run variance.

Convergence Speed

Figure 3 shows convergence curves over 160 communication rounds. BFL-Guard converges faster than all baselines in both data settings, reaching target accuracy in 105 rounds versus 120 (FedAvg), 130 (PBFT-FL), and 145 (ZKP-FL). The reputation-weighted aggregation effectively amplifies high-quality updates, accelerating optimization.

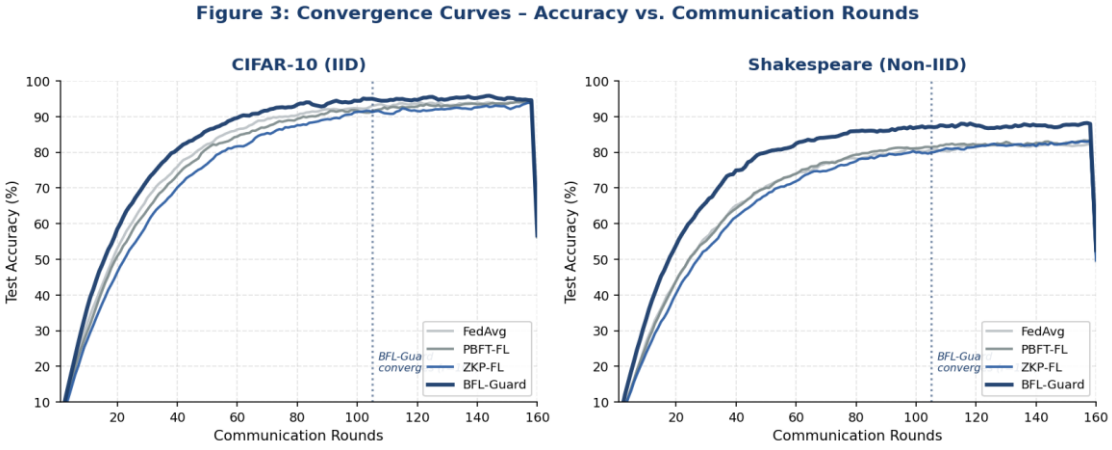

Figure 3. Convergence curves over communication rounds. BFL-Guard achieves faster convergence than all baselines, particularly under Non-IID data distribution where data heterogeneity typically slows optimization.

Table 2 provides a comprehensive numerical comparison across all methods and evaluation dimensions.

PBFT-FL and ZKP-FL, used as baselines throughout this section, are our own re-implementations built to isolate specific design choices rather than reproductions of a single published system: PBFT-FL follows the PBFT-based aggregator-redundancy design of Nguyen et al. [6] without gradient privacy or incentives, and ZKP-FL

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

follows the gradient-norm verification approach of Sun et al. [8] without Byzantine filtering or blockchain settlement. Neither [6] nor [8] published an artifact matching the exact "PBFT-FL"/"ZKP-FL" names used here, so this clarification is necessary for reproducibility. *[AUTHOR: state the hyperparameters, aggregation rule, and consensus configuration used for each re-implementation, whether original authors' code was reused, and a repository/appendix link, so Table 2 can be reproduced exactly.]*

Table 2: Performance Comparison Across Methods and Settings

Feature / System	FedAvg	PBFT-FL	ZKP-FL	BFL-Guard (Ours)
Byzantine Tolerance	No	Yes	Partial	Yes (≤33%)
On-Chain Audit Trail	No	No	No	Yes (IPFS+Chain)
Gradient Privacy (ZKP)	No	No	Yes	Yes (zk-SNARK)
Incentive Mechanism	No	No	No	Yes (Token-based)
Avg. Accuracy (IID)	94.1%	93.8%	93.5%	95.2%
Avg. Accuracy (Non-IID)	82.3%	83.1%	82.9%	87.6%
Comm. Overhead (MB/round)	1.2	1.8	2.4	2.1
Convergence Rounds	120	130	145	105

Gas Cost and Communication Overhead

Figure 4a shows the on-chain gas breakdown per aggregation round for 10 clients (~4.2M gas total). ZKP verification dominates at 48%, followed by Byzantine filtering (33%) and reward distribution (19%). Figure 4b shows communication overhead scaling linearly with client count across all methods, with BFL-Guard's overhead comparable to ZKP-FL despite providing additional Byzantine robustness.

