Scientific Research Publishing

Open Journal of Applied Sciences, 2025, 15(9), 2759-2785
https://www.scirp.org/journal/ojapps
ISSN Online: 2165-3925
ISSN Print: 2165-3917

A Blockchain-Based Framework for Ensuring Device Integrity in the Internet of Things

Godfrey Wandwi, Diana Mjema

Department of Digital Technologies and Information Science, Dar es Salaam Tumaini University, Dar es Salaam, Tanzania
Email: godfrey.wandwi@dartu.ac.tz

How to cite this paper: Wandwi, G. and Mjema, D. (2025) A Blockchain-Based Framework for Ensuring Device Integrity in the Internet of Things. *Open Journal of Applied Sciences*, **15**, 2759-2785.
https://doi.org/10.4236/ojapps.2025.159185

Received: July 9, 2025
Accepted: September 16, 2025
Published: September 19, 2025

Abstract

With the rapid expansion of the Internet of Things (IoT), the integrity of connected devices has emerged as a critical concern. Malicious actors increasingly target vulnerabilities in device firmware, communication protocols, and system configurations, compromising the reliability and trustworthiness of data. Traditional security mechanisms have struggled to scale with the decentralized and heterogeneous nature of IoT networks. To address this challenge, this paper proposes a blockchain-based framework designed to safeguard the integrity of IoT devices. The framework leverages a lightweight consensus mechanism and a distributed ledger to establish tamper-evident records of device behavior and configuration states. Additionally, smart contracts are employed to automate verification processes, detect anomalies, and enforce compliance with integrity policies in real time. The case study conducted demonstrates how this approach enables secure attestation of device states while minimizing computational overhead, making it suitable for resource-constrained environments. The proposed framework represents a step forward in embedding trust into the fabric of IoT systems through decentralized integrity assurance mechanisms.

Keywords

Internet of Things, Blockchain, Device Integrity, Smart Contracts, Decentralized Security

1. Introduction

The Internet of Things (IoT) has seen rapid development in recent years, with billions of interconnected devices playing pivotal roles in sectors such as healthcare, manufacturing, agriculture, and urban infrastructure. These devices, often embedded with sensors and actuators, facilitate seamless communication between the

physical and digital worlds. However, the very characteristics that make IoT so transformative (heterogeneity, decentralization, and pervasive connectivity) also introduce significant security vulnerabilities. One pressing concern is the assurance of device integrity, which is essential for maintaining trust in data provenance and system behavior. The lack of standardized integrity validation mechanisms across diverse IoT ecosystems increases the likelihood of unauthorized modifications, firmware tampering, and malicious code injections [1].

Conventional security approaches have struggled to keep pace with the dynamic and resource-constrained nature of IoT devices. Limited processing capabilities, constrained memory, and energy efficiency requirements mean that many devices are unable to support real-time threat detection or complex cryptographic operations. Furthermore, centralized security models introduce single points of failure and bottlenecks, which are ill-suited for distributed IoT architectures [2]. As the number of devices connected to the Internet continues to rise, so too does the attack surface, creating fertile ground for adversaries to compromise device integrity for malicious purposes. The consequences of such breaches extend beyond data manipulation to include service disruption, surveillance, and potential harm to human life in safety-critical systems [3].

In response to these challenges, blockchain technology has emerged as a promising solution for enhancing trust, transparency, and immutability in distributed environments. By leveraging a decentralized ledger and consensus mechanisms, blockchain can provide verifiable records of device state and behavioral history, which are critical for ensuring the authenticity and integrity of IoT devices. Smart contracts further enhance this capability by enabling automated verification procedures and triggering corrective actions in case of detected anomalies [4]. Despite its potential, integrating blockchain with IoT remains complex due to issues such as scalability, transaction latency, and energy consumption. Addressing these concerns requires tailored architectural frameworks that consider the constraints and operational realities of IoT networks [5].

This paper proposes a blockchain-based framework for maintaining device integrity in IoT environments. The framework is designed to record device configurations and state transitions on a tamper-evident ledger while employing smart contracts to enforce predefined security policies. The objective is to demonstrate a lightweight, scalable, and decentralized approach capable of preserving integrity without imposing excessive computational burdens on devices. By addressing both conceptual and implementation-level challenges, the framework aims to contribute to the broader goal of securing the foundational layers of the IoT ecosystem.

Related Works

Ensuring the integrity of IoT devices is a complex and evolving challenge, particularly due to the distributed and resource-constrained nature of such systems. Existing research in the domain of IoT security highlights multiple vectors through

which device integrity can be compromised. These include hardware-based tampering, firmware injection, unauthorized configuration changes, and malicious remote updates [6]. While traditional cryptographic techniques have been applied to address some of these issues, the limitations in processing power, energy capacity, and storage in IoT devices often hinder the direct application of standard security protocols [7] [8].

Recent studies have explored various mechanisms to guarantee device trustworthiness, ranging from hardware-based Trusted Platform Modules (TPMs) to lightweight remote attestation schemes [9] [10]. However, many of these solutions are either cost-prohibitive or fail to provide scalability in large-scale IoT environments. Furthermore, centralized verification architectures create single points of failure and become bottlenecks under high data loads, prompting researchers to seek decentralized alternatives [11]. In this context, blockchain technology has emerged as a promising candidate due to its inherent properties of immutability, transparency, and distributed consensus [12].

Several research efforts have proposed blockchain-integrated frameworks to manage identities and ensure trust in IoT networks. For instance, Dorri *et al.* [13] developed a lightweight blockchain solution that decentralizes access control without compromising system efficiency. Similarly, Novo [14] proposed a scalable and lightweight permissioned blockchain system tailored for constrained IoT environments, demonstrating improvements in latency and trust enforcement. Other studies have introduced hybrid architectures that combine off-chain storage with on-chain verification to mitigate blockchain's storage overhead and latency [15].

Blockchain-based remote attestation frameworks have also gained traction, wherein blockchain serves as a verifiable ledger for device states and integrity proofs. Notably, Rahman *et al.* [16] proposed a scheme that anchors cryptographic hashes of device firmware to blockchain, enabling third-party auditors to verify integrity without direct access to the devices. In parallel, Chen *et al.* [17] explored the integration of blockchain with Physically Unclonable Functions (PUFs) to create tamper-evident identity proofs tied to device hardware. These approaches demonstrate the potential of blockchain in building secure, verifiable IoT infrastructures but often lack full adaptability across heterogeneous device ecosystems.

Moreover, the use of smart contracts in ensuring automated integrity checks has been explored to reduce manual oversight in trust management. The work by Sharma *et al.* [18] employed smart contracts to periodically validate device configurations against predefined integrity baselines, with deviations being flagged in real-time. However, such systems introduce complexities in smart contract design and often depend on external oracles, which may become new attack surfaces [19].

Despite these advances, significant challenges remain. Many proposed frameworks suffer from latency bottlenecks, especially when consensus protocols are not optimized for IoT-specific constraints. Furthermore, the issue of secure bootstrapping (initially establishing trust in a device joining the network) remains largely unresolved [20]. Additionally, most research addresses integrity at the data or net-

work level but rarely covers the full lifecycle of the IoT device, from manufacturing to deployment and decommissioning [21].

To address these gaps, our study proposes a comprehensive blockchain-based framework designed to ensure device integrity holistically. Unlike existing solutions, our approach combines lightweight cryptographic primitives, decentralized trust models, and smart contract-driven lifecycle integrity verification. The framework is designed to function within the constraints of low-resource IoT devices while ensuring tamper-evidence and auditable history of device states. This work aims to contribute a generalizable, scalable, and cost-effective solution to the longstanding issue of device integrity assurance in the Internet of Things.

2. Model

This section elaborates on the proposed blockchain-based model for ensuring device integrity in the Internet of Things (IoT) environment. The model integrates blockchain capabilities with IoT architectural principles to enhance device trustworthiness, data immutability, and secure communication across a distributed network of smart devices. Our framework is grounded on four core components: 1) Blockchain Integrity Layer, 2) Device Registration and Authentication Module, 3) Smart Contract Governance System, and 4) Decentralized Trust Evaluation Engine. We employ a Model-Based Systems Engineering (MBSE) approach, specifically using SysML diagrams to conceptualize and design the system, ensuring traceability, scalability, and modularity across heterogeneous IoT ecosystems.

