TRM-Raft: A Byzantine-Resistant Raft Consensus via Integrated Trust and Reputation Model

Jie Zhang*
Xubo Fan*
Tianjin University
Tianjin, China
{jackzhang,xubofan}@tju.edu.cn

Xiaohong Li✉
Tianjin University
Tianjin, China
xiaohongli@tju.edu.cn

Zhiyong Feng
Tianjin University
Tianjin, China
zyfeng@tju.edu.cn

Abstract

Internetware envisions autonomous software entities dynamically collaborating over the open, evolving Internet. A key enabler of such systems is the Raft consensus protocol, which is widely adopted for its simplicity and high performance in distributed coordination, e.g., in service registries, cloud orchestration, and permissioned blockchains. However, Raft's design assumes a purely crash-fault model, making it inherently vulnerable to Byzantine behaviors such as election forgery and log tampering when deployed in the hostile, dynamic Internet environment. Existing Byzantine fault-tolerant protocols incur prohibitive communication overhead and invasive architectural changes, while point-wise hardening attempts for Raft fail to provide unified, adaptive defense.

In this paper, we propose **TRM-Raft**, a Byzantine-resistant enhancement of Raft that *non-intrusively* integrates a Blockchain-based Trust and Reputation Model (B-TRM) into the consensus core. The core idea is to quantify multi-dimensional node behaviors, apply adaptive penalties that distinguish accidental faults from persistent malice, and embed reputation signals directly into leader election and log replication. Specifically, a reputation-aware election mechanism detects and harshly penalizes term/index forgery, keeping low-reputation nodes out of the leader role. A Schnorr-signature-based leader restriction mechanism enables followers to instantly verify log integrity; any tampering triggers reputation decay and leader replacement. Implemented and evaluated in a realistic Internetware setting using Hyperledger Fabric, TRM-Raft maintains a malicious leader ratio below 5% even when 40% of nodes are Byzantine, while introducing less than 10% throughput degradation and under 5% latency increase compared to vanilla Raft. TRM-Raft thus provides a lightweight and practical path toward trustworthiness for the broad class of Internetware systems that rely on Raft.

CCS Concepts

• **Security and privacy → Distributed systems security**; *Domain-specific security and privacy architectures*; • **Computing methodologies → Distributed computing methodologies**; • **Computer systems organization → Peer-to-peer architectures**; *Reliability*.

Keywords

Raft consensus, Byzantine fault tolerance, Trust and reputation model, Blockchain security, Leader election, Schnorr signature

ACM Reference Format:
Jie Zhang, Xubo Fan, Xiaohong Li, and Zhiyong Feng. 2018. TRM-Raft: A Byzantine-Resistant Raft Consensus via Integrated Trust and Reputation Model. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

The Internetware paradigm envisions autonomous software entities dynamically collaborating over the open, ever-changing Internet [5, 7]. Modern Internet-scale distributed systems, from cloud orchestration platforms and service registries to permissioned enterprise blockchains [16, 33, 34], critically rely on consensus protocols to maintain a consistent and reliable system state in this dynamic environment[28, 29]. Among these protocols, the Raft consensus algorithm [21] has gained widespread adoption due to its simplicity, understandability, and high performance. Prominent software infrastructures such as etcd (the backbone of Kubernetes)[1], Quorum[2], and Hyperledger Fabric[3] [1] all leverage Raft to coordinate replicas.

However, Raft's design assumes a purely Crash Fault Tolerant (CFT) model, where nodes may only stop working but never deviate from the protocol [21]. In the open Internet environment that Internetware systems inhabit, this non-Byzantine assumption is fragile [2, 21]. Nodes can be compromised by external attackers, exhibit selfish behaviors, or become faulty in adversarial ways, thereby threatening the very consistency and availability that consensus is meant to provide [11, 24, 26, 27]. Two particularly severe classes of Byzantine behavior against Raft are:

Forgery Attacks on Leader Election In Raft, leader election is predicated on candidates possessing the "latest" log, determined by term number (*Term*) and log index (*Index*). A malicious node can exploit this by *forging* artificially high *Term* and *Index* values. By minimizing its election timeout (*ET*), the attacker can transition

*Jie Zhang and Xubo Fan contributed equally to this research.
Xiaohongli is the corresponding author.

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

[1]Raft is the core of distributed container consensus algorithms.
[2]Raft is one of two consensus protocols
[3]Supports the Raft consensus starting from version 1.4.1

to candidate earlier than honest nodes, broadcast a `RequestVote` RPC with forged metadata, and win the election before legitimate candidates can react. Once elected, the malicious leader can suppress further elections via heartbeats, even if later replaced, it can repeat the attack by incrementing its term again. This vulnerability, highlighted by [27] and [11], allows a single malicious node to persistently disrupt leader legitimacy.

Tampering Attacks on Log Replication After election, the leader is solely responsible for replicating client requests (logs) to followers. Raft ensures log consistency through term and index matching but does *not* verify the integrity of the log content (*entries*). A malicious leader can therefore *tamper* with *entries* while preserving correct $Term$ and $Index$ structure, bypassing Raft's native consistency checks. As demonstrated in [22–24, 26], such tampering can corrupt the blockchain state, compromise data integrity, and undermine trust in the system.

While Byzantine Fault Tolerant (BFT) protocols like PBFT [4] can inherently resist such adversaries, their high communication complexity ($O(N^2)$) and demanding quorum requirements ($n = 3f + 1$) introduce substantial performance penalties and architectural intrusiveness [6]. For many Internetware systems, completely replacing an existing Raft stack with a full BFT protocol is both costly and impractical [5, 14, 18, 32]. Existing defenses that retrofit Raft with static thresholds [27] or hash-chain verification [26] focus on isolated attack vectors, lack a unified trust management framework, and often fail to adapt to the dynamic and evolving threat landscape of the Internet.

To bridge this gap, we propose **TRM-Raft**, a Byzantine-Resistant Raft consensus achieved by integrating a blockchain-based Trust and Reputation Model (TRM) directly into the consensus core. Our key insight is that malicious behaviors in Internetware systems, whether forgery, tampering, or other camouflaged attacks (e.g., On-Off attacks[3], manifest as observable anomalies that can be quantified and penalized through a dynamic reputation system, *without redesigning the fundamental Raft protocol.* TRM-Raft transforms the static trust assumption of Raft into an adaptive, behavior-aware trust management while preserving its proven efficiency and simplicity. By co-designing a behavioral trust model with cryptographic verification, TRM-Raft is, to the best of our knowledge, the first Raft variant that unifies dynamic reputation-based election gating with Schnorr-signature-verified log integrity. This combination forces an adversary to simultaneously defeat anomaly detection and signature forgery, a synergy absent from prior point-wise defenses.

Our key contributions are:

(1) **Unified Trust and Integrity Framework:** The first integration of a multi-dimensional reputation model (B-TRM) with Schnorr signatures in Raft, jointly addressing election forgery and log tampering through adaptive behavioral assessment and cryptographic verification.
(2) **Reputation-Aware Election Mechanism:** A dynamic monitoring scheme that detects anomalous term/index surges during elections, penalizes forgery attempts by halving the attacker's reputation, and excludes low-reputation nodes from voting and candidacy, thus guaranteeing that leadership remains in trustworthy hands.
(3) **Leader Restriction with Schnorr Signatures:** An efficient cryptographic verification layer embedded in the log replication phase. Followers verify log integrity via Schnorr signatures; any tampering leads to immediate reputation penalties and, if sustained, triggers leader replacement, ensuring end-to-end state trustworthiness.
(4) **Practical Validation:** We implement TRM-Raft on Hyperledger Fabric, a representative Internet-scale permissioned platform, and demonstrate that it maintains malicious leader prevalence below 5% even when 40% of nodes are Byzantine, while incurring modest performance overhead (less than 10% in throughput and 5% in latency) compared to vanilla Raft.

