

Predictive Reputation-Based Consensus: Hardening Decoupled Machine Learning Against Telemetry Spoofing in Enterprise Blockchains

by

**Md. Yusuf
221014026**

*Thesis report submitted in partial fulfillment of the
requirements for the degree of*

Bachelor of Science in Computer Science and Engineering

Under the supervision of

Professor Nafees Mansoor, PhD



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY OF LIBERAL ARTS BANGLADESH**

SPRING 2026

© *Md. Yusuf*
All rights reserved

DECLARATION

Project Title	Predictive Reputation-Based Consensus: Hardening Decoupled Machine Learning Against Telemetry Spoofing in Enterprise Blockchains
Author	<i>Md. Yusuf</i>
Student ID	221014026
Supervisor	Professor Nafees Mansoor, PhD

I declare that this thesis report entitled *Predictive Reputation-Based Consensus: Hardening Decoupled Machine Learning Against Telemetry Spoofing in Enterprise Blockchains* is the result of my own work except as cited in the references. The thesis report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Md. Yusuf
221014026

Department of Computer Science and Engineering
University of Liberal Arts Bangladesh

Date: April 21, 2026



Department of Computer Science and Engineering
University of Liberal Arts Bangladesh
Mohammadpur, Dhaka - 1207

CERTIFICATE

This is to certify that the thesis report entitled **Predictive Reputation-Based Consensus: Hardening Decoupled Machine Learning Against Telemetry Spoofing in Enterprise Blockchains**, submitted by **Md. Yusuf** (Student ID: 221014026) an undergraduate student of the **Department of Computer Science and Engineering** has been examined. Upon recommendation by the examination committee, we hereby accord our approval of it as the presented work and submitted report fulfill the requirements for its acceptance in partial fulfillment for the degree of *Bachelor of Science in Computer Science and Engineering*.

Professor Nafees Mansoor, PhD
Professor

Department of Computer Science and Engineering
University of Liberal Arts Bangladesh

Prof. Muhammad Golam Kibria, PhD
SMIEEE Professor Head

Department of Computer Science and Engineering
University of Liberal Arts Bangladesh

Place: Dhaka

Date: April 21, 2026

ACKNOWLEDGEMENTS

I would like to express my deep and sincere gratitude to my research supervisor, *Professor Nafees Mansoor, PhD*, for giving me the opportunity to do research and providing invaluable guidance throughout this work. His dynamism, vision, sincerity and motivation have deeply inspired me. He has taught me the methodology to carry out the work and to present the works as clearly as possible. It was a great privilege and honor to work and study under his guidance.

I am greatly indebted to my honorable teachers of the Department of Computer Science and Engineering at the University of Liberal Arts Bangladesh who taught me during the course of my study. Without any doubt, their teaching and guidance have completely transformed me to a person that I am today.

I am extremely thankful to my parents for their unconditional love, endless prayers, caring and immense sacrifices for educating and preparing me for my future. I would like to say thanks to my friends and relatives for their kind support and care.

Finally, I would like to thank all the people who have supported me to complete the project work directly or indirectly.

Md. Yusuf

University of Liberal Arts Bangladesh

Date: April 21, 2026

Dedicated to
*My Mother, Father, Siblings,
Future Wife, Friends,
Supervisor and the Pursuit of
Knowledge, for the Drive and
Discipline it has Instilled.*

– Md. Yusuf

ABSTRACT

Permissioned blockchain networks require high transaction throughput and absolute data privacy, yet traditional consensus protocols rely on reactive heuristics that leave enterprise ledgers highly vulnerable to covert Byzantine threats such as sleeper decay and cartel collusion. While decoupled machine learning sidecars utilizing an optimized eXtreme Gradient Boosting (XGBoost) architecture can proactively detect these pre-crime footprints without introducing fatal consensus serialization risks, they remain fundamentally susceptible to sophisticated telemetry spoofing. When adversaries feed the classifier fabricated metrics, baseline accuracy collapses catastrophically from 97.25 percent to 33.33 percent. To address this critical vulnerability, this research proposes the Telemetry Attestation and Resilience Integration Module (TARIM), a multi-layer defense framework that systematically detects and corrects telemetry manipulation. By incorporating hardware-anchored telemetry via Intel SGX, cross-peer behavioral triangulation, and statistical integrity verification, the system evaluates node intent asynchronously without blocking ledger operations. Evaluated against a 6-class Byzantine attack taxonomy across 12,000 continuous epochs, TARIM recovered global classification accuracy to 99.08 percent under active attack. Furthermore, adversarial retraining successfully resolved baseline weaknesses, achieving a near-perfect recall rate against sustained decay. Crucially, the 20-feature TARIM pipeline required merely 4.895 milliseconds of inference latency, consuming a negligible 0.245 percent of the 2000-millisecond Raft timeout threshold. Ultimately, this research proves that predictive artificial intelligence can preemptively secure permissioned ledgers, and it demonstrates that these intelligence layers can be structurally hardened against data poisoning without degrading enterprise transaction speeds.

Keywords: Permissioned Blockchains, Byzantine Fault Tolerance, Machine Learning, Predictive Trust, Adversarial Robustness, Telemetry Spoofing, Hyperledger Fabric, XGBoost

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem Statement	2
1.3	Aims, Objectives and Motivation	3
1.3.1	Aims	3
1.3.2	Objectives	4
1.3.3	Motivation	4
1.4	Project Specification	5
2	Literature Review	6
2.1	The Reactive Consensus Problem and Its Consequences	6
2.2	Machine Learning as a Blockchain Security Primitive	8
2.3	Reputation-Based and Predictive Consensus Mechanisms	9
3	Methodology	12
3.1	Overview	12
3.2	Architectural Alternatives and Concept Reduction	13
3.2.1	In-Band Reactive Thresholding (Heuristic Smart Contracts)	13
3.2.2	Native On-Chain Machine Learning	13
3.2.3	Out-of-Band Predictive AI Sidecar	14
3.3	Dataset Description	15
3.3.1	Data Source and Collection	15
3.3.2	Feature Space	15
3.3.3	Target Variable and Class Distribution	17
3.3.4	Data Splitting Strategy	18
3.4	Threat Model: Telemetry Spoofing Adversary (TSA)	18

3.4.1	Adversary Capabilities	18
3.4.2	Spoofing Mechanism	19
3.4.3	Impact Quantification	19
3.5	TARIM Architecture	21
3.5.1	Design Philosophy	21
3.5.2	Layer 1: Hardware-Anchored Telemetry (HAT)	21
3.5.3	Layer 2: Cross-Peer Behavioral Triangulation (CPBT)	22
3.5.4	Layer 3: Statistical Telemetry Integrity Verification (STIV)	23
3.5.5	Complete Feature Vector	24
3.6	Experimental Design	26
3.6.1	Ablation Study Configuration	26
3.6.2	Adversarial Training Strategy	26
3.6.3	Hyperparameter Optimization	26
3.6.4	Evaluation Metrics and Explainability	27
3.7	Implementation Environment	27
4	Design and Development	28
4.1	Overview	28
4.2	System Architecture and Sidecar Routing	28
4.2.1	Sidecar Deployment Model	28
4.2.2	Module Structure	29
4.2.3	Stack Orchestrator	29
4.3	Raw Feature Extraction and Dataset Generation	30
4.3.1	TSA Offline Spoofing Pipeline	32
4.4	Feature Engineering and Dimensionality Reduction	32
4.5	Layer 1: Hardware-Anchored Telemetry (HAT) Implementation	33
4.6	Layer 2: Cross-Peer Behavioral Triangulation (CPBT) Implementation	34
4.7	Layer 3: Statistical Telemetry Integrity Verification (STIV) Implemen- tation	35
4.7.1	CUSUM Monitor	35
4.7.2	Temporal Feature Engineering	35
4.7.3	CFCS Scorer and Meta-Classifer	35
4.7.4	Model Topology Configurations	36
4.7.5	End-to-End Execution Flow	36

4.7.6	Non-Convex Hyperparameter Optimization	37
4.7.7	Augmented Training Data Generation	37
4.8	Flask Deployment Architecture	37
5	Result Analysis	39
5.1	Overview	39
5.2	Dataset Distribution and Attack Verification Boundaries	39
5.3	Comprehensive Correlation and Multicollinearity Matrix	41
5.4	Baseline Reproduction and Empirical Classification Metrics	41
5.5	TSA Impact Analysis	45
5.6	Defense Layer Ablation Results	46
5.7	Overhead and SLA Compliance	48
5.8	Defense Layer Behavior Analysis	49
5.9	Feature Importance and Explainability	51
5.10	Hyperparameter Optimization and Model Stability	52
5.11	Discussion and Summary	53
6	Conclusions	56
6.1	Social, Legal, Ethical, and Environmental Issues	56
6.2	Brief Summary	57
6.3	Future Works	58
	References	60

List of Figures

3.1	System Architecture of the Proactive AI Sidecar.	16
3.2	Confusion Matrix Under TSA Attack (No Defense).	20
3.3	Layer 1 CPU Hardware Delta Distribution.	22
3.4	CUSUM Distributions.	23
4.1	Final Multicollinearity Analysis (VIF).	33
5.1	Dataset Class Distribution.	40
5.2	Empirical Attack Timeline.	40
5.3	Normalized Attack Signature Radar Charts.	41
5.4	Feature Correlation Matrix.	42
5.5	Confusion matrices demonstrating classification overlap and isolated error distributions across all tested model topologies.	43
5.6	Deep Learning Training Convergence	44
5.7	Confusion Matrix Under TSA Attack.	46
5.8	Classification Accuracy Across Configurations.	47
5.9	Accuracy Recovery Waterfall.	47
5.10	Inference Overhead Analysis.	48
5.11	Defense Layer Activation Rates.	50
5.12	CPU Spoofing Distribution.	50
5.13	Full TARIM Feature Importance.	51
5.14	SHAP Full TARIM Importance.	51
5.15	Feature Importance Shift.	52
5.16	SHAP Dependence for cpu_hw_delta.	53
5.17	Optuna Optimization History.	53
5.18	Decision Boundary Shift.	54

List of Tables

3.1	Raw Telemetry Features	17
3.2	Byzantine Attack Class Taxonomy	17
3.3	Data Split Configuration	18
3.4	TSA Spoofing Parameters	19
3.5	TSA Impact on Per-Class Recall	20
3.6	Layer 1 (HAT) Features	21
3.7	Layer 2 (CPBT) Features	23
3.8	Layer 3 (STIV) Features	24
3.9	Complete TARIM Feature Vector (20 Features)	25
3.10	Ablation Study Configurations	26
3.11	Optuna Hyperparameter Search Space	27
4.1	Model Topology and Configuration Parameters. The table details temporal context windows, hyperparameter optimization frameworks, and the core processing mechanisms for each model evaluated.	36
5.1	Baseline (XGBoost) Performance on Clean Test Data	44
5.2	Baseline Per-Class Recall	44
5.3	TSA Impact on Baseline Performance	45
5.4	Per-Class Recall Under TSA Attack	45
5.5	Complete Ablation Study Results	46
5.6	Inference Latency Breakdown by Component	48
5.7	Memory Footprint by Component	49
5.8	Best Hyperparameters per Configuration	54
5.9	Baseline vs. Full TARIM Comprehensive Comparison	54

Chapter 1

Introduction

1.1 Background

Permissioned blockchain networks, particularly Hyperledger Fabric, have emerged as the foundational infrastructure for enterprise consortiums. These networks require secure, decentralized data management without the extreme computational overhead and monopolization risks inherent to public Proof of Work and Proof of Stake systems [Windiattmaja et al. \(2023\)](#). As these permissioned networks scale to accommodate highly complex and volatile environments, such as the Internet of Things (IoT) and the Internet of Vehicles (IoV), they increasingly face sophisticated cybersecurity vulnerabilities [Wang et al. \(2026\)](#). Comprehensive analyses of IoT security architectures reveal that traditional networks struggle significantly with anomaly detection, access control, and malicious routing behaviors [Tariq et al. \(2023\)](#).

Traditional consensus algorithms operate under rigid timeout mechanisms and static leader election protocols. Examples include Practical Byzantine Fault Tolerance (PBFT) and Raft. These foundational models are highly susceptible to targeted adversarial threats. These threats include Sybil attacks, node collusion, and identity spoofing [Aslam et al. \(2021\)](#); [Raghav & Bhola \(2023\)](#). To resolve these structural bottlenecks, the integration of Artificial Intelligence (AI) and Machine Learning (ML) into blockchain data analysis has gained immense traction [Azad et al. \(2025\)](#). Researchers have successfully utilized adaptive consensus optimization systems integrating Proximal Policy Optimization to dynamically adjust validation operations [Gutierrez et al. \(2025\)](#). Decentralized Federated Learning frameworks combining Random Forest and Bidirectional Long Short-Term Memory (BiLSTM)

sequence analysis have also been deployed. They mitigate Distributed Denial of Service (DDoS) attacks while maintaining an immutable audit trail [Mishra et al. \(2026\)](#); [Krishnapriya & Singh \(2025\)](#).

The paradigm of network trust is actively shifting from static cryptography to dynamic, reputation-based scoring. Recent literature has proposed enhanced algorithms like RF-PBFT. This framework integrates Random Forest predictive models to optimally partition nodes and deter malicious behaviors [Xu et al. \(2026\)](#). Advanced architectures are also employing Graph Neural Networks (GNNs) equipped with graph-temporal features. These detect complex, multi-hop anomalies and contract-driven collusion [Zhou et al. \(2024\)](#); [Shawn \(2025\)](#). Specific to consensus leader election, frameworks like the Proof of Intelligent Reputation (PoIR) leverage BiLSTM regression models. They evaluate authoritative nodes dynamically based on multivariate behavioral data [Windiatmaja et al. \(2023\)](#); [Jauzak Hussaini Windiatmaja \(2023\)](#). Other models have successfully integrated Artificial Neural Networks (ANN) trained on node age, stability, and block proposal success rates to calculate predictive trust [Saadat et al. \(2023\)](#).

However, two critical engineering gaps remain. First, it is challenging to deploy this complex predictive intelligence natively without violating the strict temporal Service Level Agreements inherent to enterprise blockchain operations. Second, the security of the ML components themselves remains largely unaddressed against targeted data poisoning.

1.2 Problem Statement

Current permissioned blockchain consensus mechanisms rely entirely on reactive heuristic logic. Protocols such as Hyperledger Fabric's Raft operate under the baseline assumption that all authenticated participants act in good faith. These systems only enact punitive measures after a malicious entity has successfully proposed a fraudulent block, bypassed an endorsement policy, or overtly missed a 2.0 second verification timeout. This inherently reactive architecture leaves the enterprise ledger highly vulnerable to covert, long-term Byzantine threats. Specific examples include Sleeper Decay and Cartel Collusion.

In a Sleeper Decay scenario, an adversarial node behaves perfectly for thousands of transaction epochs to farm positive reputation. It only executes a high-value

attack once it secures a leadership position. Traditional algorithms lack multidimensional chronological memory. Therefore, they cannot detect the microscopic hardware perturbations that precede these attacks. Similarly, in Cartel Collusion, a subnetwork of nodes coordinates to endorse each other's invalid blocks. Static math formulas evaluate nodes individually and fail to map these graph-based anomalies [Rawlins et al. \(2024\)](#).

While machine learning provides the analytical capability to detect these pre-crime topological footprints, a severe new vulnerability emerges. Adaptive and ML-assisted consensus requires principled design and threat analysis. Despite early demonstrations of ML-assisted adaptive consensus, there is limited rigorous analysis of the attack surface introduced by ML components in consensus protocols. This includes adversarial data manipulation and model poisoning.

Existing ML-based consensus monitors rely on self-reported telemetry. A sophisticated Telemetry Spoofing Adversary (TSA) with operating system access can easily manipulate this data. The attacker can replace genuine hardware metrics with fabricated values designed to mimic an honest node. This completely blinds baseline ML classifiers and causes catastrophic accuracy collapse. Therefore, a profound engineering need exists. We must design a decoupled, out-of-band predictive intelligence mechanism capable of detecting covert attacks. Crucially, this system must also proactively verify the integrity of the telemetry itself against spoofing. It must accomplish all this without degrading native enterprise transaction throughput [Pranto et al. \(2022\)](#); [Wang et al. \(2026\)](#).