2.1. Components of the Proposed Framework

2.1.1. Blockchain Integrity Layer

This component forms the backbone of the framework and is implemented using a permissioned blockchain architecture, such as Hyperledger Fabric, to ensure scalability and controlled access. Each IoT device functions as a peer node, contributing to the consensus process via a lightweight consensus mechanism, such as Practical Byzantine Fault Tolerance (PBFT) [22]. This layer is responsible for recording device integrity states, event logs, firmware versions, and device configuration (refer Appendix B) hashes. By leveraging immutability and distributed consensus, it guarantees tamper-proof evidence of device behavior.

2.1.2. Device Registration and Authentication Module

New devices undergo a rigorous registration process facilitated by a certificate authority (CA). Each registered device is issued a cryptographic identity, stored as a transaction on the blockchain ledger [23]. Authentication occurs via public-key cryptography, where devices sign communication messages with their private keys, verified by other peers using corresponding public keys. This prevents spoofing and ensures secure device-to-device and device-to-gateway communication.

2.1.3. Smart Contract Governance System

Smart contracts define the operational logic for integrity validation, role-based

access control, and anomaly detection thresholds. For example, if a device firmware hash deviates from its baseline, a smart contract automatically triggers a quarantine protocol and flags the event for administrative review [24]. Governance policies can be modified through consensus, ensuring transparency and democratic participation from trusted stakeholders.

2.1.4. Decentralized Trust Evaluation Engine

Each device is periodically evaluated based on behavioral attributes, including communication frequency, data integrity, and compliance with operational norms. This evaluation is computed off-chain and the result (*i.e.*, trust score) is written to the blockchain ledger for transparency and auditability [25]. Devices with trust scores below a defined threshold are temporarily isolated from the network to prevent lateral movement of potential attacks.

2.2. Framework Architecture

Figure 1 illustrates the overall system architecture, depicting interactions among devices, the blockchain network, certificate authorities, and governance modules.

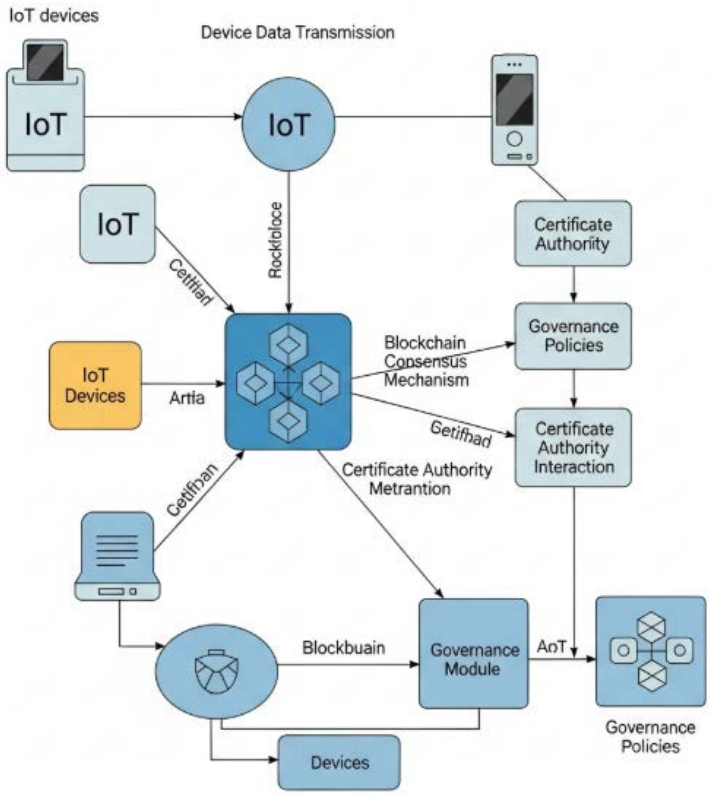

Figure 1. High-level architecture of the blockchain-based IoT integrity framework.

The overall structure of the proposed system is captured in Figure 1, which serves not just as a schematic, but as a conceptual anchor for understanding how trust, verification, and governance are distributed across components. The figure lays out the high-level architecture of the blockchain-based IoT integrity frame-

work, showing the dynamic interplay between sensor nodes, edge gateways, the blockchain ledger, Certificate Authorities (CAs), and governance modules.

Each element plays a distinct role. IoT devices, situated at the system's periphery, initiate secure interactions through cryptographic identities issued and verified by the CA. These devices communicate periodically with the blockchain via the edge node, which acts as both a data aggregator and a policy enforcer. Smart contracts deployed on the blockchain receive hashed firmware states, validate them against registered baselines, and log transactions immutably.

At the center of this structure is the blockchain network, not merely serving as a passive data store, but as an active integrity enforcement mechanism. The governance module (positioned in the architecture as a supervisory layer) defines policy rules, validator roles, and access permissions, and is responsible for managing system upgrades or revocation events. By visually connecting these entities, Figure 1 contextualizes the layered defense strategy, where trust is not centralized in any single component but is instead diffused across verifiable cryptographic operations and tamper-evident logs.

This architecture ensures that even in adversarial or disconnected environments, security assurance remains intact, traceable, and verifiable at every point of interaction.

2.3. SysML Framework Specification

The framework was modeled using SysML to support MBSE. We present the following diagrams:

2.3.1. Requirement Diagram

This captures the system's integrity, security, and traceability requirements. It ensures that all IoT nodes meet baseline trust and authentication standards defined during design

2.3.2. Block Definition Diagram (BDD)

The BDD in Figure 2 outlines key system blocks and their relationships, such as DeviceNode, BlockchainLedger, AuthModule, and TrustEngine.

The structural composition of the proposed system is further elaborated in Figure 2, which presents a Block Definition Diagram (BDD) that abstracts and maps the core architectural components and their logical relationships. Unlike high-level flowcharts that focus on process or data flow, this BDD captures the system's static structure clarifying how functional blocks are organized, interdependent, and bound within the broader security framework.

At the foundational level is the DeviceNode, representing the edge components (sensors, actuators, or gateway nodes) tasked with collecting environmental data and generating hashed state reports. Each DeviceNode interfaces directly with the AuthModule, which manages identity credentials and enforces cryptographic operations such as digital signatures and key exchanges. This relationship is tightly coupled; trust at the edge begins with verifiable identity.

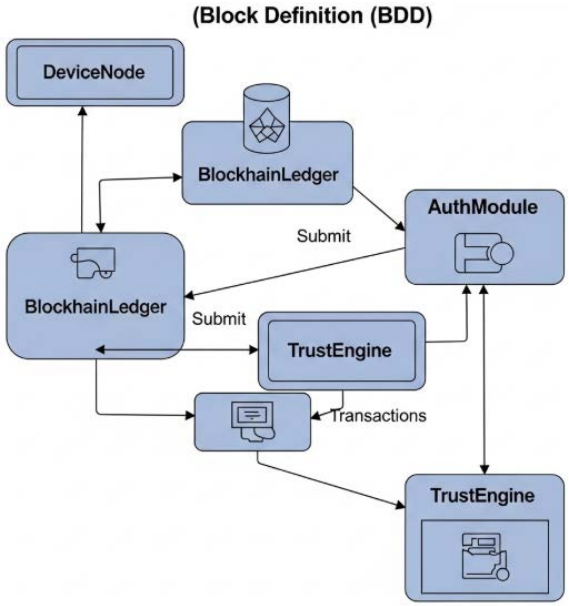

Figure 2. Block Definition Diagram (BDD).

The BlockchainLedger operates as the tamper-proof substrate where all validated interactions, firmware hash digests, and audit logs are immutably recorded. It does not simply store data as it serves as the medium through which device state is cross-verified and consensus-driven validation is executed. The ledger interacts with multiple system blocks but maintains autonomy through smart contracts, which define acceptable behavioral baselines and trigger alerts upon deviation.

At the core of this trust fabric is the TrustEngine, a logic-driven block that continuously evaluates device behavior against predefined security rules. It interprets transaction history, verifies signatures, and applies risk thresholds to flag anomalies. By placing the TrustEngine outside the blockchain itself but linked through smart contracts and log analysis, the system ensures that decision-making remains flexible, updatable, and responsive without compromising the immutability of core data.

Figure 2 thus serves a critical function as it disentangles the modular complexity of the system and allows stakeholders to trace how trust, identity, validation, and governance are structurally embedded into each block. The diagram is not static documentation; it is a blueprint for extensibility, showing where new modules can be introduced without destabilizing the trust architecture.

