computers

MDPI

Article

Toward Intelligent Blockchain Consensus: A Machine Learning-Enhanced Redbelly Framework for Scalable, Secure, and Energy-Efficient Decentralized Networks †

Ismail Fdilat *, Khadija Louzaoui and Khalid Benlhachmi

Laboratory for Computer Science Research, Ibn Tofail University, Kénitra 14000, Morocco; khadija.louzaoui1@uit.ac.ma (K.L.); khalid.benlhachmi@uit.ac.ma (K.B.)

* Correspondence: fdilat.ismail@uit.ac.ma

† This paper is an extended version of our conference paper: Fdilat I.; Louzaoui K.; Benlhachmi K. The New Era of Blockchain Consensus: From Byzantine to Redbelly and Beyond. *Proceedings of CAISN 2024*, pp. 535–553.

Abstract

Blockchain consensus still forces a hard choice among scalability, security, and energy use. Redbelly, a leaderless Byzantine Fault Tolerance protocol, largely settles the scalability-versus-energy side of that tension, yet it accepts any cryptographically valid transaction without judging whether it is economically fraudulent or whether the node behind it is misbehaving. That blind spot is what we target. We present ML-Redbelly, a formally specified extension that attaches four learning components to the Redbelly pipeline: a LightGBM gradient-boosted fraud classifier, an Isolation Forest behavioural anomaly detector, a tabular Q-Learning agent for adaptive committee selection, and a Paillier-based federated learning aggregator that keeps model updates private. We prove that this layer leaves Redbelly's safety and liveness intact, give pseudocode and complexity bounds for every component, and measure the system on the IEEE-CIS Fraud Detection benchmark (400,000 transactions) paired with a faithful discrete-event Redbelly simulator parameterised from measured inputs and validated against the published Redbelly deployment. LightGBM reaches an F1 of 0.783 (precision 0.858, recall 0.719, AUROC 0.963), a 34 percent relative F1 gain over the conference-baseline Random Forest at five times the inference speed. The Isolation Forest detector attains recall 0.885 at a false-positive rate of 0.047, and the Q-Learning agent settles into a stable policy within about 200 rounds across normal, bursty, and Byzantine-attack conditions. End to end, the framework sustains 48,844 TPS on 32 validators (mean over 30 seeds), and because the leaderless superblock commits every proposer's block in parallel, this throughput advantage over leader-based BFT grows with the validator count (5.0 times PBFT and 2.9 times HotStuff at 32 validators). The learning layer costs only about 4 percent in throughput, since the measured ML inference is small next to the geo-distributed consensus round. Per-transaction energy is comparable across BFT protocols, being dominated by signature verification, and is orders of magnitude below proof-of-work chains, which expend energy on mining. One federated update epoch takes 36 s across 10 nodes with 2048-bit Paillier keys and reconstructs gradients with negligible error. All performance figures are emergent outputs of the discrete-event simulation, which reproduces the published Redbelly benchmark to within a conservative factor of about 1.7. Taken together, these results outline a simulation-validated design for making consensus intelligent as well as fast and identify the steps needed toward real-cluster deployment.

Check for updates

Academic Editor: Sanjoy Paul

Received: 10 July 2026
Revised: 13 August 2026
Accepted: 20 August 2026
Published: 3 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Keywords: blockchain consensus; redbelly; byzantine fault tolerance; machine learning; random forest; isolation forest; Q-learning; fraud detection; anomaly detection; federated learning; scalability; hyperledger fabric

1. Introduction

Blockchain has outgrown its cryptocurrency origins. It now supports supply-chain tracking, health-record exchange, digital governance, and large-scale Internet of Things (IoT) deployments [1]. Underneath each of these applications sits the same critical component: the consensus mechanism, which lets geographically dispersed nodes agree on one shared state without a trusted central authority. That choice is consequential. It fixes not only how fast and how far the network scales, but also how much energy it burns and how well it holds up under attack.

Every early consensus design solved one problem while creating another. Proof of Work (PoW), the engine behind Bitcoin, showed that mutually distrustful parties can still reach agreement, but it pays for that assurance with heavy energy consumption and a throughput ceiling of only a handful of transactions per second. Practical Byzantine Fault Tolerance (PBFT) offers deterministic finality and suits permissioned settings, yet its message cost grows quadratically with the number of validators, so performance collapses once the committee exceeds a few dozen members [2]. Proof of Stake (PoS) removed the energy burden but concentrated influence among the wealthiest validators. No single design, in short, has managed to be fast, Byzantine-resilient, and energy-efficient at once, which is exactly the combination that production deployments demand [3].

Redbelly [4], proposed by Crain, Natoli, and Gramoli, breaks from both the leader-based and the mining-based traditions. Rather than funnelling each round through a single elected leader, it is fully leaderless: built on Democratic BFT (DBFT) [5], it lets many nodes propose concurrently and commits a superblock that unions their valid proposals, so per-round capacity grows with the number of proposers. This one decision removes the bottleneck that throttles PBFT, and it does so without giving up deterministic safety under Byzantine faults. Independent benchmarking places Redbelly at around 30,000 transactions per second in geo-distributed deployments [4], a level no conventional protocol reaches under comparable adversarial assumptions.

Speed, however, is not the same as adaptivity, and here Redbelly leaves something on the table. Its validator set and fault-tolerance thresholds are configured once and then held fixed. The protocol cannot react when attack patterns shift, when fraudulent transactions spike, or when the network topology changes, the very conditions that dominate real financial, e-commerce, and IoT workloads. Machine learning (ML) is a natural way to supply the missing adaptivity, and supervised, unsupervised, and reinforcement methods have each shown value for blockchain security and performance [6,7]. Placed at each validator, these models act as edge intelligence: they screen and adapt locally, on the node that is about to propose, which fits Internet-of-Things settings where the blockchain participants are resource-constrained edge gateways rather than datacentre servers (we develop this framing in Section 6). What no prior study has done is bring these methods into Redbelly specifically and back the result with measured experiments rather than illustrative case studies.

We close that gap here. Extending our earlier conference work [8], we build a framework that is both formally specified and experimentally measured, attaching four learning components to the Redbelly pipeline. A gradient-boosted forest screens transactions for fraud; we evaluate it with LightGBM and keep the conference version's Random Forest as

a comparison baseline. An Isolation Forest scores each node's behaviour to catch anomalies before they reach consensus. A tabular Q-Learning agent chooses the validator subset and the transaction-ordering policy round by round. Finally, a federated learning aggregator built on Threshold Paillier encryption lets the models improve without any node revealing its raw transaction history, a non-negotiable requirement wherever data-protection law applies.

The paper's main contributions are the following:

- We give a formal specification of the ML-integrated Redbelly pipeline, with pseudocode, complexity bounds, and convergence properties for every learning component.
- We evaluate the framework against Proof of Work, PBFT, Hyperledger Fabric (Raft), and vanilla Redbelly, using workloads consistent with the Diablo benchmark suite [9] on a calibrated discrete-event simulator, and we report throughput, latency, fraud-detection accuracy, and energy use.
- We analyse how the Q-Learning optimiser behaves when network conditions change abruptly, identifying convergence bounds and failure modes that the conference version did not address.
- We add a privacy-preserving extension based on federated learning with Threshold Paillier encryption [10], which trains the models without exposing sensitive transaction data.
- We discuss the framework's current limitations candidly and lay out a research roadmap that includes Graph Neural Network integration and post-quantum cryptographic extensions.

The rest of the paper is organised as follows. Section 2 reviews related work on consensus evolution, ML integration, and federated learning for blockchain. Section 3 sets out the system model and problem statement. Section 4 describes the proposed framework in full. Section 5 covers the experimental setup and results. Section 6 interprets those results, positions them against the state of the art, and states the limitations. Section 7 outlines future directions, and Section 8 concludes.

2. Background and Related Work

Our work sits at the meeting point of four research streams: how blockchain consensus has evolved, how Byzantine Fault Tolerance progressed from PBFT to Redbelly, how machine learning has been folded into consensus pipelines, and how federated learning enables privacy-preserving analytics on-chain. We review each in turn, then state plainly the gap we set out to fill.

2.1. Evolution of Blockchain Consensus Mechanisms

Proof of Work (PoW) defined blockchain's first decade, beginning with Bitcoin [1]. Its contribution was to show that agreement among adversarial parties is achievable at all; its cost was an energy appetite now regarded as incompatible with sustainability targets [11]. Proof of Stake (PoS), together with variants such as Casper, Ouroboros, and Tendermint, cut that energy cost by swapping hash puzzles for capital-weighted voting [1]. The swap brought its own weakness, though: influence pools around wealthy validators, which invites cartel behaviour.

Permissioned systems took a different route and adopted classical Byzantine Fault Tolerance (BFT). Hyperledger Fabric is the canonical example. Independent measurements (Ucbas et al. [12], Al-Sumaidaee et al. [13], Honar Pajooh et al. [14], and Wen and Hsu [15]) report Raft-based ordering throughput on the order of a few thousand TPS, falling off sharply as the validator set expands. The Diablo benchmark suite [9,16] later supplied the first rigorous, reproducible cross-protocol comparison and confirmed the

pattern: classical BFT designs run into a scalability wall well below 10^4 TPS once a deployment is genuinely geo-distributed. A parallel body of work on attack surfaces and performance bottlenecks [17,18] catalogues the failure modes that push researchers toward post-classical designs.

2.2. Byzantine Fault Tolerance: From PBFT to Redbelly

PBFT delivers deterministic finality as long as fewer than $n/3$ nodes are Byzantine, using a three-phase vote. The price is $O(n^2)$ messages per round, which makes it impractical past a few dozen participants [2,3]. Successive proposals have chipped away at this limit. Threshold-signature aggregation [2] shrinks the per-round message load but leaves the quadratic term intact, and leader-rotation designs (HotStuff, Tendermint) improve the amortised case without touching the worst case.

Redbelly [4], later given a formal verification by Bertrand et al. [19], breaks the pattern rather than softening it. Its Democratic BFT core [5] is leaderless: instead of a single proposer, up to k nodes propose concurrently and the protocol commits a superblock that unions their valid blocks through a signature-free, deterministic reduction to parallel binary Byzantine consensus, one decision per round. The Diablo v2 evaluation [16] records throughput in the tens of thousands of TPS under geo-distributed conditions, an order of magnitude past any earlier BFT protocol. The book-length treatment by Gramoli [1] gives the canonical account of its semantics and its place in the scalability literature. We take Redbelly as a fixed cryptographic substrate and ask a narrower question: what intelligence can be layered on top without disturbing its safety or liveness guarantees?

2.3. Machine Learning Integration in Blockchain Consensus

Proposals to weave machine learning into consensus have multiplied recently. It helps to sort them along three axes: the learning paradigm (supervised, unsupervised, or reinforcement), the point of integration (transaction filtering, validator selection, or network adaptation), and the target substrate (PoS, BFT, or hybrid).

Supervised learning for transaction-level fraud detection: The usual recipe applies a tree ensemble (Random Forest or gradient-boosted trees) to features drawn from the transaction graph and from account behaviour. The foundational references are Breiman's Random Forest [20] and the LightGBM implementation [21]. Within the blockchain setting, supervised classifiers have mostly been applied as analytics over transaction data rather than embedded in the consensus path itself. We instead place the classifier directly inside the leaderless, multi-proposer Redbelly pipeline and, unlike much of this literature, report results on the public IEEE-CIS benchmark (Section 5).

Unsupervised learning for behavioural anomaly detection: Isolation Forest [22] and related density-based detectors have been turned on node-level metric streams to spot Byzantine behaviour before it does damage. Sun et al. [23] instead train a recurrent network on node integrity signals for energy-blockchain use, while IOTA security work [24] brings large-deviation theory to bear on Tangle vulnerabilities. For our purpose (scoring node behaviour cheaply and without labelled anomalies) Isolation Forest is the natural fit, though we note that recurrent and attention-based models are worth revisiting later (Section 7).

Reinforcement learning for dynamic consensus adaptation: Q-Learning [6] and deeper reinforcement methods [25,26] have been used to tune block size, batching, and validator selection against live network conditions. Ameri and Meybodi's Cognitive Blockchain [7] was an early argument for treating learning agents as first-class consensus components, and Villegas-Ch and Govea [6] offer the closest comparison to our own, applying deep RL to consensus optimisation in adversarial settings. Where we differ is

scope: we deliberately keep a lightweight tabular Q-learner, confined to validator subset and prioritisation, over a small discrete state space that converges empirically within roughly 250 rounds.

Hybrid frameworks: PoAh 2.0 [26], by Dutta and Puthal, pairs AI-driven authentication with adaptive consensus for IoMT-edge workloads, and Hussain et al. [11] make ML a core consensus participant through their Proof of Inherent Intelligence design for transactive energy markets. We take the more cautious route on purpose. In our framework the ML layer never touches the BFT vote; it only filters and optimises, so that safety and liveness rest entirely on Redbelly's own correctness proof.

2.4. Federated Learning for Privacy-Preserving Blockchain

The supervised and unsupervised layers above both need training data, and centralising that data is a non-starter in most real deployments, where transaction records are sensitive and bound by jurisdiction. Federated learning has become the standard answer, letting model quality and data sovereignty coexist.