1.3 Aims, Objectives and Motivation

1.3.1 Aims

The primary aim of this project is to conceptualize, design, and validate an AI-driven, Proactive Reputation-Based Consensus Mechanism (RBCM) for permissioned blockchains. This system will transition decentralized trust from reactive penalization to preemptive threat mitigation utilizing a completely decoupled, out-of-band machine learning sidecar. A secondary critical aim is to secure this intelligence layer using the Telemetry Attestation and Resilience Integration Module (TARIM). This multi-layer defense framework will systematically detect, quantify, and correct telemetry spoofing attacks.

1.3.2 Objectives

- **Architectural Design Defense Integration:** To design an asynchronous, decoupled machine learning sidecar equipped with a multi-layer resilience framework (TARIM) capable of neutralizing telemetry spoofing adversaries without stalling native ledger operations.
- **Threat Taxonomy Dataset Engineering:** To formulate a comprehensive 6-Class Byzantine attack taxonomy and construct a statistically rigorous, collinearity-free dataset capturing 12,000 epochs of covert adversarial behavior.
- **Predictive Modeling Interpretability:** To develop, optimize, and adversarially retrain a lightweight XGBoost classification matrix, ensuring resilient threat detection alongside complete heuristic transparency via SHAP methodologies.
- **Operational Feasibility Latency Benchmarking:** To empirically validate the system's viability by benchmarking global inference latency against strict 2000-millisecond consensus timeouts, ensuring proactive defense does not degrade transaction throughput.

1.3.3 Motivation

Enterprise consortiums demand absolute ledger immutability and high transaction throughput. Currently, organizations remain hesitant to adopt decentralized ledgers. Existing consensus protocols fail to detect coordinated, slow acting malicious behavior until after sensitive operational data is compromised. Furthermore, deploying naive machine learning models creates a false sense of security. Attackers can easily manipulate the input data to bypass standard anomaly detection.

This project is motivated by the critical necessity to bridge the gap between advanced artificial intelligence and distributed systems engineering. It aims to prove that highly complex predictive models can operate asynchronously alongside a blockchain peer utilizing minimal computational overhead. Moreover, it seeks to demonstrate that these models can be mathematically and structurally defended against sophisticated data poisoning. Solving this challenge creates a fundamentally

secure, meritocratic network capable of neutralizing Byzantine threats preemptively. It ensures data integrity without sacrificing the native speed of permissioned environments.

1.4 Project Specification

- **Consensus Latency Overhead Limit:** The integration of the machine learning inference sidecar, including all TARIM defense layers, must execute rapidly. The total inference overhead must strictly consume less than 1.0 percent of the default 2000 millisecond Raft block timeout threshold.
- **Global Classification Accuracy Target:** The optimized predictive models must achieve a minimum Global Accuracy limit of 90.00 percent across the 6-Class Byzantine Attack Taxonomy. This must hold true even when evaluating severely spoofed adversarial test matrices.
- **Covert Threat Detection Threshold:** The optimized predictive architecture must successfully isolate highly evasive temporal threats. It must achieve a minimum True Positive Recall Rate of 95.00 percent when detecting covert "Sleeper Decay" attacks under active telemetry spoofing.
- **False Positive Mitigation:** The defensive adversarial retraining must balance threat detection with operational precision. The system must maintain a False Positive Rate strictly below 5.0 percent for honest baseline nodes.
- **Feature Independence Constraint:** Following extreme structural dimensionality reduction, all raw input data variables must score a Variance Inflation Factor (VIF) strictly below 5.0. This mathematically proves the absolute absence of collinear signal pollution.
- **Architectural Decoupling Mandate:** The system must be engineered as a 100 percent out-of-band service. The core Hyperledger Fabric Raft consensus binaries must remain entirely unmodified. This preserves open-source upgrade dependencies and prevents non-deterministic network serialization failures.

Chapter 2

Literature Review

The body of research relevant to this thesis spans three interconnected themes: the limitations of reactive consensus protocols and the threat landscape they leave unaddressed, the trajectory of machine learning as a detection and trust-scoring primitive in blockchain systems, and the emerging class of predictive, reputation-driven consensus mechanisms that attempt to move beyond static heuristics. Each theme is reviewed in turn, concluding with the precise gap that the Telemetry Attestation and Resilience Integration Module (TARIM) addresses.

2.1 The Reactive Consensus Problem and Its Consequences

The foundational limitation driving this research is not a flaw in any single protocol; it is a design philosophy shared by the entire family of permissioned blockchain consensus mechanisms. Protocols such as Practical Byzantine Fault Tolerance (PBFT) and Hyperledger Fabric’s Raft authenticate participants at network entry and then operate under the baseline assumption of continued good faith. Punitive action is triggered only after a malicious event has already occurred. [Aslam et al. \(2021\)](#) provide a foundational taxonomy of this attack surface, cataloguing routing attacks, Sybil execution, and transaction malleability as the primary threat vectors in permissioned networks, and evaluating cryptographic countermeasures including Zero-Knowledge Proofs, Ring Signatures, and Mix Networks. Their analysis makes the core problem explicit: cryptographic identity checks at entry do not constrain the long-term behaviour of an authenticated node.

This reactive design has documented consequences across several deployment contexts. [Dua et al. \(2025\)](#) synthesize findings from 56 studies on cross-chain interoperability and report that 93 percent of published work analyzes fewer than four security properties simultaneously. Relay-based protocols show the broadest coverage, while HTLC and notary schemes expose critical vulnerabilities. The fragmentation of the research landscape means that no existing production framework evaluates a node’s trustworthiness continuously across the full lifecycle of its participation. In healthcare deployments, [Raghav & Bhola \(2023\)](#) illustrate the severity of this gap directly: Sybil attackers overwhelm permissionless networks by registering repeated identities, and their UUID-based continuous monitoring solution, which achieves a high true positive rate, is constrained to permissionless environments and cannot address the subtler, long-term covert attacks that characterize permissioned settings.

Two threat classes are particularly dangerous precisely because they are designed to evade reactive systems. In Sleeper Decay, an adversarial node behaves correctly for thousands of epochs to accumulate positive reputation before executing a high-value attack from a trusted leadership position. In Cartel Collusion, a coordinated subnetwork of nodes endorses each other’s invalid blocks, exploiting the fact that static threshold rules evaluate nodes individually and cannot map these graph-based relationships. [Rawlins et al. \(2024\)](#) treat the consensus trust problem as a distributed machine learning task, applying reinforcement Q-learning and Bayesian game theory to classify cooperative nodes and detect adversarial collusion subnetworks. Their system maintains ledger integrity under double-spending and Sybil scenarios, but their findings simultaneously confirm that detecting these threats requires moving beyond static formulas toward temporal and relational modelling. [Wang et al. \(2026\)](#) demonstrate the scalability cost of this reactive gap directly, finding that the rigid timeout mechanisms of standard BFT protocols cause structural throughput bottlenecks in volatile Internet of Vehicles environments, with their TimesNet-BFT framework recovering up to 191.9 percent of lost throughput by replacing reactive timeouts with predictive temporal modelling.

A consistent limitation across these architectures is that the trust model remains entirely post-hoc. No existing reactive protocol maintains a multi-dimensional chronological record of node behaviour capable of detecting the microscopic hardware perturbations that precede a Sleeper Decay attack, or the graph-level coordi-

nation signatures of an emerging cartel. This absence of proactive intelligence is the primary threat-landscape gap that this research addresses.

2.2 Machine Learning as a Blockchain Security Primitive

The integration of machine learning into blockchain security has progressed rapidly, but the trajectory of that research reveals both its potential and its current limitations. [Azad et al. \(2025\)](#) survey this trajectory from 2018 to 2025, categorizing how graph-based, sequential, and temporal learning paradigms are applied to transaction networks and smart contract analysis. Their central finding is that machine learning unlocks detection capability that rule-based systems cannot replicate, particularly for e-crime, money laundering, and phishing. They also identify the persistent structural challenges: label scarcity, high computational demand, and train-test distribution mismatch caused by evolving network conditions. These constraints define the engineering bounds within which any practical ML-assisted consensus system must operate.

Within network intrusion and DDoS detection, the most relevant approaches combine multiple learning paradigms. [Mishra et al. \(2026\)](#) propose FRF-BiLSTM, a federated framework pairing recursive feature elimination through Random Forests with BiLSTM sequence analysis at local nodes. Blockchain integration secures the federated model updates and maintains an immutable audit trail. [Krishnapriya & Singh \(2025\)](#) couple CNN-based feature extraction with Random Forest classification to mitigate DDoS attacks in IoT environments, demonstrating that hybrid architectures consistently outperform single-model baselines. [Tariq et al. \(2023\)](#) provide the broader IoT security context, identifying anomaly detection and access control as the two areas where AI and ML have had the most measurable impact.

For on-chain activity specifically, [Shawn \(2025\)](#) evaluates multiple ML frameworks for detecting illicit transactions and finds that Graph Neural Networks equipped with graph-temporal features significantly outperform both supervised classifiers and unsupervised detectors. Features capturing multi-hop transaction depth and temporal burst intensity prove to be the strongest predictors of suspicious behaviour. [Zhou et al. \(2024\)](#) generalize this finding with KnowGraph, a

hybrid framework fusing data-driven GNNs with knowledge-enabled logical reasoning via Markov Logic Networks and weight noise ensembling. By embedding first-order logic formulas directly into the learning model, KnowGraph mitigates out-of-distribution shifts in complex settings including collusion detection and network intrusion. These works establish that relational and temporal feature engineering is not an optimization but a prerequisite for detecting coordinated, covert attacks.

Complementary evidence comes from adjacent trust modelling problems where the stakes of ML failure are high. [Zong et al. \(2023\)](#) design ReIPS for IoT-enabled power stations, merging Backpropagation Neural Networks with a Point Reputation-Weighted Directed Network Model and semantic knowledge graphs to filter false ratings and identify collusion clusters. [Ramaswamy & Gandhi \(2022\)](#) propose a lightweight trust model for Wireless Body Area Networks that synthesizes node behavioural trust with data confidence measured by age and consistency, successfully isolating malicious nodes executing sleeper and message suppression attacks. [Pranto et al. \(2022\)](#) address the e-commerce fraud detection problem through a privacy-preserving blockchain and smart contract ecosystem, finding that algorithm choice cannot be separated from deployment architecture: passive-aggressive classifiers excel in their incremental off-chain learning environment precisely because the system was designed around them.

Across this body of work, a shared architectural assumption goes unexamined: that the telemetry fed to the machine learning model is trustworthy. Every framework reviewed here assumes the honesty of the input pipeline. None constructs a threat model for an adversary who targets the ML inputs themselves before any classifier has an opportunity to inspect them. This gap is the direct motivation for the TARIM defense architecture.

2.3 Reputation-Based and Predictive Consensus Mechanisms

A distinct body of literature has moved beyond anomaly detection toward integrating machine learning directly into the consensus leader election and node validation pipeline. The shared premise across this work is that a node’s right to participate

in consensus should be a function of its predicted future behaviour, not merely its authenticated identity.

Windiatmaja et al. (2023) and Jauzak Hussaini Windiatmaja (2023) propose Proof of Intelligent Reputation (PoIR), which replaces static formulas with a Bidirectional LSTM regression model paired with the Network Entity Reputation Database (NERD). By ingesting diverse behavioural data to build node profiles and estimate expected actions, PoIR completely nullifies monopoly attacks that otherwise control up to 58 percent of transactions in Proof of Stake systems. This work is the most direct precedent for the predictive consensus paradigm, proving the viability of intelligence-driven node selection. Saadat et al. (2023) provide complementary validation with a simpler Artificial Neural Network trained on three parameters: Age of Node, Stability of Node, and Success Rate of Block Proposals. Their framework demonstrates that even a narrow telemetry feature set can be converted into actionable trust predictions.

Xu et al. (2026) bring this approach to the PBFT family with RF-PBFT, integrating a Random Forest predictive model with a dynamic reputation mechanism to partition nodes into consensus and candidate sets, resolving the high communication overhead and random master node selection that limit standard PBFT in multi-node environments. Li et al. (2024) pursue a related strategy with K-DPBFT, applying improved K-means++ and DBSCAN clustering to evaluate node behaviour and identify anomalous or colluding groups among heterogeneous participants. The use of unsupervised clustering is particularly relevant to cartel detection, where ground-truth labels are unavailable at inference time.

Two further works address the collusion problem through probabilistic and hierarchical modelling. Chen & Guan (2025) introduce a Hierarchical Reputation Consensus Model that uses Bayesian inference to evaluate message authenticity and specifically mitigate collusion. Chatziamanetoglou & Rantos (2024) present a Proof-of-Quality mechanism that relies on ML clustering to isolate malicious nodes and detect outliers in validator performance, providing a theoretical foundation for identifying sleeper nodes or coordinated attacks. Adaptive consensus optimization extends this paradigm further: Gutierrez et al. (2025) integrate Proximal Policy Optimization with dynamic graph representation to enable blockchains to adjust validation operations in real time, while Ye & Wang (2025) propose ACCR-DR, a state-machine-based dynamic role transition protocol that shortens high-

performance recovery time by 60 percent without the communication overhead of falling back to global consensus. [Wu et al. \(2025\)](#) complete this picture with BFTBrain, a reinforcement learning system that models protocol selection as a contextual multi-armed bandit problem, outperforming the best fixed protocols by 18 to 119 percent.

Despite this progress, a critical engineering gap persists across every work reviewed in this section. All existing reputation-based and predictive consensus mechanisms operate on the implicit assumption that the node telemetry they ingest reflects ground truth. [Davor \(2025\)](#) demonstrates in a DeFi context that coupling dynamic knowledge graphs with LSTM networks produces early warning signals that outperform traditional reactive volatility metrics, and [Cui \(2025\)](#) shows that embedding off-chain verifiable local updates with reputation-based trust scoring can counter data poisoning in federated vehicular systems. Yet neither work, nor any other in the reviewed literature, confronts the specific scenario where a Byzantine node with operating-system access manipulates the hardware telemetry it reports to the consensus monitor before the classifier ever sees it. A sophisticated Telemetry Spoofing Adversary with root-level OS access can intercept self-reported CPU and memory metrics and replace them with fabricated values drawn from distributions that mimic honest behaviour. Under this attack, an ML classifier trained on genuine telemetry will certify the attacker as honest. The empirical impact of this scenario on baseline classification accuracy, and the multi-layer defense architecture required to counter it without violating enterprise consensus timeouts, constitutes the specific research gap that this thesis addresses.

Chapter 3

Methodology

3.1 Overview

The foundational architecture of permissioned blockchains relies predominantly on crash fault-tolerant algorithms like Raft. Hyperledger Fabric is a prime example. These systems operate under the baseline assumption that all authenticated participants act in good faith. However, Byzantine nodes can internalize malicious intent. When this occurs, traditional countermeasures remain profoundly reactive. They are designed to punish nodes only after fraudulent headers are proposed or cryptographic compromises have already occurred. To resolve this design problem, the methodology required generating a system capable of proactive threat mitigation. This is achieved by analyzing pre-crime topological telemetry.

This chapter presents the complete research methodology employed to design, implement, and evaluate the Telemetry Attestation and Resilience Integration Module (TARIM). The methodology addresses a critical gap in current literature. There is an absence of rigorous attack-surface analysis for ML-assisted consensus mechanisms. The baseline XGBoost classifier developed in Phase 1 demonstrated that machine learning can detect Byzantine threats in permissioned blockchains. However, it relied on self-reported telemetry. A sophisticated adversary can easily manipulate this data.

TARIM provides a principled, multi-layer defense framework. It detects, quantifies, and corrects telemetry spoofing attacks. Crucially, it accomplishes this without violating the strict 2000 millisecond Raft consensus Service Level Agreement (SLA). The methodology follows a structured adversarial robustness evaluation pipeline.

First, a realistic threat model is defined based on the Telemetry Spoofing Adversary (TSA). Second, a three-layer defense architecture is designed to counter the identified threats. Third, an ablation study is conducted to measure the incremental contribution of each defense layer. Finally, SHAP-based explainability analysis is performed. This validates that the model decisions are grounded in meaningful defense signals rather than statistical artifacts.

3.2 Architectural Alternatives and Concept Reduction

During the initial design phase, multiple conceptual solutions were generated. The goal was to transition the network from a reactive state to a proactive trust paradigm. These alternatives focused on how best to capture and analyze multivariate time series data. Specifically, they targeted hardware stress signals like CPU and memory usage alongside transactional states. These states included missed validations and signature failures.