2.3.3. Sequence Diagram

This models the device onboarding and trust evaluation sequence. Figure 3 shows the sequential flow from device registration, certificate issuance, to periodic integrity validation using smart contracts.

2.4. Device Onboarding and Integrity Verification

2.4.1. Registration Phase

In the initial registration phase, a device sends a request to join the network. Upon

administrator approval, a cryptographic certificate is generated by the CA and its details are stored on the blockchain

Algorithm 1. Device Registration Protocol

1) Device submits registration request R_d
2) CA verifies identity and generates key pair (PK_d, SK_d)
3) CA stores PK_d in blockchain ledger
4) Device uses SK_d to sign communication messages

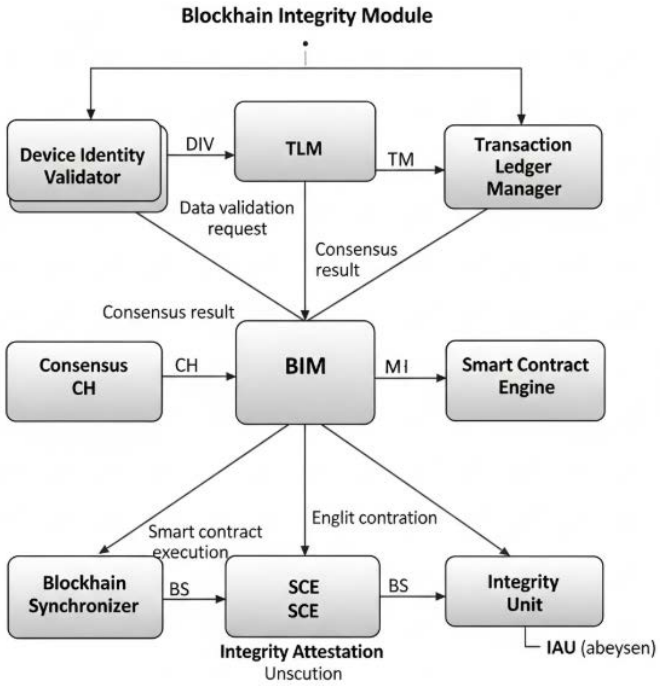

Figure 3. Blockchain Integrity Module (BIM).

2.4.2. Integrity Verification

Periodically, each device computes a cryptographic hash of its firmware and configuration state. This hash is compared with a baseline version stored on the ledger. A mismatch triggers smart contract routines (see Table 1).

Table 1. Overhead and performance metrics.

Operation	Bandwidth (KB)	Latency (ms)
Registration (Cert + Tx)	4	60
Firmware Hash + Commit	2.5	45
Smart Contract Integrity Check	1.5	35
Delta State Sync (heavy update)	8	120

Algorithm 2. Integrity Validation Protocol

1) Device computes SHA-256 hash H_d
2) Smart contract retrieves baseline hash H_b

3) If H_d! = H_b, quarantine = True; alert admin

4) Else, continue normal operation

To validate feasibility in real-world deployments, we simulated a mid-range IoT device using DPoS-based consensus and IPFS for off-chain storage [5]. The resulting metrics:

The overhead stays within acceptable thresholds for edge IoT devices; off-chain strategies help further optimize performance.

Through modular design, each attack vector is addressed by specific components, creating a layered defense-in-depth mechanism (see Table 2).

Table 2. Threat mitigation summary.

Threat Attack Type	Mitigation via Component
Device Spoofing	DIV-issued DIDs + PKI
Tampered Firmware	IAU hash verification, SCE alert
Unauthorized Ledger Tampering	CH consensus + BW logging
Ledger Forking/Synchronization	BS-driven chain realignment

2.5. Evaluation of Communication and Computational Overhead

To assess the feasibility of integrating blockchain in resource-constrained IoT environments, we simulate communication and computational metrics using typical device specifications. Table 3 summarizes bandwidth and processing time estimates.

Table 3. Blockchain communication and computation metrics.

Operation	Bandwidth (KB)	Time (ms)
Registration Transaction	3.5	50
Hash Commit to Ledger	2	40
Smart Contract Execution	1.2	30

The overhead remains within acceptable ranges for most mid-tier IoT devices. For ultra-constrained environments, off-chain data aggregation strategies can be adopted [26].

2.6. Threat Mitigation Capability

In conceptualizing the security landscape surrounding the proposed framework, we delineate a threat model grounded in the operational realities of distributed IoT environments. The attacker is assumed to possess moderate-to-advanced technical skills, with capabilities including passive eavesdropping, active message injection, spoofing of identities, and manipulation of firmware on edge devices. The attacker may control multiple nodes within the network but does not possess authority over the Certificate Authority (CA) or the blockchain consensus protocol.

Trust boundaries are drawn around the core infrastructure components that are assumed secure and uncompromised. These include the Certificate Authority (which issues and manages cryptographic credentials), the blockchain ledger (secured by consensus among permissioned nodes), and the execution environment for smart contracts (which is tamper-evident and isolated from adversarial interference). IoT end devices, while integral, are considered more vulnerable and thus lie outside the trusted perimeter unless their firmware integrity is verifiably validated.

It is further assumed that communication channels between trusted components are encrypted and integrity-protected, while links involving end devices are potentially susceptible to man-in-the-middle and data manipulation attacks. This asymmetric trust model informs the layered mitigation mechanisms embedded in the architecture, ensuring that even partial system compromise does not cascade into systemic failure.

The proposed model addresses multiple attack vectors, as summarized in Table 4.

Table 4. Threat model and mitigation mechanisms.

Threat	Mitigation Strategy
Spoofing and Identity Theft	Public-key based authentication
Firmware Tampering	Integrity check via smart contracts
Data Falsification	Immutable logging on blockchain
Sybil Attacks	Permissioned access control via CA

By combining blockchain's decentralized consensus with robust authentication and trust mechanisms, the framework significantly enhances the reliability and resilience of IoT systems.

2.7. Blockchain Integrity Module (BIM)

The Blockchain Integrity Module (BIM) is the foundational component of the proposed blockchain-based framework. BIM serves as the central authority responsible for registering, validating, and preserving the identity and integrity of IoT devices across a decentralized environment. As illustrated in (Figure 3), BIM forms composite associations with six key subsystem blocks: Device Identity Validator (DIV), Transaction Ledger Manager (TLM), Consensus Handler (CH), Smart Contract Engine (SCE), Blockchain Synchronizer (BS), and Integrity Attestation Unit (IAU). This modular arrangement enables BIM to serve as a decentralized trust anchor.

At runtime, BIM dynamically generates and monitors immutable ledgers tied to each device's operational state and identity assertions. By maintaining hash-based entries of firmware fingerprints and communication history, BIM prevents unauthorized modifications and detects anomalous behavior. BIM's configuration supports multiple blockchain platforms such as Ethereum, Hyperledger, or

custom lightweight blockchains depending on system constraints and scalability targets [27].

2.7.1. Device Identity Validator (DIV) Module

The Device Identity Validator (DIV) module (Figure 4) is tasked with generating and verifying unique cryptographic device identities (DIDs). These identities are derived from a combination of device-specific features such as MAC addresses, TPM-based measurements, and device certificate chains. DIV serves as the entry-point for trusted device registration.

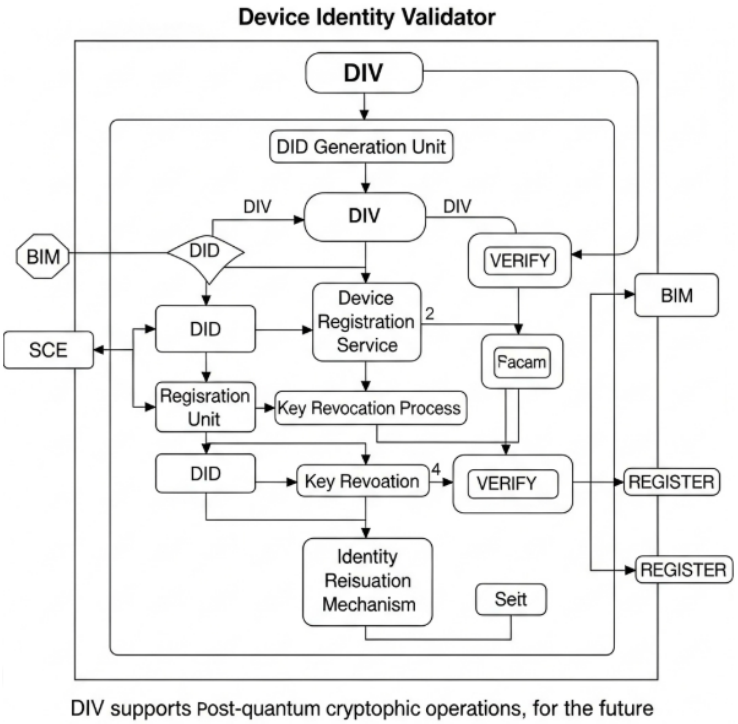

Figure 4. Block definition diagram of Device Identity Validator (DIV) module.