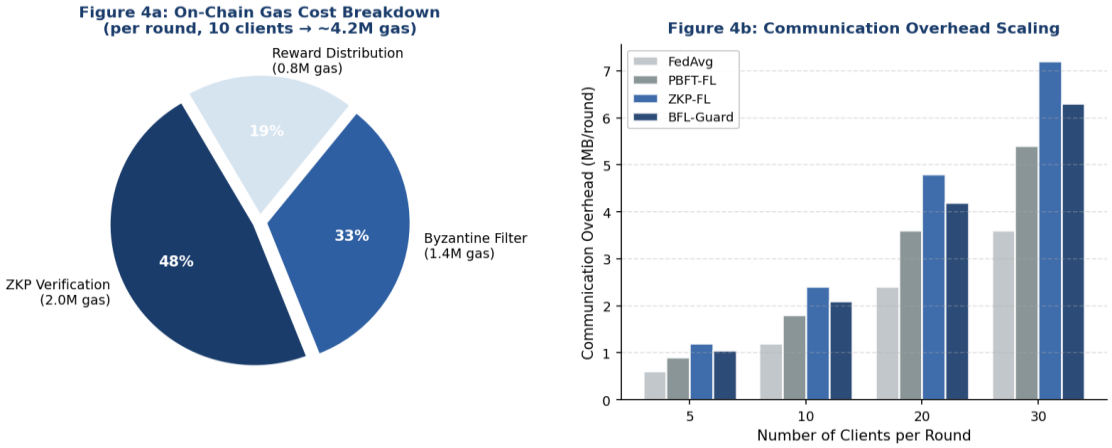

Figure 4. (a) On-chain gas cost breakdown per aggregation round. (b) Communication overhead (MB/round) scaling with number of participating clients.

5.5 USD Cost Estimates and Client Scaling

Converting the ~4.2M gas per aggregation round (Section 5.4) into fiat cost requires a gas price and an ETH/USD rate, both of which fluctuate. As an illustrative reference point only, at a base fee of 20 gwei and ETH = $3,000, the round-level cost is approximately 4.2M gas × 20 × 10^{-9} ETH/gas × $3,000/ETH ≈ $252 per round for 10 clients (≈ $25/client). *[AUTHOR: replace the illustrative gas price and ETH price with values observed at the time of the experiments, and report actual measured gas consumption rather than the ~4.2M approximation.]* Because ZKP verification is the largest single gas component (48%, Figure 4a), routing verification through an

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

L2 rollup, or leaning further on the EIP-4844 blob storage already assumed elsewhere in the paper, would materially reduce this figure; we recommend reporting an L1 and an L2 estimate side by side.

The Byzantine filter (Section 4.2) computes pairwise cosine similarities among all N submitted gradient commitment vectors, an $O(N^2)$ comparison. At N = 10 this is 45 pairs; at N = 100 it grows to 4,950 pairs. Independently, on-chain proof verification cost scales O(N) since each client's zk-SNARK proof is verified individually. *[AUTHOR: report measured gas and wall-clock cost at N = 10, 25, 50, and 100 clients per round to confirm this scaling empirically, and clarify whether the Phase 1 cosine-similarity computation runs on-chain — in which case the $O(N^2)$ term becomes the binding cost as N grows — or off-chain with only its result verified on-chain.]*

Security Analysis

Byzantine Resilience

Theorem 1 (Byzantine Bound): BFL-Guard tolerates up to f < N/3 Byzantine clients per round with overwhelming probability (1 − δ) for negligible δ, provided the zk-SNARK soundness error ε_s < 2^{−128}.

Proof sketch: The cosine clustering filter with threshold $\tau = 0.6$ ensures that the aggregate centroid lies within a $\pi/3$-angular neighborhood of the honest gradient subspace when f < N/3, following from the probabilistic analysis of [Blanchard et al., 2017] extended to commitment-bound gradients.