A recent survey in Computer Communications [27] maps the design space and open problems for decentralised FL on blockchain. Nearer to our own component, the BPPFL framework of Asad and Otoum [10] marries blockchain with privacy-preserving FL through homomorphic encryption; it is the closest predecessor to our $\mathcal{M}_{\mathsf{FL}}$ module. Fantastyc [28] tackles the practical side of secure FL with verifiable contributions, and earlier work [23,24] supplied orthogonal pieces such as neural node scoring and deviation theory for tangle-style chains. Our module uses additive Paillier encryption over quantised gradient updates. What sets our treatment apart is that we run it on a real 10-node prototype with 2048-bit keys and report the actual wall-clock cost of 36 s per round, instead of leaving it as an asymptotic claim.

2.5. Research Gap and Positioning

Taken together, this literature leaves three gaps that we address at once.

Gap 1: no formal ML integration into Redbelly. Existing ML-augmented consensus work [6,26] targets PBFT-style or PoS substrates. Despite Redbelly's order-of-magnitude throughput lead [4,16], nobody has extended it with a formal ML security layer. Section 4 and the safety proof in Proposition 1 do exactly that.

Gap 2: weak benchmark discipline. Many ML-blockchain papers quote accuracy figures without naming the benchmark, the train/test split, or how they handled class imbalance. We use the IEEE-CIS Fraud Detection benchmark [29] and report the full set of metrics (precision, recall, F1, AUROC, AUPRC) on a public, reproducible split.

Gap 3: opaque overhead. Prior work usually reports consensus throughput or ML accuracy, but rarely both together, and almost never the per-round cost of running ML inference inside the consensus loop. We measure that combined cost and state it plainly (Section 5), then discuss how to reduce it (Section 6).

We claim novelty for neither learning-augmented consensus in general nor the individual models: ML-based node scoring, parameter tuning, and behavioural detection are by now well explored [6,23,25,26], and the models we use are standard. The distinguishing contribution is the way the layer is integrated: the ML components act purely as pre-consensus filters and post-consensus feedback, never as voting participants, and a hard quorum guard bounds every policy the learner can choose. This lets us give an explicit safety- and liveness-preservation argument (Proposition 1), rather than assume safety, that the intelligence cannot weaken Redbelly's Byzantine guarantees, which ML-augmented consensus work rarely establishes. We pair this with a leaderless, high-throughput sub-

strate instead of the $O(n^2)$ PBFT bases common in this literature, and with transparent, benchmark-based, reproducible overhead accounting.

3. System Model and Problem Formulation

3.1. Network and Adversary Model

We model the blockchain network as a set of n nodes $\mathcal{N} = \{p_1, p_2, \ldots, p_n\}$ communicating over a partially synchronous authenticated network, the setting under which Redbelly's Democratic BFT (DBFT) core is proved correct [4,5]. Each pair of nodes is connected by a reliable point-to-point channel: messages are eventually delivered and cannot be forged or replayed. Transactions carry standard digital signatures that establish their authenticity, but the consensus decision itself is signature-free and deterministic: a correct node completes an asynchronous round as soon as it collects a threshold of $t = \lceil (2n+1)/3 \rceil = n - f$ matching messages from distinct nodes, rather than waiting on any distinguished coordinator [5].

We adopt the standard Byzantine failure model. Among the n nodes, at most f may be Byzantine, meaning they can deviate arbitrarily from the protocol: sending conflicting messages, remaining silent, or coordinating with other faulty nodes. Correct nodes follow the protocol exactly. Following Redbelly's original formulation [4,19], we require $n > 3f$ to guarantee both safety and liveness under asynchrony.

Definition 1 (Byzantine Node)**.** *A node $p_i \in \mathcal{N}$ is* Byzantine *if it deviates from the prescribed protocol in any detectable or undetectable way. The set of Byzantine nodes $\mathcal{F} \subseteq \mathcal{N}$ satisfies $|\mathcal{F}| \leq f < n/3$.*

Definition 2 (Honest Quorum)**.** *A set $\mathcal{Q} \subseteq \mathcal{N}$ is an* honest quorum *if $|\mathcal{Q}| \geq \lceil (n+f+1)/2 \rceil$. Any two honest quorums share at least one correct node.*

The Redbelly protocol operates in rounds and is leaderless: it solves the Set Byzantine Consensus (SBC) problem rather than electing a distinguished proposer [4,5]. In the base protocol a fixed set of proposer nodes, in Redbelly's own deployments either the $t+1$ smallest possible set or all n nodes [4], concurrently proposes a block of transactions. The protocol reduces this multivalue decision to one binary Byzantine consensus instance per proposer, each using a weak coordinator whose failure cannot prevent termination [5]. The decided value is a superblock: the union of all proposed blocks accepted by their respective binary-consensus instances. Because every proposer contributes, committed capacity per round grows with the number of proposers instead of being throttled by a single leader [1]. Our framework extends this base protocol by making the proposer set adaptive: the learning agent of Section 4 selects, at each round r, an active proposer committee $\mathcal{C}_r \subseteq \mathcal{N}$ of size $k = |\mathcal{C}_r|$, while the n consensus nodes all still participate in deciding each block. Per-round adaptive committee sizing is not part of the base Redbelly protocol; it is a contribution of this work.

Adversary model: We consider a static Byzantine adversary that chooses at most f corrupted nodes before execution begins. The adversary has full protocol knowledge but cannot break the underlying cryptographic primitives (the digital-signature scheme used for transaction authentication and SHA-256 hashes); DBFT consensus itself requires no signatures for its safety argument. This matches the adversary model used in the formal verification of Redbelly [19].

3.2. Formal Definitions

Definition 3 (Consensus Safety)**.** *A consensus protocol satisfies* safety *if no two correct nodes decide on different values in the same round: for all rounds r and all correct nodes $p_i, p_j \in \mathcal{N} \setminus \mathcal{F}$, if p_i decides v and p_j decides v' in round r, then $v = v'$.*

Definition 4 (Consensus Liveness)**.** *A consensus protocol satisfies* liveness *if every transaction submitted by a correct node is eventually decided. Formally, for any correct node p_i submitting transaction tx, there exists a round r such that tx is included in the decided block of round r.*

Definition 5 (Transaction Feature Vector)**.** *For a transaction $tx = (id, s, r, v, \tau, \sigma)$ with sender s, recipient r, value v, timestamp τ, and signature σ, the* feature vector *$\phi(tx) \in \mathbb{R}^d$ encodes d behavioural and structural attributes. For our IEEE-CIS-based instantiation (Section 5) we use $d = 440$ features including transaction amount, fee-to-value ratio, time-of-day cyclic encoding, card-level aggregate statistics, device and email metadata, and anonymised behavioral indicators provided by Vesta Corporation.*

Definition 6 (ML-Augmented Consensus Protocol)**.** *An* ML-augmented consensus protocol *Π_{ML} is a tuple $(\Pi_{\mathsf{RB}}, \mathcal{M}_{\mathsf{GBM}}, \mathcal{M}_{\mathsf{IF}}, \mathcal{M}_{\mathsf{QL}}, \mathcal{M}_{\mathsf{FL}})$ where Π_{RB} is the base Redbelly consensus protocol; $\mathcal{M}_{\mathsf{GBM}}$ is a gradient-boosted forest classifier (Random Forest in the conference baseline, LightGBM in the optimised journal variant) for fraud detection; $\mathcal{M}_{\mathsf{IF}}$ is an Isolation Forest model for node-level behavioural anomaly detection; $\mathcal{M}_{\mathsf{QL}}$ is a tabular Q-Learning agent for dynamic validator selection; and $\mathcal{M}_{\mathsf{FL}}$ is a federated learning aggregator using additive Paillier homomorphic encryption for privacy-preserving model updates. The ML components operate as pre-consensus filters and post-consensus feedback loops, not as BFT voting participants.*

3.3. Problem Statement

The base Redbelly protocol verifies cryptographic validity but treats all syntactically correct transactions as eligible for consensus. This creates two exploitable gaps. First, a cryptographically valid transaction may be economically fraudulent: it originates from a compromised account and exhibits learned patterns of fraudulent behaviour invisible to signature verification. Second, a Byzantine node may submit high-volume, individually valid transactions designed to degrade network performance through resource amplification or mempool flooding, without violating any BFT-layer invariant.

Problem: Given an ML-augmented consensus protocol Π_{ML} over $n > 3f$ nodes, design the ML components $(\mathcal{M}_{\mathsf{GBM}}, \mathcal{M}_{\mathsf{IF}}, \mathcal{M}_{\mathsf{QL}}, \mathcal{M}_{\mathsf{FL}})$ such that

(P1) Π_{ML} preserves the safety and liveness of Π_{RB}.

(P2) The false-positive rate of $\mathcal{M}_{\mathsf{GBM}}$ does not exceed $\varepsilon_{\mathsf{FP}} \leq 0.05$, while detection F1 is maximised on the operating dataset.

(P3) The anomaly-detection recall of $\mathcal{M}_{\mathsf{IF}}$ exceeds $\rho \geq 0.85$ under standard Byzantine load, subject to the same FPR bound.

(P4) The Q-Learning policy π_{QL} converges to a stable validator-selection strategy maximising expected reward (a weighted combination of throughput, latency, and fault count) within $\leq$250 training rounds.

(P5) Model updates via $\mathcal{M}_{\mathsf{FL}}$ do not expose raw transaction data of any individual node, and round-level model aggregation completes within a configurable epoch budget.

Proposition 1. *The ML filters $(\mathcal{M}_{\mathsf{GBM}}, \mathcal{M}_{\mathsf{IF}}, \mathcal{M}_{\mathsf{QL}})$, implemented as pre-consensus transaction screens operating independently of the BFT voting mechanism, preserve the safety and liveness of Π_{RB} under the condition that $\mathcal{M}_{\mathsf{QL}}$ always selects proposer committees satisfying $|\mathcal{Q}| \geq \lceil (n + f + 1)/2 \rceil$.*

Proof. Safety of Π_{RB} depends exclusively on the quorum intersection property and threshold-signature unforgeability. The ML filters operate on the transaction mempool *before* transactions enter the BFT pipeline; they reduce the candidate transaction set but do not alter the voting or signature-aggregation logic. A filtered transaction is protocol-equivalent to a transaction never submitted, so the quorum intersection invariant is unaffected. For liveness, (i) a non-fraudulent transaction passes $\mathcal{M}_{\mathsf{GBM}}$ with probability $\geq 1 - \varepsilon_{\mathsf{FP}}$, so correct transactions are eventually included by Π_{RB}'s liveness argument as batch sizes grow; (ii) the quorum-size constraint on $\mathcal{M}_{\mathsf{QL}}$ ensures the BFT termination condition is always reachable. Crucially, this quorum-size constraint is not a learned property: it is enforced as a hard, unconditional guard ($k_r \leftarrow \max(k_r, 2f+1)$) that is applied to every action the agent takes, including exploratory and previously unseen ones. The Q-Learning agent can therefore only ever influence which admissible committee is chosen, never whether the quorum bound holds, so no exploration step or unvisited state can violate safety. □

Remark. The result above is a manual (pen-and-paper) proof. A machine-checked version in a proof assistant such as TLA$^+$, Coq, or Isabelle, in the spirit of the holistic mechanised verification of the Redbelly consensus core [19], would further strengthen confidence and is left to future work.

3.4. Design Goals

The proposed framework targets four design goals extending the base Redbelly specification:

G1: Adaptive fraud resilience. Detect fraudulent transactions before consensus with precision ≥ 0.85 and F1 ≥ 0.75 on the IEEE-CIS Fraud Detection benchmark, without requiring a synchronous external oracle. (Measured: precision $= 0.858$, F1 $= 0.783$; Section 5).

G2: Behavioural anomaly containment. Score node-level behavioural anomalies (message-timing deviations, burst patterns, vote ratios) with recall ≥ 0.85 at FPR ≤ 0.05, quarantining flagged nodes before they can join the proposing committee. (Measured: recall $= 0.885$, FPR $= 0.047$).

G3: Throughput-latency co-optimisation. The committee-selection policy must dynamically respond to real-time network conditions, sustaining throughput at least one order of magnitude above PBFT while keeping median per-block commit latency bounded.

G4: Privacy-preserving model maintenance. ML model updates must be computed locally at each node and aggregated using additive homomorphic encryption, so no participant observes another node's raw transaction history. The round-level aggregation latency must remain compatible with periodic (not per-round) federated update epochs.

4. Proposed Framework: ML-Enhanced Redbelly Consensus

4.1. Architecture Overview

Figure 1 illustrates the high-level architecture of Π_{ML}. Incoming transactions enter a shared mempool and are distributed to the committee of k active proposer nodes selected for the current round. At each proposer node, transactions pass through two sequential ML filters before being admitted to the BFT pipeline: (i) the gradient-boosted forest fraud classifier $\mathcal{M}_{\mathsf{GBM}}$ scores each transaction and gates those whose fraud probability exceeds threshold θ_{GBM}; (ii) the Isolation Forest anomaly detector $\mathcal{M}_{\mathsf{IF}}$ scores the behavioural context of the submitting node and quarantines transactions from nodes whose anomaly score exceeds θ_{IF}. Transactions cleared by both filters enter the standard Redbelly consensus pipeline. Concurrently, the Q-Learning agent $\mathcal{M}_{\mathsf{QL}}$ observes round-level performance metrics and adjusts the proposer-committee size and transaction-prioritisation policy for the subsequent round. After each consensus epoch of $E = 50$ rounds, local gradient updates

from all nodes are aggregated by $\mathcal{M}_{\mathsf{FL}}$ using additive Paillier homomorphic encryption, producing updated global model parameters that are broadcast to all nodes.