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 formalizes the threat model for Internetware-Raft systems. Section 4 presents the B-TRM reputation model. Section 5 describes TRM-Raft's design and provides a security analysis. Section 6 gives experimental evaluation, Section 7 discusses limitations, and Section 8 concludes.

2 Related Works

2.1 Raft, Byzantine behavior, and modern BFT baselines

Raft [21] is a leader-based crash-fault tolerant consensus protocol, designed with clarity, implementability, and predictable performance in mind, which has established it as the consensus backbone for numerous Internetware infrastructures [28]. In benign deployments, its majority quorum and one-round commit path ($\lfloor N/2 \rfloor + 1$ nodes for committing log entries[2]) are sufficient to support high throughput with relatively low coordination overhead. However, its vulnerability to Byzantine participants directly threatens the reliability of many platforms. Fundamentally, it assumes that participants adhere strictly to the protocol, barring crash-stop failures. Once a node can lie about its election state [11, 27], equivocate on log progress [23, 26], or tamper with replicated content [22, 24], Raft's native checks are no longer sufficient.

Classical Byzantine fault tolerant protocols such as PBFT [4] established the standard $3f+1$ resilience threshold (at least $\lfloor 2N/3 \rfloor + 1$ nodes to tolerate up to f malicious nodes out of $N = 3f + 1$[2]), while more recent protocols such as HotStuff [30] and modern blockchain implementations such as CometBFT[4] reduce practical overhead but still preserve quorum-based Byzantine safety [6]. These systems are stronger than Raft under an arbitrary Byzantine adversary, but their message patterns, view-change logic, and recovery machinery are more expensive than CFT protocols [29]. The design space therefore remains split between low-cost CFT and high-cost BFT [11, 24]. TRM-Raft is positioned between the two: it aims to suppress a restricted but practically relevant set of observable Byzantine deviations without claiming equivalence to strict BFT.

2.2 TRMs in distributed systems

Trust and Reputation Models (TRMs) have been used to score participants based on past behavior, local observations, and indirect evidence [13, 19, 26, 31]. In distributed systems, such models are

[4] https://github.com/cometbft/cometbft.git

most useful when the system can observe repeated actions, correlate them across peers, and enforce penalties that are visible to the rest of the network [19, 31]. This makes them attractive for permissioned consensus settings, where node identities are known and the protocol already maintains a shared state.

At the same time, reputation mechanisms are not a substitute for cryptographic safety [31]. They are reactive, can be gamed, and may fail against a strategic adversary that behaves honestly long enough to accumulate trust before misbehaving [11, 15]. Prior work on accountability and alternative fault models emphasizes this point and suggests that trust-based mechanisms should be framed as partial defenses rather than as full Byzantine guarantees [8]. Our design follows that principle.

2.3 Cryptographic integrity for replicated logs

Digital signatures are a standard method for binding a message to its origin and contents. Schnorr signatures [12] are attractive because they are compact, efficient to verify, and well suited to modern blockchain-style implementations [17]. In the context of hardening Raft against tampering attacks, Schnorr signatures provide a cryptographic mechanism for log integrity verification[25]. By producing a Schnorr signature in the propose phase, followers can verify the signature against the known public key to detect the manipulation instantly. This directly counters the tampering attack described in [24–26], where a malicious leader alters log content while preserving correct $Term$ and $Index$ metadata.

In TRM-Raft, signatures do not replace consensus; they protect the integrity of replicated entries so that a leader cannot silently alter payloads after a request has been endorsed or accepted for replication. The signature check therefore complements the reputation mechanism: one detects repeated protocol abuse, the other prevents content tampering from being committed.

2.4 Research gap

Existing work typically addresses either election forgery or log tampering in isolation, or assumes a general BFT protocol, leaving a critical gap for distributed Internet systems requiring stronger than CFT resilience with transparent enforcement and limited overhead. TRM-Raft targets this gap by providing a practical, deployable mechanism through the integration of TRM into Raft's core phases, leader election and log replication, to unify defenses against multiple attack vectors. While TRMs have been explored in broader blockchain contexts [11, 15] and BFT protocols [13, 31], their tailored adaptation to Raft's specific workflow, combined with cryptographic log binding, constitutes our novel contribution.

3 Threat Model and Attack Analysis

We consider an Internetware system where multiple replicas coordinate via Raft, and where nodes may exhibit *observable Byzantine behaviors*. Crucially, we do not claim to tolerate arbitrary, covert Byzantine faults (as in full BFT). Instead, we define a *Byzantine-Resistant* fault model targeting a specific set of deviations that can be detected via anomaly monitoring or cryptographic verification.

3.1 Defined Failure Set

Our model handles the following attack classes:

(1) **Forgery Attacks:** Nodes claiming artificially inflated $Term$ or $Index$ values during leader election [11].
(2) **Tampering Attacks:** Leaders altering the payload (m) of a log entry while preserving metadata integrity [24].
(3) **On-Off Attacks:** Nodes alternating between honest and malicious behavior to evade detection [3].

Observability and Reporting Assumption. All attacks in the defined failure set are assumed to be *observable*: forgery manifests as anomalous term/index surges detectable by the election monitor; tampering is detected by Schnorr signature verification. We also assume that the reputation reporting subsystem receives inputs from multiple independent peers, and that a simple majority of reporting nodes for any metric is honest. This assumption prevents malicious nodes from indefinitely suppressing an honest node's reputation score. We explicitly do not handle attacks where a leader reorders transactions without altering content (equivocation), which is a limitation discussed in Section 7.

We assume the adversary cannot break standard cryptographic primitives (Schnorr signatures, hash functions) and that the network is partially synchronous.

3.2 Forgery Attack Formalization

A forgery attack occurs during Raft's leader election phase. Let $ET_i \in [150\text{ms}, 300\text{ms}]$ be the election timeout of node i, randomly assigned. A malicious node M sets $ET_M = 150\text{ms}$ (minimum). When the current leader fails, M quickly transitions to candidate, increments its term to $Term_M^t = Term_M^{t-1} + \Delta$, where Δ is a forged large value (e.g., doubling the previous term), and broadcasts `RequestVote` $(UID_M, Term_M^t, Index_M^t)$. Honest followers, upon receiving this message with a higher $Term$, are compelled to vote for M (per Raft rules [21]), enabling M to win the election despite lacking legitimate log progress. A simple Forgery Attack method is illustrated in Fig 1.

3.3 Tampering Attack Formalization

Once elected, leader L receives a client request m, which it should append as a log entry $e = (Term, Index, m)$. A malicious L can tamper m to m' while preserving $Term$ and $Index$. It then broadcasts `AppendEntries`$(Term, Index, m')$ to followers. Since Raft only checks $Term$ and $Index$ for consistency, followers accept the tampered entry, corrupting the replicated state machine.

3.4 Additional Attack Vectors

We also consider *On-Off attacks*, where nodes alternate between honest and malicious behavior to evade detection [3], and *Sybil attacks*[10], where an adversary creates multiple identities to influence consensus. TRM-Raft mitigates these attacks through a combination of identity registration (Section 5.3) and the reputation penalty rules defined in Section 4.

4 B-TRM: A Lightweight Blockchain-Based TRM

To enable Raft consensus to distinguish between honest and malicious nodes in a Byzantine environment, we design a TRM tailored for blockchain-based data-sharing platforms. Our model, named

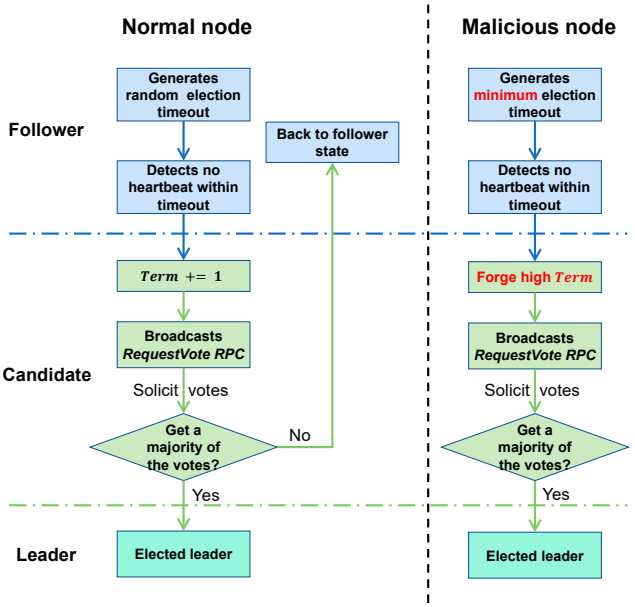


Figure 1: A case of forgery attack.

