Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Split-Plane Decentralized Middleware For Secure Multi-Protocol Iot Communication: Architecture, Formal Verification, And Experimental Validation

Ahmed Swar[1]*, Mohammed Belal[1], Soha Ahmed Ehssan Aly[1]

[1]Computer Science Department, Faculty of Computers and Artificial Intelligence, Capital University (Formerly Helwan University), Cairo 12317, Egypt
ahmed.swar@fci.helwan.edu.eg , belal@fci.helwan.edu.eg , dr.soha@fci.helwan.edu.eg

Abstract

The rapid expansion of Internet of Things (IoT) deployments across smart cities, industrial automation, and critical infrastructure has exposed fundamental limitations in prevailing IoT security architectures. Centralized cloud-based solutions introduce single points of failure and incur latency incompatible with real-time operation, while fully decentralized blockchain-based approaches suffer from scalability and throughput constraints. This paper proposes a decentralized middleware for secure multi-protocol IoT communication and analytics based on a split-plane architecture that explicitly decouples real-time security enforcement from auditability and analyt-ics. The middleware decomposes into four independently optimized planes: a Security Plane with distributed Policy Decision Points synchronized via gossip protocols for sub-40 ms admission decisions; a Data Plane using RabbitMQ Quorum Queues and Apache NiFi for fault-tolerant, high-throughput protocol mediation; an Analytics Plane employing a two-stage Random Forest and Deep Autoen-coder intrusion detection pipeline with ADWIN-based concept drift detection and Krum-robust federated aggregation; and an Audit Plane backed by Hyperledger Fabric with periodic Merkle-root anchoring to Polygon PoS at a cost of approximately $2.16 USD per month. Formal verification using TLA+ exhaustively explores over 1.84 million states to confirm that the audit-layer consensus pre-serves agreement and validity under crash-fault assumptions, with zero safety violations detected. Extensive experimental evaluation across five public benchmark datasets on both cloud-tier (Intel i7) and edge-tier (Raspberry Pi 4) hardware demonstrates that the system maintains peak throughputs exceeding 10,000 transactions per second, incurs less than 15% security overhead, achieves aver-age enforcement latencies below 40 ms, and sustains detection F1-scores up to 0.97. A systematic ablation study confirms that each plane contributes necessarily: disabling audit-plane decoupling increases latency by 379%, while removing adaptive trust decay raises the false positive rate by 158%. These results confirm that decentralized trust and real-time IoT performance can be reconciled through explicit architectural separation

Keyword: Internet of Things (IoT), Zero Trust Architecture, Decentralized Middleware, Split-Plane Architecture, IoT Security, Trust Manage-ment, Blockchain-based Auditability, Intrusion Detection, Concept Drift, Formal Verification, Microservices.

Introduction

The Internet of Things (IoT) has matured from a conceptual framework into the backbone of modern digital infrastructure. Today, vast networks of geographically scattered, heterogeneous devices drive large-scale automation and data-driven decision-making. From smart transportation to healthcare monitoring, these systems completely rely on continuous, low-latency communication.

Because these deployments increasingly dictate physical safety, core properties like reliability and security have transitioned from mere design choices to mandatory operational limits [1], [2]. Yet, securing this expanded footprint presents a formidable challenge. The sheer diversity of IoT hardware, combined with long operational lifecycles and heavily constrained edge resources, makes enforcing uniform security policies practically impossible. Furthermore, because nodes frequently operate in physically accessible environments,

they are uniquely exposed to hardware compromise, cloning, and impersonation. When multiplied across millions of deployed nodes, these localized vulnerabilities create a massive surface attack. Adversaries routinely exploit this scale, weaponizing weakly protected edge devices to orchestrate massive botnets and distributed denial-of-service (DDoS) campaigns [3], [4].

For years, the industry leaned on centralized, cloud-hosted services to handle IoT authentication and oversight. We understand the appeal: centralization makes policy management easy. Unfortunately, it also introduces fatal architectural flaws. The most obvious is the single point of failure. Beyond that, forcing a constrained edge device to communicate with a remote server adds severe round-trip latency. If the network partitions or connectivity drops, system resilience shatters. In industrial cyber-physical systems, waiting for a remote security decision isn't just an inconvenience. It directly risks physical damage, stalls production, and creates life-safety hazards [5], [6].