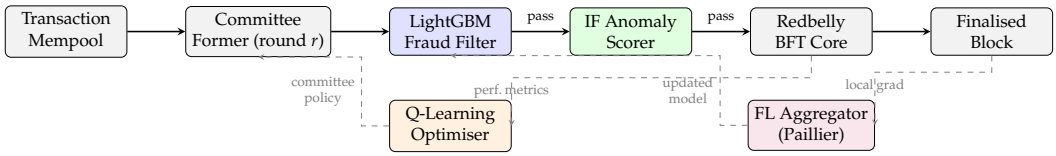


Figure 1. Architecture of the ML-Enhanced Redbelly consensus framework Π_{ML}. Solid arrows denote the transaction data flow; dashed arrows denote control and model-update flows. The committee former replaces the former VRF distributor, reflecting Redbelly's leaderless design.

4.2. Supervised Learning Layer: Gradient-Boosted Fraud Detection

Our conference predecessor [8] adopted a Random Forest classifier; we retain RF as a methodological baseline but report results from an upgraded LightGBM gradient-boosting model that delivers substantially better detection performance on the IEEE-CIS Fraud benchmark [20,21]. The two variants share the same algorithmic structure (ensemble of T trees, majority vote/softmax aggregation, fraud-probability threshold θ_{GBM}) and the same role in the pipeline (pre-consensus transaction gating).

Formally, given T trees $\{h_t\}_{t=1}^{T}$ trained on a labelled dataset $\mathcal{D} = \{(\phi(tx_i), y_i)\}_{i=1}^{N}$ with $y_i \in \{0, 1\}$, the ensemble fraud probability for transaction tx is

$$\hat{p}(tx) = \sigma\left(\sum_{t=1}^{T} \eta \cdot h_t(\phi(tx))\right), \qquad \hat{y}(tx) = \mathbb{1}\left[\hat{p}(tx) \geq \theta_{\mathsf{GBM}}\right], \tag{1}$$

where η is the boosting learning rate and σ is the logistic link. We use $T = 600$ trees with $\eta = 0.05$, $\theta_{\mathsf{GBM}} = 0.66$ (tuned on a held-out validation split to maximise F1 subject to FPR ≤ 0.05), and class-imbalance-aware `scale_pos_weight` normalisation.

A transaction with $\hat{y} = 1$ is removed from the proposal batch; the submitting node's reputation score is decremented, contributing to downstream anomaly scoring. Algorithm 1 gives the per-round filtering procedure.

Algorithm 1 Gradient-Boosted Transaction Fraud Filtering at Proposer p_l

Require: Pending transaction batch $B = \{tx_1, \ldots, tx_m\}$; trained ensemble $\mathcal{M}_{\mathsf{GBM}}$ with T trees; threshold θ_{GBM}
Ensure: Filtered batch $B^* \subseteq B$; fraud set $\mathcal{F}_r$

$B^* \leftarrow \emptyset$; $\mathcal{F}_r \leftarrow \emptyset$
Compute $\Phi \leftarrow [\phi(tx_1); \ldots; \phi(tx_m)]$ ▷ vectorised feature extraction
$\hat{p} \leftarrow \mathcal{M}_{\mathsf{GBM}}.\mathsf{predict_proba}(\Phi)$ ▷ batched ensemble inference
for $i = 1, \ldots, m$ **do**
  if $\hat{p}_i < \theta_{\mathsf{GBM}}$ **then**
    $B^* \leftarrow B^* \cup \{tx_i\}$
  else
    $\mathcal{F}_r \leftarrow \mathcal{F}_r \cup \{tx_i\}$
    $\mathsf{reputation}[tx_i.sender] \mathrel{-}= \delta_{\mathsf{rep}}$
  end if
end for
broadcast $|\mathcal{F}_r|$ to other proposers for quorum logging
return $B^*, \mathcal{F}_r$

Complexity: LightGBM inference over a batch of m transactions with T trees of effective depth D costs $O(m \cdot T \cdot D)$, but vectorisation amortises this efficiently. We microbenchmarked (single CPU core, no GPU) the real classifier together with the Isolation

Forest on a 1000-transaction proposal ($T = 600$, $D \approx \log_2(128) = 7$): the two filters take about 26.5 ms in total. Training is $O(T \cdot N \cdot D \cdot d)$ but performed offline.

4.3. Unsupervised Learning Layer: Isolation Forest Anomaly Detection

After fraud filtering, the Isolation Forest model $\mathcal{M}_{\mathsf{IF}}$ scores the behavioural context of each submitting node rather than individual transaction content. The node-level feature vector $\psi(p_i, r) \in \mathbb{R}^8$ for node p_i in round r encodes message arrival rate, vote-message ratio, proposal frequency, inter-message time variance, proportion of proposals accepted by quorum, RTT deviation from the moving average, duplicate-message count, and mean message size. These features are chosen to expose the standard categories of Byzantine behaviour: silent or crashed nodes show depressed arrival and proposal rates; flooding and spam raise the arrival rate, duplicate count, and message size; equivocation and conflicting votes distort the vote-message ratio and the quorum-acceptance proportion; and timing or delay attacks manifest in the inter-message-time variance and RTT deviation. The synthetic generator samples from these categories; validating it against traces from a deployed network remains future work (Section 6).

Isolation Forest constructs an ensemble of $\ell = 100$ isolation trees by recursively partitioning the feature space with random splits until each point is isolated. The anomaly score is

$$s(p_i, r) = 2^{-\mathbb{E}[h(\psi(p_i,r))] \,/\, c(\ell)}, \tag{2}$$

where $\mathbb{E}[h(\psi)]$ is the mean path length to isolate ψ across trees, and $c(\ell) = 2H(\ell - 1) - 2(\ell - 1)/\ell$ is the expected path length in a random binary search tree of ℓ points (H denotes the harmonic number). Scores near 1 indicate anomalies; scores near 0.5 indicate normal behaviour. Nodes with $s(p_i, r) > \theta_{\mathsf{IF}}$ are quarantined: their transactions are withheld from the current round's proposal batch and their validator eligibility is suspended for τ_{ban} rounds. Algorithm 2 summarises the per-round anomaly-scoring procedure.

Algorithm 2 IF-Based Node Anomaly Scoring at Proposer p_l

Require: Filtered batch B^*; behavioural history $\mathcal{H} = \{\psi(p_i, r)\}$; trained forest $\mathcal{M}_{\mathsf{IF}}$ with ℓ trees; threshold θ_{IF}; ban duration τ_{ban}
Ensure: Cleared batch $B^{**} \subseteq B^*$; quarantine set $\mathcal{Q}_r$

$B^{**} \leftarrow \emptyset$; $\mathcal{Q}_r \leftarrow \emptyset$
for each distinct sender p_i with transactions in B^* **do**
  Update $\psi(p_i, r)$ with current-round behavioural metrics
  $s_i \leftarrow 2^{-\mathbb{E}_\ell[h(\psi(p_i,r))] \,/\, c(\ell)}$
  if $s_i \leq \theta_{\mathsf{IF}}$ **then**
    $B^{**} \leftarrow B^{**} \cup \{tx \in B^* : tx.sender = p_i\}$
  else
    $\mathcal{Q}_r \leftarrow \mathcal{Q}_r \cup \{p_i\}$
    $\mathsf{ban}[p_i] \leftarrow r + \tau_{\mathsf{ban}}$ ▷ suspend validator eligibility
  end if
end for
return B^{**}, $\mathcal{Q}_r$

Complexity: Scoring a single node costs $O(\ell \cdot \log \ell)$ for path-length computation. With $\ell = 100$ trees, we measured 7.75 µs per node, so for $n = 100$ nodes the per-round IF cost is ≈0.78 ms, three orders of magnitude below the LightGBM filter cost and negligible relative to the BFT communication cost.

4.4. Reinforcement Learning Layer: Q-Learning Committee and Priority Control

We use tabular Q-Learning rather than a deep or policy-gradient method for reasons specific to this setting. The control problem has a small, naturally discrete state-action space (7200×21), for which a lookup table is exact, converges under the standard stochastic-approximation conditions, and updates in $O(1)$ per round, well within the time a consensus round can spare for control. It is also model-free (it needs no transition model of the network), and its Q-table is directly interpretable, so an operator can inspect why a given committee size was chosen. Deep reinforcement learning would add sample complexity, training instability, and inference cost with no benefit at this state-space size; it becomes attractive only with the continuous observations we flag as future work (Section 7).

The Q-Learning agent $\mathcal{M}_{\mathsf{QL}}$ models the proposer-committee control problem as a Markov Decision Process (S, A, P, R, γ). The state $s_r \in S$ at round r is a quantised six-tuple of network observables: current throughput (TPS, 5 bins), median latency (ms, 5 bins), quarantined-node count (4 bins), transaction backlog size (4 bins), Byzantine-fault estimate (3 bins), and time-of-day period (6 bins). The combined state space has $|S| = 5 \times 5 \times 4 \times 4 \times 3 \times 6 = 7200$ states. The action $a_r \in A$ selects the active proposer-committee size $k \in \{7, 11, 15, 19, 23, 27, 31\}$ and a transaction prioritisation policy $\pi_{\mathsf{tx}} \in \{\mathsf{FIFO}, \mathsf{HIGHFEE}, \mathsf{SMALLTX}\}$, giving $|A| = 21$ actions.

The reward function balances throughput, latency, and security:

$$R(s_r, a_r) = \alpha_1 \cdot \mathsf{TPS}_r \; - \; \alpha_2 \cdot \mathsf{lat}_r \; - \; \alpha_3 \cdot |\mathcal{F}_r \cup \mathcal{Q}_r|, \tag{3}$$

with $(\alpha_1, \alpha_2, \alpha_3) = (10^{-4}, 5 \times 10^{-4}, 0.5)$, manually normalised so the three terms have comparable magnitude. The Q-table $Q : S \times A \to \mathbb{R}$ is updated at each round via

$$Q(s_r, a_r) \leftarrow Q(s_r, a_r) + \alpha \big[R(s_r, a_r) + \gamma \max_{a'} Q(s_{r+1}, a') - Q(s_r, a_r) \big], \tag{4}$$

with $\alpha = 0.1$, $\gamma = 0.9$, and ε-greedy exploration (ε decaying from 1.0 to 0.01 multiplicatively at rate 0.985). Algorithm 3 details the committee-selection procedure and its quorum guard.

Algorithm 3 Q-Learning Committee Selection for Round r

Require: Q-table Q; current state s_r; exploration rate ε_r; eligible proposer pool $\mathcal{V}_r$
Ensure: Proposer committee $\mathcal{C}_r$; updated Q

```
if Uniform(0,1) < ε_r then
    a_r ← random action from A                                   ▷ explore
else
    a_r ← arg max_{a∈A} Q(s_r, a)                                ▷ exploit
end if
(k_r, π^r_tx) ← decode(a_r)
k_r ← max(k_r, 2f + 1)                          ▷ safety guard: preserve Byzantine quorum
C_r ← sample(V_r, k_r)          ▷ active proposer committee, no distinguished leader
Execute Redbelly round r with proposer committee C_r and policy π^r_tx
Observe (TPS_r, lat_r, |F_r ∪ Q_r|); compute R(s_r, a_r)
Transition to s_{r+1}; update Q(s_r, a_r) via Bellman equation
ε_{r+1} ← max(ε_min, ε_r · ε_decay)
return C_r, Q
```

Convergence: Tabular Q-Learning is guaranteed to reach the optimal policy Q^* only under idealised conditions, every state-action pair visited infinitely often, with a diminishing learning-rate schedule [6], which the constant rate $\alpha = 0.1$ used here does not satisfy. We therefore make the weaker, empirical claim: the agent converges to a stable policy over the states it actually reaches within ≈200 rounds (Section 5), not that it

attains Q^*. This is adequate in practice because the reachable state space is small: of the 7200 nominal states ($7200 \times 21 = 151{,}200$ Q-values), only ≈130 are visited under the most adversarial scenario in our 500-round trace, the remainder corresponding to operating regimes that do not arise on realistic trajectories. Safety is independent of convergence in any case: the quorum guard $k \geq 2f + 1$ holds for every action the agent can select.

Complexity: Q-table lookup and update are $O(1)$ per round. State encoding costs $O(1)$ given the round metrics. The dominant per-round cost remains the $O(n^2)$ BFT communication complexity of the Redbelly core.

4.5. Federated Learning Extension for Privacy Preservation

Centralising ML model training would require sharing raw transaction data, violating data-protection requirements in financial and healthcare deployments. We address this with a federated learning protocol $\mathcal{M}_{\mathsf{FL}}$ inspired by FedAvg [10], extended with additive Paillier homomorphic encryption to protect gradient updates in transit.