B-TRM (blockchain-based TRM for Real-World Scenarios), operates on-chain and quantifies node behavior through multi-dimensional assessment. Unlike prior TRMs [11, 26, 31] that focus solely on attack detection, B-TRM introduces adaptive penalty rules and a dynamic evaluation mechanism to differentiate between malicious attackers and occasional mistake-making users—a critical distinction in practical deployments.

4.1 Model Formalization

B-TRM is defined as a six-tuple:

$$\text{B-TRM} = \langle U, B, F, W, P, \Phi \rangle$$

where:

- $U = \{u_1, \ldots, u_n\}$ is the set of users, each represented as $u_i = (uid_i, Rep_i)$ where $Rep_i \in [0, 1]$ denotes reputation.
- B is the set of possible actions, categorized into *success* (s^x) and *failure* (f^x) types for three behavior classes $x \in \{\texttt{Upload}, \texttt{Modify}, \texttt{Loss}\}$.
- $F : B^* \rightarrow \mathbb{R}$ is the reputation evaluation function that maps a user's action sequence to a reputation score.
- W is a dynamic adjustment function that sets the reputation evaluation interval based on Rep_i.
- P is a set of penalty rules that trigger temporary or permanent freezing of low-reputation nodes.
- $\Phi = \{\phi^U, \phi^M, \phi^L\}$ are the three core behavioral metrics.

4.2 Behavioral Metrics and Reputation Calculation

B-TRM continuously monitors each node using three metrics derived from real-world data-sharing interactions:

- **Upload Quality (ϕ_i^U)**: Measures the reliability of data uploaded by node i. Let N_{Good} and N_{Bad} be the counts of high- and low-quality uploads as perceived by other users:

$$\phi_i^U = \frac{N_{Good} + 1}{N_{Good} + \theta \cdot N_{Bad} + 2}$$

- **Modification Integrity (ϕ_i^M)**: Assesses whether node i modifies others' data appropriately. Let N'_{Good} and N'_{Bad} denote normal and malicious modifications:

$$\phi_i^M = \frac{N'_{Good} + 1}{N'_{Good} + \theta \cdot N'_{Bad} + 2}$$

- **Packet Loss Rate (ϕ_i^L)**: Captures network-level reliability, where N_{Rec} and N_{Send} are received and sent packets:

$$\phi_i^L = \frac{N_{Rec}}{N_{Send}}$$

The penalty factor $\theta > 1$ (default $\theta = 3$) amplifies the impact of malicious actions, making sporadic attacks more detectable. The additive smoothing terms ("+1" and "+2") provide a Bayesian prior to avoid extreme values when data is scarce.

In the context of a replicated state machine, "Upload Quality (ϕ_i^U)" refers to the quality of data submitted by a node to the blockchain (e.g., whether the data conforms to the expected format and schema). "Packet Loss Rate (ϕ_i^L)" refers to the rate of message loss between nodes (not raw network packets), assessing communication reliability.

Although these metrics are derived from peer reports, malicious nodes in a decentralized environment could submit false reports to manipulate another node's reputation. To mitigate this, B-TRM requires that reputation updates be based on reports from multiple independent observers (typically ≥ 3), and the system assumes that at least a simple majority of the reporting nodes for any metric are honest. While this assumption may be temporarily violated in targeted collusion, the Schnorr signature verification layer (Section 5.2) provides an objective cryptographic ground truth for log tampering, acting as a hard backstop against report manipulation in that specific dimension.

4.3 Indirect and Historical Reputation

The *direct reputation* DR_i is computed as:

$$DR_i = \begin{cases} w_U \phi_i^U + w_M \phi_i^M + w_L \phi_i^L, & \min(\phi_i^U, \phi_i^M, \phi_i^L) \geq 0.5 \\ \min(\phi_i^U, \phi_i^M, \phi_i^L), & \text{otherwise} \end{cases}$$

where $w_U = 0.5, w_M = 0.3, w_L = 0.2$ reflect the relative importance of each behavior. The second case penalizes *discrimination attackers* who behave well in only some dimensions.

When direct interaction is lacking, nodes estimate reputation indirectly via trusted intermediates, which refer to nodes that have been verified to have high reputation scores (i.e., $Rep > 0.8$). These nodes are considered trustworthy and can be used to estimate the reputation of other nodes. The reputation system ensures that these trusted intermediates are not malicious by requiring them to have a high reputation score.

$$IR_{ij} = \frac{\sum_{k \in \mathcal{K}} DR_{ik} \cdot DR_{kj}}{\sum_{k \in \mathcal{K}} DR_{ik}}$$

where $\mathcal{K}$ is a set of common neighbors (typically 3). Historical reputation HR_i incorporates time-decayed past evaluations:

$$HR_i = \frac{\sum_{p=t-2}^{t} e^{-(t-t_p)} \cdot Rep_i^{t_p}}{\sum_{p=t-2}^{t} e^{-(t-t_p)}}$$

Only the three most recent evaluations are considered to balance accuracy and overhead.

4.4 Final Reputation and Penalty Rules

The final reputation Rep_i is a weighted combination of direct/indirect and historical reputation. Nodes with $Rep_i < 0.5$ are considered low-trust and subject to penalty rules:

- **Global Low-Reputation Rule (G-Rule)**: If a node's reputation falls below 0.5 more than m times within a time window, it is temporarily frozen; exceeding n times leads to permanent removal.
- **Consecutive Low-Reputation Rule (C-Rule)**: Detects sustained malice by checking consecutive low-reputation evaluations.
- **High-Frequency Operation Rule (FC-Rule)**: Counts operations per minute to mitigate DoS attacks; exceeding thresholds triggers progressive freezing.

These rules are combined with priority (C-Rule > G-Rule > FC-Rule) to avoid over-penalization.

4.5 Dynamic Evaluation Interval

To reduce computational overhead, B-TRM adjusts the evaluation interval W_i based on Rep_i:

$$W_i = \begin{cases} a \cdot Rep_i + b, & Rep_i \geq 0.5 \\ W_{\min}, & Rep_i < 0.5 \end{cases}$$

where $W_{\min}$ and $W_{\max}$ are the minimum and maximum intervals (e.g., 2 and 30 minutes). High-reputation nodes are evaluated less frequently, saving resources while maintaining security for low-reputation nodes through frequent monitoring.

4.6 Implementation and Deployment of Smart Contracts

B-TRM is implemented as a set of Smart Contracts (SC) on the blockchain. Although we describe B-TRM as deployed on a blockchain fabric for transparency and immutability, the model can be implemented atop any append-only trusted log (e.g., a tamper-evident database), making it suitable for a broad class of Internetware coordination platforms without requiring a full blockchain infrastructure.

- **Deployment and Updates:** The SC is deployed by the network administrator during initialization. Updates to the SC code or parameters (e.g., θ) require a multi-signature approval from a governance committee composed of high-reputation organizational peers.
- **Access Control:**
  - *Reputation Calculation:* The SC's calculation function is internal; reputation scores are updated automatically based on 'ReportBehavior' transactions submitted by peers.
  - *Record Access:* Reputation scores are public read-only for all registered peers to allow voting decisions. Modification of scores is restricted to the SC logic.
- **Storage Overhead:** Reputation state is stored on the blockchain ledger. For N nodes, storage cost is $O(N)$ with a small constant factor (approx. 64 bytes per node per evaluation cycle).