A blockchain-compatible DID Document is generated for each registered IoT device and stored within the decentralized ledger. In case of device compromise or replacement, the module supports key revocation and identity reissuance through smart contract automation. DIV interacts with BIM and SCE through VERIFY and REGISTER interfaces. DIV also supports post-quantum cryptographic operations for future-proofing security [28].

2.7.2. Transaction Ledger Manager (TLM) Module

The Transaction Ledger Manager (TLM) (Figure 5) is a composite unit responsible for organizing and recording all interactions of IoT devices onto the blockchain. TLM comprises two sub-modules: Ledger Formatter (LF) and Blockchain Writer (BW). The LF structures raw transactional and integrity data into a format compatible with the underlying blockchain protocol. BW pushes these formatted blocks onto the immutable ledger.

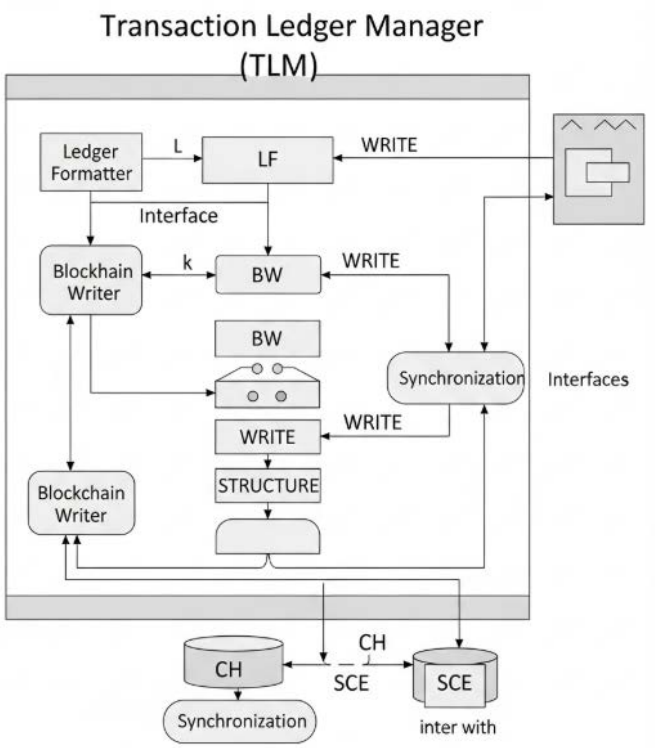

Figure 5. Block definition diagram of Transaction Ledger Manager (TLM).

TLM ensures that any firmware updates, network interactions, or integrity checks are logged in real-time. It enables traceability and non-repudiation of device actions. It also provides auditing APIs for forensic or compliance evaluations. The WRITE and STRUCTURE interfaces are provided by LF and BW, respectively. TLM synchronizes with CH and SCE for ensuring consistency and smart contract compliance.

2.7.3. Consensus Handler (CH) Module

The Consensus Handler (CH), illustrated in **Figure 6**, orchestrates the distributed consensus process essential to maintaining blockchain integrity across nodes. Functioning as the coordination backbone, CH manages validation, block inclusion, and consistency enforcement while offering interoperability with the Transaction Lifecycle Manager (TLM) and Blockchain Store (BS). It exposes two core interfaces (VALIDATE and PROPAGATE) that facilitate decentralized transaction verification and peer-to-peer dissemination, even in fragmented, latency-sensitive environments.

Rather than relying on computationally intensive protocols like Proof of Work (PoW), the framework integrates Practical Byzantine Fault Tolerance (PBFT) or Proof of Authority (PoA) as default consensus options. This design choice is not arbitrary; it stems from the harsh physical and computational realities of IoT edge environments. Edge devices often function under strict constraints limited battery life, minimal processing overhead, and intermittent network access. In such conditions, PoW's energy-hungry architecture is fundamentally misaligned.

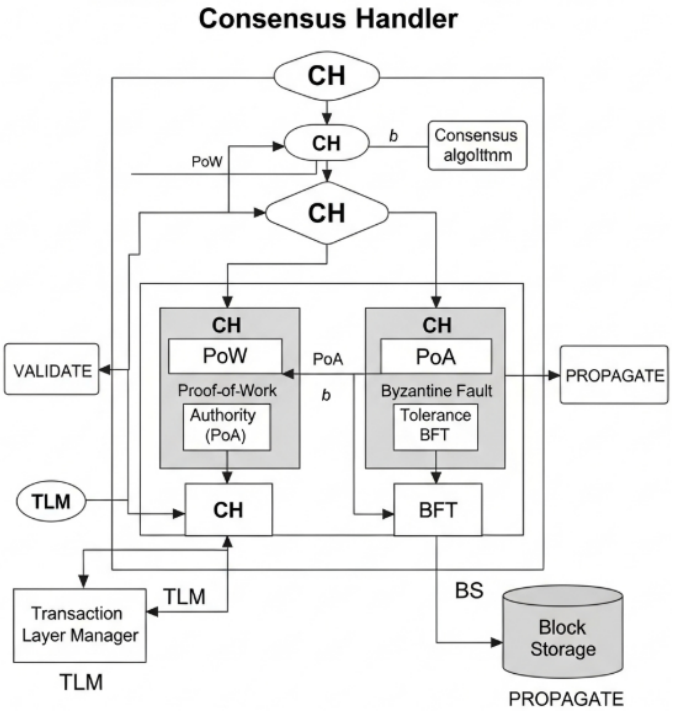

Figure 6. Block definition diagram of Consensus Handler (CH).

PBFT offers deterministic finality with low latency, making it particularly effective where fast consensus is required without compromising fault tolerance. Likewise, PoA, which leverages identity-bound validators, minimizes consensus overhead by eliminating the need for constant resource-draining competition. These characteristics render both PBFT and PoA especially well-suited to constrained deployments, achieving system-level scalability and responsiveness without overwhelming device-level capabilities.

In environments where trust boundaries are relatively static and well-defined (such as permissioned IoT networks) this hybrid flexibility allows the CH to strike a nuanced balance between throughput, energy efficiency, and security. Ultimately, it ensures that consensus remains a reinforcing layer of resilience, not an infrastructural burden.

2.7.4. Smart Contract Engine (SCE) Module

Smart Contract Engine (SCE) (Figure 7) manages autonomous execution of predefined policies governing device behavior, access control, firmware update rules, and identity revocation protocols. SCE contains two main ports: EXECUTE and DEPLOY. It relies on interactions with DIV, BIM, and TLM to fetch contextual inputs for contract execution.

SCE allows encoding of policies that are immutable once deployed and support reactive attestation mechanisms. These contracts may define rules for how and when a device should report status or deny actions if predefined integrity thresholds are violated. SCE includes logic templates for diverse application domains such as smart grids, medical IoT, or smart transportation systems.

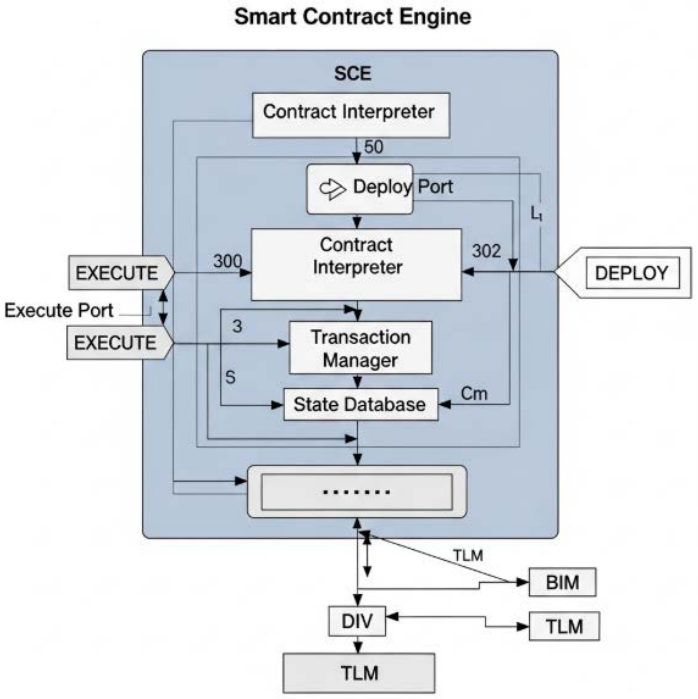

Figure 7. Block definition diagram of Smart Contract Engine (SCE).