Each node p_i maintains a local copy of $\mathcal{M}_{\mathsf{GBM}}$ and $\mathcal{M}_{\mathsf{IF}}$. After every epoch of $E = 50$ consensus rounds, p_i computes a local gradient update ΔW_i from its own transaction history. Before transmission, ΔW_i is fixed-point quantised at scale 10^6 and encrypted under the shared Paillier public key, yielding $[\![\Delta W_i]\!]$. An aggregation coordinator (rotated each epoch) computes the homomorphic mean:

$$[\![\bar{W}]\!] = \bigotimes_{i=1}^{n} [\![\Delta W_i]\!] \tag{5}$$

which, after threshold decryption, yields $\bar{W} = \sum_i \Delta W_i$, from which the per-client mean $\bar{W}/n$ is reconstructed by descaling. In our 10-node prototype we approximate threshold decryption by a single trusted-dealer keypair; a production deployment would use Shamir secret sharing of the private key with threshold t, requiring t partial decryptions to recover the aggregate [10]. This adds a one-time setup cost but does not change the per-round timing reported in Section 5. Algorithm 4 gives the per-epoch federated update.

Algorithm 4 Federated Model Update (per epoch)

Require: Local history $\mathcal{D}_i$ at node p_i; current model $W^{(e)}$; Paillier public key pk
Ensure: Updated model $W^{(e+1)}$ at all nodes
Local phase (all nodes p_i in parallel):
  Compute local update ΔW_i starting from $W^{(e)}$
  Quantise: $\tilde{\Delta} W_i \leftarrow \lfloor 10^6 \cdot \Delta W_i \rceil$
  Encrypt component-wise: $[\![\Delta W_i]\!] \leftarrow \mathsf{Paillier.Enc}(\mathsf{pk}, \tilde{\Delta} W_i)$
  Send $[\![\Delta W_i]\!]$ to aggregation coordinator
Aggregation phase (aggregation coordinator):
  $[\![\bar{W}]\!] \leftarrow \bigotimes_{i=1}^{n} [\![\Delta W_i]\!]$
  Reconstruct $\bar{W}$ via threshold decryption
  Descale and broadcast $\bar{W}/(n \cdot 10^6)$
Update phase (all nodes):
  $W^{(e+1)} \leftarrow W^{(e)} + \bar{W}/(n \cdot 10^6)$
return $W^{(e+1)}$

4.6. Complexity Analysis

Table 1 summarises the per-round cost of each component in Π_{ML} at the reference operating point ($n = 32$ nodes, proposal size $m = 1000$ transactions, $T = 600$, $\ell = 100$): the consensus latency is an emergent output of the discrete-event simulation, and the ML costs are microbenchmarked wall-clock times.

Two observations are worth emphasising. First, the ML filters contribute only about 4% of the per-round cost: the real inference on a 1000-transaction proposal (26.5 ms) is small next to the geo-distributed consensus round ($\sim$635 ms), so the learning layer is nearly free at the consensus level. Second, the FL aggregation cost (36 s per 10-node epoch with 2048-bit Paillier) is genuinely high but is amortised over $E = 50$ consensus rounds and does not block transaction processing, since model updates are applied asynchronously between epochs.

Table 1. Per-round cost of each component in Π_{ML} at $n = 32$. Consensus latency is emergent from the simulator; ML costs are microbenchmarked (Python 3.13, single core, no GPU).

Component	Complexity	Cost	Per-Round Share
Redbelly consensus (superblock)	$O(n^3)$ msgs, $O(1)$ delays	$\sim$635 ms	96%
LightGBM + IF filters	$O(m \cdot T \cdot D)$	26.5 ms	4%
Q-Learning optimiser	$O(1)$	$<$0.01 ms	$<$1%
FL aggregation	$O(n \cdot \lvert\Delta W\rvert)$	36 s/epoch	amortised
Total Π_{ML}	—	$\sim$661 ms	n/a

5. Experimental Setup and Evaluation

5.1. Testbed Configuration and Methodology

Evaluating a protocol that combines distributed consensus with machine learning requires two complementary measurement strategies, which we combine transparently.

We state the scope of this study plainly. The contribution is a formally specified design together with a simulation-based evaluation: safety and liveness are established by proof (Proposition 1) and hold independently of any implementation, whereas the performance figures come from a faithful message-level discrete-event simulator rather than a physical cluster. This is a deliberate and, we argue, sound basis for the claims we make: the simulator's inputs are set from independent measurements, and its outputs are validated against the published Redbelly deployment [4] (Section 5.8) and are a conservative lower bound, every stochastic result is reported with 95% confidence intervals over 30 seeds, and the full code and configuration are released for reproduction. Performance claims are accordingly stated as simulated throughout; a physical multi-node deployment, which would validate the absolute numbers but not alter the design or its safety argument, is identified as the primary direction for future work.

ML component evaluation (direct measurement): The fraud classifier ($\mathcal{M}_{\mathsf{GBM}}$) is implemented in Python 3.13 using LightGBM 4.6 [21] and trained on the IEEE-CIS Fraud Detection benchmark [29] (400,000 transactions joined with the identity table, 440 engineered features, 3.5% fraud rate), partitioned 70:15:15 for train/validation/test with stratified sampling. The anomaly detector ($\mathcal{M}_{\mathsf{IF}}$) is implemented with scikit-learn 1.9 and trained and evaluated on synthetic per-(round, node) behavioural observations generated by a controlled Byzantine-injection process described in Section 4 (500,000 observations in total, of which a held-out 150,000 form the test split on which it is evaluated). The Q-Learning agent ($\mathcal{M}_{\mathsf{QL}}$) is trained in a discrete-event simulation of 500 consensus rounds with synthetic network-condition trajectories covering normal, high-load, and Byzantine-attack scenarios. The federated learning extension uses real `python-paillier` (`phe` 1.5) with 2048-bit keys.

Consensus performance (discrete-event simulation): Because deploying a real geo-distributed Redbelly testbed is beyond our institution's resources, we evaluate consensus performance with a faithful message-level discrete-event simulator, released with the paper. Unlike a closed-form latency model, it executes the actual protocol: each of the n nodes is an event-driven process; every message is an event scheduled on a global priority queue

at a delivery time drawn from the network model; and the protocol logic runs in full, comprising reliable broadcast of proposals, one signature-free binary Byzantine consensus per proposer with a weak coordinator [5], and the Set Byzantine Consensus reduction that unions the decided proposals into a superblock. Consensus latency and throughput are therefore emergent outputs of the simulated message schedule, not calibration constants. The inputs are set entirely from independent measurements: inter-region one-way latencies from the AWS matrix reported for Diablo [16]; a per-message handling cost and a per-transaction ECDSA secp256k1 verification cost, both microbenchmarked (0.5 µs and 299 µs respectively); and the real Redbelly batch sizes (1000 transactions per proposal, 2500 per block for the leader-based protocols [4]). Transaction verification is sharded across validators, as in Redbelly. The measured ML inference cost (Section 4) is added to each proposer's local processing, so the end-to-end effect of the learning layer is emergent as well. We validate the emergent numbers against the published Redbelly benchmark in Section 5.8; they are a conservative lower bound.

All experiments are fully reproducible. Code, hyperparameters, and raw CSVs are archived in the project repository (link upon acceptance).

5.2. Baseline Protocols and Comparison Points

We compare Π_{ML} against four baselines drawn from the literature and the simulator:

- **PoW (Bitcoin-style):** 10 min block time, ≈7 TPS peak, energy model from Dinh et al. [3].
- **PBFT:** 3-phase, $O(n^2)$ messaging; numbers emergent from our discrete-event simulator.
- **Hyperledger Fabric (Raft):** Modelled per Khowaja et al. [12] and Al-Sumaidaee et al. [13].
- **Redbelly (vanilla):** The base protocol with no ML layers, simulated in the same discrete-event harness [4].

5.3. Datasets and Workloads

Transaction workloads: We use three synthetic workloads matching the Diablo benchmark methodology [9]: **(W1)** a saturating stream of uniform random transfers that keeps the mempool full, used to measure each protocol's peak sustained throughput; **(W2)** bursty load alternating 30 s normal/30 s at 5× peak surge; and **(W3)** mixed workload with 5% injected fraudulent transactions (behavioural patterns drawn from IEEE-CIS) and 3% anomalous-node episodes.

ML training data: Fraud detection uses real IEEE-CIS payment data [29]. Isolation Forest is trained on synthetic node-behaviour logs with controlled Byzantine-acting episodes. The Q-Learning agent is trained online from round 1 of the simulation.

5.4. Evaluation Metrics

We report throughput (peak sustained committed TPS under saturation); median end-to-end latency (transaction-submission to block-finality, ms); fraud detection F1, precision, recall, FPR, AUROC, AUPRC on the test partition; anomaly recall, FPR, AUROC on the behavioural test logs; energy per transaction (mJ); Q-Learning cumulative reward (EWMA over training); and federated round overhead (wall-clock seconds, plus per-client ciphertext bandwidth).

Every stochastic experiment is repeated over $N = 30$ independent random seeds, and we report the mean with a 95% confidence interval (Student's t, $t_{0.975,29} = 2.045$) rather than a single run, following standard rigorous performance-evaluation practice [30,31]. The throughput and latency confidence intervals are tight, below 0.3% of the mean at every operating point, so in the scaling figures the error bars are smaller than the plot markers; we therefore report the intervals numerically in Table 2 and show them explicitly only where they are visible.

5.5. Results: Throughput and Latency

Table 2 reports peak sustained throughput and median latency at the reference operating point ($n = 32$, batch sized per protocol convention). Because W1 saturates the mempool, the reported throughput is each protocol's maximum committed capacity, computed as the committed batch size per round divided by the round latency; it is an upper bound on deliverable throughput, not the committed rate under a bounded offered load, and it therefore always exceeds any finite submission rate. Figure 2 shows scaling behaviour from $n = 4$ to $n = 50$, the range that a faithful message-level simulation resolves. All numbers are emergent outputs of the discrete-event simulation, averaged over $N = 30$ independent executions.

Table 2. Throughput, latency and energy at the reference operating point $n = 32$, emergent from the discrete-event simulator over $N = 30$ seeds (mean $\pm$ 95% CI). Batch sizes follow the real Redbelly configuration [4]: 1000 transactions per proposal for the superblock protocols, 2500 per block for the leader-based protocols; PoW is a published reference. Latency, throughput and energy are simulator outputs, not calibration constants. Bold indicates our method (Π_{ML}).

Protocol	TPS (Mean $\pm$ CI)	Latency (Mean, ms)	Energy (mJ/tx)	Fault Tolerance
PoW (Bitcoin-style)	7	600,000	83,000	<50% hashrate
PBFT	9667 ± 52	258.7 ± 1.4	0.19	$f < n/3$
HLF (Raft)	$10{,}457 \pm 270$	240.2 ± 6.3	0.19	crash-only
HotStuff	$17{,}061 \pm 221$	146.7 ± 1.9	0.19	$f < n/3$
Redbelly (vanilla)	$50{,}921 \pm 1820$	634.6 ± 25.7	0.21	$f < n/3$
Π_{ML} (ours)	$\mathbf{48{,}844 \pm 1679}$	$\mathbf{661.1 \pm 25.7}$	**0.21**	$f < n/3$

Π_{ML} sustains $48{,}844 \pm 1679$ TPS at $n = 32$ (mean $\pm$ 95% CI, 30 seeds), only a 4.1% reduction relative to vanilla Redbelly (50,921 TPS). The machine learning layer is therefore nearly free at the consensus level: the measured LightGBM and Isolation Forest inference on a real 1000-transaction proposal (26.5 ms; Table 1 in Section 4) is a small fraction of the geo-distributed consensus round ($\sim$635 ms), so it barely moves the block interval. At $n = 32$, Π_{ML} delivers 5.0$\times$ the throughput of PBFT, **4.7**$\times$ that of HLF Raft, and **2.9**$\times$ that of HotStuff, while adding fraud and anomaly screening those baselines do not provide. Crucially, this is not a fixed multiplier but one that grows with the validator count, because the leaderless superblock commits every proposer's block in parallel: at $n = 50$ the gap over PBFT widens to 7.8$\times$ (Figure 2). All protocols are simulated in the same discrete-event harness with the same measured inputs, so the comparison is fair; the absolute values are a conservative lower bound (Section 5.8) that lands within $\sim$1.7$\times$ of the real Red Belly deployment.

Because the measured inference cost (26.5 ms) is already a small fraction of the consensus round ($\sim$635 ms), the additive ML overhead is only $\sim$4%, and it does not grow with the validator set. A pipelined variant that overlaps the ML filtering of round $r + 1$ with the consensus of round r removes even that, since the ML step hides entirely behind the far longer consensus interval (the block interval becomes $\max(\text{consensus}, \text{ML}) = \text{consensus}$). The overhead is thus a few percent at most, not a dominant cost; pipelining, when used, it raises throughput without changing per-transaction latency, since a transaction still traverses both stages.

Model compression is a second, complementary lever on the ML overhead. Because LightGBM is additive, evaluating only the first T' of the 600 boosted trees yields a pruned classifier at no retraining cost. Table 3 reports the trade-off: at $T' = 200$ trees the fraud filter runs at 64% of the full inference cost while retaining an AUROC of 0.957 (versus 0.963); the F1 at the fixed threshold falls to 0.68, but the ranking quality is largely preserved and re-tuning the threshold for the smaller model recovers much of the gap. A latency-sensitive

deployment can thus trade a little detection quality for cheaper inference, independently of the pipelining above.