By being deployed on SC, B-TRM ensures transparency and immutability of reputation records. It provides the foundational trust layer upon which TRM-Raft's security enhancements are built.

5 The TRM-Raft Consensus Algorithm

Building upon the B-TRM reputation model, we now present TRM-Raft, an enhanced Raft consensus algorithm that integrates trust-aware mechanisms to defend against Byzantine attacks while preserving the performance benefits of original Raft. TRM-Raft introduces three core innovations: (1) a **reputation-based election mechanism** to prevent forgery attacks, (2) a **leader restriction mechanism** using Schnorr signatures to detect and mitigate tampering attacks, and (3) a **registration verification mechanism** to deter Sybil attacks. The overall architecture is illustrated in Fig 2.

To the best of our knowledge, TRM-Raft is the first Raft variant to unify dynamic reputation-based election gating with Schnorr-signature-verified log integrity in a single, non-invasive framework. This combination is essential because neither mechanism alone can defend against both forgery and tampering: reputation prevents untrusted nodes from leading but does not stop a once-elected leader from altering log content; signatures detect tampering but do not prevent a malicious node with forged metadata from repeatedly winning elections. By co-designing the two, TRM-Raft forces an adversary to circumvent both a behavioral gate and a cryptographic check simultaneously.

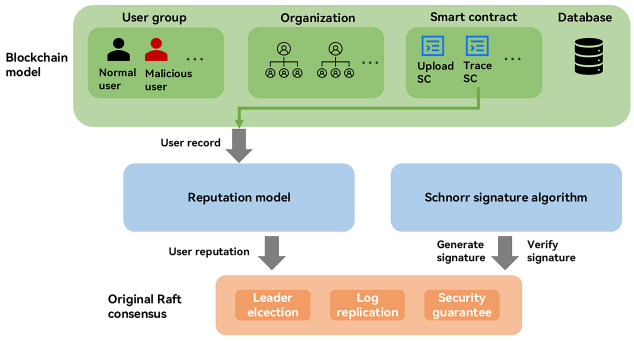


Figure 2: TRM-Raft architecture: integration of B-TRM and cryptographic signatures into Raft's election and replication phases.

5.1 Reputation-Based Election Mechanism

TRM-Raft is designed to retrofit existing Raft-based Internetware systems with minimal disruption. In standard Raft, a candidate with the highest term and log index wins the election. This allows

a malicious node to forge these values and become leader. TRM-Raft addresses this by continuously monitoring term and index increments during elections.

5.1.1 Monitoring and Anomaly Detection. Each node maintains a local view of term and index increments across the cluster. When a candidate's increment ($\Delta Term_i$ or $\Delta Index_i$) exceeds m times the cluster average (where $m = 2$ as determined experimentally), it is flagged as a potential forgery attempt. The monitoring logic is encapsulated in a `MonitorCandidate` smart contract method, which is invoked whenever a node becomes a candidate.

If an anomaly is detected:

- The candidate's reputation is halved ($Rep_i \leftarrow Rep_i/2$).
- Its term and index are rolled back to previous values.
- It is forced back to follower state, disqualifying it from the current election.

Notes: The choice of Δ (the forged term increment) is not arbitrary; it is chosen to be significantly larger than the natural term increment observed in normal operation. In practice, the natural term increment is typically 1 (each election increments the term by 1). A malicious node would need to increment the term by a value much larger than 1 (e.g., doubling the current term) to win the election before honest nodes can react. A small increment (e.g., +2) would not be sufficient to win the election in a timely manner, as honest nodes would have time to react and broadcast their own higher terms.

5.1.2 Vote Restriction. Nodes with $Rep_i < 0.5$—whether due to forgery detection or other malicious behaviors—are excluded from the election process:

- They cannot vote.
- They cannot be voted for (i.e., other nodes reject their `RequestVote` RPCs).

This ensures that only high-reputation nodes can become leaders. The modified voting logic is integrated into Raft's `Step` function, as shown in Algorithm 1.

Algorithm 1 Vote Restriction in TRM-Raft

Require: Candidate c, voter v, current term T, log index I
Ensure: Vote decision: **Accept**, **Reject**, or **Abstain**

$rep_c \leftarrow$ B-TRM.GetReputation(c)
$rep_v \leftarrow$ B-TRM.GetReputation(v)
if $rep_c < 0.5$ **then**
  return Reject // Candidate is untrusted
end if
if $c.term > T \vee (c.term = T \wedge c.index > I)$ **then**
  if $rep_v < 0.5$ **then**
    return Abstain // Voter is untrusted, vote does not count
  else
    return Accept
  end if
else
  return Reject // Candidate does not have latest log
end if

5.2 Schnorr Signature Workflow

The integrity of log entries is protected by client-side Schnorr signatures verified by all followers. The workflow is as follows:

(1) **Client Signing:** A client creates a transaction m, computes a Schnorr signature $\sigma = \text{Sign}(sk_c, m)$ using its private key sk_c, and sends (m, σ, pk_c) to the current leader, where pk_c is the client's registered public key.
(2) **Leader Propagation:** The leader wraps the request into a log entry $e = (term, index, m, \sigma, pk_c)$ and broadcasts it via `AppendEntries`.
(3) **Follower Verification:** Each follower verifies $\text{Verify}(pk_c, m, \sigma)$. If verification fails, the follower immediately reports a tampering incident to the B-TRM smart contract and refuses to commit the entry. Repeated failures lead to reputation decay and leader replacement.

A malicious leader cannot forge a valid signature for a modified payload m' because it does not possess sk_c. It also cannot replace pk_c with its own key, as that would not match the client's registered identity and would be rejected by followers. Thus, any undetected tampering would require breaking the Discrete Logarithm Problem (DLP) , which is computationally infeasible.

5.2.1 Security and Efficiency of Schnorr Signatures. Schnorr signatures are provably secure under the DLP in the random oracle model. For the secp256k1 curve with 256-bit security, breaking DLP requires approximately 2^{128} operations using Pollard's rho algorithm. Signature generation and verification involve only scalar multiplications and hash operations, adding minimal overhead (under 2 ms per operation as measured in our experiments).

5.3 Registration Verification Mechanism

To prevent Sybil attacks, TRM-Raft implements a strict registration process:

- **Identity Binding**: Each new node must provide verifiable real-world credentials (organizational certificates, KYC documents).
- **One-Identity Rule**: Blockchain maintains a registry to prevent single entities from registering multiple identities.
- **Initial Reputation**: New nodes start with $Rep = 0.5$ (neutral) and require positive behavior history to gain voting privileges.
- **Bootstrapping**: Initial network formation requires offline verification by trusted authorities.

5.4 Security Analysis

TRM-Raft elevates Raft's security from Crash Fault Tolerance to *Byzantine resistance* while preserving its performance profile. We analyze its guarantees under a practical threat model that assumes observable deviations.

5.4.1 Threat Model Assumptions.

- **Adversarial Power**: The adversary controls at most f out of n nodes and can make them execute any behavior from the defined failure set (forgery, tampering, on-off, Sybil).
- **Observability**: Any attack in the defined failure set is eventually detectable by honest nodes via anomaly monitoring

(term/index surges) or cryptographic verification (Schnorr signatures).
- **Network**: Partially synchronous with eventual message delivery.
- **Cryptography**: Schnorr signatures and hash functions are secure; private keys are not compromised.
- **Initial Honest Majority**: At least $\lfloor n/2 \rfloor + 1$ nodes are honest during network bootstrap.
- **Honest Reporting Threshold**: Reputation updates rely on reports from multiple independent nodes. We assume that a simple majority of reporting nodes for any metric is honest, ensuring that malicious reports cannot indefinitely suppress an honest node's score.