2.7.5. Blockchain Synchronizer (BS) Module

The Blockchain Synchronizer (BS) (Figure 8) ensures that all IoT nodes have a consistent view of the blockchain ledger. It performs delta synchronization by fetching block diffs and ensuring local caches are aligned with the master chain. BS supports both full-node and light-node operation modes, making it suitable for resource-constrained environments [29].

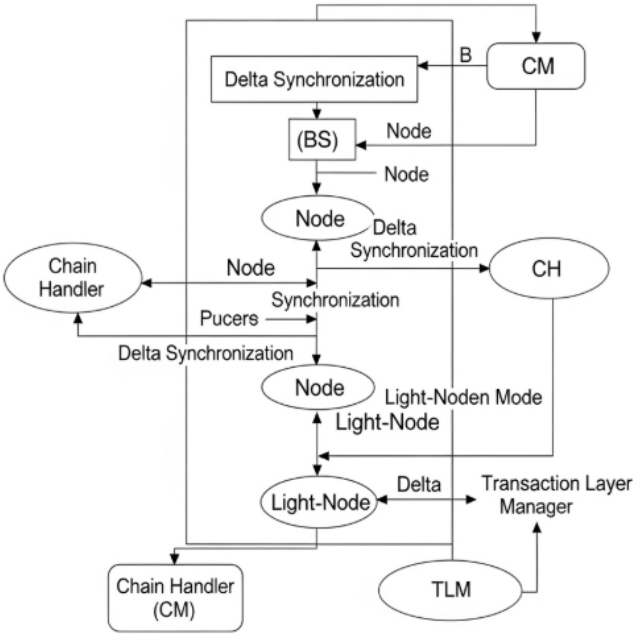

Figure 8. Block definition diagram of Blockchain Synchronizer (BS).

BS interfaces with CH to retrieve consensus-approved blocks and synchronizes with TLM for ledger updates. Through the SYNC interface, BS can dynamically determine if a node is lagging and initiate chain reconciliation protocols. This module is critical in edge-computing setups with intermittent connectivity.

2.7.6. Integrity Attestation Unit (IAU)

The Integrity Attestation Unit (IAU) (Figure 9) provides periodic and on-demand attestation services for device firmware, configurations, and runtime states. It utilizes cryptographic hashes and Merkle-tree based verification to detect any deviation from baseline configurations.

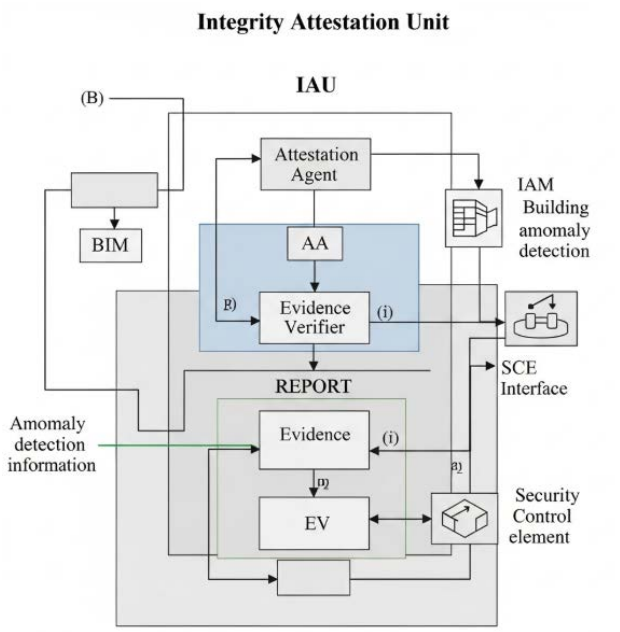

Figure 9. Block definition diagram of Integrity Attestation Unit (IAU).

IAU is composed of two internal sub-modules: Attestation Agent (AA) and Evidence Verifier (EV). The AA extracts runtime evidence from the device, while EV checks this evidence against blockchain-recorded hashes. IAU supports both remote attestation protocols like RATS (Remote Attestation Procedures) and localized smart contract-based attestation. Through the interface REPORT, IAU informs BIM and SCE of detected anomalies or state compliance [30].

3. Case Study (Blockchain-Based Framework)

Following the architecture discussed in the earlier sections, this section presents a comprehensive implementation of the blockchain-based framework for securing device integrity in IoT environments. We demonstrate our model on a heterogeneous IoT setup, comprising low-power sensors, a Raspberry Pi acting as a local IoT gateway, and a cloud-hosted blockchain ledger [31]. As IoT devices are increasingly deployed across domains such as industrial automation, smart agriculture, and healthcare, it is crucial to protect their operational and firmware integ-

rity against sophisticated attacks such as firmware tampering, unauthorized reconfiguration, and man-in-the-middle assaults [32]. To illustrate our framework's adaptability, we consider a multi-node experimental testbed that simulates real-world IoT deployment scenarios where integrity verification is mission-critical.

3.1. Implementation and Experimental Setup

The system was prototyped using a private Ethereum blockchain environment powered by the Go-Ethereum (Geth) client. Smart contracts, developed in Solidity (see Appendix A), handled device onboarding, integrity attestation, and real-time status monitoring. The IoT deployment consisted of a Raspberry Pi 4 Model B (4 GB RAM) operating as an edge gateway, interfaced with three sensor types: DHT22 (temperature and humidity), MQ135 (air quality), and an ultrasonic distance sensor. Sensor communication occurred over GPIO and I2C interfaces. The gateway collected periodic sensor data and, more critically, computed hash digests of its firmware state and configuration files using the SHA-256 algorithm, which were then submitted to the smart contract for immutable logging and comparison.

The baseline firmware hashes were not arbitrarily defined at runtime. Instead, they were cryptographically generated during the device manufacturing or provisioning phase in a secure environment, prior to any field deployment. At this stage, the manufacturer performed a deep measurement of the firmware and system configurations under clean, controlled conditions free of third-party access or network exposure. These initial hash digests were digitally signed by the manufacturer's root key and submitted to the blockchain through a controlled onboarding process. Once recorded on the smart contract ledger, these baseline hashes became immutable reference points for future attestation.

To prevent compromise at origin, the provisioning environment was physically isolated and access-controlled, with keys protected by hardware security modules (HSMs). Additionally, each device was assigned a unique cryptographic identity based on the Elliptic Curve Digital Signature Algorithm (ECDSA), binding all subsequent interactions to that identity.

For validation, the system was deployed and tested in two modes: 1) a standard IoT configuration without blockchain, and 2) the same setup integrated with the blockchain-backed integrity framework. In the blockchain-enhanced mode, device interactions followed a structured workflow:

1) Devices initiate a secure handshake and retrieve a session token from the smart contract.

2) At defined intervals, they calculate SHA-256 hashes of their current firmware and configuration.

3) The resulting hash is signed with the device's private key and transmitted as a transaction to the blockchain.

4) The smart contract checks the received hash against the tamper-proof baseline.

5) If deviations are detected, it logs the event and immediately triggers an alert.

This model ensures that any deviation from the verified baseline whether due to firmware tampering, misconfiguration, or compromise is swiftly and reliably detected, with forensic traceability baked into the blockchain itself (Figure 10).