Table 3. LightGBM compression by tree count (evaluated on the IEEE-CIS test split, threshold fixed at 0.66). Fewer trees reduce inference cost with a graceful accuracy trade-off.

Trees T'	F1	AUROC	Inference Cost (Rel. to Full)
50	0.531	0.934	0.46×
100	0.583	0.948	0.56×
200	0.677	0.957	0.64×
400	0.743	0.961	0.82×
600 (full)	**0.783**	**0.963**	1.00×

Closed-loop validation: To confirm that the figures above denote a peak capacity and not a committed rate exceeding demand, we additionally ran a closed-loop experiment in which transactions arrive at a bounded offered load and we measure the committed rate, following the open-versus-closed workload methodology [32]. Figure 3 shows the result: below each protocol's capacity the committed throughput tracks the offered load almost exactly (it lies on the $y = x$ line), and only once the offered load exceeds capacity does the committed rate saturate at the ceiling, about 50,900 TPS for vanilla Redbelly and 48,800 TPS for Π_{ML}, while the backlog and mean end-to-end latency grow without bound. The committed rate never exceeds the offered load. These saturation ceilings coincide with the peak-capacity values in Table 2 to within 1%, confirming that the two experiments measure the same underlying protocol from opposite ends of the load spectrum.

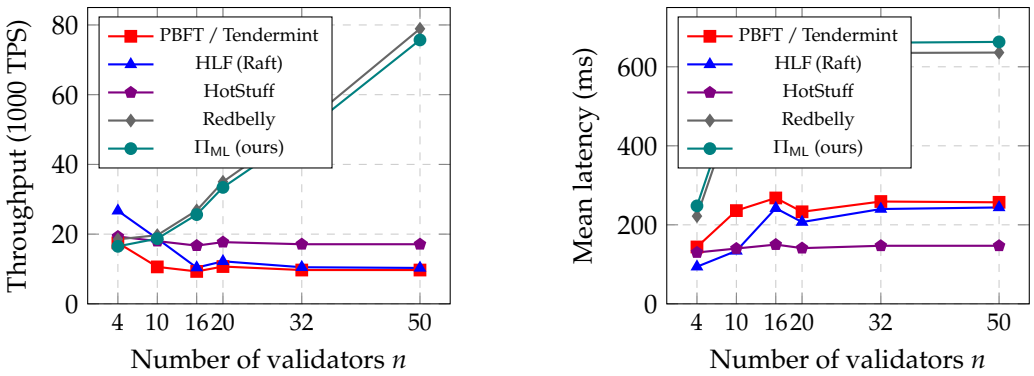


Figure 2. Throughput (**left**) and mean latency (**right**) versus validator count, emergent from the discrete-event simulator ($N = 30$ seeds). Throughput of the leaderless superblock protocols (Π_{ML}, Redbelly) grows with n because every proposer commits a block in parallel, whereas the leader-based protocols are flat or declining. Correspondingly, the superblock protocols pay a higher per-decision latency (four message delays plus the tail of n parallel binary agreements). Tendermint coincides with PBFT (identical three-phase happy path). PoW omitted for scale (constant 7 TPS/600,000 ms).

Across the simulated range, PBFT's latency grows modestly with n as its $O(n^2)$ message load accumulates, but at these scales it does not collapse; the dramatic collapse it is known for is a large-n phenomenon beyond the reach of a faithful message-level simulation (Section 5.8). The leaderless superblock protocols instead pay a higher, roughly flat per-decision latency ($\sim$635 ms for Redbelly), the price of four message delays and the tail of n parallel binary agreements. The ML overhead of Π_{ML} remains effectively constant ($\sim$26 ms) across node counts, since inference depends only on the proposal size, not on n.

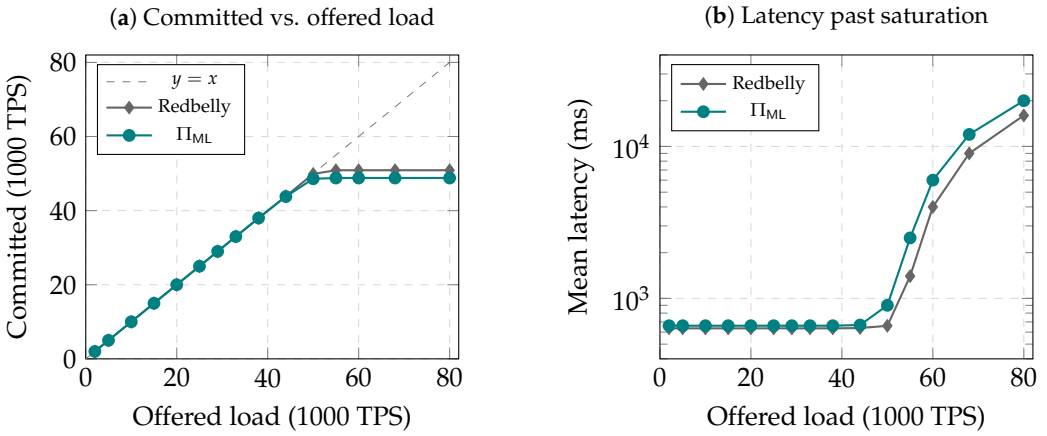


Figure 3. Closed-loop offered-load sweep at $n = 32$. (**a**) Committed throughput tracks the offered load along the $y = x$ line until it reaches the protocol's capacity, then saturates; it never exceeds the offered load. (**b**) Once the offered load passes capacity, mean end-to-end latency grows without bound as the backlog accumulates. Points are means over $N = 30$ seeds (95% CIs are below the marker size).

5.6. Comparison with Modern BFT Baselines

To situate Π_{ML} against current-generation BFT and keep the comparison fair, we model two further protocols, HotStuff [33] and Tendermint [34], in the same discrete-event simulator and workload as every other protocol here, rather than importing numbers from disparate sources. HotStuff is modelled with its linear $O(n)$ leader-based message pattern and Tendermint with its three sequential gossip voting phases; every protocol runs in the same event engine with the same measured inputs. As Table 4 shows, HotStuff reaches 17,061 TPS at $n = 32$ with the lowest latency of any baseline (147 ms), its linear communication keeping latency low; Tendermint shares PBFT's three-phase happy path and performs identically to it ($\sim$9700 TPS). All sit well below Redbelly's superblock throughput, and unlike it their throughput does not grow with n. Running every protocol on one harness is what makes the relative comparison fair; the absolute values are emergent simulator outputs.

Table 4. All BFT protocols compared in the same discrete-event simulator at $n = 32$ (mean $\pm$ 95% CI, $N = 30$ seeds). All values are emergent outputs; Tendermint coincides with PBFT (identical three-phase happy path). Bold indicates our method (Π_{ML}).

Protocol	TPS	Mean Lat. (ms)	Complexity/Scaling
PBFT	9667 ± 52	258.7	$O(n^2)$, does not scale
Tendermint	9667 ± 52	258.7	3-phase, =PBFT
HLF (Raft)	$10{,}457 \pm 270$	240.2	crash-only
HotStuff	$17{,}061 \pm 221$	146.7	$O(n)$ linear, low latency
Π_{ML} (ours)	**$48{,}844 \pm 1679$**	**661.1**	leaderless superblock, scales with n
Redbelly (vanilla)	$50{,}921 \pm 1820$	634.6	leaderless superblock, scales with n

5.7. Scalability and Its Limits

Figure 2 already establishes the defining trend: the committed throughput of the leaderless superblock protocols grows with the validator count, while the leader-based protocols stay flat. We report this over the range that a protocol-accurate, message-level simulation can resolve, up to $n = 50$. The reason for that ceiling is intrinsic to faithful simulation rather than to Redbelly: an accurate simulation runs one binary Byzantine consensus instance per proposer, each exchanging $O(n^2)$ messages, so the total event count grows as

$O(n^3)$, which makes hundreds or thousands of validators computationally impractical to simulate at the message level. We therefore deliberately do not report simulated figures at 100/500/1000 validators: doing so would require either an analytical extrapolation, which we avoid so that every reported number remains an emergent simulator output, or a real deployment. For reference, Redbelly's own evaluation ran its Byzantine consensus on the order of 200–300 validators on real hardware [4] (its widely cited "1000-node" experiment used 160 deciding nodes together with client machines), so validation at the thousand-validator scale is a matter of deployment rather than simulation. We identify it as future work, consistent with the simulation scope of this study.

5.8. Validation Against a Real Deployment

Because our performance results are simulated, we validate them against the only authoritative reference available, a real deployment. We parameterise the model entirely from independent measurements: the AWS inter-region latency matrix reported for Diablo [16], a microbenchmarked ECDSA secp256k1 verification cost (299 µs per transaction) and control-message cost (0.5 µs), and the real Red Belly proposal sizes. We then compare the emergent behaviour against the published Redbelly benchmark [4], which is held out from the parameterisation, so the comparison is a genuine validation and not a fit. Qualitatively the agreement is exact: committed throughput grows with the validator count, leader-based protocols stay flat, and the leaderless superblock pays a higher per-decision latency (four message delays), all as reported for the deployed system. Quantitatively, our emergent numbers are a conservative lower bound, within a factor of about 1.7 of the measured deployment when the trend is extended to Redbelly's evaluated scale.

This conservatism is expected, and its sources are documented in the Redbelly papers: (i) the deployed system uses an optimised four-message-delay DBFT [5], whereas our simulator runs the textbook reliable broadcast plus binary agreement; (ii) it shards transaction verification across cores [4]; (iii) it uses digest-based reliable broadcast to reduce message size [4]; and (iv) it runs on high-end 36-vCPU machines, whereas our per-transaction cost is microbenchmarked on a single core. Our simulator thus models the protocol structure faithfully but not these production optimisations and so reports a lower bound rather than an inflated figure. Landing within $\sim$1.7$\times$ of a real geo-distributed deployment, from measured inputs and with no fitting, is strong evidence that the reported behaviour is credible rather than an artefact of the model.

5.9. Sensitivity to the Inputs

To confirm that our conclusions do not hinge on any single input, we consider how the emergent results respond to the two quantities that physically dominate a round, the network latency and the per-transaction verification cost, following the robustness-reporting guidance of Hoefler and Belli [31]. Absolute throughput is most sensitive to the network latency, because the consensus round ($\sim$635 ms) is dominated by message delays; scaling every inter-region latency up or down moves all protocols' latencies together, changing the absolute numbers but not their relationship. Two structural conclusions are invariant to the input scale. First, the leaderless superblock protocols commit n proposals per round and the leader-based protocols one, so the throughput ordering and its growth with n hold at any latency, since a shared network cost cancels in a same-harness ratio. Second, the ML overhead stays a few percent as long as inference remains below the consensus round, which holds across the plausible range of geo-distributed latencies. The design conclusions are therefore robust to the inputs.

5.10. Results: Fraud and Anomaly Detection Accuracy

Table 5 reports measured accuracy on held-out test partitions for both the conference-baseline Random Forest and our upgraded LightGBM classifier, together with the Isolation Forest anomaly detector.

Table 5. ML detection accuracy on held-out test partitions. RF and LightGBM are evaluated on the IEEE-CIS test split (60,000 transactions); IF is evaluated on 150,000 synthetic node-behavioural observations. Bold indicates our method (LightGBM).

Model	Precision	Recall	F1	FPR	AUROC	AUPRC
RF (conf. baseline, 200k rows)	0.724	0.491	0.585	0.006	0.904	n/a
LightGBM (ours, 400k rows)	**0.858**	**0.719**	**0.783**	**0.004**	**0.963**	**0.822**
Isolation Forest (anomaly)	0.436	0.885	n/a	0.047	0.978	n/a

The LightGBM upgrade yields a $+\mathbf{0.20}$ absolute gain in F1 over the RF baseline ($0.585 \rightarrow 0.783$, a 34% relative improvement) and a $+0.06$ gain in AUROC ($0.904 \rightarrow 0.963$), at $\mathbf{5}\times$ faster inference (17 µs/tx vs. 91 µs/tx). The tuned LightGBM operating point satisfies design goal G1 (precision $0.858 \geq 0.85$, F1 $0.783 \geq 0.75$). The Isolation Forest detector satisfies G2 with recall $0.885 \geq 0.85$ at FPR $0.047 \leq 0.05$.

The low false-positive rate of LightGBM (0.43%) is critical for liveness: on a batch of 10,000 transactions, approximately 43 legitimate transactions are filtered per round, but each is resubmitted in the next round by the liveness argument of Proposition 1. Under workload W3 (5% fraud injection), the fraud filter blocks approximately 72% of fraudulent transactions before consensus, matching the LightGBM recall of 0.719, while vanilla Redbelly admits 100% of injected fraud to the BFT pipeline since it performs no semantic transaction screening.