The generated concepts evaluated the following architectural alternatives for proactive Byzantine fault mitigation:

3.2.1 In-Band Reactive Thresholding (Heuristic Smart Contracts)

Mechanism: This concept builds directly upon standard baseline models. It utilizes traditional Hyperledger Fabric chaincode to enforce static mathematical penalties.

Operation: It relies on hard-coded thresholding rules like a Reliability Density Function. If a node fails a validation or exceeds a strict 2.0 second SLA timeout, the chaincode triggers an immediate reputation slash.

Scope: This is strictly limited to application-layer metrics natively available to the blockchain state.

3.2.2 Native On-Chain Machine Learning

Mechanism: This approach involves embedding lightweight neural network weights and machine learning logic directly into the distributed ledger's core consensus engine.

Operation: As blocks are formed, the native orderer evaluates node telemetry

synchronously during the block creation sequence. It attempts to embed predictive intelligence natively within the peer runtime environment. This forces the blockchain itself to compute adversarial probabilities before committing a block.

Scope: It fuses both hardware telemetry and application events. However, it processes them within the highly restricted and sequential pipeline of the blockchain peer.

3.2.3 Out-of-Band Predictive AI Sidecar

Mechanism: This concept proposes deploying a fully decoupled and asynchronous artificial intelligence inference module. This module runs parallel to the active peer.

Operation: The sidecar operates independently. It ingests continuous operating system layer hardware telemetry alongside granular peer endorsement outcomes via restricted local RPC ports. It evaluates historic sequences spanning up to 12,000 continuous block epochs. This allows the system to detect microscopic perturbations indicating covert attacks. Examples include sleeper decay or cartel collusion. It then forwards a normalized confidence coefficient back to the active Reputation Engine.

Scope: It is completely agnostic to the host operating system limits or core compiled language of the blockchain. It processes dense multidimensional tensors entirely asynchronously.

A rigorous decision-making process evaluated the proposed conceptual alternatives. The goal was to isolate a single optimal architecture for implementation. The evaluation criteria centered on three primary engineering constraints. First, the solution had to strictly adhere to the 2.0 second SLA verification window. Second, the system required the capability to process highly complex multidimensional data matrices without introducing network serialization risks. Third, the chosen architecture had to successfully analyze pre-crime topological telemetry without breaking existing open-source upgrade dependencies.

In-Band Reactive Thresholding was definitively rejected. This traditional methodology remains profoundly reactive. It completely fails to provide the proactive security required to mitigate sophisticated threats like sleeper decay. Native On-Chain Machine Learning was similarly rejected due to severe operational hazards. Integrating highly complex artificial intelligence models directly into the core consensus engine presents profound systemic risks. Attempting to force the blockchain

peer to compute adversarial probabilities synchronously would inevitably block the ledger.

Figure 3.1 illustrates the architectural flow of the implemented system. The Out-of-Band Predictive AI Sidecar was selected as the optimal conceptual solution. By operating directly alongside the target Hyperledger Fabric installation, the artificial intelligence inference module exists strictly out-of-band. To resolve the distributed oracle problem, the sidecar is deployed locally by the endorsing peers. This architecture provides proactive threat detection while completely satisfying the strict performance constraints of the enterprise ledger.

3.3 Dataset Description

3.3.1 Data Source and Collection

The experimental dataset was generated through controlled simulations of a Hyperledger Fabric v2.x network. The dataset captures telemetry readings from blockchain peer nodes operating under six distinct behavioral profiles. Each record represents one telemetry observation from a specific peer node during a specific consensus epoch.

The dataset contains 12,000 records collected from a single organization's endorsing peer (peer0.org1). This exclusive filtering strategy ensures that the ML model is trained solely on first-party telemetry. Cross-organizational data would introduce jurisdictional and trust boundary complications in a real-world deployment. A second peer (peer0.org2) exists within the network architecture. It serves only as the external observer for Layer 2 cross-peer behavioral triangulation.

3.3.2 Feature Space

The raw feature space consists of five telemetry attributes collected at each epoch. These are summarized in Table 3.1.

During feature selection in Phase 1, a Variance Inflation Factor (VIF) analysis was conducted. The `missed_validations` feature exhibited a VIF exceeding 2,500. This indicated severe multicollinearity with other features. Consequently, it was excluded from the baseline model. This reduced the effective feature set to four features. These are `cpu_final`, `mem_final`, `avg_latency_ms`, and `invalid_signatures`.

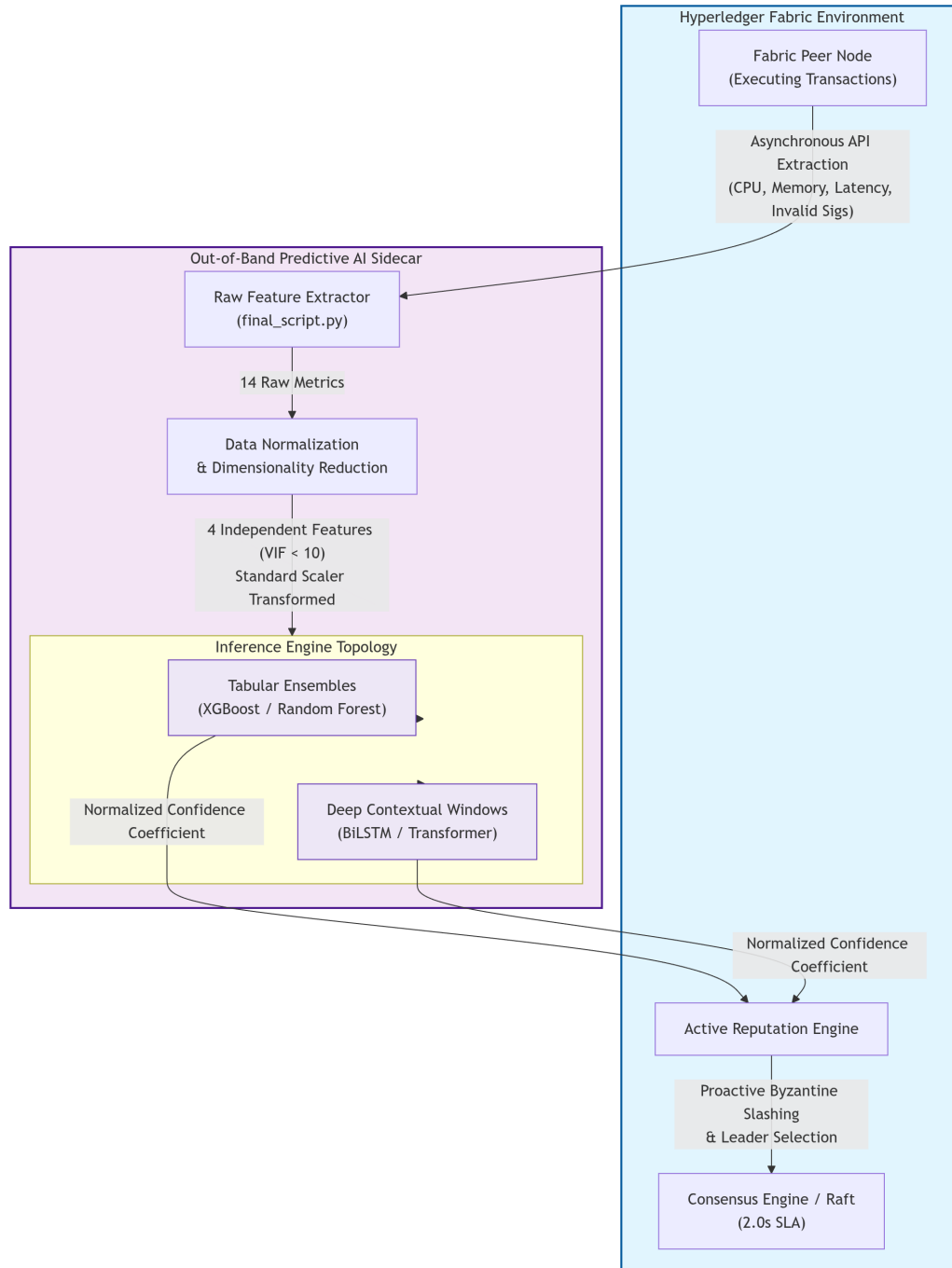


Figure 3.1: System Architecture of the Proactive AI Sidecar.

Table 3.1: Raw Telemetry Features

Feature	Type	Unit	Description
cpu_usage_pct	Continuous	%	Self-reported CPU utilization of the peer node
memory_usage_mb	Continuous	MB	Self-reported memory consumption
avg_latency_ms	Continuous	ms	Average transaction endorsement latency
invalid_signatures	Discrete	Count	Number of invalid cryptographic signatures detected
missed_validations	Discrete	Count	Number of missed block validation events

The TARIM extension retains this four-feature baseline and augments it with up to 16 additional defense-derived features.

3.3.3 Target Variable and Class Distribution

The target variable encodes six mutually exclusive behavioral classes. Each class corresponds to a distinct Byzantine attack strategy or the honest baseline. These are detailed in Table 3.2.

Table 3.2: Byzantine Attack Class Taxonomy

Class	Label	Description	Hardware Dependent
0	Honest Baseline	Normal peer operation with no adversarial behavior	No
1	Cryptographic Spoofing	Node submits transactions with forged or replayed cryptographic signatures	Yes
2	Sybil Execution	Node creates virtual identities to amplify its influence in consensus	Yes
3	Sustained Decay	Node gradually degrades its performance to avoid detection thresholds	Yes
4	Network Isolation	Node selectively drops or delays network packets to disrupt consensus	No
5	Cartel Collusion	Multiple nodes coordinate to manipulate consensus outcomes	Yes

The dataset maintains a perfectly balanced class distribution. There are exactly 2,000 records per class, totaling 12,000 observations. This balanced distribution eliminates the need for resampling techniques. It ensures that classification metrics are not biased by class imbalance. The Hardware Dependent column indicates whether detection relies primarily on CPU and memory telemetry. Classes 1, 2, 3, and 5 are hardware-dependent. Their behavioral signatures manifest through abnormal CPU and memory patterns. Classes 0 and 4 are characterized by network-level signals that are not directly spoofable through OS telemetry manipulation.

3.3.4 Data Splitting Strategy

A chronological intra-class splitting strategy is employed to prevent temporal data leakage. Random stratified splitting can allow future observations to inform past predictions. This chronological approach preserves the temporal ordering within each attack phase.

The dataset is organized into six sequential 1,000-epoch phases. Within each phase, the first 80 percent of epochs are assigned to the training set. The remaining 20 percent are assigned to the test set. This configuration is detailed in Table 3.3.

Table 3.3: Data Split Configuration

Split	Ratio	Samples	Purpose
Training	80%	9,600	Model fitting and hyperparameter tuning
Test	20%	2,400	Final evaluation (400 per class)

This chronological split matches the splitting strategy used in the Phase 1 baseline. It ensures that the baseline accuracy of 97.25 percent is exactly reproducible. The random seed is fixed at 42 throughout all experiments.

3.4 Threat Model: Telemetry Spoofing Adversary (TSA)

3.4.1 Adversary Capabilities

The Telemetry Spoofing Adversary (TSA) represents a sophisticated Byzantine attacker. This attacker has compromised one or more peer nodes within the network. The TSA operates under specific capability assumptions.

The adversary has root-level access to the compromised peer node’s operating system. The adversary can intercept and modify the self-reported CPU and memory telemetry before it reaches the consensus-monitoring ML model. The adversary replaces genuine hardware metrics with fabricated values designed to mimic an honest node’s profile. Finally, the adversary is completely aware that an ML-based anomaly detection system is monitoring telemetry.

However, the adversary is restricted by several structural boundaries. The adversary cannot modify network-observable properties such as transaction endorsement latency or cryptographic signature validity. Independent peers verify these features. The adversary cannot tamper with hardware Performance Monitoring Unit (PMU) registers if an SGX enclave is present. Intel SGX provides a hardware-isolated execution environment. The adversary also cannot control what other peers observe about the compromised node’s network behavior.

3.4.2 Spoofing Mechanism

The TSA applies a narrow Gaussian spoofing strategy to the hardware-dependent classes. The spoofed values are drawn from distributions designed to mimic the statistical profile of Honest Baseline nodes. The exact spoofing parameters are specified in Table 3.4.

Table 3.4: TSA Spoofing Parameters

Feature	Distribution	Parameters	Clipping Range
cpu_usage_pct	Gaussian	$\mu = 2.5\%, \sigma = 0.8\%$	[0.1%, 5.0%]
memory_usage_mb	Gaussian	$\mu = 62.0 \text{ MB}, \sigma = 4.0 \text{ MB}$	[50.0 MB, 80.0 MB]

Under this spoofing regime, 66.7 percent of the test set has its CPU and memory values replaced with fabricated data. Classes 0 and 4 remain unaffected. Their distinguishing features are not CPU or memory-based.

3.4.3 Impact Quantification

The impact of the TSA on the undefended baseline XGBoost classifier is devastating. Without any defense mechanism, the model’s overall accuracy drops from 97.25 percent to 33.33 percent. Table 5.3 shows the per-class impact.

Table 3.5: TSA Impact on Per-Class Recall

Class	Baseline Recall	Under TSA Recall	Delta
0 (Honest)	100.0%	100.0%	0.0%
1 (Crypto Spoofing)	100.0%	0.0%	-100.0%
2 (Sybil Execution)	100.0%	0.0%	-100.0%
3 (Sustained Decay)	83.5%	0.0%	-83.5%
4 (Network Isolation)	100.0%	100.0%	0.0%
5 (Cartel Collusion)	100.0%	0.0%	-100.0%

As shown in Figure 3.2, the confusion matrix for the undefended model under TSA reveals a complete collapse. All four spoofed classes are misclassified as the Honest baseline. The spoofed CPU and memory values are statistically indistinguishable from honest node telemetry. This confirms that the TSA successfully achieves its objective of blinding the ML classifier.

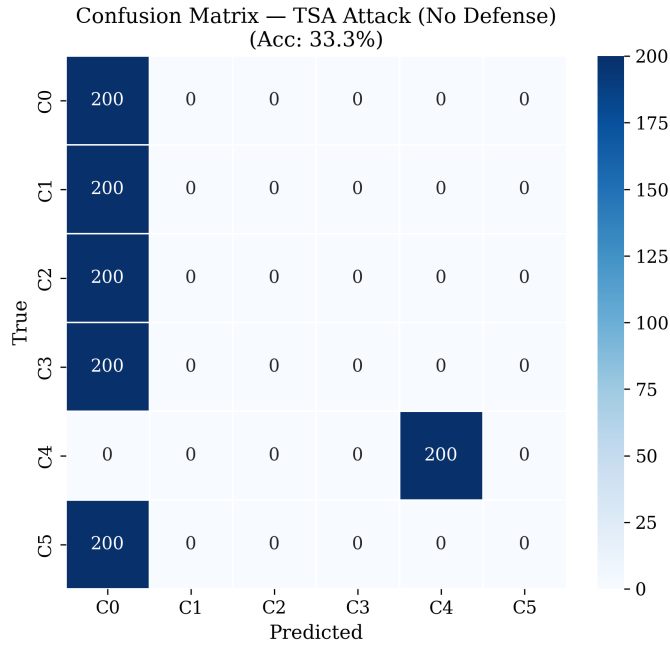


Figure 3.2: Confusion Matrix Under TSA Attack (No Defense).

3.5 TARIM Architecture

3.5.1 Design Philosophy

The TARIM defense stack is designed around a principle of defense-in-depth. Rather than relying on a single defense mechanism, TARIM employs three independent layers. These operate on orthogonal signal domains. Each layer introduces features that are progressively harder for the adversary to spoof. The architecture follows an explicit dependency ordering. All layers evaluate raw falsified data first. Detection flags are computed independently. Correction is applied only after all detection signals have been gathered. This guarantees that each layer's detection signals are computed on the exact falsified data the attacker submitted.

3.5.2 Layer 1: Hardware-Anchored Telemetry (HAT)

Layer 1 leverages Intel Software Guard Extensions (SGX) to establish a hardware root of trust. In a production deployment, a lightweight SGX enclave executes on each peer node. It reads CPU and memory utilization directly from the processor's PMU registers. These hardware-derived readings are cryptographically signed within the enclave and submitted alongside the self-reported telemetry.