Looking for a way out, the research community recently turned into fully decentralized, blockchain-backed security. The theoretical benefits are strong, as distributed ledgers provide immutable, verifiable trust. However, our analysis shows this approach simply trades a latency bottleneck for a throughput bottleneck. Tying operational IoT data flows directly to cryptographic consensus mechanisms destroys system performance. Even if we strip the ledger down to a highly optimized, permissioned model, the mandatory consensus overhead creates a hard ceiling on throughput. In the end, these constraints make traditional blockchain models fundamentally incompatible with the high-frequency, real-time control loops required by modern IoT [7], [8].

This fundamental tension between scalability and trust motivates the need for new architectural approaches to IoT security. Zero Trust Architecture (ZTA), which enforces continuous verification and eliminates implicit trust assumptions, has gained significant attention as a modern security paradigm. Unlike perimeter-based models, ZTA requires every access request to be evaluated dynamically based on identity, context, and behavior. Although ZTA has proven effective in enterprise IT environments, its direct application to IoT systems remains challenging because of device resource constraints, protocol diversity, and stringent latency requirements at the network edge [9], [10]. We term this challenge the "Zero-Trust Delay Penalty": the latency cost that continuous per-request verification imposes on systems where decisions must be completed within single-digit-millisecond control loops.

The current state of the art reveals a significant research gap, as existing IoT security solutions commonly address isolated aspects of the problem, rather than providing end-to-end security middleware suitable for heterogeneous, real-time environments. Specifically, three recurring deficiencies emerge from the literature:

1. **Decoupling gap:** Centralized platforms prioritize operational simplicity at the cost of resilience, while blockchain frameworks strengthen integrity but incur overheads incompatible with latency-sensitive workloads. No existing middleware explicitly separates enforcement from audit persistence at the architectural level.
2. **Analytics integration gap:** AI-driven intrusion detection systems enhance visibility but are frequently deployed as standalone components with no closed-loop feedback to enforcement. Concept drift the gradual evolution of IoT traffic distributions is widely acknowledged but rarely addressed within operational middleware pipelines.
3. **Verification gap:** Distributed audit and consensus components are increasingly adopted in IoT middleware, yet their correctness properties are almost never formally verified, relying instead on informal arguments or empirical testing alone.

To bridge these gaps, this paper proposes decentralized middleware for secure multi-protocol IoT communication and analytics based on a Split-Plane Architecture. The proposed middleware reconciles the competing demands of low latency and decentralized trust by explicitly decoupling the real-time security enforcement from data processing, analytics, and auditability. This architectural separation enables the integration of zero-trust principles, adaptive trust management, and federated analytics, without compromising the performance required by modern IoT applications. The middleware is implemented as a reproducible, containerized microservice system with explicit technology bindings for every architectural

plane. This study makes the following key contributions:

- **Architecture:** A decentralized split-plane middleware architecture is proposed that explicitly separates real-time security enforcement from data routing, analytics, and auditing. This allows each plane to be independently optimized for latency, throughput, and integrity. The architecture is implemented using RabbitMQ 3.13 (Quorum Queues) for fault-tolerant data routing, Apache NiFi 2.x for protocol normalization, and Docker/Kubernetes for orchestration.
- **Trust Management:** We introduce an adaptive trust decay mechanism governed by an exponential decay function that incorporates device type, expected reporting behavior, and temporal context to significantly reduce false positives (by 158% compared to static thresholds, as demonstrated in ablation) while maintaining the timely isolation of compromised devices.
- **Analytics Integration:** Federated analytics is integrated with ADWIN-based lightweight concept drift detection and a two-stage Random Forest + Deep Autoencoder IDS pipeline to enable adaptive security monitoring under nonstationary IoT behavior, thereby minimizing sensitive data centralization. Krum-based robust aggregation is employed to protect against federated learning poisoning attacks.
- **Integrity Assurance:** We design a hybrid blockchain anchoring mechanism that combines high throughput Hyperledger Fabric v2.5 permissioned ledger logging with periodic public-chain anchoring via Merkle root commitments to Polygon PoS, achieving verifiable auditability at an amortized cost of approximately $2.16 USD per month over three orders of magnitude cheaper than per-event on-chain writes.
- **Formal Verification:** TLA+ formal verification is employed to validate the safety properties of the audit-layer consensus model. The TLC model checker exhaustively explored over 1.84 million states and confirmed zero safety violations under crash-fault assumptions ($f \leq 2$ of $N = 5$ nodes). Two specification bugs were discovered and corrected during development, demonstrating the practical value of formal methods beyond documentation.
- **Comprehensive Evaluation:** Experimental validation is conducted using five public IoT security datasets on both cloud-tier (Intel i7) and edge-tier (Raspberry Pi 4) hardware, reporting performance metrics including throughput, latency, and detection accuracy across diverse attack profiles. A systematic ablation study quantifies the contribution of each architectural plane.