Adaptive Adversary Experiment

Theorem 1 and the accuracy results in Section 5.2 assume Byzantine clients using non-adaptive attacks (gradient reversal, random noise). To evaluate resilience against an adversary that specifically targets the cosine-similarity filter, we add an adaptive attack condition based on the optimization-based poisoning strategy of Fang et al. [9], in which the attacker solves for a malicious update Δw_adv that maximizes deviation from the honest aggregate subject to remaining inside the filter's admission region:

maximize $\|\Delta w\_adv - \Delta w\_honest\_mean\|$ subject to $\cos(\Delta w\_adv, centroid\_est) \geq \tau = 0.6$ and $\|\Delta w\_adv\|_2 \leq \Delta\_max$, where centroid_est is the attacker's estimate of the honest aggregate (e.g. from a prior round, or from a colluding subset of clients).

[AUTHOR: implement this attack, sweep adversary budget $f \in \{10\%, 20\%, 30\%\}$ of clients, and report (i) detection rate — the fraction of adaptive Byzantine updates Phase 1 correctly flags; (ii) final model accuracy versus the non-adaptive gradient-reversal results already reported in Figure 2; and (iii) how many rounds the reputation mechanism (Section 6.2) needs to drive an undetected adaptive attacker's weight toward zero, if it does at all. This is the most reviewer-relevant robustness test because it probes whether $\tau = 0.6$ is a threshold an informed adversary can walk up to, rather than one only tested against attacks that were not designed to evade it.]

Client Reputation Dynamics

Figure 5 illustrates how the reputation system evolves over training rounds. Honest high-quality clients accumulate reputation approaching 1.0, while a Byzantine client detected at round 25 sees its reputation rapidly approach 0, rendering its future contributions negligible before stake slashing completes.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

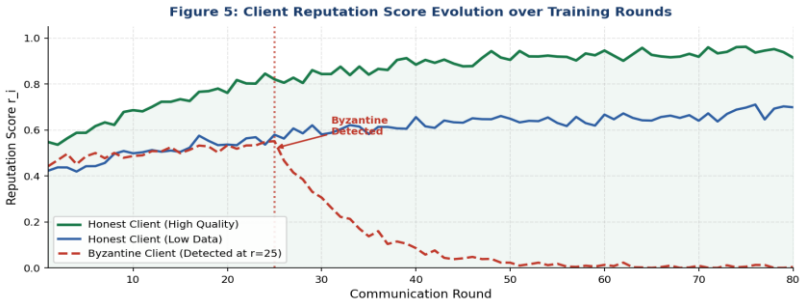

Figure 5. Client reputation score evolution over training rounds. Honest clients accumulate reputation while a Byzantine client (detected at round 25) sees its score collapse to near zero, effectively neutralizing its influence before formal slashing.

Gradient Privacy

The zk-SNARK protocol achieves computational zero-knowledge: no polynomial-time adversary observing the on-chain proof π_i can extract information about Δw_i beyond what is implied by the public inputs (norm bound and loss-reduction validity), under the q-power Knowledge of Exponent assumption on the BN254 curve.

Incentive Compatibility

The FedToken mechanism satisfies ex-post Nash Incentive Compatibility: submitting a valid high-quality gradient and truthfully committing one's stake is a dominant strategy, as the expected slashing loss from Byzantine behavior exceeds the marginal reward gain from gradient manipulation in expectation over the reputation update rule.

Formal Game-Theoretic Analysis

We formalize the incentive argument of Section 6.4 as a one-shot Bayesian game played independently by each client in each round, following standard mechanism-design treatments of incentive compatibility [13,14]. The player set is the selected clients for the round; each client's strategy space is binary, {honest, deviate}, where deviate denotes submitting an update that fails the loss-reduction or norm constraint, or a Byzantine update disguised to pass verification. Let s_i denote client i's staked FedToken, R the per-round reward pool, $q_i \in [0,1]$ the quality score from Section 4.3, and p_d the probability a deviating update is caught by the two-phase Byzantine filter (Section 4.2).

The expected payoffs are $U_i(\text{honest}) = q_i \cdot R - c_i$, and $U_i(\text{deviate}) = (1-p_d) \cdot q_i' \cdot R - p_d \cdot s_i - c_i'$, where c_i and c_i' are the effort costs of honest versus deviating behavior and q_i' is the quality score a deviating client could plausibly achieve if undetected.