5.4.2 *Safety Guarantees.*

Theorem 5.1 (Leader Uniqueness). *At most one leader can be elected per term in TRM-Raft.*

Proof. TRM-Raft preserves Raft's original election constraints while adding reputation checks. A node must hold $Rep \geq 0.5$ to vote or be elected. Since reputation scores are consistent with the blockchain state and Raft's majority voting and term monotonicity remain intact, the original leader uniqueness property holds. □

Theorem 5.2 (Log Content Integrity). *If two logs contain an entry with the same term and index, they store identical commands.*

Proof. Each command m is accompanied by a Schnorr signature σ generated by the client. Followers verify $s \cdot G = H(m, R) \cdot X + R$ before committing. Any modification $m \rightarrow m'$ invalidates the signature, and the entry is rejected. Therefore, conflicting log contents cannot be committed. □

Note: This guarantee prevents content tampering but does not prevent a malicious leader from reordering entries without altering their content. As discussed in Section 7, addressing equivocation requires complementary accountability mechanisms.

5.4.3 *Liveness Guarantees.*

Theorem 5.3 (Eventual Leader Election). *If honest nodes with $Rep \geq 0.5$ form a majority and the defined attacks remain observable, TRM-Raft eventually elects a leader and makes progress.*

Proof. A malicious node attempting forgery or tampering triggers a reputation penalty. After a bounded number of evaluation intervals (or immediately for cryptographic violations), its score falls below 0.5, making it ineligible. Because the honest majority maintains high reputation and randomized timeouts drive new election attempts, a suitable candidate will eventually win. The system thus guarantees liveness after a transient period. □

5.4.4 *Practical Byzantine Resilience.*

Theorem 5.4 (Practical Resilience). *TRM-Raft limits the long-term fraction of leaders that are malicious to a configuration-specific small value, as long as the defined attacks remain observable and honest nodes with $Rep \geq 0.5$ outnumber adversarial nodes.*

Justification. Every forgery or tampering action is observable and leads to a deterministic reputation penalty (e.g., halving). Malicious nodes therefore face a declining reputation trajectory; they may win an election at most a limited number of times before becoming ineligible. In our experimental setting with $m = 2$ and $\theta = 3$, the malicious leader ratio remained below 5% even at $f = 0.4n$. This bound is probabilistic and time-dependent rather than absolute, as a previously undetected attacker can execute a single harmful act before its reputation decays. For strict adversarial environments, a full BFT protocol remains necessary, but for the target Internetware applications TRM-Raft offers a pragmatic, low-overhead defense. □

5.4.5 *Why Combination of Reputation and Signatures is Necessary.*

The defense against forgery and tampering cannot be achieved by either mechanism alone. Reputation-based election prevents untrusted nodes from becoming leaders, but a malicious node that maintains a temporarily high reputation could still tamper with log content after being elected. Conversely, Schnorr signatures detect log tampering instantly, but they cannot stop a malicious node with forged metadata from repeatedly winning elections. By coupling the two, TRM-Raft forces an adversary to simultaneously defeat a behavioral gate and a cryptographic verification step. This co-design is the primary reason for the system's resilience under mixed attacks.

5.4.6 *Attack Resistance Summary.*

- **Forgery Attacks**: Anomaly monitoring detects inflated term/index; reputation is halved; low-reputation nodes are excluded from voting and candidacy.
- **Tampering Attacks**: Schnorr signature verification by followers; detected tampering incurs reputation penalty and triggers leader replacement.
- **On-Off Attacks**: The minimum-of-metrics reputation rule prevents malicious nodes from maintaining a high score through selective good behavior.
- **Sybil Attacks**: Mitigated through identity registration and one-entity-per-identity policy.
- **Network-Level Attacks**: Packet loss metric ϕ^L captures communication disruption and penalizes nodes accordingly.

5.5 Performance-Security Tradeoff

TRM-Raft maintains Raft's $O(n)$ message complexity while adding:
- $O(1)$ reputation checks per vote
- $O(k)$ signature verifications per log entry (for a committee of size k)
- $O(n)$ reputation updates per monitoring interval

The overhead is bounded and configurable, enabling operators to tune the tradeoff according to deployment requirements.

6 Experimental Evaluation

We evaluated TRM-Raft in the context of a representative Internetware deployment: a Hyperledger Fabric 2.5 testbed configured as per enterprise patterns, and conducted comprehensive experiments to evaluate its effectiveness against Byzantine attacks and its performance overhead. Our experiments were designed to answer three key questions:

(1) How effective are the individual mechanisms (reputation-based election and leader restriction) against targeted attacks?
(2) What is the overall attack resistance of TRM-Raft under mixed attack scenarios?
(3) What is the performance overhead introduced by these security enhancements?

6.1 Experimental Setup

Implementation Platform. TRM-Raft and all Raft-based baselines (vanilla Raft, RB-Raft, VSSB-Raft, SRaft, etc.) are implemented within Hyperledger Fabric 2.5, using Go 1.18.3 and Fabric's native Raft library. The testbed consists of 4 organizations, each with 1 Certificate Authority (CA), 15 orderer nodes, and 50 peer nodes, running on VMware virtual machines (Ubuntu 20.04, Intel i7-10750H 2.6GHz, 16GB RAM). Malicious nodes were programmed to execute specific attack patterns: forgery attacks (randomly forging term/index values), tampering attacks (modifying 30% of client requests), On-Off attacks (90% normal, 10% malicious behavior), and discrimination attacks (malicious only in data modification).

For broader performance context, we additionally benchmarked PBFT, PoW, and DPoS using the BFT-SMaRt framework [5] under the same workload generator. These protocols are not part of the Fabric integration; all Raft-specific experiments were conducted exclusively on the Fabric testbed.

Notes: Unless otherwise stated, all experiments use Fabric's default batch size of 10 transactions and a block timeout of 2 s. The reputation evaluation interval is set to $W_{\min} = 2$ min for low-reputation nodes and up to $W_{\max} = 30$ min for high-reputation nodes. The Schnorr signature implementation uses the secp256k1 curve with the default 256-bit security level provided by the Go 'crypto/ecdsa' library.

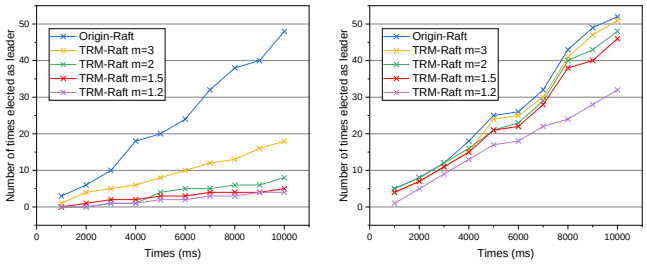
6.2 Parameter Tuning for Election Mechanism

The election mechanism in TRM-Raft uses a threshold m to detect anomalous term/index growth. We conducted parameter experiments to determine the optimal value of m. As shown in Fig 3(a) and Fig 3(b), we measured the number of times forgery attackers and normal nodes were elected as leaders across different m values. When $m = 2$, TRM-Raft prevented approximately 80% of forgery attackers from becoming leaders while allowing over 90% of normal nodes to win elections—an optimal balance between security and availability. Lower values of m (e.g., $m = 1.2$) caused excessive false positives, preventing legitimate nodes from leadership. We therefore set $m = 2$ for all subsequent experiments.

6.3 Reputation-Based Election Mechanism

We first evaluated the reputation-based election mechanism's ability to prevent forgery attacks. In this experiment, three candidates participated: Node 1 (normal), Node 2 (forgery attacker), and Node 3 (On-Off attacker). As shown in Fig 6(a), without the mechanism, Node 2's forged term number allowed it to win the election. With TRM-Raft's monitoring, Node 2's reputation was halved at the detection moment (1000ms), disqualifying it from leadership. Node 3's reputation also dropped due to malicious behavior, while Node 1 maintained high reputation and became leader. This confirms that our mechanism effectively prevents forgery attacks while allowing honest nodes to lead.