Related work

While significant strides have been made in individual domains, such as blockchain for trust, Zero Trust for access control, and machine learning for threat detection, the existing literature frequently highlights a fundamental tension between security rigor and operational performance. Most current frameworks treat these components as standalone solutions or suffer from tight architectural coupling, which introduces latency bottlenecks and single points of failure. This review examines these disparate research threads to identify the specific gaps in multi-protocol integration, real-time enforcement, and scalable auditability that the proposed split-plane middleware aims to resolve. The following subsections review the current landscape of IoT security, focusing on the evolution of perimeter-based models into decentralized, data-driven architectures.

Zero Trust for IoT and Industrial IoT (IIoT): Authentication, Policy Decision Point (PDP) Placement, and Continuous Verification

Zero Trust has been gaining ground in IoT circles, and for good reasons. Perimeter-based models were never a great fit for environments where devices can be physically accessed, cloned, or re-flashed. ZTA's insistence on verifying every request regardless of where it comes from is a better match for the threat model [9], [11]. That said, most ZTA proposals for IoT still evaluate policies at a centralized cloud service or a single edge gateway. This setup works for moderate traffic, but it buckles under scale. Every access request has to make a round trip to the PDP, and at thousands of requests per second, even a well-provisioned PDP becomes a chokepoint. A few studies have explored continuous authentication using lightweight cryptographic primitives, which helps with replay attacks but does not solve the latency problem at the middleware level [10], [11]. What is largely missing is a middleware-level pattern that preserves zero-trust semantics continuous, context-aware verification without funneling every decision through a single bottleneck. Our middleware addresses this by distributing policy decision logic to federated gateways that maintain in-memory policy caches synchronized via gossip. This cuts enforcement latency by 80–85% compared to centralized PDP designs.

Blockchain-Based IoT Middleware: Trust, Access Control, and Throughput Limits

Blockchain technology has been extensively explored as a foundation for trust management in IoT systems, primarily because of its append-only auditability and decentralized consensus, which can mitigate insider threats and prevent log tampering in distributed environments. Recent surveys show that blockchain is mostly applied to device identity management, access control, data provenance, and integrity preservation, with permissioned ledgers often preferred in IoT and IIoT settings, where governance control and performance requirements outweigh full decentralization. These properties are particularly attractive in smart cities and industrial deployments where accountability, traceability, and post-incident forensics are critical operational requirements [7], [8].

Despite these advantages, the dominant architectural pattern in many blockchain-enabled IoT middleware designs retains a tight coupling between operational events and ledger writes, such as write-per-event or write-per-packet-hash approaches. This coupling effectively pushes the consensus and commits overhead into the hot data path, resulting in reduced throughput and increased end-to-end latency, even when permissioned blockchains and optimized consensus protocols are employed. Empirical studies have consistently identified consensus latency and transaction throughput as the primary scalability bottlenecks under realistic IoT traffic rates.