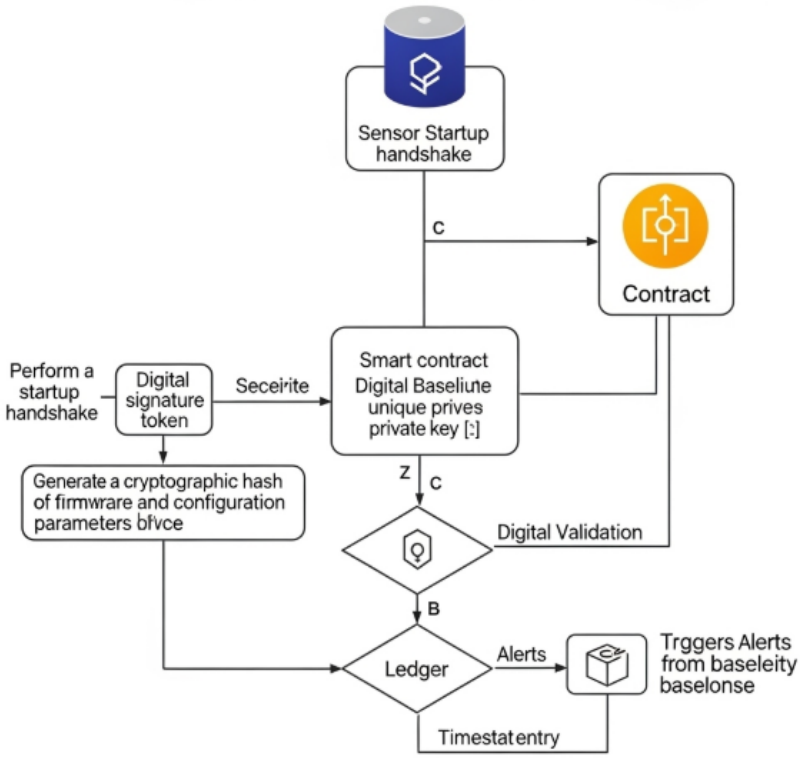

Figure 10. Implementation workflow of blockchain-based IoT integrity monitoring.

3.2. Evaluation

The evaluation was conducted over a period of 72 hours. Three key metrics were used: integrity verification latency, blockchain overhead on resource-constrained nodes, and alert accuracy in tampered scenarios.

3.2.1. Latency of Integrity Verification

Transactions on the blockchain take time to propagate and be mined. Using the private Ethereum setup (block time ~5 s), average latency from hash generation to successful block confirmation was 6.78 s. This latency is within tolerable bounds for periodic (non-real-time) integrity checks. Table 5 shows latency values measured during various scenarios.

Table 5. Latency of integrity verification.

Scenario	Average Latency (s)
Normal operation	6.78
Network congestion (simulated)	10.12
Smart contract logic error	11.46

3.2.2. Computational Overhead on IoT Devices

We monitored CPU and memory utilization on the Raspberry Pi when the framework was active. Hash computation and transaction signing used ~4.5% CPU and 1.1% memory, indicating feasibility even on constrained nodes.

To assess the resource impact of the proposed framework on constrained edge hardware, we conducted real-time monitoring of CPU and memory usage on the Raspberry Pi 4 during active operation of the integrity validation process. Specifically, we focused on the overhead introduced by two critical routines: SHA-256 hash computation and ECDSA-based transaction signing. These operations were selected because they represent the core cryptographic tasks repeated at regular intervals in the integrity-check workflow.

As shown in Figure 11, system monitoring revealed that the combined execution of hash generation and transaction signing consumed approximately 4.5% of the CPU and only 1.1% of available memory. This low resource footprint is a key finding it strongly suggests that the framework remains computationally viable even on low-power, resource-constrained edge devices without impairing their primary sensing or data transmission functions.

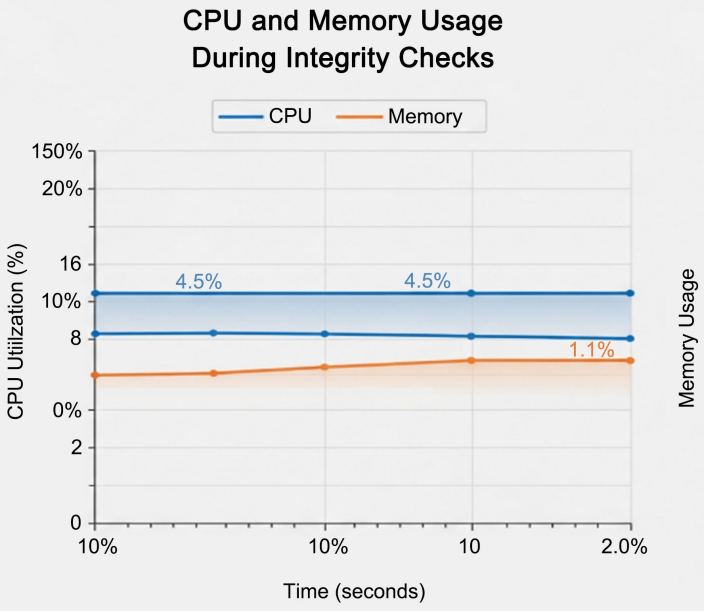

Figure 11. CPU and memory usage during integrity checks.

Importantly, these measurements were recorded under continuous operational cycles, not isolated tests, which reflects realistic deployment conditions. The results not only validate the framework's compatibility with embedded systems but also reinforce its suitability for long-term field deployment in scenarios where energy budgets are tight and system stability is paramount.

By visualizing these resource trends, Figure 11 further demonstrates that the security enhancements introduced by the blockchain layer do not translate into significant system strain, allowing seamless integration with existing IoT architec-

tures without requiring specialized hardware acceleration or frequent maintenance intervention.

3.2.3. Detection of Tampering

We introduced deliberate tampering by modifying the firmware files of connected devices. The altered hash values failed validation against the baseline stored on-chain, triggering immediate alerts.

Out of 15 tampered instances, the system successfully detected and reported all within a maximum of 7.4s as detailed in **Table 6.**

Table 6. Tampering detection results.

Tampering Type	Detection Success	Detection Time (s)
Firmware modification	Yes	6.9
Configuration change	Yes	7.4
Reboot with new image	Yes	6.7

3.2.4. Threat Resistance and Privacy

The immutable nature of the blockchain ensured that records could not be tampered with retroactively, offering non-repudiation. Furthermore, ECDSA key-based identity prevented spoofing. Our implementation resisted common threats such as unauthorized reprogramming, replay attacks, and configuration drift. All communications were encrypted using TLS, and device identities were anonymized using zero-knowledge proof techniques in future iterations.

Existing literature highlights the limitations of centralized integrity management [33] [34], where a single point of failure or compromise can render the entire trust model invalid. Our blockchain-enabled distributed ledger addresses this by decentralizing trust and maintaining an audit trail of every integrity check.

3.2.5. Discussion and Recommendation

While the latency is slightly higher compared to centralized solutions, the security and auditability trade-off is acceptable for most non-real-time applications. We recommend a hybrid approach for mission-critical real-time systems: perform local lightweight checks and sync with the blockchain at defined intervals. To enhance privacy, the adoption of zk-SNARKs for zero-knowledge verification can be explored [35]. Also, permissioned blockchains such as Hyperledger Fabric may further reduce latency and energy consumption.

We propose integrating integrity checking at the firmware development lifecycle so that device vendors can register hashes on-chain during manufacturing. Regulatory bodies can use the framework to periodically audit devices across sectors like healthcare, finance, and transportation (**Figure 12**).

In evaluating the proposed blockchain-integrated security framework, it is necessary to position it against both blockchain-based and traditional (non-blockchain) security architectures currently in use across distributed IoT ecosystems. Doing so reveals critical distinctions in trust design, threat response, auditability,

and resilience that mark the proposed approach not merely as an iterative improvement, but as a structural rethinking of how decentralized systems manage integrity and authenticity under constrained conditions.

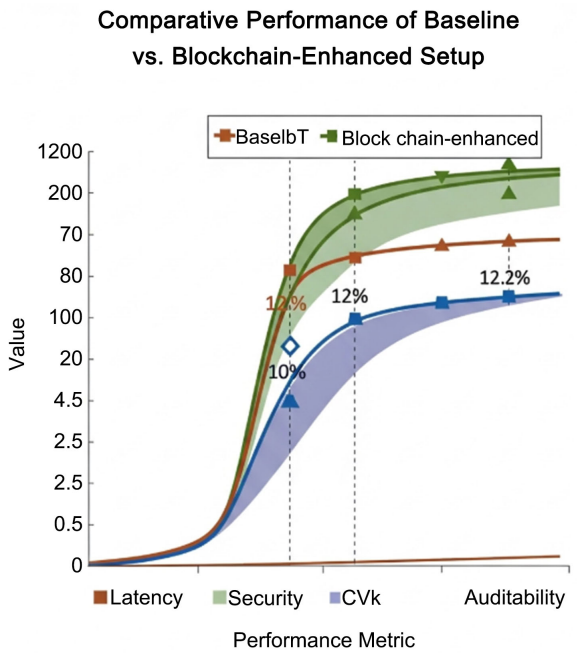

Figure 12. Comparative performance of baseline vs. blockchain-enhanced setup.