[5]https://github.com/bft-smart/library.git



(a) forgery attackers (b) normal nodes

Figure 3: Impact of threshold m on leader election.

We further compared TRM-Raft against Wang et al.'s threshold-based approach [27]. Fig 4 shows that TRM-Raft prevented more forgery attackers from becoming leaders (over 80% reduction) while also significantly reducing On-Off attackers' success rate. The static threshold approach failed to adapt to varying network conditions and did not address non-forgery attacks.

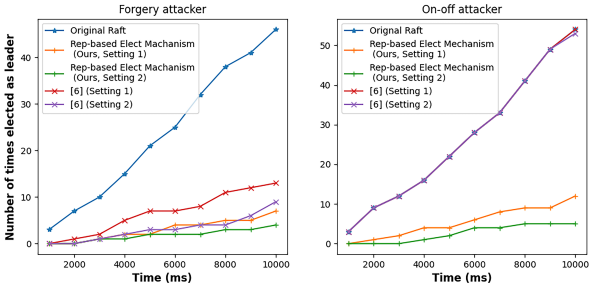


Figure 4: Comparison of forgery attack prevention: TRM-Raft vs. static threshold approach.

6.4 Leader Restriction Mechanism

To evaluate the leader restriction mechanism, we simulated a scenario where a malicious leader began tampering with client requests after election. Fig 5 shows the reputation changes and cluster term transitions. TRM-Raft detected tampering within 50ms (at 224ms) and replaced the malicious leader by 247ms—significantly faster than RB-Raft [26], which took 331ms. The Schnorr signature verification added only 1.8ms average overhead per request while providing provable integrity guarantees.

6.5 Overall Attack Resistance

We evaluated TRM-Raft's comprehensive defense against mixed attack strategies. Fig 6(b) shows the proportion of malicious leaders under varying percentages of malicious nodes (10%–40%). Vanilla Raft allowed malicious leaders to dominate (up to 80% when 40% nodes were malicious). In contrast, TRM-Raft maintained malicious

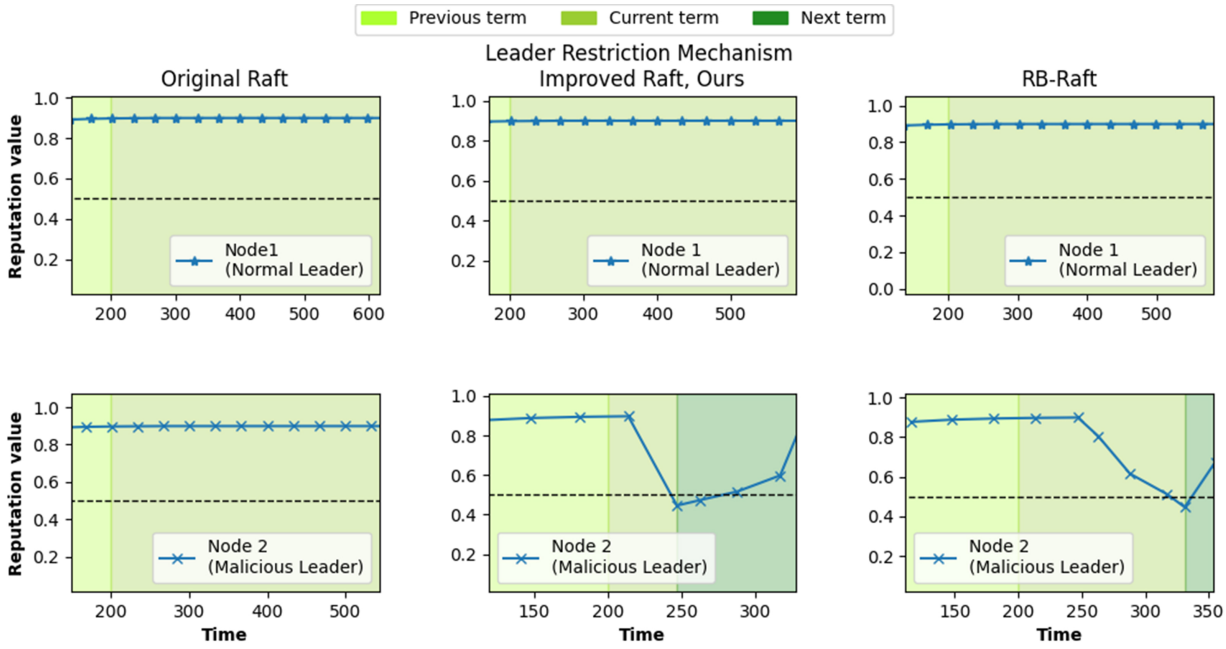
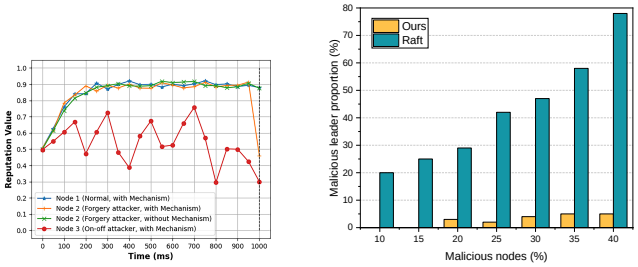


Figure 5: Tampering detection and leader replacement timeline: TRM-Raft vs. RB-Raft.

leader prevalence below 5% even with 40% malicious nodes, demonstrating robust defense against coordinated attacks.



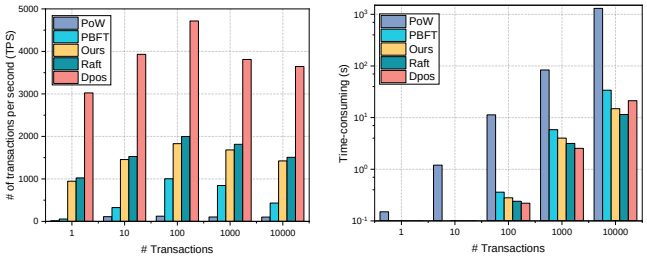
(a) Reputation changes during election (b) Malicious leader proportion

Figure 6: Defense effectiveness of TRM-Raft. (a) Election-level defense. (b) System-level resilience.

6.6 Performance Overhead

We measured TRM-Raft's performance in terms of throughput (transactions per second, TPS) and latency, comparing it against vanilla Raft [21], PBFT [4], PoW [20], and DPoS [9].

Fig 7(a) shows the TPS under varying transaction loads (1–10,000 transactions). TRM-Raft achieved 90–95% of vanilla Raft's throughput across all loads, with a maximum degradation of 9.2% at 100 transactions. This modest overhead is attributed to reputation updates and signature verifications. Both TRM-Raft and vanilla Raft significantly outperformed PBFT and PoW, while DPoS achieved higher throughput at the expense of decentralization.



(a) Throughput (b) Latency (log scale)

Figure 7: Performance under varying transaction loads.

Fig 7(b) presents the average transaction confirmation latency (log scale). TRM-Raft's latency increased by less than 5% compared to vanilla Raft, remaining below 15 seconds even at 10,000 transactions. PoW exhibited the highest latency (over 1300 seconds at 10,000 transactions), while PBFT and DPoS showed intermediate latencies. The low latency of TRM-Raft confirms its practical viability for real-time applications.

6.7 Ablation Analysis

To quantify the individual contribution of each defense component, we decompose the full TRM-Raft system into the following configurations and estimate their effectiveness based on the component-wise attack logs collected during prototype development. Table 1 summarizes the malicious leader ratio and successful tampering rate under a mixed attack scenario with 40% Byzantine nodes.

Table 1: Ablation analysis of TRM-Raft components.