More recent work has attempted to alleviate these limitations by integrating messaging middleware (e.g., message queues or stream brokers) with blockchain components to improve reliability and absorb bursty traffic. While such designs reduce perceived delay, asynchronous messaging is typically treated as a performance optimization rather than an explicit architectural separation of concerns. Consequently, security enforcement, data transport, and audit persistence remain logically intertwined, and blockchain interactions can still influence real-time behavior under load.

Consequently, many blockchain-based IoT middleware proposals lack a systematic split-plane architecture that (i) preserves real-time enforcement in the operational path, (ii) relegates auditability and integrity guarantees to an asynchronous plane, and (iii) maintains cryptographically verifiable linkage between enforcement decisions and immutable audit records [12], [13].

Gap addressed by our work: The proposed Audit Plane uses Hyperledger Fabric as a high-throughput private ledger with asynchronous Merkle-root anchoring to Polygon PoS. Our ablation study demonstrates that this decoupling is the single most impactful design decision: coupling audit writes synchronously to the hot path (as in pure blockchain designs) increases latency by 379% and reduces throughput by 68%.

Auditability at Scale: Merkle Proofs, Off-Chain Summaries, and Anchoring Patterns

To address the scalability limitations of blockchain-based systems, many recent architectures have adopted on-chain summaries combined with off-chain storage and computation. In these designs, Merkle trees, hash chains, or other cryptographic commitments are used to summarize large volumes of events, enabling efficient integrity verification without committing raw telemetry or logs to the ledger. This pattern has been widely adopted in high-event-rate environments because it preserves tamper evidence while significantly reducing the on-chain storage requirements and consensus overhead. In IoT settings, such approaches are particularly attractive because they allow integrity guarantees to be scaled with the device count and message frequency [14], [15].

In parallel, a growing body of work has focused on blockchain-integrated logging and log-auditing frameworks to provide tamper-evident accountability for security events, system logs, and access records. These studies consistently identified storage overhead, verification latency, and committed throughput as fundamental bottlenecks when blockchain is naively used for log persistence. Consequently, many proposals emphasize batch commitments, delayed anchoring, and hierarchical logging structures to avoid per-record blockchain writes while retaining verifiable integrity.

However, despite these advances, most existing solutions fall into one of two categories:
(i) integrity-focused storage systems that provide verifiable logs but remain largely disconnected from real-

time enforcement and access control decisions, or (ii) logging and auditing mechanisms developed in isolation from the operational constraints of IoT middleware, such as multi-protocol routing, real-time admission control, and adaptive trust management.

Consequently, what remains underexplored is a unified IoT middleware architecture that treats audit anchoring as a first-class audit plane is explicitly integrated with security enforcement and analytics rather than as a standalone integrity component appended to the data flow. Bridging this gap requires architectural decoupling that preserves real-time performance, while maintaining a cryptographically verifiable linkage between enforcement actions and immutable audit records [16], [17].

Gap addressed by our work: The Audit Plane is architecturally integrated with the Security and Analytics Planes trust-score changes, policy violations, and drift-triggered retraining events are automatically recorded while remaining fully decoupled from the Data Plane hot path. Our cost analysis shows this is financially viable at ~$2.16/month for public-chain anchoring.

Analytics for IoT Security: IDS, Concept Drift, Federated Learning, and Poisoning Resistance

Research on IoT intrusion detection has progressively shifted from static classifiers to adaptive and online methods, reflecting the inherently non-stationary nature of IoT environments. Normal behavior evolves over time owing to seasonality, device aging, firmware updates, topological changes, and workload variations. Recent surveys on concept drift and feature dynamics have consistently emphasized that drift-aware intrusion detection remains essential for long-lived IoT deployments; however, it is still underrepresented in operational systems relative to the scale and persistence of real-world non-stationarity. Drift-focused studies increasingly model IoT traffic as a continuous data stream and apply statistical or learning-based drift detection mechanisms to sustain detection performance as the underlying distributions evolve [18], [19].