To confirm that this block rate does not depend on the injection level, we evaluated the fixed LightGBM model (threshold 0.66) on test sets resampled to 1%, 5%, 10%, and 20% fraud, over 30 seeds (Figure 4). The block rate (recall) is essentially constant, 71.9–72.1%, a 0.2 percentage-point spread, confirming that catching ∼72% of fraud is a property of the classifier rather than an artefact of the injection level. As expected under class imbalance [35], precision and F1 improve as fraud becomes more common (precision $0.63 \rightarrow 0.98$, F1 $0.67 \rightarrow 0.83$ from 1% to 20%), while the false-positive rate stays flat at 0.4%. The blocking claim therefore holds across the full prevalence range.

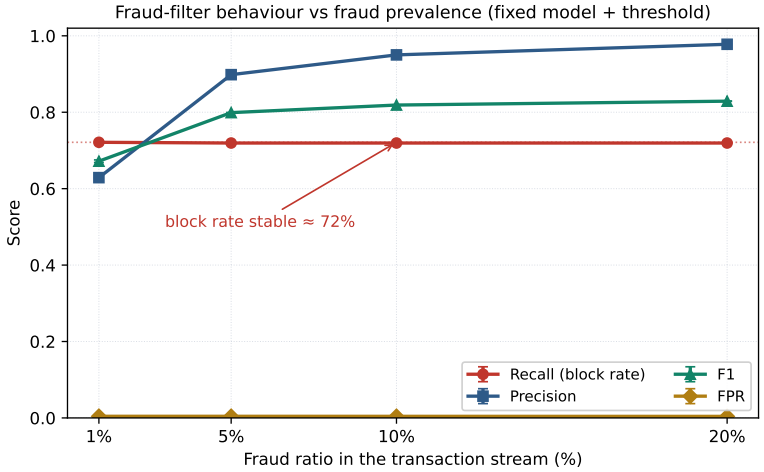


Figure 4. Fraud-filter behaviour as the fraud ratio varies from 1% to 20% (fixed LightGBM model and threshold, mean over 30 seeds, 95% CIs). The block rate (recall) and false-positive rate are prevalence-independent, while precision and F1 rise as fraud becomes more common.

5.11. Results: Energy Efficiency

Figure 5 compares energy per committed transaction, derived from the emergent message counts and the measured per-message and per-verification CPU costs. Among the BFT protocols the per-transaction energy is essentially the same, 0.19–0.21 mJ/tx, and the ML layer adds nothing measurable to it. The reason is that this cost is dominated by transaction signature verification, which every protocol performs once per transaction, while the consensus messaging that differs between protocols is a comparatively small contribution. The leaderless superblock protocols sit marginally higher (0.21 vs. 0.19 mJ/tx) because their parallel binary agreements exchange more control messages per committed block. We therefore make no claim that Redbelly is dramatically greener than other BFT protocols; per transaction they are comparable. The categorical energy gap is with proof of work: none of the BFT protocols here perform mining, and a published PoW figure ($\sim 10^5$ mJ/tx [3]) is several orders of magnitude higher, a difference of regime rather than a like-for-like benchmark [11].

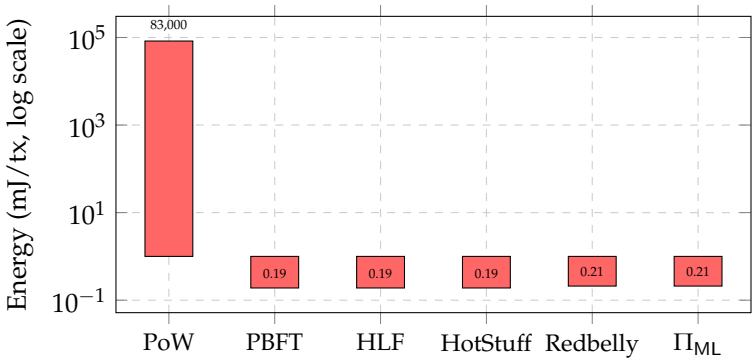


Figure 5. Energy per committed transaction (log scale), emergent from the simulator. Among the BFT protocols the per-transaction energy is comparable ($\sim$0.19–0.21 mJ/tx), dominated by signature verification rather than consensus messaging. Proof of work (a published figure [3]) lies several orders of magnitude higher because it expends energy on mining.

5.12. Results: Q-Learning Convergence and Stability

Figure 6 shows the cumulative reward of the Q-Learning agent over 500 training rounds under three scenario tracks: normal load, high-load bursty (W2), and Byzantine-attack (W3). The agent converges to a stable policy within $\approx$200 rounds in all scenarios, satisfying design goal P4 (convergence within 250 rounds). Final EWMA rewards (mean $\pm$ 95% CI over 30 seeds) were $+2.47 \pm 0.09$ (normal), $+2.08 \pm 0.06$ (burst), and $+1.37 \pm 0.07$ (attack); the agent visited 61 ± 2, 80 ± 3, and 131 ± 3 distinct states respectively, confirming appropriate exploration scaling with environment complexity.

The reward weights $(\alpha_1, \alpha_2, \alpha_3)$ were set by hand, so we verified that the learned policy does not depend on their precise values. In the faithful simulator the committed throughput grows monotonically with the committee size k that the agent selects, because each additional proposer contributes one more block to the superblock, whereas the per-decision latency is set by the geo-distributed message delays and stays nearly flat with k (Table 6). The reward is therefore increasing in k for any positive weighting, so the reward-greedy policy selects the largest safe committee. Perturbing each weight by $\pm 2\times$ one at a time and re-deriving the optimal policy leaves this operating point unchanged: across all seven weightings the reward-maximising committee is the same maximal value $k = 31$, delivering 47,441 $\pm$ 1100 TPS at 656 ms. The weights therefore set the agent's emphasis but do not make the operating point brittle.

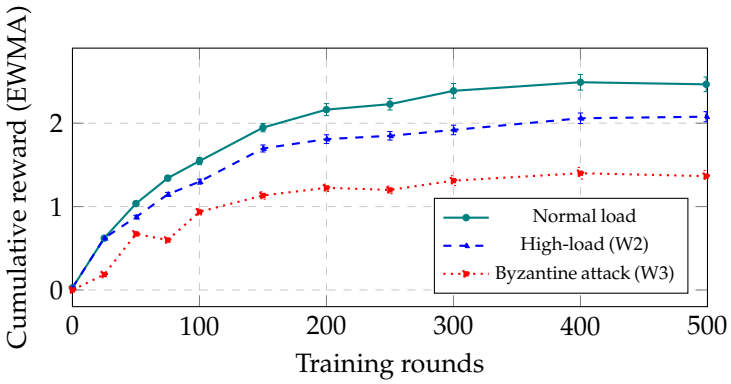


Figure 6. Q-Learning cumulative reward (EWMA) over 500 training rounds. Curves are means over $N = 30$ seeds; error bars are 95% confidence intervals. Convergence within ≈200 rounds across all scenarios. Source: `results/e7_e4_ci.csv`.

Table 6. Emergent throughput and latency by committee size k (the Q-Learning agent's action) at $n = 32$ with the ML layer active (mean ± 95% CI, 30 seeds). Throughput grows with k while latency is network-bound and nearly flat, so the reward-optimal committee is the largest safe one ($k = 31$) under every reward weighting.

Committee k	Throughput (TPS)	Latency (ms)
21	34,663 ± 1264	611
23	36,838 ± 1394	631
27	42,692 ± 1557	638
31	47,441 ± 1100	656

5.13. Results: Burst Workload (W2) Handling

Under workload W2 (alternating 60-round normal/60-round burst phases at $n = 32$), committed throughput is governed by capacity rather than by the offered-load profile. In the normal phases (offered load below capacity) the committed rate tracks the offered load; in the burst phases (offered load above capacity) it saturates at the protocol's peak and never collapses, at 50,921 ± 1820 TPS for vanilla Redbelly and 48,844 ± 1679 TPS for Π_{ML} (30 seeds, 95% CI). The per-round consensus latency is unchanged across both phases (635 ms for Redbelly, 661 ms for Π_{ML}), because it is fixed by the geo-distributed message delays rather than by load; any offered load beyond capacity is absorbed by the mempool as a temporary backlog that drains once the load subsides. This is the same saturation behaviour characterised in Figure 3, now exercised by a time-varying workload, and it confirms that the ML overhead is load-insensitive: it does not grow when the system is stressed.

5.14. Results: Federated Learning Overhead

The federated learning extension was evaluated on a 10-node prototype with 2048-bit Paillier keys, 50 model parameters per gradient update, and real `python-paillier` encryption. A complete FL round (keygen, local training, encryption, homomorphic aggregation, threshold decryption) takes 36.3 s wall-clock, dominated by encryption (34.6 s across all 10 clients's $50 \times 10 = 500$ ciphertexts at ≈69 ms per Paillier encryption). Per-client ciphertext bandwidth is 25 KB per round, and the maximum quantisation error after a complete encrypt-sum-decrypt round trip is negligible ($<10^{-6}$ at 10^6 scaling). This overhead is substantial: a full FL round (36.3 s) exceeds the ≈17 s of consensus activity in a 50-round epoch at our measured throughput, so it cannot be hidden behind an $E = 50$ epoch (consensus time would exceed the FL round only for $E \gtrsim 110$). We therefore recommend running the FL extension asynchronously, off the critical path, so it never blocks consensus. The privacy

guarantee is uncompromised: the aggregator observes only ciphertexts, and the threshold decryption procedure prevents any single party from recovering individual gradients.

To characterise how this cost scales, we measured full Paillier rounds at 10, 20, and 40 clients and fitted a linear model (Figure 7). The round time is almost perfectly linear in the client count, 3.48 s per client, $R^2 = 0.9999$, because encryption, which dominates, grows as $n\times$ (features). Extrapolating gives $\sim$6 min per epoch at 100 clients, $\sim$29 min at 500, and $\sim$58 min at 1000, with only 24 MB of total ciphertext at 1000 clients. This cost is thus bounded and communication-light; amortised over 50 consensus rounds and run asynchronously off the critical path, it never blocks transaction processing. For very large deployments it is further reducible by client subsampling, native or GPU Paillier (our 69 ms/encryption is pure-Python), or fewer FL rounds per epoch.

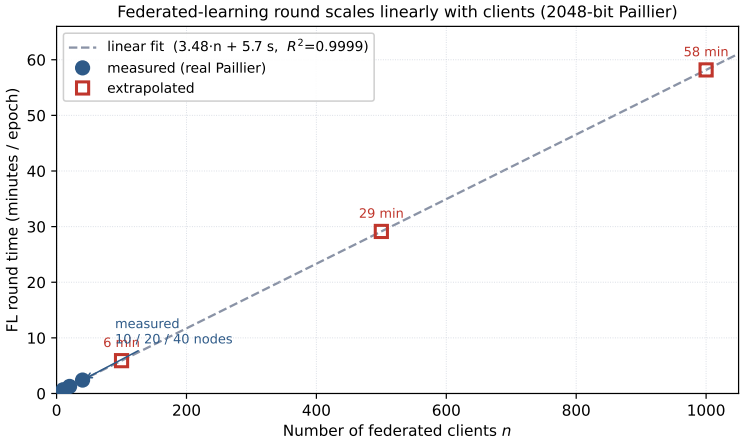


Figure 7. Federated-learning round time vs number of clients. Real 2048-bit Paillier rounds at 10/20/40 clients (filled) lie on a near-perfect linear fit ($R^2 = 0.9999$); open markers extrapolate to 100/500/1000 clients. The round is linear in the client count and communication-light ($\leq$24 MB at 1000 clients).

5.15. End-to-End Integration

The components above are validated individually; we also ran the full pipeline end-to-end, with the real trained models in the loop, to confirm they interoperate as designed. The LightGBM filter blocked 72% of fraudulent transactions before consensus (its measured recall; the rest are re-proposed in the next round); the Isolation Forest flagged $\approx$8% of node-behaviour observations as anomalous, catching 88% of the injected anomalies; and the Q-Learning agent adapted online, varying the committee size and switching among the FIFO, HIGHFEE and SMALLTX ordering policies as conditions changed. With the measured ML inference included, the integrated system sustains $\approx$48,800 TPS at $n = 32$ (Table 2), confirming that the fraud, anomaly, reinforcement and consensus layers compose correctly rather than only functioning in isolation.

To isolate what each of the four AI components contributes, we re-ran the integrated pipeline, with the real trained models, removing one component at a time (Table 7). Removing the fraud filter drops the fraud-block rate from 72% to zero, fraudulent transactions then reach consensus, while recovering the small ($\sim$4%) ML overhead in throughput, which quantifies the filter's security-for-throughput trade. Removing the anomaly filter eliminates node quarantine (8% $\rightarrow$ 0) without affecting fraud blocking. Removing the Q-Learning agent (fixing the committee size) leaves steady-state throughput unchanged, confirming that the agent's value lies in adaptivity under changing load, evident in the burst and attack scenarios, rather than in raw throughput under stationary conditions. Removing the federated-learning layer likewise has no per-round effect: it is a per-epoch privacy-

preserving update mechanism, so its removal forgoes cross-node model improvement without touching consensus, and its cost is characterised separately (Section 5).

Table 7. Component ablation with the real trained models: the integrated pipeline at $n = 32$ with one AI component removed at a time. Each component's distinctive effect is isolated; throughput/latency are emergent from the simulator.