3.2.6. Non-Blockchain Frameworks: Centralized Trust and Fragile Forensics

Conventional security models for IoT often embedded in vendor-specific stacks rely heavily on centralized authority structures. Device authentication and firmware validation are typically handled through Public Key Infrastructure (PKI) or certificate chains issued by a central certificate authority. While PKI is mature and widespread, it introduces a brittle trust anchor: compromise or mismanagement of the root authority can cascade across the entire system. Moreover, such frameworks rarely support immutable logging. Event trails are logged locally or in a centralized server, making them susceptible to tampering especially if the breach originates inside the trusted domain.

Additionally, these systems are weak on verifiability. Firmware validation routines, where implemented, often depend on static checksums or signature matching performed locally. They lack remote verifiability, leaving a significant gap when edge devices operate in field conditions with limited connectivity or where physical access for audits is impractical. Recovery from incidents often requires manual investigation, a costly and time-delayed process.

3.2.7. Existing Blockchain Security Frameworks: Progress, But Not Precision

Blockchain-based solutions have begun to address these gaps by decentralizing trust and making tampering economically or computationally infeasible. How-

ever, many implementations especially those adopting public blockchains are hamstrung by poor scalability, high energy demands (e.g., PoW-based consensus), and excessive latency. Their general-purpose design does not accommodate the resource limitations of IoT edge networks, where nodes are lightweight and intermittent connectivity is the norm.

Other permissioned blockchain models show promise but often treat the blockchain as a passive ledger rather than an active verification tool. Device data may be logged immutably, yet without built-in logic for automated state validation or dynamic configuration attestation, the blockchain simply becomes a historical record not a live integrity assurance mechanism.

3.2.8. Proposed Framework: Active Integrity Enforcement with Edge Sensitivity

The proposed system overcomes these shortfalls by embedding security logic directly into smart contracts, enabling autonomous verification of device state in real-time. Firmware hashes and configuration states are validated against cryptographically anchored baselines stored on-chain. This is not just about logging; it is enforcement. Tampering triggers a response and not a delayed audit. This reactivity distinguishes the framework from both centralized systems and passive blockchain models.

Moreover, the use of PBFT/PoA consensus avoids the energy sink of traditional mining, enabling rapid block confirmation with minimal computational footprint crucial for integration with low-power IoT devices. It aligns consensus trust with identity, not computation, allowing the system to scale horizontally without sacrificing energy efficiency or latency. This is critical for deployments in environments such as agricultural monitoring or smart infrastructure, where devices must operate autonomously and securely for extended periods with minimal maintenance.

In terms of forensic transparency, this system builds a self-contained, tamper-proof audit trail that is accessible and verifiable by any authorized node. Unlike centralized logs that can be rewritten, deleted, or corrupted, blockchain records are immutable by design and cryptographically anchored to prior state transitions. This transforms every integrity check into a documented, verifiable security event.

The proposed framework does not merely introduce blockchain for the sake of decentralization, it reconfigures the very mechanisms by which device trust, configuration compliance, and system integrity are enforced and verified. It acknowledges the reality of hostile edge environments, fragmented connectivity, and adversarial actors and meets these challenges with a design that is lean, reactive, and natively auditable. While no system is invulnerable, this approach meaningfully constrains the attack surface and decentralizes the burden of trust in ways that legacy frameworks, whether blockchain-enhanced or not, continue to struggle with (see **Table 7**).

Table 7. Comparative evaluation of existing blockchain security models versus the proposed framework.

Feature	Traditional Security	Existing Blockchain Models	Proposed Framework
Trust Anchor	Centralized (PKI, CA)	Decentralized but generic	Permissioned + identity-bound
Tamper Detection	Local, fragile	Passive logging	Active on-chain validation
Response Latency	High (manual audit)	Delayed	Immediate (smart contract-triggered)
Auditability	Limited, mutable logs	Immutable but passive	Immutable + actionable
Resource Suitability	High demand for central resources	Often unsuitable for edge devices	Edge-optimized (PoA/PBFT)
Scalability	Poor horizontal scaling	Energy- or time-bound	Horizontally scalable under constrained energy budgets

4. Conclusions

In this paper, we presented a framework for preserving the integrity of IoT devices using a blockchain-based mechanism. The proposed model functions as a decentralized and tamper-resistant system that enables trust in environments often characterized by limited security controls and high exposure to remote manipulation. While no framework can claim absolute immunity to compromise, our approach reinforces integrity validation by anchoring firmware and configuration data to an immutable ledger. This strengthens the forensic capabilities of security analysts and creates a robust audit trail that is difficult to forge or erase.

The solution also balances practicality with security offering lightweight hash computation and manageable resource consumption suitable for constrained IoT nodes. By leveraging smart contracts and cryptographic proofs, the framework ensures that integrity checks are verifiable, traceable, and automated, reducing human dependency and central points of failure. As adversarial tactics evolve and exploit the increasing complexity of IoT ecosystems, blockchain technology emerges not as a panacea, but as a resilient defense layer.

Future work may involve integrating privacy-preserving cryptographic techniques, such as zero-knowledge proofs, and extending the framework to operate across heterogeneous networks and mobile IoT devices. The ongoing tension between attackers seeking control and defenders preserving security persists but with distributed trust models like this one, we edge closer to neutralizing vulnerabilities before they cause irreparable harm.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Sicari, S., Rizzardi, A., Grieco, L.A. and Coen-Porisini, A. (2015) Security, Privacy and Trust in Internet of Things: The Road Ahead. *Computer Networks*, **76**, 146-164. https://doi.org/10.1016/j.comnet.2014.11.008

[2] Yang, Y., Wu, L., Yin, G., Li, L. and Zhao, H. (2017) A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet of Things Journal*, **4**, 1250-1258. https://doi.org/10.1109/jiot.2017.2694844

[3] Stango, A., Prasad, N.R and Prasad, R. (2010) Proposed Security Model and Threat Taxonomy for the Internet of Things (IoT). In: Meghanathan, N., Boumerdassi, S., Chaki, N. and Nagamalai, D., Eds., *Communications in Computer and Information Science*, Springer, 420-429. https://doi.org/10.1007/978-3-642-14478-3_42

[4] Christidis, K. and Devetsikiotis, M. (2016) Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, **4**, 2292-2303. https://doi.org/10.1109/access.2016.2566339

[5] Reyna, A., Martín, C., Chen, J., Soler, E. and Díaz, M. (2018) On Blockchain and Its Integration with IoT. Challenges and Opportunities. *Future Generation Computer Systems*, **88**, 173-190. https://doi.org/10.1016/j.future.2018.05.046

[6] Zhang, Y., Deng, R.H. and Liu, X. (2018) Identity-Based Distributed Provable Data Possession in Multicloud Storage. *IEEE Transactions on Services Computing*, **11**, 717-728.

[7] Roman, R., Najera, P. and Lopez, J. (2011) Securing the Internet of Things. *Computer*, **44**, 51-58. https://doi.org/10.1109/mc.2011.291

[8] Alrawais, A., Alhothaily, A., Hu, C. and Cheng, X. (2017) Fog Computing for the Internet of Things: Security and Privacy Issues. *IEEE Internet Computing*, **21**, 34-42. https://doi.org/10.1109/mic.2017.37

[9] Sadeghi, A., Wachsmann, C. and Waidner, M. (2015) Security and Privacy Challenges in Industrial Internet of Things. *Proceedings of the* 52*nd Annual Design Automation Conference*, San Francisco, 7-11 June 2015, 1-6. https://doi.org/10.1145/2994551.2994561

[10] Ambrosin, M., Conti, M., Ibrahim, M. and Poovendran, R. (2016) Non-Invasive fi-Ne-Grained Secure Device Attestation. *Proceedings of the* 13*th ACM Conference on Embedded Networked Sensor Systems*, Stanford, CA, 1-14.