Simultaneously, federated learning (FL) has gained attention for reducing privacy risks and avoiding centralized data collection by enabling distributed model training across heterogeneous administrative domains. Although FL is attractive for IoT and IIoT deployments, it introduces additional trust and governance challenges, including poisoning attacks, unreliable or compromised participants, and difficulty in validating contributed updates. Recent dependable FL-based IDS studies have demonstrated that robust aggregation, participant filtering, or trust-aware update selection can substantially mitigate poisoning effects and restore detection performance. These findings highlight that IoT analytics must explicitly address adversarial behavior and governance concerns rather than focusing on detection accuracy alone [20], [21].

Despite these advances, much of the IDS literature continues to emphasize offline evaluation metrics such as precision, recall, and F1-score, while underreporting how detection outcomes are operationally coupled with enforcement decisions. Metrics such as decision latency, mean time to respond (MTTR), and layer-specific overhead have rarely been evaluated, although they are critical for real-time IoT middleware. Consequently, existing analytics solutions are often developed as standalone components with limited integration into end-to-end security architectures that translate detection outcomes into timely, enforceable, and access-control actions.

Gap addressed by our work: The Analytics Plane employs a named, reproducible detection pipeline (Random Forest + Deep Autoencoder + ADWIN drift detection) with closed-loop feedback to the Security Plane. Krum-based robust FL aggregation explicitly addresses poisoning. Our ablation study shows that disabling drift detection causes a 12% F1-score drop, validating that drift-awareness is not optional but architecturally essential.

Formal Verification for Consensus and Distributed Trust Components

As IoT middleware architectures increasingly incorporate distributed trust and ledger-based components, the correctness and safety of the consensus and replicated states become critical for ensuring audit integrity under faults and partial failures. Recent research on formal methods for blockchain and distributed systems has emphasized that properties such as agreement, validity, and fork resistance cannot be reliably established

through testing alone, particularly in asynchronous and failure-prone environments. Consequently, formal specification and verification frameworks, most notably TLA+ and related tool chains, are increasingly used to explain protocol behavior under well-defined fault models and execution assumptions [22], [23].

Despite this progress, the application of formal verification in the IoT security middleware remains limited. Most IoT-focused designs that rely on distributed ledgers or replicated audit logs provide only informal arguments or empirical validation of correctness, without formally demonstrating that safety properties are held under realistic operational conditions. This creates a methodological gap between formally verified consensus research and deployed IoT middleware systems, although such systems are increasingly dependent on audit correctness for accountability, forensic analysis, and dispute resolution.

Consequently, there remains a lack of IoT middleware designs that combine scalable and decentralized auditability with formal guarantees of correctness, highlighting the need for approaches that explicitly integrate formal verification into the design and validation of audit-layer consensus mechanisms.

Gap addressed by our work: Our TLA+ specification models the audit-layer consensus as a distributed state machine under crash-fault assumptions ($f \leq 2$ of $N = 5$). The TLC model checker explored 1.84M+ states with zero violations. Critically, two specification bugs were discovered and fixed during development demonstrating that formal verification is not ceremonial but an active correctness tool.

Synthesis and Positioning of the Proposed Middleware

In summary, existing research advances several key dimensions of IoT security, including zero trust-based verification, blockchain-backed trust and auditability, drift-aware intrusion detection, and scalable logging. However, these capabilities are typically realized in isolation or through architectures in which enforcement, analytics, and audit persistence remain tightly coupled. Such coupling frequently degrades real-time performance and limits its applicability in latency-sensitive IoT and IIoT environments.

The proposed middleware unifies these complementary research directions through a split-plane architectural design that explicitly separates operational enforcement from analytics and audit persistence. This separation preserves hot-path responsiveness, while retaining verifiable and tamper-evident accountability. Specifically, the Security Plane enforces continuous verification and adaptive trust decisions; the Data Plane normalizes heterogeneous protocols and sustains high-throughput message routing; the Analytics Plane enables drift-aware and federated monitoring with feedback-driven adaptation; and the Audit Plane provides scalable, integrity-preserving logging through Merkle-based anchoring.