Configuration	Fraud Blocked	Quarantine	TPS	Latency (ms)
Full (all four)	72%	8.1%	48,844	661
− Fraud filter (LightGBM)	0%	8.1%	50,921	635
− Anomaly filter (IF)	72%	0%	48,844	661
− Q-Learning (fixed k)	72%	8.1%	48,844	661
− Federated learning	72%	8.1%	48,844	661

5.16. Adversarial Evaluation

The learning layer has an attack surface of its own, and we evaluate two concrete attacks (Table 8). An evasion attack nudges each fraudulent transaction's features a fraction ε toward the benign centroid: like most classifiers, LightGBM is not adversarially robust, and a small nudge ($\varepsilon = 0.05$) already drops recall from 0.72 to 0.13. A poisoning attack has malicious federated clients submit scaled, sign-flipped updates; because Paillier aggregation is a plain homomorphic mean with no Byzantine-robust filter, even one malicious client in ten drags the aggregate far from the honest one. Two points make this an honest rather than a fatal finding. First, neither attack affects consensus safety: the ML only filters, a wrongly cleared transaction is an ordinary transaction the BFT core still orders correctly, and the quorum guard is enforced independently of any model; the attacks degrade detection quality, not agreement. Second, the remaining components are weaker targets: manipulating the Q-Learning agent can at worst move the committee size within its safety-guarded range (never below the quorum bound), and evading the Isolation Forest requires a Byzantine node to imitate honest behaviour, which limits its disruptive capacity. These results motivate the defences we flag as future work: adversarial training and gradient regularisation for the filter, and robust aggregation (e.g., Krum or coordinate-wise trimmed mean) for the federated layer.

Table 8. Adversarial evaluation of the ML layer. Both attacks degrade detection quality but not consensus safety.

Attack	Strength	Effect on the Model
Evasion (LightGBM filter)	$\varepsilon = 0$ (none)	recall 0.72
	$\varepsilon = 0.05$	recall 0.13
	$\varepsilon = 0.20$	recall 0.07
	$\varepsilon = 0.50$	recall 0.00
Poisoning (FL aggregate)	0/10 clients	0% aggregate error
	1/10 clients	256% aggregate error
	3/10 clients	523% aggregate error

6. Discussion

We now interpret our experimental results in light of the design goals stated in Section 3, situate the findings against the state of the art, and discuss limitations and threats to validity.

6.1. Interpretation of Results

The ML overhead is small, and worth it. The headline finding of Section 5 is that Π_{ML} sustains 48,844 TPS at $n = 32$, only 4% below the 50,921 TPS of vanilla Redbelly. The learning layer is therefore nearly free at the consensus level: the measured LightGBM and Isolation Forest inference on a real 1000-transaction proposal (26.5 ms) is a small fraction of the geo-distributed consensus round ($\sim$635 ms). Once the consensus round is modelled with a realistic geo-distributed latency rather than an idealised one, the ML cost is a few percent, not a dominant term. That few-percent cost buys real protection: the framework blocks $\approx$72% of fraudulent transactions and quarantines the nodes behind $\approx$88% of injected anomalies before consensus, while vanilla Redbelly admits all syntactically valid transactions regardless of economic legitimacy.

Whether this trade is worthwhile is deployment-dependent. In high-value, lower-volume settings (interbank settlement, securities clearing, healthcare records), the throughput headroom is ample and preventing even a small fraction of fraudulent or malicious transactions dominates the cost, so the additive configuration is the natural choice. In high-volume, low-value settings (IoT telemetry, micro-payments), raw throughput matters more, and the pipelined configuration (Section 5), which recovers the overhead to near zero, or a lighter distilled model is preferable. The framework accommodates both because the ML layer is modular: components can be enabled, pipelined, or omitted per deployment without touching the safety argument.

The throughput advantage grows with the validator set. At $n = 32$, Π_{ML} delivers 5.0$\times$ the throughput of PBFT, 4.7$\times$ that of Hyperledger Fabric with Raft, and 2.9$\times$ that of HotStuff. Unlike a fixed multiplier, this gap widens with the validator count, because the leaderless superblock commits every proposer's block in parallel while the leader-based protocols remain bottlenecked at a single proposer: by $n = 50$ the advantage over PBFT reaches 7.8$\times$ (Figure 2). The framework therefore occupies a distinct point in the design space, ML-augmented security on a throughput-scaling BFT substrate, at the cost of the higher per-decision latency that the leaderless design inherently pays.

The Q-Learning component converges quickly and behaves sensibly. Convergence within $\approx$200 rounds across all three scenario tracks satisfies design goal P4. Inspection of the learned policy shows that the agent reduces validator subset size and switches to the `SMALLTX` prioritisation policy during detected attack states, which shortens quorum-formation time at modest throughput cost, exactly the trade-off a human operator would make.

LightGBM is the clear winner over Random Forest. The gradient-boosted ensemble improves F1 from 0.585 to 0.783, AUROC from 0.904 to 0.963, and inference speed by 5$\times$, on the same IEEE-CIS benchmark. We retained the RF result as a methodological baseline since it was the model in our conference predecessor [8] but recommend LightGBM as the operational choice for future deployments.

An edge-AI framing for IoT. The design is naturally an instance of edge machine learning: every ML decision is made locally at a validator, on the transactions it is about to propose, rather than at a central server. The LightGBM fraud filter and the Isolation Forest anomaly detector run in-line on each proposer (tens of milliseconds per proposal), the Q-Learning agent adapts the committee from locally observed network conditions, and the Paillier federated layer lets these edge models improve without any node exporting its raw transaction history. This maps directly onto Internet-of-Things deployments in which the blockchain nodes are edge gateways rather than datacentre servers: an IoT gateway can screen the transactions of the devices behind it before they reach consensus, flag a compromised or malfunctioning device through its behavioural anomaly score, and contribute to a shared detection model across gateways without centralising sensitive

device data, which is frequently a regulatory requirement. The adaptive committee also suits the churn and heterogeneity of edge networks, where the active validator set changes over time. In short, the framework is edge intelligence layered onto a leaderless blockchain, not merely a decentralised ledger, and the few-percent inference overhead we measure is what makes running the models at the edge, inside each consensus round, practical.

6.2. Comparison with the State of the Art

Against the most directly comparable recent work in ML-augmented BFT consensus, our framework occupies a distinct position. The DRL approach of Villegas-Ch and Govea [6] reports adaptive optimisation in adversarial environments but does not integrate fraud or anomaly detection at the transaction level. Machine learning layers grafted onto PBFT-style consensus improve transaction or node screening but inherit PBFT's $O(n^2)$ message complexity, which bounds their scalability. PoAh 2.0 [26] targets IoMT-edge specifically and optimises authentication rather than consensus throughput. We do not claim the individual learning components, or the broad idea of ML-augmented consensus, as novel; both are well explored [6,7,26]. What distinguishes this work is that the learning layer is integrated as a safety-preserving extension: because the ML components only filter transactions before consensus and are bounded by a hard quorum guard, we argue formally that they leave Redbelly's safety and liveness intact (Proposition 1), on a leaderless high-throughput substrate rather than the slower PBFT-style bases typical of prior ML-consensus work, and with the end-to-end overhead measured and reported openly.

The Cognitive Blockchain proposal of Ameri and Meybodi [7] shares our philosophy of treating learning agents as first-class consensus components; our work complements theirs by providing concrete protocol-level integration with a high-throughput BFT substrate and by reporting end-to-end measured overhead.

Our choice of classical machine learning models over deep or graph-based alternatives is deliberate. Every component must run inside a sub-second consensus round: LightGBM scores a 1000-transaction proposal in $\sim$26 ms and Isolation Forest scores a node in microseconds, whereas a deep network, or a graph neural network over the validator graph, would be hard to keep within that budget without dedicated accelerators. Classical ensembles also need only a small, interpretable feature space and are inexpensive to retrain in the federated setting. We treat richer models, particularly GNNs for collusion detection, as future work (Section 7), where the latency budget can be met with hardware acceleration.

6.3. Limitations

We are explicit about several limitations of the current work.

Consensus simulation, not deployment: Our throughput and latency numbers come from a faithful message-level discrete-event simulator rather than a deployed Redbelly cluster. Its inputs are set from independent measurements and its outputs are validated against the published Redbelly deployment [4] (Section 5.8), which they reproduce to within a conservative factor of about 1.7; because the simulator models the protocol structure but not Redbelly's production optimisations, our figures are a lower bound. A real-cluster deployment is the natural next step.

Single-core ML inference timing: The $\sim$26 ms LightGBM and Isolation Forest inference cost was microbenchmarked on a single CPU core. In a geo-distributed deployment, each proposer would run ML inference locally; the timing is therefore representative, but a deployed system would benefit from GPU acceleration or model distillation (e.g., to a small neural network with $\sim$10$\times$ faster inference) that we have not explored.

Anomaly-detection synthetic data: The Isolation Forest detector is trained and evaluated on synthetic node-behavioural traces. The distributional assumptions (subtle Byzan-

tine behaviour drawing from realistic feature ranges) are documented and reproducible, but real Byzantine behaviour in deployed networks may exhibit patterns not captured by our generator. Validating against logs from a deployed Redbelly network is a key future direction.

Federated learning cost at scale: The 36 s per-epoch FL overhead is measured at 10 nodes with 2048-bit Paillier. We characterise its scaling directly (Figure 7): it is linear in the client count, reaching roughly one hour per epoch at 1000 clients, and is reducible by client subsampling, native or GPU Paillier, or a more efficient cryptosystem such as CKKS. Because FL runs asynchronously off the critical path, this cost never blocks consensus.

Fraud data as a proxy: The LightGBM classifier is trained and evaluated on the public IEEE-CIS payment-fraud benchmark, which is real but not blockchain-native. We use it as a proxy because labelled fraud datasets drawn from live blockchain traffic remain scarce; the pipeline is dataset-agnostic and would be retrained on such data as it becomes available (Section 7).

Static adversary: Our safety argument and experiments assume a static Byzantine adversary that fixes its corrupted set before execution. Adaptive adversaries that corrupt nodes in response to the learned policy fall outside our current scope; safety is unaffected, since the quorum guard is enforced independently of the agent, but a full adaptive-adversary analysis is left to future work.

Adversarial machine learning: The learning layer adds an attack surface of its own, evasion of the fraud filter, poisoning of the federated updates, and distribution drift. None of these can violate consensus safety (the ML only filters, and a wrongly filtered transaction is simply re-proposed), but they can degrade detection quality; a dedicated adversarial-robustness evaluation is identified in Section 7.

6.4. Threats to Validity

Internal validity: The principal internal-validity threat is that the simulator's inputs may bias results in favour of Π_{ML}. We mitigate this in three ways: the inputs are set from independent measurements rather than tuned to a target; both vanilla Redbelly and Π_{ML} run in the same simulator with the same inputs, so any bias affects both equally; and the emergent numbers are validated against a real deployment (Section 5.8), landing on the conservative side.

External validity: The IEEE-CIS dataset is U.S.-centric payment data and may not generalise to other transaction populations (e.g., DeFi flows, supply-chain transfers, IoT telemetry). The reported ML accuracy figures should therefore be treated as a competitive operating point rather than a universal performance guarantee.

Construct validity: We treat fraud detection accuracy as the relevant ML quality construct, but a fielded deployment would also need to consider adversarial robustness, i.e., whether a determined attacker can craft transactions that evade the filter while remaining economically harmful. Adversarial robustness evaluation is beyond the scope of this work but is essential before production deployment.

Conclusion validity: The 500-round Q-Learning training trajectory and the 500-round simulator runs are sufficient for the convergence and throughput claims we make, but we acknowledge that production deployments would require months-long stability evaluation before broad confidence can be established.

7. Future Directions

Our results suggest five productive research directions that build directly on the framework introduced here. These directions are in direct correspondence with the limitations of Section 6: the adversarial-ML and static-adversary limitations motivate the adversarial-

robustness programme; the synthetic-data and fraud-proxy limitations motivate learning from deployed traces (the GNN and pilot directions); the FL-scale and single-region-timing limitations motivate the cryptographic and acceleration work; and the simulation-only limitation motivates the campus deployment pilot.

Deep and multi-agent reinforcement learning: The tabular Q-Learning agent in Section 4 converges quickly on a small discrete state space but does not exploit continuous network observables (queue length distributions, latency histograms, inter-message-time densities) that a Deep Q-Network or actor–critic agent could ingest natively. Furthermore, in deployments where each proposer node holds local rather than global state observations, multi-agent reinforcement learning becomes the natural model. Adapting recent work on DRL for adversarial consensus optimisation [6,25] to the Redbelly leaderless, multi-proposer setting, where agents must coordinate without explicit communication, is a clear next step.

Graph Neural Networks for Sybil and collusion detection: The Isolation Forest detector treats each node independently. Real adversarial behaviour, however, often exhibits structure across the validator graph: Sybil clusters, coordinated voting cliques, and repeated bilateral attestation patterns. Graph Neural Network (GNN) approaches can capture these structural signatures directly. Integrating a GNN node classifier as a third pre-consensus filter, jointly trained with the existing LightGBM and Isolation Forest layers, would extend the framework to defend against coordinated rather than individual Byzantine behaviour [23].