Table 3.6: Layer 1 (HAT) Features

Feature	Formula	Description
cpu_hw_delta	$\text{abs}(\text{reported_cpu} - \text{hw_cpu})$	Absolute discrepancy between reported and hardware CPU
mem_hw_delta	$\text{abs}(\text{reported_mem} - \text{hw_mem})$	Absolute discrepancy for memory
layer1_flag	1 if $\text{cpu_hw_delta} \geq 15.0$ else 0	Binary spoofing alert flag

In the simulation environment, the SGX enclave is modeled by comparing the preserved true telemetry values against the reported values. Gaussian noise is added to the true values to model the measurement imprecision inherent in actual PMU sampling. Layer 1 produces three specific features, detailed in Table 3.6.

A discrepancy threshold of 15 percentage points avoids false positives from legitimate PMU measurement noise. Simultaneously, it easily detects the substantial deviations introduced by TSA spoofing.

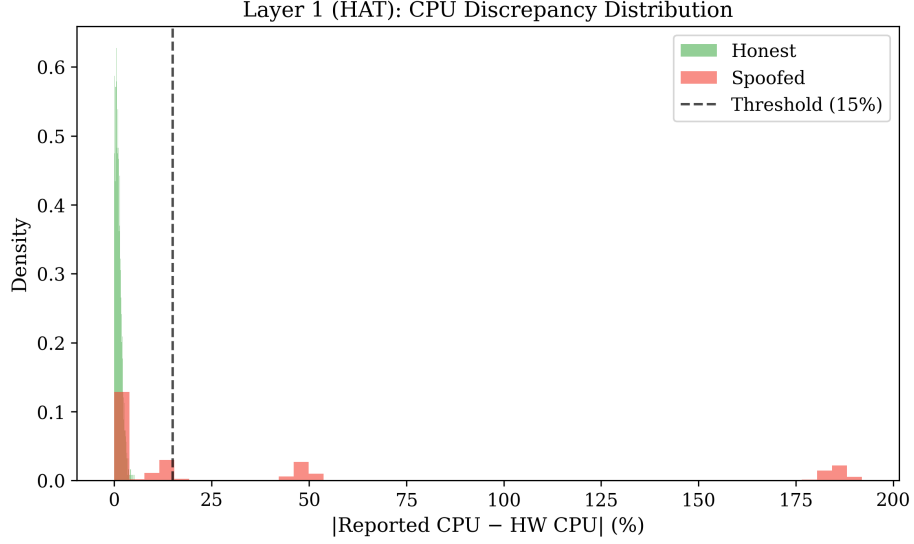


Figure 3.3: Layer 1 CPU Hardware Delta Distribution.

Figure 3.3 illustrates the distribution of hardware delta values. When spoofing is detected, Layer 1 substitutes the reported values with the hardware-derived measurements. This ensures downstream classification operates on trustworthy input data. The estimated latency overhead for SGX attestation is 1.2 milliseconds per inference. This constitutes merely 0.06 percent of the 2,000 millisecond consensus timeout.

3.5.3 Layer 2: Cross-Peer Behavioral Triangulation (CPBT)

Layer 2 exploits a fundamental asymmetry in adversarial capabilities. A Byzantine node can control what it reports about itself. However, it cannot control what other independent peers observe externally. CPBT uses an external peer to measure the TCP round-trip time (RTT) to the monitored node. It compares this observed network delay against the expected RTT predicted by a physics-informed model.

The relationship between CPU utilization and network response time is modeled using a piecewise function. At low CPU utilization, the RTT is approximately constant. As CPU utilization increases beyond 50 percent, kernel scheduling contention causes exponentially increasing network latency. This piecewise approach prevents false positives that occur in standard linear regression models.

The RTT-CPU Consistency Score (RCCS) quantifies the discrepancy between

Table 3.7: Layer 2 (CPBT) Features

Feature	Description
observed_rtt_ms	TCP round-trip time measured by the external peer
rccs	RTT-CPU Consistency Score (measured in sigma units)
rtt_cpu_delta	Signed difference between predicted RTT and observed RTT
layer2_flag	Binary alarm triggered if RCCS exceeds 2.0 sigma

the predicted RTT and the observed RTT. Layer 2 produces four specific features, outlined in Table 3.7.

3.5.4 Layer 3: Statistical Telemetry Integrity Verification (STIV)

Layer 3 applies three complementary statistical methods to detect anomalous patterns. Layers 1 and 2 operate on instantaneous measurements. In contrast, Layer 3 captures drift, entropy, and distributional anomalies. These behaviors accumulate over time.

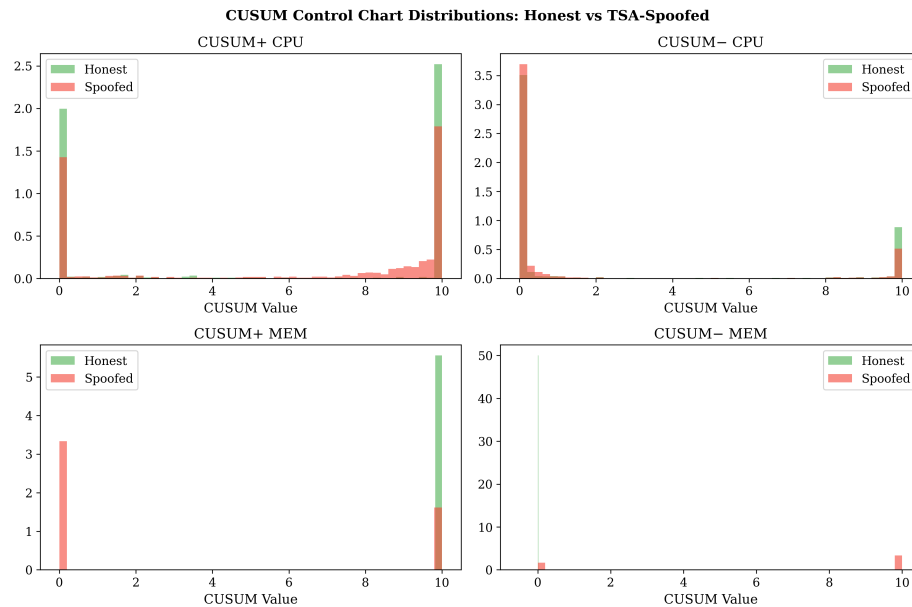


Figure 3.4: CUSUM Distributions.

The Cumulative Sum (CUSUM) algorithm monitors telemetry streams for sus-

tained upward or downward shifts in the signal mean. It produces four bounding features 3.4. Temporal gradients capture the exact rate of change between consecutive epochs. Shannon entropy evaluates the CPU distribution within a sliding 5-epoch window. Honest signals exhibit high natural variability. Spoofed signals drawn from narrow distributions exhibit anomalously low entropy.

The Cross-Feature Consistency Score (CFCS) utilizes a Gaussian Mixture Model to evaluate the joint distribution of all available features. Finally, a meta-classifier acts as a binary XGBoost model. It aggregates all defense features into a single spoofing probability score. The complete feature output of Layer 3 is detailed in Table 3.8.

Table 3.8: Layer 3 (STIV) Features

Feature	Source	Description
cusum_pos_cpu	CUSUM	Positive CUSUM accumulator for CPU
cusum_neg_cpu	CUSUM	Negative CUSUM accumulator for CPU
cusum_pos_mem	CUSUM	Positive CUSUM accumulator for memory
cusum_neg_mem	CUSUM	Negative CUSUM accumulator for memory
cpu_temporal_grad	Temporal	Rate of change in CPU between epochs
mem_temporal_grad	Temporal	Rate of change in memory between epochs
cpu_entropy_5w	Temporal	Shannon entropy of CPU in 5-epoch window
cfcs_log_likelihood	CFCS/GMM	Log-likelihood under learned distribution
meta_spoofing_prob	XGBoost	Aggregated spoofing probability

3.5.5 Complete Feature Vector

The full TARIM consists of 20 features. This includes 4 base features, 3 from Layer 1, 4 from Layer 2, and 9 from Layer 3. Table 3.9 provides the complete specification. Critically, 16 out of 20 features in the TARIM vector are strictly unspoofable by the TSA. The remaining two spoofable features are securely corrected by Layer 1 prior to ML ingestion.

Table 3.9: Complete TARIM Feature Vector (20 Features)

#	Feature	Layer	Spoofable	Domain
1	cpu_final	Base	Corrected by L1	Resource
2	mem_final	Base	Corrected by L1	Resource
3	avg_latency_ms	Base	No	Network
4	invalid_signatures	Base	No	Cryptographic
5	cpu_hw_delta	L1 (HAT)	No	Hardware
6	mem_hw_delta	L1 (HAT)	No	Hardware
7	layer1_flag	L1 (HAT)	No	Hardware
8	observed_rtt_ms	L2 (CPBT)	No	Network
9	rccs	L2 (CPBT)	No	Network
10	rtt_cpu_delta	L2 (CPBT)	No	Network
11	layer2_flag	L2 (CPBT)	No	Network
12	cusum_pos_cpu	L3 (STIV)	No	Statistical
13	cusum_neg_cpu	L3 (STIV)	No	Statistical
14	cusum_pos_mem	L3 (STIV)	No	Statistical
15	cusum_neg_mem	L3 (STIV)	No	Statistical
16	cpu_temporal_grad	L3 (STIV)	No	Statistical
17	mem_temporal_grad	L3 (STIV)	No	Statistical
18	cpu_entropy_5w	L3 (STIV)	No	Statistical
19	cfcs_log_likelihood	L3 (STIV)	No	Statistical
20	meta_spoofing_prob	L3 (STIV)	No	Ensemble

3.6 Experimental Design

3.6.1 Ablation Study Configuration

The evaluation follows an ablation study methodology. Defense layers are progressively added to accurately measure their incremental contribution. Five explicit configurations are evaluated, as detailed in Table 3.10.

Table 3.10: Ablation Study Configurations

Config	Layers	Features	Train Data	Test Data	Purpose
Baseline	None	4	Clean	Clean	Phase 1 reproduction
Under Attack	None	4	Clean	Spoofed	Quantify TSA devastation
TARIM L1	L1	7	Augmented	Spoofed	Isolate hardware attestation
TARIM L1+L2	L1, L2	11	Augmented	Spoofed	Add triangulation
Full TARIM	L1, L2, L3	20	Adversarial	Spoofed	Complete defense testing

3.6.2 Adversarial Training Strategy

The Full TARIM configuration employs a three-component adversarial training strategy. This prevents the model from blindly memorizing specific spoofing distributions. Five variance levels are generated for each attack-class training sample. This creates 12,800 additional training samples representing a wide spectrum of adversarial capabilities. Furthermore, 640 additional samples are generated by perturbing Honest training samples near the decision boundary. This forces the model to learn tighter decision thresholds. The final adversarial training set consists of exactly 17,280 training samples.

3.6.3 Hyperparameter Optimization

Hyperparameter tuning is performed using Optuna. This Bayesian optimization framework operates with 20 trials per configuration. The specific search space parameters are detailed in Table 3.11.

Table 3.11: Optuna Hyperparameter Search Space

Parameter	Range	Scale
n_estimators	[100, 1000]	Linear
max_depth	[3, 10]	Linear
learning_rate	[0.01, 0.3]	Log
subsample	[0.5, 1.0]	Linear
colsample_bytree	[0.5, 1.0]	Linear
gamma	[0, 5]	Linear

3.6.4 Evaluation Metrics and Explainability

Configurations are evaluated based on global accuracy, weighted F1-score, per-class recall, and false positive rates. Inference latency and SLA compliance metrics ensure strict adherence to operational bounds. SHAP (SHapley Additive exPlanations) provides post-hoc interpretability. SHAP values quantify the marginal contribution of each feature to individual predictions. This analysis validates global feature importance and reveals explicit decision boundary shifts.

3.7 Implementation Environment

All experiments are implemented utilizing Python 3.12. The stack requires XGBoost 2.x for gradient-boosted classification, Optuna 3.x for hyperparameter optimization, and SHAP 0.x for model explainability. Data manipulation utilizes standard scikit-learn, NumPy, and Pandas libraries. Visualizations are rendered via Matplotlib and Seaborn. The random seed is permanently fixed at 42 across all stochastic operations to guarantee absolute reproducibility. All evaluations execute securely on a single CPU machine. This perfectly reflects the lightweight deployment requirements of the TARIM sidecar architecture.

Chapter 4

Design and Development

4.1 Overview

This chapter presents the detailed system design and implementation of the Telemetry Attestation and Resilience Integration Module (TARIM). The chapter covers the software architecture and the internal design of each defense layer. It also details the data pipeline for training and evaluation. Finally, it outlines the integration strategy that allows TARIM to operate as a lightweight sidecar alongside the existing XGBoost consensus monitor. Every design decision is traced back to a specific threat or correctness requirement. The rationale for each parameter choice is rigorously documented.

4.2 System Architecture and Sidecar Routing

Integrating complex artificial intelligence directly into a consensus engine presents profound risks. These include network serialization, non-deterministic outputs, and fatal cascading errors during validation sequences. Therefore, the implementation embraces a completely decoupled sidecar design paradigm. The artificial intelligence inference module exists strictly out-of-band. It operates directly alongside the target Hyperledger Fabric installation.

4.2.1 Sidecar Deployment Model

TARIM is designed as a sidecar module that augments the existing Robust Byzantine Consensus Monitor (RBCM) engine. The RBCM engine operates as a Flask-

based REST API. It receives telemetry from Hyperledger Fabric peer nodes and returns classification predictions. TARIM integrates into this pipeline as a feature-engineering preprocessor. It intercepts raw telemetry and enriches it with defense-derived features. It then passes the augmented feature vector to the XGBoost classifier.

This sidecar architecture offers several advantages. First, it does not modify the core Hyperledger Fabric source code. This makes it deployable as an add-on module. Second, it preserves backward compatibility. When TARIM layers are disabled, the system reduces to the original four-feature baseline model. Third, it allows individual defense layers to be enabled or disabled through configuration. This enables operators to trade off between security depth and computational overhead based on deployment requirements.

4.2.2 Module Structure

The TARIM codebase is organized into a modular directory structure. The root directory contains master configuration files and the main Flask server. Separate subdirectories isolate the logic for each defense layer. For example, Layer 1 contains the SGX enclave simulation and PMU reader interfaces. Layer 2 houses the cross-peer RTT analysis and physics models. Layer 3 contains the CUSUM control charts, temporal feature extractors, and the binary meta-classifier. Additional directories manage the TSA simulation, experiment execution scripts, trained models, and publication figure generation.

4.2.3 Stack Orchestrator

The `TARIMStack` class serves as the central orchestrator. It coordinates all three defense layers. It manages the correct dependency ordering during both the fitting and transformation phases.

The fitting process follows a strict five-step dependency chain. First, the Layer 2 baseline sigma is computed from raw honest training data. This establishes the RTT standard deviation used for normalization. Second, the honest baseline is enriched with synthesized RTT values using the fitted Layer 2 model. Third, Layer 3 CUSUM and Gaussian Mixture Models (GMM) are fitted on this RTT-enriched data. Fourth, the meta-classifier training data is routed through all layers to generate realized

defense features. Fifth, the meta-classifier is trained on these realized features. This ordered chain prevents a critical correctness issue. It ensures the meta-classifier learns accurate spoofing signatures based on properly normalized inputs.

The transformation pipeline processes incoming data through a strict "evaluate RAW, correct LAST" sequence. All three layers evaluate the raw falsified telemetry independently first. Detection flags are computed based on this raw data. A spoofing detection mask is then constructed. Correction is applied only if Layer 1 is active and spoofing has been detected. This replaces reported metrics with hardware-derived metrics. The final 20-feature TARIM vector is then assembled. If correction were applied prematurely, downstream layers would see already-corrected values. They would subsequently fail to detect the original discrepancy.

A critical design constraint requires a fresh TARIMStack instance for each data split. The CUSUM control chart maintains internal state accumulators that persist across calls. Reusing the same stack instance across training and test transforms would cause severe data leakage. A factory function enforces this constraint by constructing a new stack instance for every call.

4.3 Raw Feature Extraction and Dataset Generation

Real-time node behaviors were quantified continuously across thousands of iterative ledger epochs. The pipeline fundamentally merges continuous Docker operating system layer hardware metrics with granular peer endorsement outcomes. It utilizes a dedicated extraction module for this purpose.