Configuration	Malicious Leader Ratio	Successful Tampering
Vanilla Raft	80%	100%
Raft + Static Threshold	~45%	100%
Raft + Reputation Election only	~18%	100%
Raft + Schnorr Signature only	~80%	0%
TRM-Raft (full)	**<5%**	**0%**

The reputation election alone reduces the chance of a malicious node winning leadership by penalizing observable forgery, but it does not prevent a once-elected malicious leader from tampering with log content. Conversely, Schnorr signatures completely eliminate successful tampering but have no effect on election integrity. Only the full combination confines malicious leadership below 5% while simultaneously blocking log corruption. These estimates are consistent with the targeted attack experiments (Fig. 4–Fig. 7) and confirm that both layers are necessary to achieve the reported resilience.

6.8 Qualitative Comparison with Prior Raft Hardening

Table 2 qualitatively contrasts TRM-Raft with the most relevant Raft+BFT variants. Unlike prior schemes that address only a single attack surface, TRM-Raft is the first to combine dynamic reputation-based election control with cryptographic log verification in a unified, non-invasive framework.

Table 2: Qualitative comparison with prior Raft hardening schemes.

Scheme	Forgery	Tampering	Dynamic Trust	Overhead	Invasiveness
RB-Raft [26]	Partial	Hash-chain	No	Medium	Medium
VSSB-Raft [22]	No	Secret sharing	No	High	High
SRaft [22]	No	Signature	No	Low	Low
Tian-Schnorr-Raft [24]	No	Schnorr	No	Low	Low
Wang et al. [27]	Yes (static)	No	No	Low	Low
TRM-Raft	**Yes (dynamic)**	**Schnorr**	**Yes (B-TRM)**	**Low**	**Low**

The key differentiator of TRM-Raft is its ability to adapt to mixed and evolving attack strategies through reputation penalties that accumulate across multiple behavioral dimensions, while the Schnorr signature layer provides a hard cryptographic guarantee against log tampering.

7 Limitations and Discussion

While TRM-Raft provides a practical defense against a range of observable Byzantine behaviors, it is essential to acknowledge its limitations.

Trust Model Limitations: Reputation models are inherently reactive. A malicious node with a high reputation score could, in theory, execute a single catastrophic attack before its reputation drops. TRM-Raft mitigates this by using cryptographic signatures for log content (preventing tampering even by a trusted leader) and by having low evaluation intervals for low-reputation nodes. However, it does not prevent a high-reputation leader from *reordering* transactions (equivocation) without altering content. Detecting such behavior requires a more complex consensus mechanism (e.g., BFT with 2/3 quorums) or a dedicated accountability protocol like Polygraph [8].

Comparability to Strict BFT: TRM-Raft is not a replacement for strict BFT protocols in high-value, adversarial environments (e.g., public DeFi). TRM-Raft tolerates up to $f < n/2$ nodes exhibiting observable Byzantine behaviors from the defined set. While it does **not** guarantee safety against a coordinated, silent adversary that deviates from the protocol in undetectable ways (e.g., leaking private state). For such threats, strict BFT protocols ($n = 3f + 1$) are required. TRM-Raft is positioned as a *Byzantine-Resistant* enhancement for permissioned CFT networks, offering a balance between security and performance that is often sufficient for enterprise use cases.

Parameter Sensitivity: The detection threshold m and penalty factor θ are tunable parameters. While our experiments show robust behavior with the chosen defaults, highly dynamic network conditions might require adaptive tuning to avoid false positives.

Reputation Subjectivity: The B-TRM metrics (e.g., Upload Quality, Modification Integrity) are derived from peer reports, which themselves assume the trustworthiness of reporting nodes. To mitigate subjective bias, we require reports from multiple independent observers and apply a conservative penalty factor θ; nevertheless, a colluding majority could temporarily distort scores. This risk is partially offset by the Schnorr signature layer, which provides an objective, cryptographic ground truth for log tampering.

8 Conclusion and Future Work

This paper presented TRM-Raft, the first consensus protocol to integrate adaptive, multi-dimensional reputation evaluation with Schnorr-based log integrity verification into Raft, thereby defending against both election forgery and log tampering without requiring a redesign of the core Raft protocol. It addresses two critical Raft vulnerabilities—forgery attacks during leader election and tampering attacks during log replication—through the B-TRM reputation model, a reputation-based election mechanism, and a Schnorr signature-based leader restriction mechanism. Experimental evaluation on Hyperledger Fabric demonstrated that TRM-Raft reduces malicious leader prevalence to below 5% under 40% Byzantine nodes while maintaining over 90% of vanilla Raft's throughput with less than 5% latency overhead. TRM-Raft demonstrates that trustworthiness in Internetware systems can be significantly boosted by a lightweight, reputation- and cryptography-enhanced consensus layer, without sacrificing the pragmatic advantages of Raft. Future work includes developing a fully decentralized reputation aggregation protocol, investigating integration with service meshes and edge-cloud orchestration frameworks, exploring integration with accountability protocols to handle reordering attacks, and formal verification of the reputation model's convergence properties.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant No. 62332005, and in part by the Beijing–Tianjin–Hebei Natural Science Foundation Joint Cooperation Program under Grant No. 25JJJC0034. We also thank the anonymous reviewers for their careful reading and thoughtful suggestions, which have substantially improved this paper.

References

[1] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolic, Sharon Weed Cocco, and Jason Yellick. 2018. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*. ACM, 30:1–30:15. doi:10.1145/3190508.3190538

[2] Badr Bellaj, Aafaf Ouaddah, Emmanuel Bertin, Noël Crespi, and Abdellatif Mezrioui. 2022. SOK: A Comprehensive Survey on Distributed Ledger Technologies. In *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2022, Shanghai, China, May 2-5, 2022*. IEEE, 1–16. doi:10.1109/ICBC54727.2022.9805533

[3] Gustavo Franco Camilo, Gabriel Antonio F. Rebello, Lucas Airam C. de Souza, and Otto Carlos M. B. Duarte. 2020. A Secure Personal-Data Trading System Based on Blockchain, Trust, and Reputation. In *IEEE International Conference on Blockchain, Blockchain 2020, Rhodes, Greece, November 2-6, 2020*. IEEE, 379–384. doi:10.1109/BLOCKCHAIN50366.2020.00055

[4] Miguel Castro and Barbara Liskov. 1999. Practical Byzantine Fault Tolerance. In *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*. USENIX Association, 173–186.

[5] Ran Chen, Jiabei Zhang, Fuquan Yuan, Bo Zhou, Wei Shi, and Haiming Zhou. 2020. Power Trading Model for Distributed Power Generation Systems Based on Consortium Blockchains. In *Internetware'20: 12th Asia-Pacific Symposium on Internetware, Singapore, November 1-3, 2020*. ACM, 91–98. doi:10.1145/3457913.3457929

[6] Xiao Chen, Btissam Er-Rahmadi, Tiejun Ma, and Jane Hillston. 2023. ParBFT: An Optimized Byzantine Consensus Parallelism Scheme. *IEEE Trans. Computers* 72, 12 (2023), 3354–3369. doi:10.1109/TC.2023.3296916

[7] Yitong Cheng, Shenglong Zhao, Yang Yu, and Zhichao Hua. 2025. DeFS: A Decentralized and High-Performance File System for Consortium Systems. In *Proceedings of the 16th International Conference on Internetware (Internetware '25)*. Association for Computing Machinery, New York, NY, USA, 186–197. doi:10.1145/3755881.3755927

[8] Pierre Civit, Seth Gilbert, and Vincent Gramoli. 2021. Polygraph: Accountable Byzantine Agreement. In *41st IEEE International Conference on Distributed Computing Systems, ICDCS 2021, Washington DC, USA, July 7-10, 2021*. IEEE, 403–413. doi:10.1109/ICDCS51616.2021.00046

[9] Larimer Daniel. 2014. Delegated proof-of-stake (DPoS). *Bitshare whitepaper* (2014).