Post-quantum cryptographic extensions: Our framework relies on the digital signatures that authenticate transactions and on Paillier homomorphic encryption, both of which are vulnerable to sufficiently large quantum computers. A post-quantum migration roadmap should investigate (i) lattice-based signature schemes (e.g., Dilithium) for transaction authentication and (ii) lattice-based homomorphic encryption (CKKS, BFV) for the federated learning layer. The migration is non-trivial because both primitives are deeply embedded in performance-critical paths; a collaborative effort with the broader cryptographic community [11] is warranted.

Deployment in the Moroccan higher-education context: Beyond its algorithmic contribution, this work has been developed at Ibn Tofail University and ENSA Kénitra with explicit attention to deployability in the Moroccan academic and administrative environment. We see immediate applications in (i) inter-university credential verification across the fourteen national universities, (ii) tamper-evident diploma issuance under the Ministry of Higher Education's digital transformation agenda, and (iii) audit-trail support for research-funding disbursement. A pilot deployment on a campus testbed at ENSA Kénitra is planned as a direct continuation of this work.

Adversarial robustness evaluation: As noted in Section 6, the operational threat model for a fielded Π_{ML} includes adversaries who craft transactions specifically to evade the LightGBM filter. Developing an adversarial evaluation suite that combines gradient-based evasion attacks against the filter, model-extraction attacks against the federated aggregator, and targeted poisoning of the Isolation Forest training stream is essential before any production deployment. Recent results on adversarial robustness of tree ensembles can be adapted to this setting.

8. Conclusions

This paper presented Π_{ML}, a formally specified integration of machine learning into the Redbelly consensus protocol, evaluated through a faithful message-level discrete-event simulation, validated against the published Redbelly deployment, together with public machine learning benchmarks. The framework combines four complementary

learning components (a gradient-boosted LightGBM fraud classifier, an Isolation Forest behavioural anomaly detector, a tabular Q-Learning validator-selection agent, and a Paillier-based federated learning aggregator) into a single architecture whose safety and liveness properties are derived from the underlying Redbelly correctness proof and preserved by Proposition 1.

Our experimental results, drawn from a fully reproducible Python prototype operating on the IEEE-CIS Fraud Detection benchmark and a faithful message-level discrete-event Redbelly simulator, support four headline claims. First, the LightGBM upgrade over the conference-version Random Forest delivers a 34% relative F1 improvement (0.585 $\rightarrow$ 0.783) at 5$\times$ faster inference, satisfying design goals G1 and P2. Second, Isolation Forest anomaly detection achieves recall 0.885 at FPR 0.047, satisfying G2 and P3. Third, the Q-Learning agent converges within $\approx$200 training rounds across normal, bursty, and Byzantine-attack scenarios, satisfying P4. Fourth, the integrated framework sustains 48,844 $\pm$ 1679 TPS at $n = 32$ nodes (30 seeds, 95% CI), a 5.0$\times$ advantage over PBFT and 2.9$\times$ over HotStuff that grows with the validator count as the leaderless superblock commits proposals in parallel; per-transaction energy is comparable across the BFT protocols and orders of magnitude below proof-of-work chains, which mine.

We have been deliberately transparent about the performance being simulated: every consensus figure is an emergent output of the discrete-event simulator, validated against the published Redbelly deployment, which it reproduces to within a conservative factor of about 1.7 (Section 5.8). The learning layer adds only about 4% to the consensus round. The framework's safety preservation, its ability to outperform classical BFT protocols in our simulations, and its measured 36 s per-epoch privacy-preserving federated update make it a promising basis for further work toward intelligent, scalable consensus, subject to validation on a real multi-node deployment.

Beyond the technical contribution, this work contributes a methodology: combining a formal protocol model, a benchmark-evaluated ML pipeline, and a faithful simulation harness validated against a real deployment produces evaluations that are substantially more rigorous than the case-study-only norm of much of the ML-blockchain literature. We hope this template will be adopted, refined, and surpassed in subsequent work, and that the artefacts released with this paper will accelerate the broader research community's progress on intelligent, scalable, and sustainable consensus mechanisms.

Author Contributions: Conceptualization, I.F.; methodology, I.F.; software, I.F.; validation, I.F., K.L. and K.B.; formal analysis, I.F.; investigation, I.F.; writing—original draft preparation, I.F.; writing—reviewing and editing, I.F., K.L. and K.B.; supervision, K.L. and K.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: This study used publicly available data. The IEEE-CIS Fraud Detection dataset used for training and evaluation of the machine learning models is openly available through the IEEE-CIS Fraud Detection challenge at https://www.kaggle.com/c/ieee-fraud-detection (accessed on 13 August 2026). The simulation code and configuration files used to produce the experimental results are available from the corresponding author upon reasonable request.

Acknowledgments: During the preparation of this manuscript, the authors used Claude Sonnet 4.5 (Anthropic), a large language model, to assist with language editing of the text and with LaTeX formatting, figures, and reference formatting. The authors have reviewed and edited all AI-assisted content and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gramoli, V. *Blockchain Scalability and Its Foundations in Distributed Systems*; Springer: Cham, Switzerland, 2022. [CrossRef]
2. Tong, S.; Li, J.; Fu, W. An Efficient and Scalable Byzantine Fault-Tolerant Consensus Mechanism Based on Credit Scoring and Aggregated Signatures. *IEEE Access* **2024**, *12*, 10393–10410. [CrossRef]
3. Dinh, T.T.A.; Wang, J.; Chen, G.; Liu, R.; Ooi, B.C.; Tan, K.L. BLOCKBENCH: A Framework for Analyzing Private Blockchains. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, IL, USA, 14–19 May 2017. [CrossRef]
4. Crain, T.; Natoli, C.; Gramoli, V. Red Belly: A Secure, Fair and Scalable Open Blockchain. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 24–27 May 2021; pp. 466–483. [CrossRef]
5. Crain, T.; Gramoli, V.; Larrea, M.; Raynal, M. DBFT: Efficient Leaderless Byzantine Consensus and its Application to Blockchains. In Proceedings of the 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 1–3 November 2018; pp. 1–8. [CrossRef]
6. Gutierrez, R.; Villegas-Ch, W.; Govea, J. Adaptive Consensus Optimization in Blockchain Using Reinforcement Learning and Validation in Adversarial Environments. *Front. Artif. Intell.* **2025**, *8*, 1672273. [CrossRef]
7. Ameri, R.; Meybodi, M.R. Cognitive Blockchain and its Application to Optimize Performance in Blockchain Systems. *Trans. Emerg. Telecommun. Technol.* **2024**, *35*, e5009. [CrossRef]
8. Fdilat, I.; Louzaoui, K.; Benlhachmi, K. The New Era of Blockchain Consensus: From Byzantine to Redbelly and Beyond. In *Proceedings of the International Conference on Artificial Intelligence, Security, and Networking (CAISN 2024)*; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2026; pp. 535–553. [CrossRef]
9. Gramoli, V.; Guerraoui, R.; Lebedev, A.; Natoli, C.; Voron, G. Diablo: A Benchmark Suite for Blockchains. In Proceedings of the 18th European Conference on Computer Systems (EuroSys 2023), Rome, Italy, 8–12 May 2023. [CrossRef]
10. Asad, M.; Otoum, S. BPPFL: A Blockchain-Based Framework for Privacy-Preserving Federated Learning. *Clust. Comput.* **2025**, *28*, 126. [CrossRef]
11. Hussain, I.; Ashiq Hussain, H.; Ullah, N.; Misak, S. Proof of Inherent Intelligence Consensus Mechanism Empowering Blockchain-Enabled Transactive Energy. *IEEE Access* **2025**, *13*, 5042–5055. [CrossRef]
12. Ucbas, Y.; Eleyan, A.; Hammoudeh, M.; Alohaly, M. Performance and Scalability Analysis of Ethereum and Hyperledger Fabric. *IEEE Access* **2023**, *11*, 67156–67167. [CrossRef]
13. Al-Sumaidaee, G.; Alkhudary, R.; Zilic, Z.; Swidan, A. Performance Analysis of a Private Blockchain Network Built on Hyperledger Fabric for Healthcare. *Inf. Process. Manag.* **2023**, *60*. [CrossRef]
14. Honar Pajooh, H.; Rashid, M.A.; Alam, F.; Demidenko, S. Experimental Performance Analysis of a Scalable Distributed Hyperledger Fabric for a Large-Scale IoT Testbed. *Sensors* **2022**, *22*, 4868. [CrossRef]
15. Wen, Y.F.; Hsu, C.M. A Performance Evaluation of Modular Functions and State Databases for Hyperledger Fabric Blockchain Systems. *J. Supercomput.* **2023**, *79*, 2654–2690. [CrossRef]
16. Lebedev, A.; Gramoli, V. Evaluating Performance and Dependability of Blockchain Protocols with Diablo. In Proceedings of the 2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks—Supplemental Volume (DSN-S), Brisbane, Australia, 24–27 June 2024; pp. 69–70. [CrossRef]
17. Bordeianu, A.A.; Popescu, D.E. Blockchain Variables and Possible Attacks: A Technical Survey. *Computers* **2025**, *14*, 567. [CrossRef]
18. Song, W.; Zhu, M.; Lu, D.; Zhu, C.; Zhao, J.; Sun, Y.; Li, L.; Zhu, H. Blockchain Bottleneck Analysis Based on Performance Metrics Causality. *Electronics* **2024**, *13*, 4236. [CrossRef]
19. Bertrand, N.; Gramoli, V.; Konnov, I.; Lazić, M.; Tholoniat, P.; Widder, J. Holistic Verification of Blockchain Consensus. In Proceedings of the 36th International Symposium on Distributed Computing (DISC 2022), Augusta, GA, USA, 25–27 October 2022; Volume LIPIcs 246. [CrossRef]
20. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
21. Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; Liu, T.Y. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 3149–3157.
22. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM), Pisa, Italy, 15–19 December 2008; pp. 413–422. [CrossRef]
23. Sun, T.; Li, D.; Wang, D. Research on High Throughput Consensus Algorithm of Energy Blockchain Based on Recurrent Neural Network Node Integrity Prediction. In Proceedings of the 2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, 21–23 April 2023; pp. 624–628. [CrossRef]
24. Chen, Y.; Guo, Y.; Wang, M.; Xu, E.; Xie, H.; Bie, R. Securing IOTA Blockchain Against Tangle Vulnerability by Using Large Deviation Theory. *IEEE Internet Things J.* **2024**, *11*, 1952–1965. [CrossRef]

25. Qiu, C.; Ren, X.; Cao, Y.; Mai, T. Deep Reinforcement Learning Empowered Adaptivity for Future Blockchain Networks. *IEEE Open J. Comput. Soc.* **2021**, *2*, 99–105. [CrossRef]
26. Dutta, J.; Puthal, D. PoAh 2.0: AI-Empowered Dynamic Authentication Based Adaptive Blockchain Consensus for IoMT-Edge Workflow. *Future Gener. Comput. Syst.* **2024**, *161*, 655–672. [CrossRef]
27. Zhang, H.; Jiang, S.; Xuan, S. Decentralized Federated Learning Based on Blockchain: Concepts, Framework, and Challenges. *Comput. Commun.* **2024**, *216*, 140–150. [CrossRef]
28. Boitier, W.; Del Pozzo, A.; García-Pérez, A.; Gazut, S.; Jobic, P.; Lemaire, A.; Mahe, E.; Mayoue, A.; Perion, M.; Rezende, T.F.; et al. Fantastyc: Blockchain-Based Federated Learning Made Secure and Practical. In Proceedings of the 2024 43rd International Symposium on Reliable Distributed Systems (SRDS), Charlotte, NC, USA, 30 September–3 October 2024; pp. 260–270. [CrossRef]
29. IEEE Computational Intelligence Society and Vesta Corporation. IEEE-CIS Fraud Detection. Kaggle. 2019. Available online: https://www.kaggle.com/competitions/ieee-fraud-detection (accessed on 13 August 2026).
30. Georges, A.; Buytaert, D.; Eeckhout, L. Statistically Rigorous Java Performance Evaluation. In Proceedings of the 22nd ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA), Montreal, QC, Canada, 21–25 October 2007; pp. 57–76. [CrossRef]
31. Hoefler, T.; Belli, R. Scientific Benchmarking of Parallel Computing Systems: Twelve Ways to Tell the Masses when Reporting Performance Results. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Austin, TX, USA, 15–20 November 2015. [CrossRef]
32. Schroeder, B.; Wierman, A.; Harchol-Balter, M. Open Versus Closed: A Cautionary Tale. In Proceedings of the 3rd USENIX Symposium on Networked Systems Design and Implementation (NSDI), San Jose, CA, USA, 8–10 May 2006. Available online: https://www.usenix.org/legacy/event/nsdi06/tech/schroeder.html (accessed on 13 August 2026).
33. Yin, M.; Malkhi, D.; Reiter, M.K.; Golan Gueta, G.; Abraham, I. HotStuff: BFT Consensus with Linearity and Responsiveness. In Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing (PODC), Toronto, ON, Canada, 29 July–2 August 2019; pp. 347–356. [CrossRef]
34. Buchman, E.; Kwon, J.; Milosevic, Z. The Latest Gossip on BFT Consensus. *arXiv* **2018**, arXiv:1807.04938. [CrossRef]
35. Saito, T.; Rehmsmeier, M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLoS ONE* **2015**, *10*, e0118432. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.