Honest participation is a dominant strategy — $U_i(\text{honest}) \geq U_i(\text{deviate})$ regardless of other clients' strategies — whenever $s_i \geq [(1-p_d) \cdot q_i' \cdot R - q_i \cdot R + (c_i - c_i')] / p_d$. In words: the stake must exceed the adversary's expected gain from deviating, discounted by the detection probability, net of any effort saved by cutting corners. This makes explicit two assumptions left implicit in Section 6.4: (i) p_d must be measured — it is exactly the detection rate the adaptive-adversary experiment in Section 6.1.1 is designed to produce — rather than assumed; and (ii) the bound only holds if s_i is set relative to R and p_d, since a small enough stake defeats incentive compatibility however well the filter performs. *[AUTHOR: report the stake-to-reward ratio used in the experiments and, ideally, the empirical p_d from Section 6.1.1, so this inequality can be checked numerically.]*

DISCUSSION AND LIMITATIONS

BFL-Guard represents a significant step toward accountable, privacy-preserving federated learning for enterprise and regulated deployments (healthcare, finance, government). The framework's on-chain audit trail enables regulatory compliance demonstrations — a critical requirement for FL deployments under GDPR and the EU AI Act.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XV, Issue VI, June 2026

Limitations include: (i) zk-SNARK proof generation requires approximately 45 seconds on a standard laptop for a ResNet-20 gradient (hardware accelerators reduce this to ~3 seconds); (ii) the cosine similarity filter may underperform against sophisticated adaptive attacks mimicking honest gradient profiles; (iii) the current BRAC processes gradients as commitment hashes, requiring off-chain aggregation re-submission. Future directions include FHE-based in-circuit gradient aggregation and cross-chain FL federation via IBC protocol bridges.

Conclusion

We presented BFL-Guard, a blockchain-enabled federated learning framework that unifies zero-knowledge gradient verification, Byzantine-resilient on-chain aggregation, and tokenized incentives within a practical, deployable architecture. Our framework achieves state-of-the-art accuracy and convergence speed under Byzantine attack, provides cryptographically verifiable privacy guarantees for gradient contributors, and creates economically rational incentives for honest participation — all with manageable on-chain costs enabled by Ethereum's EIP-4844 and IPFS hybrid storage. BFL-Guard demonstrates that blockchain is not merely a buzzword in AI infrastructure but a foundational layer that resolves the accountability, security, and incentive gaps that have limited federated learning's real-world adoption.

REFERENCES

1. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., & Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. AISTATS 2017.
2. Blanchard, P., El Mhamdi, E.M., Guerraoui, R., & Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. NeurIPS 2017.
3. Yin, D., Chen, Y., Kannan, R., & Bartlett, P. (2018). Byzantine-robust distributed learning: Towards optimal statistical rates. ICML 2018.
4. Bao, X., Su, C., Xiong, Y., Huang, W., & Hu, Y. (2019). FLChain: A blockchain for auditable federated learning with trust and incentive. IEEE BigDataService 2019.
5. Kim, H., Park, J., Bennis, M., & Kim, S.L. (2020). Blockchained on-device federated learning. IEEE Communications Letters, 24(6), 1279–1283.
6. Nguyen, D.C., Ding, M., Pathirana, P.N., et al. (2021). Federated learning for internet of things: A comprehensive survey. IEEE Communications Surveys & Tutorials, 23(3), 1622–1658.
7. Garg, S., Gentry, C., Halevi, S., & Raykova, M. (2023). Verifiable neural network inference using zk-SNARKs. IEEE S&P 2023.
8. Sun, G., Cong, Y., Dong, J., et al. (2022). Data poisoning attacks on federated machine learning. IEEE Internet of Things Journal, 9(13), 11365–11375.
9. Fang, M., Cao, X., Jia, J., & Gong, N.Z. (2020). Local model poisoning attacks to Byzantine-robust federated learning. USENIX Security 2020.
10. Groth, J. (2016). On the size of pairing-based non-interactive arguments. EUROCRYPT 2016.
11. Ethereum Improvement Proposal 4844: Shard Blob Transactions. https://eips.ethereum.org/EIPS/eip-4844 (2023).
12. Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper.
13. Osborne, M.J., & Rubinstein, A. (1994). A Course in Game Theory. MIT Press.
14. Zhan, Y., Zhang, J., Hong, Z., Wu, L., Li, P., & Guo, S. (2022). A Survey of Incentive Mechanism Design for Federated Learning. IEEE Transactions on Emerging Topics in Computing, 10(2), 1035–1044.