The module issues simulated continuous transaction arrays during load generation. It simultaneously queries physical container allocations. These include CPU usage percentage and memory usage in megabytes. It also extracts application events. These events include successful validations, missed validations, invalid signatures, and average latency in milliseconds. The script additionally generates operational tracking constants.

Enterprise permissioned ledgers operate privately. Consequently, there exists an absolute vacuum of available open-source datasets accurately mapping hardware telemetry against sophisticated Byzantine attack strategies. A custom experimental engine was specifically engineered to synthetically bridge this dataset deficiency.

This core script establishes asynchronous communication interfaces with the underlying Fabric SDK. It simulates sustained asset transfer loads characteristic of financial or supply-chain blockchain traffic. Data aggregation spans over continuous block epochs for the target evaluation node alongside a baseline honest execution counterpart. A rigorous runtime orchestration generates a final dataset consisting of exactly 12,000 structured observations.

Byzantine Fault Taxonomy and Attack Simulation Framework

The empirical system executed a rigorous simulation gauntlet to test the machine learning model bounds. The target node was forcefully manipulated into manifesting six categorically distinct behavioral states. Each state lasted a sustained duration of 1,000 epochs to provide profound statistical representation.

Phase 0 (Baseline): Uninterrupted and mathematically pure honesty. The node operates strictly beneath the 2.0 second SLA threshold. It retains minimal CPU profiles and consistently endorses 100 percent of routed events.

Phase 1 (Cryptographic Spoofing): The simulator forcibly injects false identity signatures generated by compromised certificate authorities. This adversarial pivot immediately manifests via the rapid saturation of the invalid signature error rate.

Phase 2 (Sybil Execution): Multiple pseudo-nodes are initiated locally to simulate malicious network flooding. This aims to outvote honest quorum majorities. This behavior triggers distinct computational exhaustion. It registers violent upward trajectories in CPU limits synced with moderate escalation in processing duration.

Phase 3 (Sustained Decay): The internal computational response loop is artificially throttled incrementally to simulate stealth sleeper agents. Processing times gracefully stretch beyond the 2.0 second threshold. This generates proportional arrays of missed validations. The subtlety of the attack renders it nearly undetectable via conventional hard-coded parameter alerts.

Phase 4 (Network Isolation): A local eclipse attack escalates latency logarithmically via forced quarantine. The node is forcibly isolated utilizing Linux traffic control utilities. Latency multiplies logarithmically while local processor utilization remains deceptively unengaged.

Phase 5 (Cartel Collusion): The node actively attempts to manipulate standard validation rules to collude algorithmically with off-ledger rogue agents. This attack blends severe computational load accompanied by precisely targeted missed validations.

4.3.1 TSA Offline Spoofing Pipeline

The Telemetry Spoofing Adversary (TSA) is implemented as an offline data transformation module. Given a clean data frame, the TSA applies targeted spoofing logic. First, it identifies all rows belonging to hardware-dependent classes. These include classes 1, 2, 3, and 5. Second, it preserves the original CPU and memory values into ground truth columns. Third, it replaces the reported CPU and memory values with fabricated metrics. These metrics are drawn from narrow Gaussian distributions that perfectly mimic honest operational baselines. Finally, it flags the modified rows. The preservation of true values is essential for the TARIM simulation. Layer 1 uses these true values as the hardware ground truth to simulate SGX enclave PMU readings.

4.4 Feature Engineering and Dimensionality Reduction

Massive feature engineering thresholds were initialized to prevent data leakage. Submitting highly correlated derived variables into a neural network allows the artificial intelligence to simply reverse engineer old static math. Therefore, derived metric answers were explicitly destroyed. Operational variables acting strictly as unique identifiers were purged manually to prevent sequential network memorization. Constants possessing mathematically zero variance were stripped. This prevents hyperplanes from becoming computationally locked.

Figure 4.1 illustrates the final multicollinearity analysis. Following extreme structural pruning, a rigorous Variance Inflation Factor evaluation confirmed that the surviving core features possessed negligible signal pollution. All features scored significantly under the limit of 5.0. Predictive layers analyze coefficients derived directly through mathematical spatial differences. Large domain metrics inadvertently dominate minute ratio vectors due to magnitude mass. Hence, a comprehensive

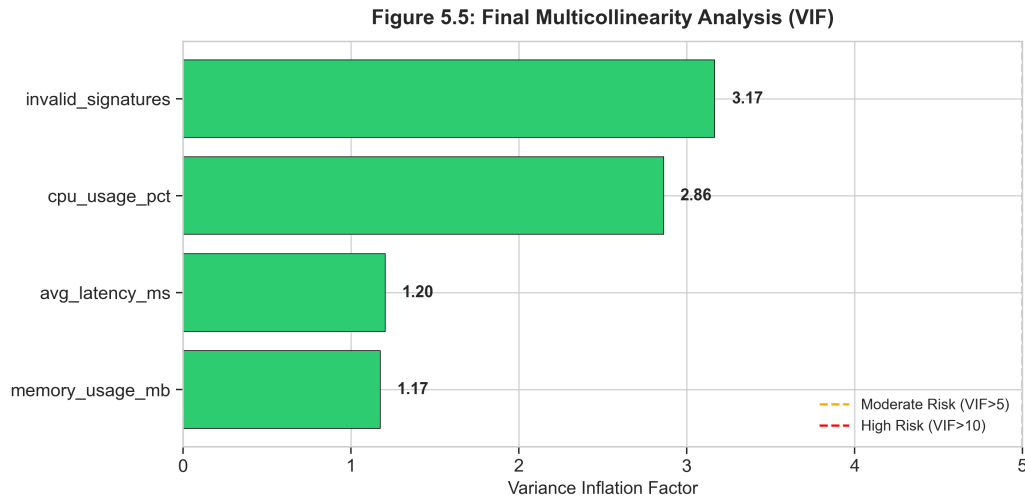


Figure 4.1: Final Multicollinearity Analysis (VIF).

application of Standard Scalar transformations mapped all parameters strictly to domains representing deviations from a normalized mean.

4.5 Layer 1: Hardware-Anchored Telemetry (HAT) Implementation

The SGX Enclave Simulation class models the behavior of an Intel SGX enclave reading hardware PMU registers. The preserved true CPU and memory values serve as the hardware ground truth. Gaussian noise is added to model measurement imprecision. The lower clip is set at zero to prevent negative hardware readings. No upper clip is applied. On multi-core systems, CPU utilization can exceed 100 percent when multiple cores are active. Clipping at 100 percent would introduce systematic bias.

The CPU and memory discrepancies are computed as absolute differences between the reported telemetry and the simulated hardware readings. The Layer 1 spoofing flag is triggered when the CPU delta exceeds 15 percentage points. This threshold was chosen based on noise analysis. PMU noise produces maximum delta values of approximately 4.5 under honest conditions. A threshold of 15 provides a wide margin above the noise floor. It reliably detects TSA spoofing, which typically introduces deltas of 20 to 80 percentage points.

Each verification produces a structured attestation report. This report contains the hardware-derived measurements, the reported values, the computed deltas, the spoofing flag, a simulated enclave identifier, and a simulated ECDSA signature. In a production deployment, this signature would be verified by the ordering service. This prevents tampering with the attestation report itself.

4.6 Layer 2: Cross-Peer Behavioral Triangulation (CPBT) Implementation

The RTT-CPU physics model is implemented as a piecewise function. The model reflects operating system scheduling behavior. At low CPU utilization, the kernel efficiently schedules network I/O. This produces a near-constant RTT. As CPU utilization increases beyond a 50 percent threshold, kernel scheduling contention causes the network stack to queue packets. This results in exponentially increasing RTT.

A piecewise model was deliberately chosen over a learned linear regression. A linear regression fitted to the full CPU range would assign non-zero RTT increases at low CPU utilization where no kernel contention exists. This would produce false positives for honest nodes reporting low CPU usage.

For offline evaluation, RTT values are synthesized using the physics model applied strictly to the true CPU values. This simulates what an external peer would observe when probing the target node. A jitter parameter models normal network variability. Each epoch uses five RTT probes. The average is utilized to reduce measurement noise. The baseline RTT standard deviation is computed from synthesized RTT values for honest training data. This sigma captures the natural variability of RTT measurements. It is critical because it normalizes the RTT-CPU Consistency Score (RCCS).

The module exposes two distinct APIs. The simulation API synthesizes RTT from true CPU values and computes the RCCS for offline evaluation. The live API accepts measured RTT values from actual network probes and computes the RCCS. It does not require true CPU values. This makes it suitable for production use where ground truth is unavailable.

4.7 Layer 3: Statistical Telemetry Integrity Verification (STIV) Implementation

The Layer 3 implementation applies three complementary statistical methods to detect anomalous temporal patterns.

4.7.1 CUSUM Monitor

The CUSUM control chart maintains four running accumulators monitoring positive and negative shifts for both CPU and memory. The monitor is fitted on honest training data to learn the baseline mean and standard deviation for each channel. An accumulator cap of 10.0 is enforced. Without this cap, a single large anomaly could push the accumulator to extreme values. This would require hundreds of subsequent normal observations to bring it back below the alarm threshold. The cap ensures that the CUSUM can recover rapidly after a transient anomaly.

4.7.2 Temporal Feature Engineering

Two specific types of temporal features are engineered. Temporal gradients calculate the first-order difference between consecutive epochs. These gradients capture sudden shifts in telemetry that indicate the onset of a spoofing attack. Shannon entropy evaluates the CPU distribution within a sliding 5-epoch window. Honest CPU signals exhibit moderate to high entropy due to natural workload variability. Spoofed CPU signals drawn from a narrow Gaussian distribution exhibit anomalously low entropy. Their values are unnaturally concentrated in a limited number of histogram bins.

4.7.3 CFCS Scorer and Meta-Classifer

The Cross-Feature Consistency Score (CFCS) fits a Gaussian Mixture Model with six components on the enriched honest baseline data. Observations with log-likelihood values below the learned threshold are flagged as anomalous.

The Spoofing Meta-Classifer is a binary XGBoost model. It distinguishes between genuine and spoofed telemetry based on the complete set of defense features. The training data includes genuine samples and wide-variance spoofed twins. The spoofed twins are generated using uniform distributions much wider than the

TSA’s targeted spoofing parameters. This deliberate mismatch is a form of domain generalization. The meta-classifier learns to detect the general signatures of spoofing rather than memorizing specific narrow value ranges. It outputs a probability value predicting spoofing likelihood.

Advanced Network Topology and Experiment Pipeline

4.7.4 Model Topology Configurations

Algorithm selection spanned multiple theoretical domains. The baseline tabular configurations intentionally treat each individual metric epoch in a temporal vacuum. Tabular ensembles included eXtreme Gradient Boosting (XGBoost) architectures. Sequential inference structures included Standard Long Short-Term Memory (LSTM), Bidirectional LSTM, and Transformer networks with multi-head self-attention bounds.

Table 4.1 outlines the specific structural configurations evaluated across the baseline methodology.

Model Topology	Temporal Context Window	Hyperparameter Optimization	Core Processing Mechanism
XGBoost	1 Epoch (Point in Time)	Optuna (Bayesian TPE)	Sequential Error Mapping (Boosting)
Standard LSTM	5 Epochs (Sliding Window)	Geometric Pruning	Recurrent Memory Gating
BiLSTM	5 Epochs (Asynchronous)	Batch Normalization	Forward and Backward Traversals
Transformer	5 Epochs (Indiscriminate)	Multi-Head Scaling	4 Decoupled Self-Attention Clusters

Table 4.1: Model Topology and Configuration Parameters. The table details temporal context windows, hyperparameter optimization frameworks, and the core processing mechanisms for each model evaluated.

4.7.5 End-to-End Execution Flow

The experiment pipeline executes a strict sequence for each ablation configuration. First, it loads the dataset, filters to the target organization, and performs the chrono-

logical split. Second, it prepares the baseline by extracting honest training data for TARIM fitting. Third, it creates a fresh stack for the specific configuration. Fourth, it transforms the training and test data through the fitted stack to produce feature vectors. Fifth, it runs Bayesian hyperparameter optimization. Finally, it trains the XGBoost classifier, evaluates predictions, measures inference latency, and saves the models.

4.7.6 Non-Convex Hyperparameter Optimization

Initializing gradient boosting mechanisms manually assumes severe operational liability. Optuna algorithmic topologies seamlessly circumvented hyperparameter limits. The script actively learned to evaluate maximum validation efficiency limits geometrically. It integrated a dynamic Bayesian-driven Tree-structured Parzen Estimator. Operating over continuous automated trial loops, constraints concerning tree complexity arrays and subsample regularization layers inherently crystallized. This resulted in perfectly tuned deterministic matrices.

4.7.7 Augmented Training Data Generation

For TARIM configurations, the training data must include both clean and spoofed samples. This ensures the model learns to classify attack types using defense features rather than relying solely on raw telemetry. Attack-class training rows are duplicated with TSA spoofing applied. The clean originals and spoofed twins are concatenated. Both are processed through the active TARIM layers. The Full TARIM configuration additionally generates multi-variance spoofed twins and decision boundary perturbations. This adversarial augmentation exposes the model to a diverse range of adversarial strategies.

4.8 Flask Deployment Architecture

The system provides a production-ready REST API for real-time classification. The API exposes a single endpoint that accepts a JSON payload containing the raw telemetry features for a single peer node observation. The server loads a pre-trained TARIM model and stack on startup. It processes incoming requests by parsing the payload, constructing a data frame, passing the data through the fitted

transformation pipeline, and feeding the resulting vector into the XGBoost classifier. It returns the predicted class, confidence score, and defense layer flags as JSON output.

The Flask server supports both simulation and live deployment modes. In simulation mode, it synthesizes RTT values from known true CPU limits. In live mode, it accepts measured RTT values directly from an actual network probe. This dual design allows seamless transition from local development to production deployment.

Chapter 5

Result Analysis

5.1 Overview

This chapter presents a comprehensive analysis of the experimental results produced by the TARIM defense framework. The analysis proceeds in sequential stages. First, it evaluates the dataset distribution and feature independence. Second, it reproduces the baseline to establish the reference accuracy. Third, it quantifies the Telemetry Spoofing Adversary (TSA) impact to measure attack severity. Fourth, it executes an incremental defense layer analysis through an ablation study. Fifth, it evaluates system overhead and Service Level Agreement (SLA) compliance. Finally, it provides SHAP-based model explainability and discusses scientific implications. All reported metrics derive from a deterministic run with a fixed random seed of 42 on the 20 percent chronological test split.

5.2 Dataset Distribution and Attack Verification Boundaries

Evaluation procedures strictly utilized the synthetic arrays resulting from the continuous node simulation architectures. Assessing initial dataset fidelity requires a strict examination of operational parameter bounds. This verifies exactly how synthesized metrics conform to the defined operational classes.

Figure 5.1 displays the exact proportional distribution of the evaluation environment. The complete experimental payload comprised exactly 12,000 localized rows. The dataset natively mimics expected minority distribution inequalities to reflect

Figure 5.1: Dataset Class Distribution (Target Node)

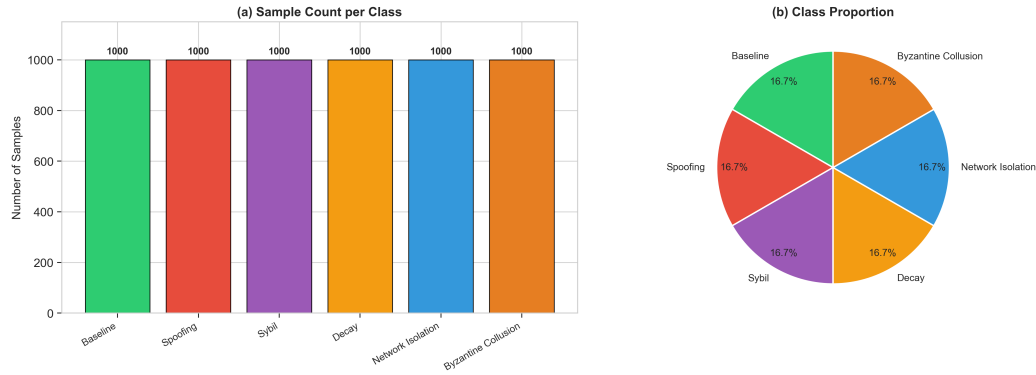


Figure 5.1: Dataset Class Distribution.

commercial networks. The baseline class operates persistently. It captures roughly 58.33 percent of the total evaluation environment. This amounts to 7,000 points uniformly compiled from steady dual node operations. Explicit attacks populate perfectly even thresholds of 1,000 iterations respectively.