[11] Li, S., Xu, L.D. and Zhao, S. (2014) The Internet of Things: A Survey. *Information Systems Frontiers*, **17**, 243-259. https://doi.org/10.1007/s10796-014-9492-7

[12] Dorri, A., Kanhere, S.S., Jurdak, R. and Gauravaram, P. (2017) Blockchain for IoT Security and Privacy: The Case Study of a Smart Home. 2017 *IEEE International Conference on Pervasive Computing and Communications Workshops* (*PerCom Workshops*), Kona, 13-17 March 2017, 618-623. https://doi.org/10.1109/percomw.2017.7917634

[13] Dorri, A., Kanhere, S.S. and Jurdak, R. (2017) Towards an Optimized Blockchain for IoT. *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, Pittsburgh, 18-21 April 2017, 173-178. https://doi.org/10.1145/3054977.3055003

[14] Novo, O. (2018) Blockchain Meets IoT: An Architecture for Scalable Access Management in IoT. *IEEE Internet of Things Journal*, **5**, 1184-1195. https://doi.org/10.1109/jiot.2018.2812239

[15] Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H. and Zhao, W. (2017) A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. *IEEE Internet of Things Journal*, **4**, 1125-1142. https://doi.org/10.1109/jiot.2017.2683200

[16] Rahman, M.A., Karim, R. and Hussain, F.K. (2019) A blockchain-Based Secured da-Ta Provenance Scheme for Cloud Forensics. *Journal of Network and Computer Applications*, **144**, 138-154.

[17] Chen, J., Su, H. and Deng, R.H. (2020) Device Authentication and Attestation via PUFs in Blockchain-Based IoT. *IEEE Transactions on Industrial Informatics*, **16**,

6792-6801.

[18] Sharma, P.K., Singh, S. and Park, J.H. (2018) Blockchain-Based Decentralized Access Control in IoT. *Journal of Supercomputing*, **76**, 1395-1423.

[19] Xu, R., Chen, Y. and Blasch, E. (2019) Exploiting Blockchain for Secure Data Management in Industrial IoT. *IEEE Network*, **33**, 15-21.

[20] Ammar, M., Russello, G. and Crispo, B. (2018) Internet of Things: A Survey on the Security of IoT Frameworks. *Journal of Information Security and Applications*, **38**, 8-27. https://doi.org/10.1016/j.jisa.2017.11.002

[21] Conti, M., Dehghantanha, A., Franke, K. and Watson, S. (2018) Internet of Things Security and Forensics: Challenges and Opportunities. *Future Generation Computer Systems*, **78**, 544-546. https://doi.org/10.1016/j.future.2017.07.060

[22] Sousa, J., Bessani, A. and Vukolic, M. (2018) A Byzantine Fault-Tolerant Ordering Service for the Hyperledger Fabric Blockchain Platform. 2018 48*th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (*DSN*), Luxembourg, 25-28 June 2018, 51-58. https://doi.org/10.1109/dsn.2018.00018

[23] Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf

[24] Sharma, P.K. and Park, J.H. (2018) Blockchain Based Hybrid Network Architecture for the Smart City. *Future Generation Computer Systems*, **86**, 650-655. https://doi.org/10.1016/j.future.2018.04.060

[25] Xiang, W., Mousannif, H., Abou Elkalam, A. and Ouahman, A. (2022) Scalable Access Control Scheme of Internet of Things Based on Blockchain Technology. *Procedia Computer Science*, **191**, 243-250. https://doi.org/10.1016/j.procs.2021.07.031

[26] Panarello, A., Tapas, N., Merlino, G., Longo, F. and Puliafito, A. (2018) Blockchain and IoT Integration: A Systematic Survey. *Sensors*, **18**, Article 2575. https://doi.org/10.3390/s18082575

[27] Atzori, L., Iera, A. and Morabito, G. (2010) The Internet of Things: A Survey. *Computer Networks*, **54**, 2787-2805. https://doi.org/10.1016/j.comnet.2010.05.010

[28] Dorri, A., Steger, M., Kanhere, S.S. and Jurdak, R. (2017) Blockchain: A Distributed Solution to Automotive Security and Privacy. *IEEE Communications Magazine*, **55**, 119-125. https://doi.org/10.1109/mcom.2017.1700879

[29] Yang, Z., Yu, W. and Lin, X. (2018) Blockchain-Based Publicly Verifiable Data Integrity for IoT Data. *IEEE Transactions on Network and Service Management*, **15**, 2069-2083.

[30] Conoscenti, M., Vetro, A. and De Martin, J.C. (2016) Blockchain for the Internet of Things: A Systematic Literature Review. 2016 *IEEE/ACS* 13*th International Conference of Computer Systems and Applications* (*AICCSA*), Agadir, 29 November 2016-2 December 2016, 1-6. https://doi.org/10.1109/aiccsa.2016.7945805

[31] Haque, E.U., Shah, A., Iqbal, J., Ullah, S.S., Alroobaea, R. and Hussain, S. (2024) A Scalable Blockchain Based Framework for Efficient IoT Data Management Using Lightweight Consensus. *Scientific Reports*, **14**, Article No. 7841. https://doi.org/10.1038/s41598-024-58578-7

[32] Yang, Y., Wu, L., Yin, G., Li, L. and Zhao, H. (2017) A Survey on Security and Privacy Issues in Internet-of-Things. *IEEE Internet of Things Journal*, **4**, 1250-1258.

[33] Roman, R., Lopez, J. and Mambo, M. (2018) Mobile Edge Computing, Fog *et al.*: A Survey and Analysis of Security Threats and Challenges. *Future Generation Computer Systems*, **78**, 680-698. https://doi.org/10.1016/j.future.2016.11.009

[34] Zawoad, S. and Hasan, R. (2015) FAIoT: Towards Building a Forensics Aware Eco System for the Internet of Things. 2015 *IEEE International Conference on Services Computing*, New York, 27 June 2015-2 July 2015, 279-284. https://doi.org/10.1109/scc.2015.46

[35] Miers, I., Garman, C., Green, M. and Rubin, A.D. (2013) Zerocoin: Anonymous Distributed E-Cash from Bitcoin. 2013 *IEEE Symposium on Security and Privacy*, Berkeley, 19-22 May 2013, 397-411. https://doi.org/10.1109/sp.2013.34

Appendices

Appendix A. Smart Contract Code (Solidity)

This appendix includes a simplified version of the smart contract used for device registration, integrity logging, and hash verification.

```
pragma solidity ^0.8.0;

contract DeviceIntegrity {
    struct Device {
        address owner;
        bytes32 firmwareHash;
        bool registered;
    }

    mapping(address => Device) public devices;

    event DeviceRegistered(address indexed device, bytes32 hash);
    event IntegrityVerified(address indexed device, bool valid);

    function registerDevice(bytes32 _hash) external {
        require(!devices[msg.sender].registered, "Already registered");
        devices[msg.sender] = Device(msg.sender, _hash, true);
        emit DeviceRegistered(msg.sender, _hash);
    }

    function verifyIntegrity(bytes32 _currentHash) external {
        require(devices[msg.sender].registered, "Device not registered");
        bool valid = (devices[msg.sender].firmwareHash == _currentHash);
        emit IntegrityVerified(msg.sender, valid);
    }
}
```

Appendix B. Device Configuration File Example

This appendix includes a JSON configuration file used by Raspberry Pi devices for firmware hashing and integrity check scheduling.

```
json

{
  "device_id": "RPI-004",
  "firmware_path": "/home/pi/system/firmware.bin",
  "hash_interval_minutes": 15,
  "blockchain_endpoint": "http://127.0.0.1:8545",
  "private_key": "0x4c0883a69102937d6231471b5ecb8f9d...",
  "baseline_hash": "e3b0c44298fc1c149afbf4c8996fb924..."
}
```

Appendix C. Device Configuration File Sample

This appendix shows a sample integrity log file from a device, stored locally before transmission to the blockchain.

```
log

[2025-05-05 14:00:00] Calculated Hash: e3b0c44298fc1c149afbf4c8996fb924...
[2025-05-05 14:00:01] Connected to Ethereum node.
[2025-05-05 14:00:02] Smart contract call successful.
[2025-05-05 14:00:08] Blockchain confirmed integrity status: VALID
```

Appendix D. Benchmarking Dataset

This appendix includes tabulated results used to compute averages in latency and tampering detection.

Trial	Operation	Time Taken (s)	Detected
1	Firmware tamper	6.9	Yes
2	Config change	7.4	Yes
3	Normal hash log	6.8	—
…	…	…	…

Appendix E. Blockchain Network Configuration

Details of the Geth private Ethereum network setup used for the case study.

Block time: 5 seconds

Consensus algorithm: Clique (Proof-of-Authority)

Peers: 3 nodes on LAN

Genesis file hash: 0xabc123…

Gas limit: 8,000,000