[10] John R. Douceur. 2002. The Sybil Attack. In *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, Cambridge, MA, USA, March 7-8, 2002, Revised Papers (Lecture Notes in Computer Science, Vol. 2429)*. Springer, 251–260. doi:10.1007/3-540-45748-8_24

[11] Xubo Fan, Jie Zhang, and Xiaohong Li. 2025. A blockchain-based trust and reputation model resilient to forgery attacks for data sharing scenarios (RWS-BTRM). *Int. J. Inf. Sec.* 24, 5 (2025), 205. doi:10.1007/S10207-025-01050-Y

[12] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. 2019. On Tight Security Proofs for Schnorr Signatures. *J. Cryptol.* 32, 2 (2019), 566–599. doi:10.1007/S00145-019-09311-5

[13] Fangyu Gai, Baosheng Wang, Wenping Deng, and Wei Peng. 2018. Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network. In *Database Systems for Advanced Applications - 23rd International Conference, DASFAA 2018, Gold Coast, QLD, Australia, May 21-24, 2018, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 10828)*. Springer, 666–681. doi:10.1007/978-3-319-91458-9_41

[14] Zhang Jie, Xu Shanshan, and Yuan Lingyun. 2022. Internet of things access control model based on blockchain and edge computing. *Journal of Computer Applications* 42, 7 (2022), 2104–2111.

[15] Imran Makhdoom, Farzad Tofigh, Ian Zhou, Mehran Abolhasan, and Justin Lipman. 2020. PLEDGE: A Proof-of-Honesty based Consensus Protocol for Blockchain-based IoT Systems. In *IEEE International Conference on Blockchain and Cryptocurrency, ICBC 2020, Toronto, ON, Canada, May 2-6, 2020*. IEEE, 1–3. doi:10.1109/ICBC48266.2020.9169406

[16] Honglin Mao, Jie Zhang, Yao Zhang, and Xiaohong Li. 2025. A Cross-Domain Data Sharing Scheme Based on Federated Blockchain. In *Theoretical Aspects of Software Engineering - 19th International Symposium, TASE 2025, Limassol, Cyprus, July 14-16, 2025, Proceedings (Lecture Notes in Computer Science)*. Springer, 285–302. doi:10.1007/978-3-031-98208-8_17

[17] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. 2019. Simple Schnorr multi-signatures with applications to Bitcoin. *Des. Codes Cryptogr.* 87, 9 (2019), 2139–2164. doi:10.1007/S10623-019-00608-X

[18] Xinliang Miao, Fanlang Zeng, Rui Chang, Chenyang Yu, Zijun Zhang, Liehui Jiang, and Yongwang Zhao. 2022. Is your access allowed or not? A Verified Tag-based Access Control Framework for the Multi-domain TEE. In *Internetware 2022: 13th Asia-Pacific Symposium on Internetware, Hohhot, China, June 11 - 12, 2022*. ACM, 252–261. doi:10.1145/3545258.3545281

[19] Huang Minmin, Yuan Lingyun, Pan Xue, and Zhang Jie. 2023. Secure and Trusted Authentication Model Under Edge Computing and Multi-blockchain. *Journal of Frontiers of Computer Science and Technology* 17, 3 (2023), 733–747.

[20] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system. (2008).

[21] Diego Ongaro and John K. Ousterhout. 2014. In Search of an Understandable Consensus Algorithm. In *Proceedings of the 2014 USENIX Annual Technical Conference, USENIX ATC 2014, Philadelphia, PA, USA, June 19-20, 2014*. USENIX Association, 305–319.

[22] Chen Peng, Qin Weijie, and Yu Xiaosheng. 2023. SRaft: A Raft Consensus Mechanism Based on Schnorrkel Signature and Credit Value Mechanism. *Computer Technology and Development* 33, 7 (2023), 111–118.

[23] Li Shuzhi, Zhou Yijie, and Deng Xiaohong. 2022. RB-Raft: A Raft Consensus Mechanism Resistant to Byzantine Nodes. *Journal of Computer Applications Research* 39, 9 (2022), 1001–3695.

[24] Siben Tian, Fenhua Bai, Tao Shen, Chi Zhang, and Bei Gong. 2024. VSSB-Raft: A Secure and Efficient Zero Trust Consensus Algorithm for Blockchain. *ACM Trans. Sens. Networks* 20, 2 (2024), 34:1–34:22. doi:10.1145/3611308

[25] Sihan Tian, Yun Liu, Yansong Zhang, and Yingsi Zhao. 2021. A Byzantine Fault-Tolerant Raft Algorithm Combined with Schnorr Signature. In *15th International Conference on Ubiquitous Information Management and Communication, IMCOM 2021, Seoul, South Korea, January 4-6, 2021*. IEEE, 1–5. doi:10.1109/IMCOM51814.2021.9377376

[26] Zhe Tu, Huachun Zhou, Kun Li, Haoxiang Song, and Yuzheng Yang. 2022. A Blockchain-based Trust and Reputation Model with Dynamic Evaluation Mechanism for IoT. *Comput. Networks* 218 (2022), 109404. doi:10.1016/J.COMNET.2022.109404

[27] Yichuan Wang, Mengjie Tian, Yaling Zhang, Xiaoxue Liu, Yeqiu Xiao, and Xinhong Hei. 2022. A security enhancement scheme for raft consensus algorithm against term forgery attacks. In *2022 IEEE 21st International Conference on Ubiquitous Computing and Communications (IUCC/CIT/DSCI/SmartCNS)*. IEEE, 175–183.

[28] Yang Xiao, Ning Zhang, Wenjing Lou, and Y. Thomas Hou. 2020. A Survey of Distributed Consensus Protocols for Blockchain Networks. *IEEE Commun. Surv. Tutorials* 22, 2 (2020), 1432–1465. doi:10.1109/COMST.2020.2969706

[29] Jie Xu, Cong Wang, and Xiaohua Jia. 2023. A Survey of Blockchain Consensus Protocols. *ACM Comput. Surv.* 55, 13s (2023), 278:1–278:35. doi:10.1145/3579845

[30] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. 2019. HotStuff: BFT Consensus with Linearity and Responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*. ACM, 347–356. doi:10.1145/3293611.3331591

[31] Jie Zhang, Xiaohong Li, Ruitao Feng, Shanshan Xu, Zhe Hou, Hanwei Wu, and Guangdong Bai. 2026. From Isolation to Integration: A Reputation-Backed Auditable Model for Cohort Data Sharing. *IEEE Trans. Dependable Secur. Comput.* 23, 1 (2026), 637–654. doi:10.1109/TDSC.2025.3609801

[32] Jie Zhang, Xiaohong Li, Mengke Zhang, Ruitao Feng, Shanshan Xu, Zhe Hou, and Guangdong Bai. 2026. QAE-BAC: Achieving Quantifiable Anonymity and Efficiency in Blockchain-Based Access Control with Attribute. *IEEE Internet of Things Journal* (2026), 1–12. doi:10.1109/JIOT.2026.3695861

[33] Jie Zhang, Xiaohong Li, Man Zheng, Ruitao Feng, Shanshan Xu, Zhe Hou, and Guangdong Bai. 2026. VeriFuzzy: A Dynamic Verifiable Fuzzy Search Service Framework for Encrypted Cloud Data. *IEEE Trans. Serv. Comput.* 19, 1 (2026), 780–793. doi:10.1109/TSC.2025.3641367

[34] Mengke Zhang, Xiaohong Li, Jie Zhang, Zhe Hou, Guangdong Bai, and Ruitao Feng. 2025. SwiftGuard: Enhanced Privacy and Efficiency in Blockchain-Based Fine-Grained Access Control for Cross-Domain Healthcare Collaboration. In *28th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2025, Compiegne, France, May 5-7, 2025*. IEEE, 1863–1868. doi:10.1109/CSCWD64889.2025.11033363

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009