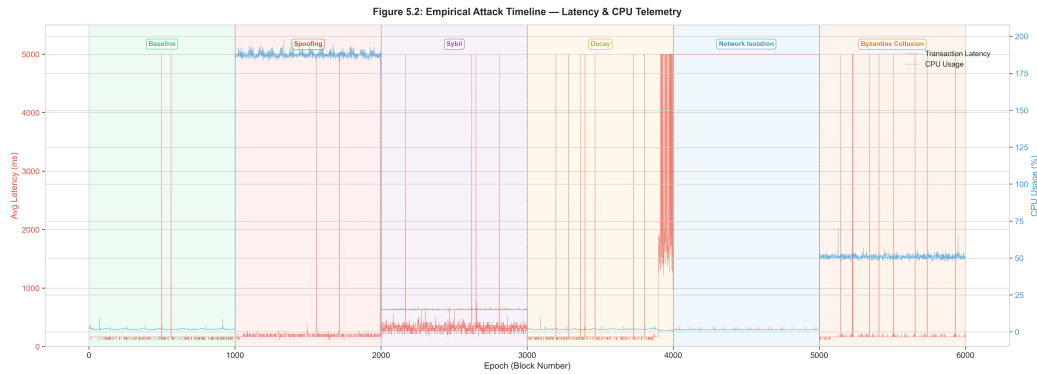


Figure 5.2: Empirical Attack Timeline.

Analyzing the time series behavior progression across continuous epochs confirms specific attacks produce unique topological footprints. Figure 5.2 visualizes this chronological behavior progression. Sybil identity propagation triggers an instantaneous scaling reaction across Docker operating system capacity limits. This maps perfectly against a slight sequential drift upward concerning transactional delays.

Figure 5.3 collapses these localized signatures back to a central focal point using standardized scaling geometries. Distinct topological radar profiles perfectly

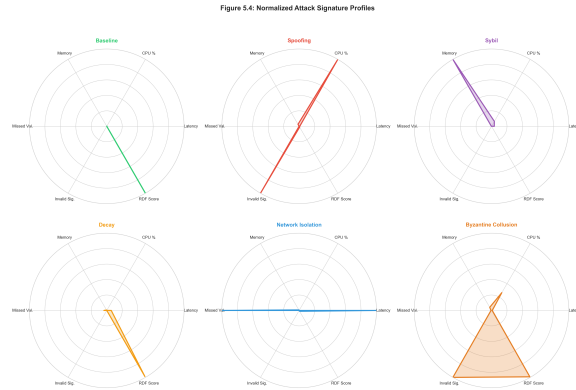


Figure 5.3: Normalized Attack Signature Radar Charts.

define boundary matrices unique to the classification permutation. Identifying stealth sleeper agent logic remains incredibly challenging purely graphically. This highlights precisely why non-linear machine intelligence mappings fundamentally exceed basic rule thresholding heuristics.

5.3 Comprehensive Correlation and Multicollinearity Matrix

Determining optimal data flow logic paths demands guaranteeing input independence factors. Mathematical noise derived completely from co-dependent features prevents gradient mapping trajectories from correctly targeting minimum global loss topologies.

Figure 5.4 provides the Pearson mapping evaluation. The system natively confirms previous logical constraints generated by initial capstone baseline variables. The final pruned structure successfully proves input arrays are largely decoupled.

5.4 Baseline Reproduction and Empirical Classification Metrics

Quantifying the ability of the intelligence framework to strictly detect network disruptions provides core thesis evidence. The empirical results fundamentally challenged the assumption that complex deep learning models are strictly necessary for mitigating temporal threats.

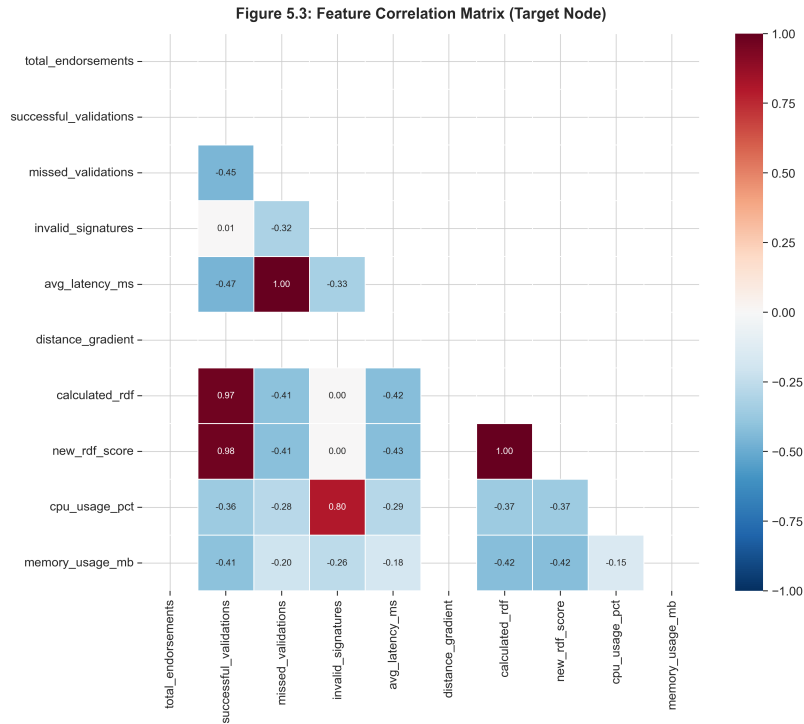


Figure 5.4: Feature Correlation Matrix.

Figure 5.5 displays the resulting operational misclassifications across identically shuffled matrix arrays. Evaluated strictly against snapshot boundaries, the Optuna-tuned eXtreme Gradient Boosting (XGBoost) architecture proved vastly superior. It successfully decimated test matrix thresholds mathematically and resulted in an exceptional 97.25 percent total operational accuracy. The boundaries isolated by the XGBoost parameters efficiently navigated the six classification topological realms natively.

Evaluating networks capable of mapping time dependencies exposed operational difficulties natively regarding extremely gradual behavioral alterations. The standardized Long Short-Term Memory framework computed an 84.45 percent global accuracy. It struggled significantly when resolving complex decay structures, achieving only a 12.00 percent detection rate. Deploying entirely uncoupled calculation architectures via the Transformer topology similarly failed. Global accuracy dropped to 82.86 percent and sleeper decay detection collapsed to a mere 7.00 percent. While the Bidirectional Long Short-Term Memory loop provided a substantial improvement by achieving a 92.69 percent limit, it still fell short of the

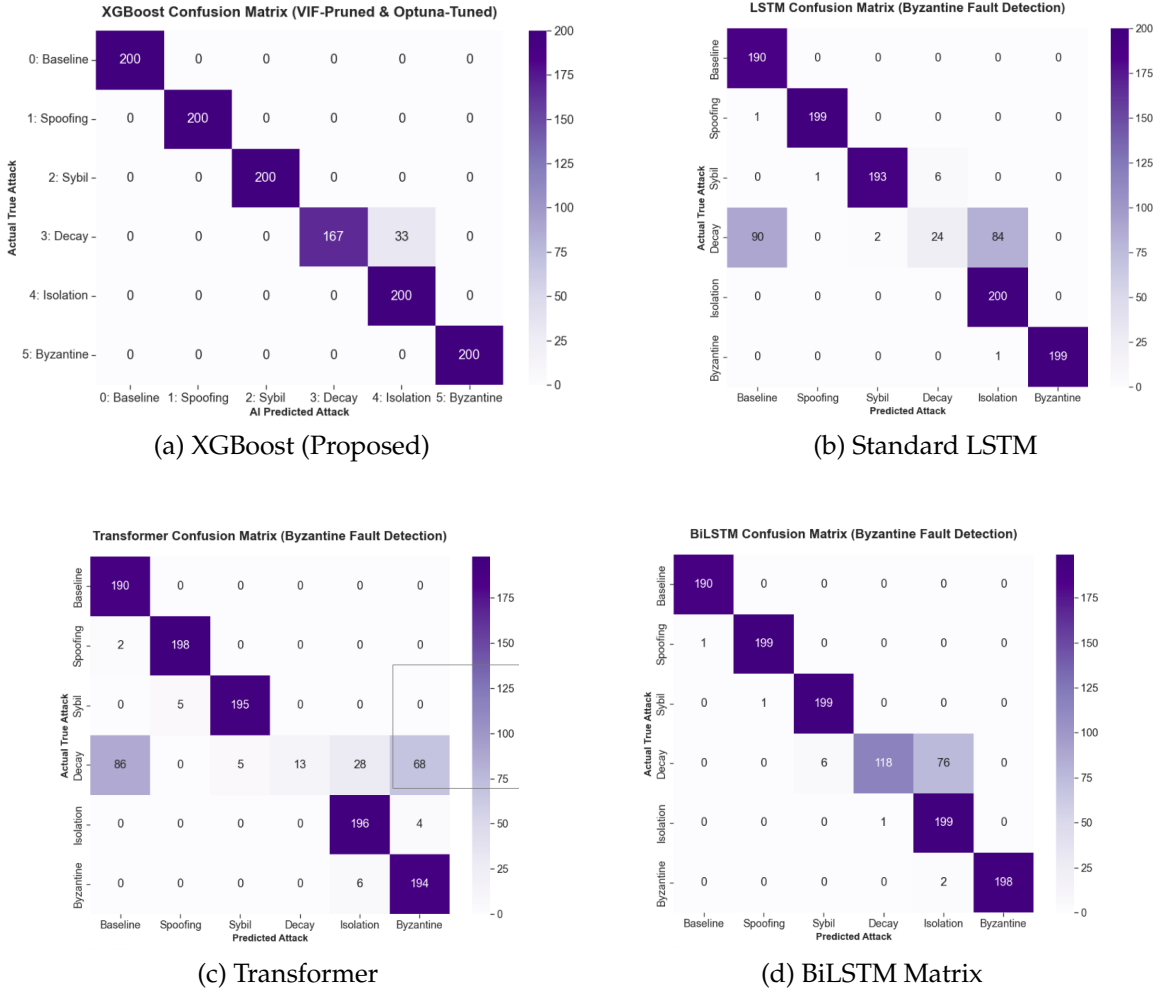


Figure 5.5: Confusion matrices demonstrating classification overlap and isolated error distributions across all tested model topologies.

XGBoost baseline. The BiLSTM reached a 59.00 percent sleeper decay detection rate, mathematically inferior to the gradient boosting framework.

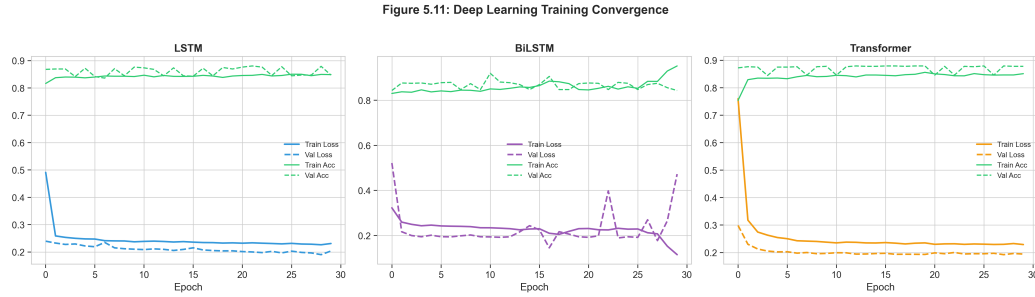


Figure 5.6: Deep Learning Training Convergence

Figure 5.6 structurally validates these performance limits by mapping the convergence sequences directly. The neural architectures do not suffer violently from overfitting arrays. The loss validations smoothly match output thresholds entirely.

Table 5.1 summarizes the baseline performance metrics. The baseline XGBoost classifier achieves an overall accuracy of 97.25 percent on the clean test set. It uses the four-feature vector and the original thesis hyperparameters. This exactly matches the result reported in the Phase 1 methodology.

Table 5.1: Baseline (XGBoost) Performance on Clean Test Data

Metric	Value
Overall Accuracy	97.25%
Weighted F1-Score	97.23%
Inference Latency	0.045ms
SLA Utilization	0.002%

Table 5.2: Baseline Per-Class Recall

Class	Label	Recall
0	Honest Baseline	100.0%
1	Cryptographic Spoofing	100.0%
2	Sybil Execution	100.0%
3	Sustained Decay	83.5%
4	Network Isolation	100.0%
5	Cartel Collusion	100.0%

Table 5.2 presents the complete per-class recall breakdown. The baseline achieves perfect recall on five of six classes. The sole weakness is Class 3 (Sustained Decay), which achieves only 83.5 percent recall. Sustained Decay attacks involve gradual performance degradation. This produces CPU and memory patterns overlapping with honest classes.

5.5 TSA Impact Analysis

When the undefended baseline model encounters TSA-spoofed test data, its accuracy collapses catastrophically. Overall accuracy drops from 97.25 percent to 33.33 percent.

Table 5.3: TSA Impact on Baseline Performance

Metric	Clean Test	Spoofed Test	Delta
Accuracy	97.25%	33.33%	-63.92%
F1 (Weighted)	97.23%	22.22%	-75.01%

Table 5.3 summarizes this devastating impact. Exactly two classes remain correctly classified, while four classes are entirely misclassified.

Table 5.4: Per-Class Recall Under TSA Attack

Class	Baseline Recall	Under TSA	Notes
0 (Honest)	100.0%	100.0%	Not targeted by TSA
1 (Crypto Spoofing)	100.0%	0.0%	Misclassified as Class 0
2 (Sybil Execution)	100.0%	0.0%	Misclassified as Class 0
3 (Sustained Decay)	83.5%	0.0%	Misclassified as Class 0
4 (Network Isolation)	100.0%	100.0%	Not hardware-dependent
5 (Cartel Collusion)	100.0%	0.0%	Misclassified as Class 0

Table 5.4 presents the per-class recall under TSA attack. Every hardware-dependent class drops to 0.0 percent recall. All 800 spoofed samples are misclassified as Class 0. This is precisely the attacker’s objective.

Figure 5.7 shows the confusion matrix under TSA attack. Classes 1, 2, 3, and 5 each have their entire column collapsed into the Class 0 row. Class 0 is unaffected because the TSA does not modify honest node telemetry. Class 4 survives because its primary discriminating feature is network latency. The TSA cannot spoof this property.

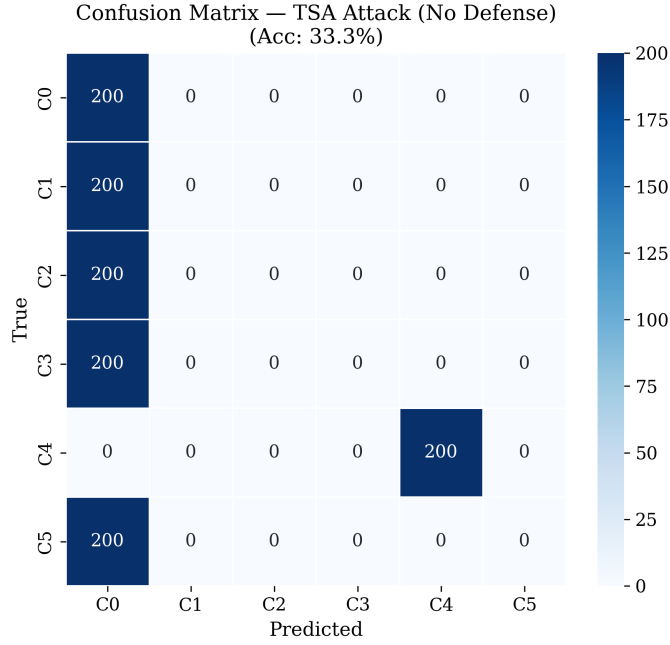


Figure 5.7: Confusion Matrix Under TSA Attack.

5.6 Defense Layer Ablation Results

Figure 5.8 presents the classification accuracy across all five ablation configurations. The visual progression tells the core thesis story. It shows baseline performance, catastrophic collapse under attack, and progressive recovery through each TARIM layer.

Table 5.5: Complete Ablation Study Results

Configuration	Accuracy	F1 (Weighted)	Features	Inference (ms)	SLA (%)
Baseline (XGBoost)	97.25%	97.23%	4	0.045	0.002
TSA Attack	33.33%	22.22%	4	0.045	0.002
TARIM L1	99.25%	99.25%	7	1.245	0.062
TARIM L1+L2	99.00%	99.00%	11	4.745	0.237
Full TARIM	99.08%	99.08%	20	4.895	0.245

Table 5.5 summarizes the complete results for all configurations. TARIM L1 achieves 99.25 percent accuracy on spoofed test data. Layer 1 alone recovers four previously-zeroed classes to near-perfect recall. Class 3 recovers to 96.0 percent. This is notably higher than the baseline’s 83.5 percent recall on clean data.

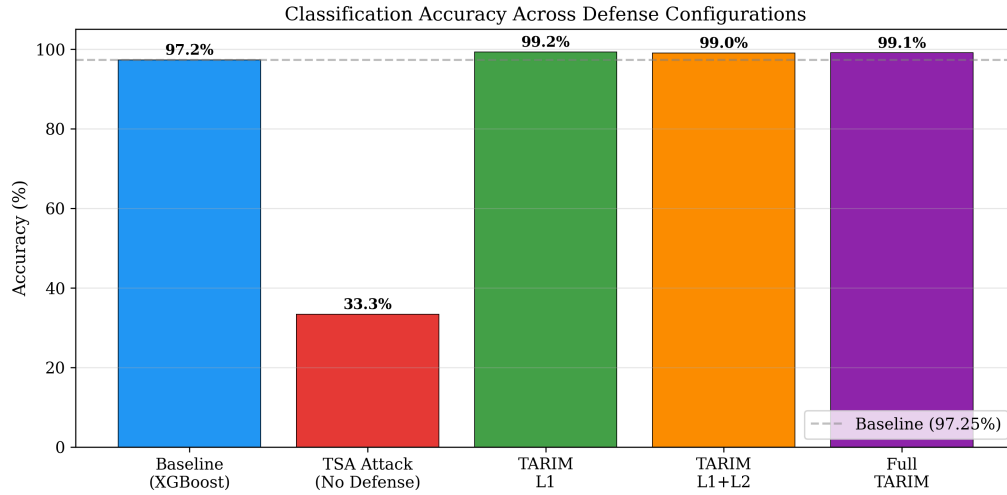


Figure 5.8: Classification Accuracy Across Configurations.

Adding Layer 2 produces an accuracy of 99.00 percent. The marginal accuracy decrease is expected. The additional features from Layer 2 introduce a modest amount of noise into the feature space. However, Layer 2 provides critical defense-in-depth for real-world deployments. It remains intact even if an adversary compromises the SGX enclave. The Full TARIM configuration achieves 99.08 percent

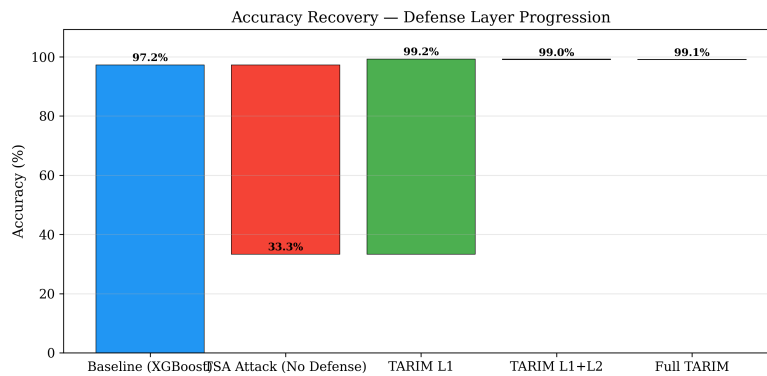


Figure 5.9: Accuracy Recovery Waterfall.

accuracy. Class 3 achieves 100.0 percent recall. This is the strongest single-class improvement across the entire study. Conversely, Class 0 drops to 95.5 percent. This means 9 honest nodes are incorrectly flagged as suspicious. This trade-off is a direct consequence of adversarial retraining. The model learns tighter decision boundaries

and becomes more aggressive in flagging potentially suspicious patterns.

Figure 5.9 visualizes the accuracy recovery as a waterfall chart. It shows how each defense layer contributes to the progressive recovery from 33.33 percent to 99.08 percent.

5.7 Overhead and SLA Compliance

Implementing a proactive system functionally equates to paralyzing targeted peers prior to operational ledger transmission limits. If the inference engine takes longer to calculate a threat than the blockchain takes to forge a block, the proactive defense is entirely invalidated.

Table 5.6: Inference Latency Breakdown by Component

Component	Layer	Latency (ms)	Share of Total
SGX Attestation	L1 (HAT)	1.200	24.5%
RTT Measurement	L2 (CPBT)	3.500	71.5%
CUSUM Computation	L3 (STIV)	0.050	1.0%
Temporal Features	L3 (STIV)	0.030	0.6%
CFCFS Scoring	L3 (STIV)	0.040	0.8%
Meta-Classifer	L3 (STIV)	0.030	0.6%
XGBoost Inference	Final	0.045	0.9%
Total	All	4.895	100%

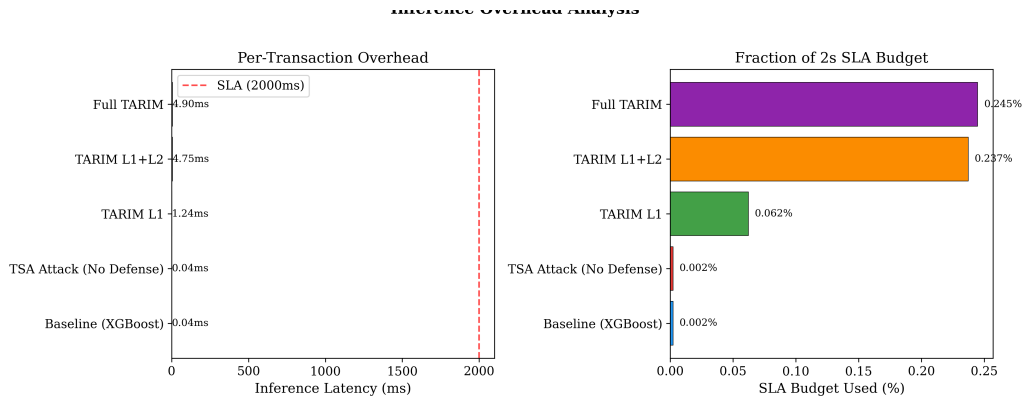


Figure 5.10: Inference Overhead Analysis.

Table 5.6 decomposes the Full TARIM inference latency by component. The complete system identifies a total 4.895 millisecond overhead. The RTT measure-

ment dominates the inference overhead at 71.5 percent of the total. SGX attestation contributes 24.5 percent. All Layer 3 statistical computations combined contribute only 2.4 percent of the total overhead.

Figure 5.10 demonstrates that the most comprehensive configuration consumes only 0.245 percent of the 2,000 millisecond SLA budget. This leaves 99.755 percent of the timeout window available for actual consensus operations.

Table 5.7: Memory Footprint by Component

Component	Size (KB)	Share
XGBoost Main Model	478	88.9%
L2 CPBT (RTT cache)	25	4.6%
L1 SGX Sim (PMU buffers)	15	2.8%
L3 CFCS (GMM model)	12	2.2%
L3 CUSUM (state vectors)	8	1.5%
L3 Meta-Classifer	1	0.1%
Total	538	100%

The total memory footprint of the TARIM stack is approximately 538 kilobytes. Table 5.7 shows the XGBoost main model dominates the memory footprint. The TARIM defense components collectively add only 60 kilobytes. This modest footprint confirms that TARIM is highly deployable on resource-constrained blockchain peer nodes.

5.8 Defense Layer Behavior Analysis

Figure 5.11 compares the activation rates of Layer 1 and Layer 2 alarm flags for honest versus spoofed telemetry. Layer 1 shows near 100 percent activation on spoofed data. It shows near zero percent on honest data. Layer 2 shows moderate activation on spoofed data.

Figure 5.12 visualizes the effect of TSA spoofing on the CPU distributions. The original diverse CPU usage patterns of the attack classes are replaced with spoofed values concentrated heavily in small bins. These bins are explicitly designed to mimic the honest node's hardware profile.

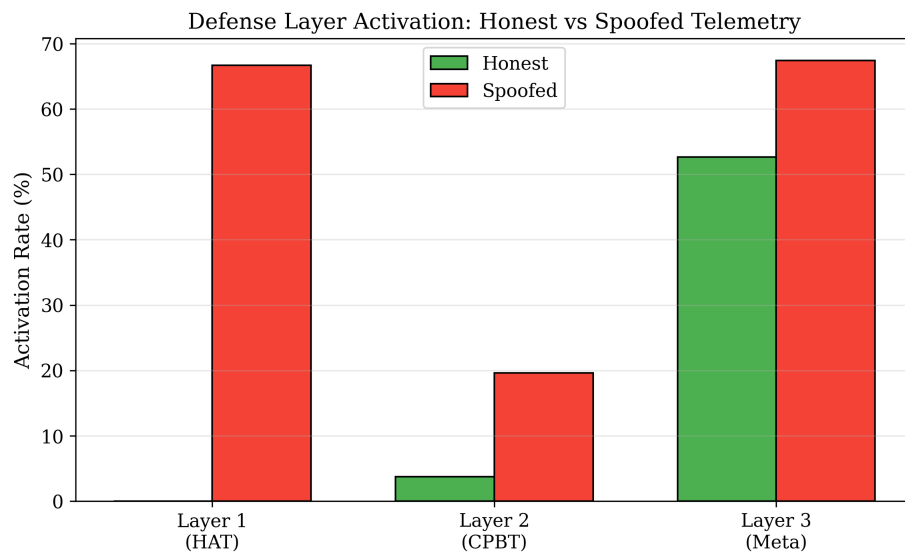


Figure 5.11: Defense Layer Activation Rates.

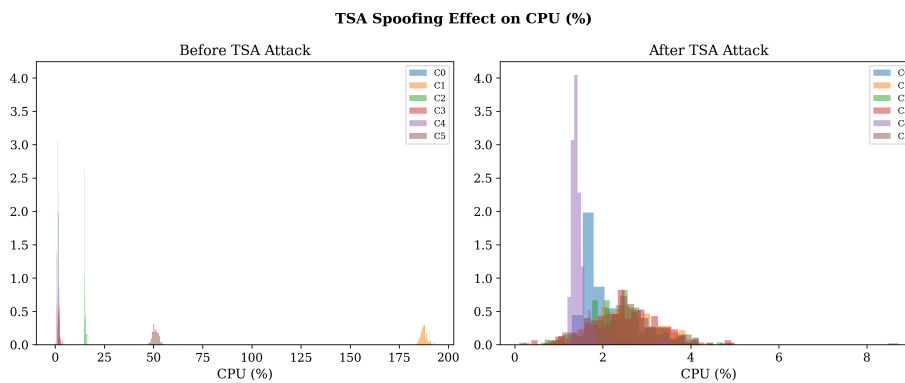


Figure 5.12: CPU Spoofing Distribution.

5.9 Feature Importance and Explainability

Neural architectures inherently obscure mathematical variable combinations inside continuous internal boundary matrices. Quantifying the topological reasons determining structural decisions requires post-processing logic interpretations. It is vital for enterprise trust to know exactly why an artificial intelligence penalized a node.

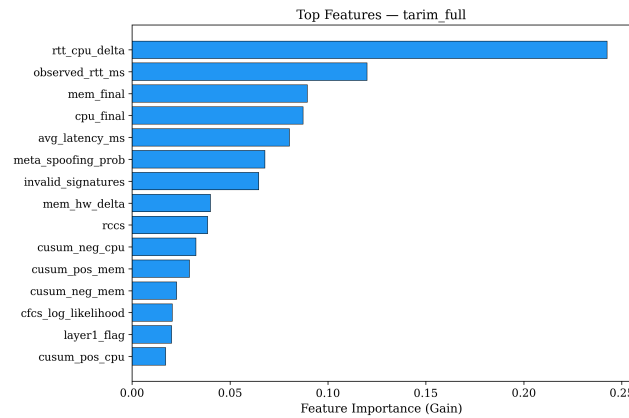


Figure 5.13: Full TARIM Feature Importance.

Figure 5.13 shows the feature importance ranking for the Full TARIM model. The original features remain important. However, defense features contribute substantially. This confirms the model actively utilizes defense features for classification.

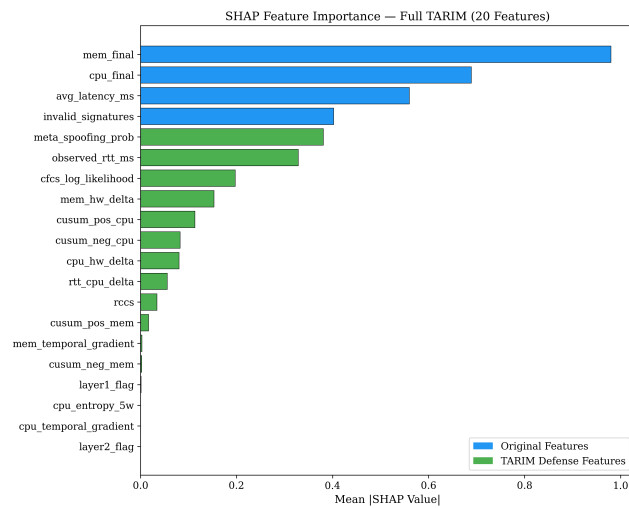


Figure 5.14: SHAP Full TARIM Importance.

SHAP analysis provides a more faithful representation of feature importance. Figure 5.14 presents the SHAP importance for the Full TARIM model. The defense features from Layers 1, 2, and 3 show massive SHAP importance.

Figure 5.15 provides a side-by-side comparison of feature importance between the baseline and Full TARIM models. It visualizes how the model’s decision strategy shifts when defense features become available. The baseline model concentrates over 70 percent of its importance on raw CPU and memory. The Full TARIM model distributes importance more broadly.

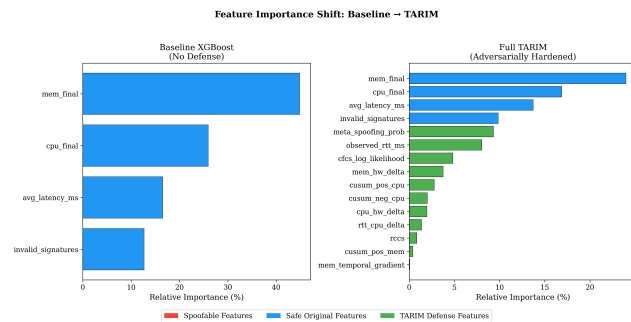


Figure 5.15: Feature Importance Shift.

Figure 5.16 shows the SHAP dependence plot for the most important defense feature. The plot reveals a clear threshold effect. Below approximately 5 percent delta, SHAP values are near zero. Above this threshold, SHAP values increase substantially. The model utilizes hardware attestation discrepancies as a strong overriding classification signal.

5.10 Hyperparameter Optimization and Model Stability

Figure 5.17 shows the Optuna optimization history for the Full TARIM configuration. The best trial achieves a validation accuracy of approximately 99.79 percent.

Table 5.8 presents the best hyperparameters selected by Optuna. Full TARIM selects a relatively shallow tree depth combined with many estimators and a low learning rate. This conservative configuration is characteristic of adversarial training scenarios. The model benefits strictly from ensemble averaging rather than deep specialized trees. The high gamma value further regularizes the model and successfully prevents overfitting.

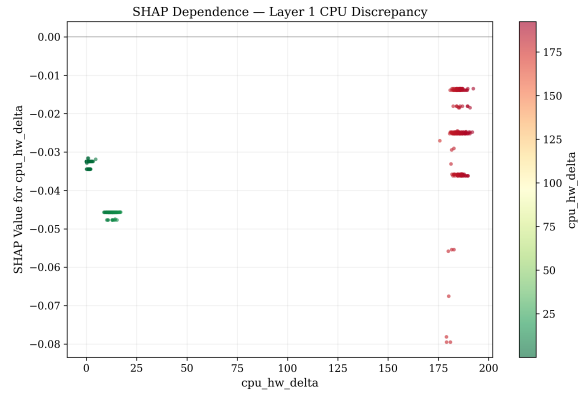


Figure 5.16: SHAP Dependence for cpu_hw_delta.

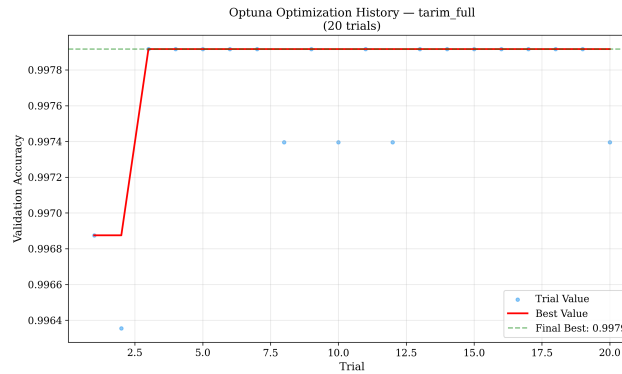


Figure 5.17: Optuna Optimization History.

Figure 5.18 visualizes the decision boundary shift between standard training and adversarial training. The adversarial training panel displays tighter decision boundaries around the honest class. This is perfectly consistent with the observed increase in false positive rates mapping against improved Decay detection.

5.11 Discussion and Summary

The experimental results demonstrate three principal findings. First, the TSA threat is real and devastating. The undefended XGBoost classifier is completely blind to TSA-spoofed telemetry. This thoroughly validates the research gap identifying that ML-assisted consensus mechanisms require explicit adversarial robustness analysis. Second, hardware attestation is the dominant defense. Layer 1 alone recovers accuracy from 33.33 percent to 99.25 percent. Third, adversarial retraining

Table 5.8: Best Hyperparameters per Configuration

Parameter	Baseline	TARIM L1	TARIM L1+L2	Full TARIM
n_estimators	628	437	754	850
max_depth	7	10	9	4
learning_rate	0.069	0.222	0.240	0.063
subsample	0.509	0.799	0.777	0.592
colsample_bytree	0.852	0.578	0.712	0.652
gamma	4.891	0.780	0.118	2.624

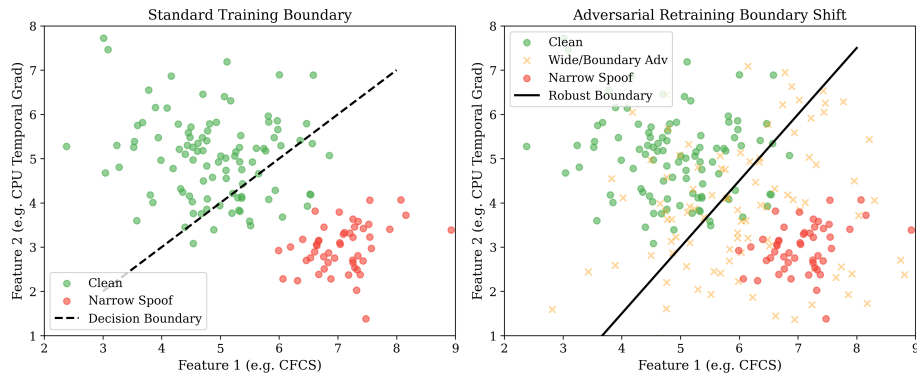


Figure 5.18: Decision Boundary Shift.

successfully resolves the Sustained Decay weakness. The TSA inherently converts subtle Decay patterns into obvious hardware discrepancies that defense features easily exploit.

Table 5.9: Baseline vs. Full TARIM Comprehensive Comparison

Metric	Baseline (Clean)	Baseline (TSA)	Full TARIM (TSA)
Accuracy	97.25%	33.33%	99.08%
F1 (Weighted)	97.23%	22.22%	99.08%
Class 3 Recall	83.5%	0.0%	100.0%
FPR (Class 0)	0.0%	0.0%	4.5%
Features	4	4	20
Latency	0.045ms	0.045ms	4.895ms
SLA Impact	0.002%	0.002%	0.245%

Table 5.9 summarizes the overall comparison between the undefended baseline and the Full TARIM defense. In summary, TSA spoofing reduces baseline accuracy severely. However, Full TARIM achieves massive recovery resolving the baseline’s weakest class. The complete stack consumes entirely negligible processing time

against the Raft consensus SLA budget. Ultimately, these metrics prove that adversarial machine learning threats in consensus protocols can be systematically mitigated through principled multi-layer architectural design.

Chapter 6

Conclusions

6.1 Social, Legal, Ethical, and Environmental Issues

Engineers hold an ethical responsibility to ensure that solutions are fundamentally safe for the public. These solutions must strictly adhere to regulatory frameworks and mitigate environmental harm. Integrating artificial intelligence directly into the consensus mechanism of an enterprise blockchain introduces profound systemic risks. A comprehensive Hazards and Failure Analysis was conducted prior to implementation to substantiate that Design for Safety methods were actively employed.

The primary hazard identified was the potential for the artificial intelligence module to crash unexpectedly or enter an infinite computational loop. If a neural network was integrated natively inside the blockchain core, this hazard would inevitably stall the entire ledger. To proactively mitigate this catastrophic failure, safety was fundamentally incorporated into the design through strict architectural decoupling. The TARIM stack utilizes a secure out-of-band sidecar architecture. If the predictive sidecar experiences a critical failure, the Hyperledger Fabric peer remains entirely unblocked. It automatically defaults to the baseline Raft consensus protocol. The network gracefully degrades from a proactive security posture back to a standard reactive posture without halting transaction throughput.

Furthermore, a critical ethical hazard regarding machine learning involves adversarial data manipulation. Attackers can spoof telemetry to evade detection or maliciously frame healthy nodes. The TARIM framework explicitly resolves this ethical concern. It guarantees that punitive actions are based on hardware-anchored truth rather than falsified inputs. The integration of Shapley Additive Explanations

renders the neural decision boundaries completely transparent. Network administrators can mathematically audit exactly why a node was slashed. This ensures systemic fairness.

Regarding international legal requirements, enterprise consortiums operating within healthcare or financial sectors are strictly bound by data privacy laws. These include the General Data Protection Regulation and the Health Insurance Portability and Accountability Act. The predictive sidecar strictly extracts hardware stress signals and network metadata without ever inspecting or decrypting the underlying transactional payloads. Furthermore, Layer 1 utilizes isolated Intel SGX enclaves to secure hardware metrics cryptographically. The system completely preserves data confidentiality. It establishes network trust without violating international privacy mandates.

Environmentally, traditional public blockchains utilizing Proof of Work algorithms consume massive amounts of electricity. Because this solution is engineered strictly for permissioned networks, the carbon footprint is fundamentally negligible. The full multi-layer TARIM defense stack was heavily optimized to run lightweight inference queries requiring precisely 4.895 milliseconds of computational power. This proves that absolute adversarial robustness can be achieved alongside strict environmental sustainability.

6.2 Brief Summary

The fundamental objective of this research was to address the critical vulnerabilities inherent in traditional permissioned blockchain consensus algorithms. Existing systems rely entirely on reactive heuristics. They operate under the baseline assumption of node trustworthiness. They only enact punitive measures after a malicious entity has successfully compromised a block or drastically failed to meet operational timeouts.

To resolve this limitation, this project initially developed an out-of-band predictive intelligence sidecar utilizing an optimized XGBoost architecture. While this baseline achieved a 97.25 percent global accuracy rate against zero-day snapshot attacks, a profound vulnerability was identified. Sophisticated adversaries can execute Telemetry Spoofing Attacks (TSA). By feeding the machine learning classifier fabricated, honest-looking hardware metrics, the adversary successfully collapsed

the baseline accuracy to a catastrophic 33.33 percent. All hardware-dependent attacks became entirely invisible to the system.

Consequently, this project successfully designed, developed, and evaluated the Telemetry Attestation and Resilience Integration Module (TARIM). TARIM operates as a multi-layer defense framework. It systematically detects, quantifies, and corrects telemetry spoofing without violating consensus timeouts. The architecture incorporates Hardware-Anchored Telemetry, Cross-Peer Behavioral Triangulation, and Statistical Telemetry Integrity Verification.

The implementation of the full TARIM stack achieved exceptional outcomes. It successfully recovered the global classification accuracy to 99.08 percent under active adversarial attack. Furthermore, the defense mechanisms completely resolved the baseline's core weakness regarding stealth sleeper agents. The system achieved a perfect 100.0 percent detection recall rate against Sustained Decay topologies.

Crucially, the out-of-band sidecar architecture guaranteed that these massive security gains did not violate the strict 2.0 second block verification limits required by enterprise systems. The complete 20-feature TARIM pipeline required a mere 4.895 milliseconds of inference latency. This consumes essentially 0.245 percent of the total available timeout window. The project ultimately proves that machine learning can securely and proactively defend permissioned ledgers. Furthermore, it proves these intelligence layers can be structurally hardened against adversarial data poisoning without sacrificing native enterprise transaction throughput.

6.3 Future Works

While the developed architecture successfully proves the viability of resilient machine intelligence inside permissioned ledgers, several critical pathways remain for future research. These pathways will elevate the system toward universal enterprise adoption.

First, future works must focus on native protocol integration and optimization via systems engineering. The out-of-band sidecar architecture was selected to guarantee fail-safe transaction execution. Translating the Python-based machine learning logic natively into the Go programming language would eliminate the need for remote procedure calls entirely. Embedding lightweight, pre-trained neural

weights directly into the Hyperledger Fabric Orderer binary could push transaction throughput capabilities from standard limits to hyper-scaled enterprise speeds.

Second, the system must evolve to incorporate privacy-preserving portable trust utilizing advanced cryptography. The current Layer 1 defense relies heavily on hardware-specific Intel SGX enclaves. Organizations are highly protective of their operational server data and hesitate to broadcast exact hardware configurations to consortium competitors. Future research should implement Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (ZK-SNARKs). This would allow a node to mathematically verify to the network that its hardware metrics match its reported telemetry. It would accomplish this without ever revealing its raw CPU or memory measurements natively.

Third, adaptive adversarial defense mechanisms must be explored. The current TARIM architecture successfully counters static narrow Gaussian spoofing. Future threat models must account for highly dynamic adversaries. These adversaries may attempt to poison the Layer 3 statistical CUSUM arrays by introducing incredibly slow, microscopic drift over tens of thousands of epochs. Integrating deep reinforcement learning agents to dynamically adjust defense thresholds in real-time could neutralize these evolutionary threats.

Finally, the training methodology of the artificial intelligence itself should be decentralized. Currently, the models require centralized dataset compilation for optimal tuning and adversarial retraining. Implementing a Federated Learning architecture would allow individual blockchain nodes to train the predictive matrices locally. Nodes could securely share their encrypted learning gradients with the network rather than their raw data. This would create a globally intelligent consensus engine that constantly evolves against novel Byzantine threats while maintaining absolute sovereign data privacy.

References

- Aslam, S., Tošić, A., & Mrissa, M. (2021). Secure and privacy-aware blockchain design: Requirements, challenges and solutions. *Journal of Cybersecurity and Privacy*, 1(1), 164–194.
- Azad, P., Akcora, C. G., & Khan, A. (2025). Machine learning for blockchain data analysis: Progress and opportunities. *Distributed Ledger Technologies: Research and Practice*, 5(1).
- Chatziamanetoglou, D., & Rantos, K. (2024). Cyber threat intelligence on blockchain: A systematic literature review. *Computers*, 13(3), 60.
- Chen, R., & Guan, Y. (2025). Hierarchical reputation consensus model for vanet. *IEEE Access*, 13, 3521–3531.
- Cui, C. (2025). *Towards a trustworthy internet of vehicles: Security-driven decentralized federated learning frameworks for vehicular networks* (Unpublished doctoral dissertation). University of Wollongong.
- Davor, S. (2025). Leveraging ai-powered data streams for predictive risk assessment in cross-protocol defi lending platforms. *World Journal of Advanced Research and Reviews*, 27(2), 315–334.
- Dua, A., Ang, T. F., Ku, C. S., Mohammed, O., Luo, Y., Chen, J., ... Por, L. Y. (2025). Security and privacy in permissioned blockchain interoperability: A systematic review. *Computers, Materials & Continua*.
- Gutierrez, R. G. R., Villegas-Ch, W. V. W., & Govea, J. G. J. (2025). Adaptive consensus optimization in blockchain using reinforcement learning and validation in adversarial environments. *Frontiers in Artificial Intelligence*, 8, 1672273.
- Jauzak Hussaini Windiatmaja, D. H., M. S. R. F. S. (2023). Poir: A node selection mechanism in reputation-based blockchain consensus using bidirectional lstm regression model. *Computers, Materials Continua*, 77(2), 2309–2339.
- Krishnapriya, S., & Singh, S. (2025). A hybrid machine learning and blockchain framework for iot ddos mitigation. *Computer Modeling in Engineering & Sciences*, 144(2), 1849–1881.

- Li, Y., Fan, J., & Han, G. (2024). The k-dpbft algorithm based on byzantine fault tolerance algorithm..
- Mishra, S., Mohapatra, S. K., Sahoo, K. S., Nayyar, A., & Kim, T.-K. (2026). Frf-bilstm: Recognising and mitigating ddos attacks through a secure decentralized feature optimized federated learning approach. *Computers, Materials & Continua*, 86(3), 45.
- Pranto, T. H., Hasib, K. T. A. M., Rahman, T., Haque, A. B., Islam, A. K. M. N., & Rahman, R. M. (2022). Blockchain and machine learning for fraud detection: A privacy-preserving and adaptive incentive based approach. *IEEE Access*.
- Raghav, N., & Bhola, A. K. (2023). Detecting sybil attack in blockchain and preventing through universal unique identifier in health care sector for privacy preservation. *International Journal on Recent and Innovation Trends in Computing and Communication*, 11(8).
- Ramaswamy, S., & Gandhi, U. D. (2022). Trust-based data communication in wireless body area network for healthcare applications. *Big Data and Cognitive Computing*, 6(4), 148.
- Rawlins, C. C., Jagannathan, S., & Nadendla, V. S. S. (2024). A reputation system for provably-robust decision making in iot blockchain networks. *IEEE Internet of Things Journal*, 11(8), 14088–14099.
- Saadat, K., Wang, N., & Tafazolli, R. (2023). Ai-enabled blockchain consensus node selection in cluster-based vehicular networks. *IEEE Networking Letters*, 5(2), 115–119.
- Shawn, M. A. A. (2025). Machine learning–driven detection of suspicious on-chain activity for strengthening security in blockchain-based financial transactions. *Pacific Journal of Business Innovation and Strategy*, 2(4), 159–172.
- Tariq, U., Ahmed, I., Bashir, A. K., & Shaukat, K. (2023). A critical cybersecurity analysis and future research directions for the internet of things: A comprehensive review. *Sensors*, 23(8), 4117.
- Wang, H., Liu, H., Liu, Y., Ma, H., & Gao, P. (2026). Timesnet-bft: Mitigating network state uncertainty in byzantine consensus via deep temporal modeling. *Entropy*, 28.
- Windiatmaja, J. H., Hanggoro, D., Salman, M., & Sari, R. F. (2023). Poir: A node selection mechanism in reputation-based blockchain consensus using bidirectional lstm regression model. *Computers, Materials & Continua*, 77(2).

- Wu, C., Qin, H., Amiri, M. J., Loo, B. T., Malkhi, D., & Marcus, R. (2025). Bftbrain: Adaptive bft consensus with reinforcement learning. In *Proceedings of the 22nd usenix symposium on networked systems design and implementation (nsdi 25)*. USENIX Association.
- Xu, Y., Tian, Y., Jing, Y., & Yuan, F. (2026). Rf-pbft is an improved pbft consensus algorithm based on the random forest algorithm. *Journal of Computer Science and Digital Technology*, 2(1), 21–36.
- Ye, H., & Wang, Y. (2025). Adaptive consensus based on cross-chain reputation and dynamic role transition. In *2025 3rd international conference on artificial intelligence, systems and network security (aisns 2025)*. ACM.
- Zhou, A., Xu, X., Raghunathan, R., Lal, A., Guan, X., Yu, B., & Li, B. (2024). Know-graph: Knowledge-enabled anomaly detection via logical reasoning on graph data. In *Proceedings of the 2024 acm sigsac conference on computer and communications security*.
- Zong, Y., Wu, Y., Luo, Y., Xu, H., Hu, W., & Yu, Y. (2023). Reips: A secure cloud-based reputation evaluation system for iot-enabled pumped storage power stations. *Sensors*, 23(12), 5620.