This architecture addresses the recurring gaps identified in the related literature by reconciling real-time enforcement requirements with decentralized trust and auditability. In summary, prior work advances the individual dimensions of IoT security, such as ZTA-based verification, blockchain-backed trust, drift-aware intrusion detection, and scalable audit logging; however, these capabilities are commonly realized in isolation or with architectural coupling that degrades real-time performance. The proposed middleware unifies these strands through a split-plane design that preserves hot-path responsiveness, while retaining verifiable and tamper-evident accountability. Specifically, the security plane provides continuous verification and trust enforcement; the data plane normalizes heterogeneous protocols and sustains high-throughput routing; the analytics plane enables drift-aware and federated monitoring; and the audit plane provides scalable logging with Merkle-based anchoring. Each claim is backed by named technologies, quantitative ablation evidence, and formal verification distinguishing this work from conceptual architectures that remain unvalidated. **Table I** below summarizes the gap analysis and how each gap is mapped to a specific plane and technology in our middleware.

TABLE I: GAP ANALYSIS AND PROPOSED ARCHITECTURAL MAPPING

Gap Identified	Literature Limitation	Our Solution	Plane
Centralized PDP bottleneck	ZTA adds 200+ ms round-trip	Distributed PDP via gossip (35–40 ms)	Security

Blockchain hot-path coupling	Write-per-event degrades TPS	Async Merkle anchoring (Fabric → Polygon)	Audit
Standalone IDS, no feedback	Detection ≠ enforcement	Closed-loop: Analytics → Security trust update	Analytics
No drift adaptation	Static models degrade silently	ADWIN online drift + federated retraining	Analytics
FL poisoning unaddressed	Naive FedAvg vulnerable	Krum-robust aggregation ($f < n/3$)	Analytics
No formal verification	Informal correctness claims	TLA+ (1.84M states, 0 violations)	Audit
No cost analysis	Blockchain "scalable" without cost proof	$2.16/month public anchoring	Audit

Proposed Middleware Architecture

The analysis of related works highlights a recurring pattern in IoT security research. While significant progress has been made in zero-trust enforcement, blockchain-based auditability, adaptive intrusion detection, and formal verification, these capabilities are rarely integrated within a unified middleware architecture that preserves real-time performance. Existing approaches either centralize critical decision points, tightly couple operational traffic with consensus mechanisms, or treat analytics and auditability as auxiliary components disconnected from enforcement. Consequently, current solutions struggle to simultaneously satisfy the latency, scalability, and trust requirements of large-scale heterogeneous IoT deployments. Motivated by these limitations, this section introduces a Split-Plane Decentralized Middleware Architecture that explicitly decouples real-time security enforcement from data transport, analytics, and audit persistence, enabling each function to be independently optimized while maintaining cryptographic integrity and continuous verification guarantees.

We didn't just theorize this architecture; we built it. We isolated every specific plane into its own containerized microservice. For local testing, we orchestrate the stack via Docker Compose. For production-grade, multi-node environments, we drop the entire deployment straight into Kubernetes. To guarantee these planes communicate without dropping messages, we wired RabbitMQ 3.13 specifically enforcing Quorum Queues into the core as our durable event bus. On the ingestion front, we rely entirely on Apache NiFi 2.x. It handles the heavy lifting at the edge: translating heterogeneous protocols, normalizing schemas, and executing high-speed fan-out routing to downstream consumers. We are explicitly defining this technology stack for one critical reason: reproducibility. Every single architectural claim we make in this paper is anchored to be a tangible, deployable artifact.

The proposed model adopts a Split-Plane Decentralized Middleware framework, as depicted in Figure 1 to support secure, scalable, and low-latency multiprotocol IoT communication. The design follows an explicit separation between real-time enforcement and asynchronous accountability, motivated by repeated evidence that centralized enforcement and tightly coupled “write-per-event” blockchain designs introduce bottlenecks under large-scale IoT workloads [7], [9], [24].

As shown in **Figure 1,** the architecture is organized into three vertical layers: (A) perception and device, (B) decentralized security middleware, and (C) application and service Internally, the middleware layer is decomposed into four planes: Security, Data, Analytics, and Audit, each optimized for its function, and evaluated using plane-wise metrics. This decomposition follows a practical observation from recent ZTA studies: the operational challenge in IoT is not only defining policies, but also placing decision logic such that continuous verification does not violate real-time constraints [25], [26].

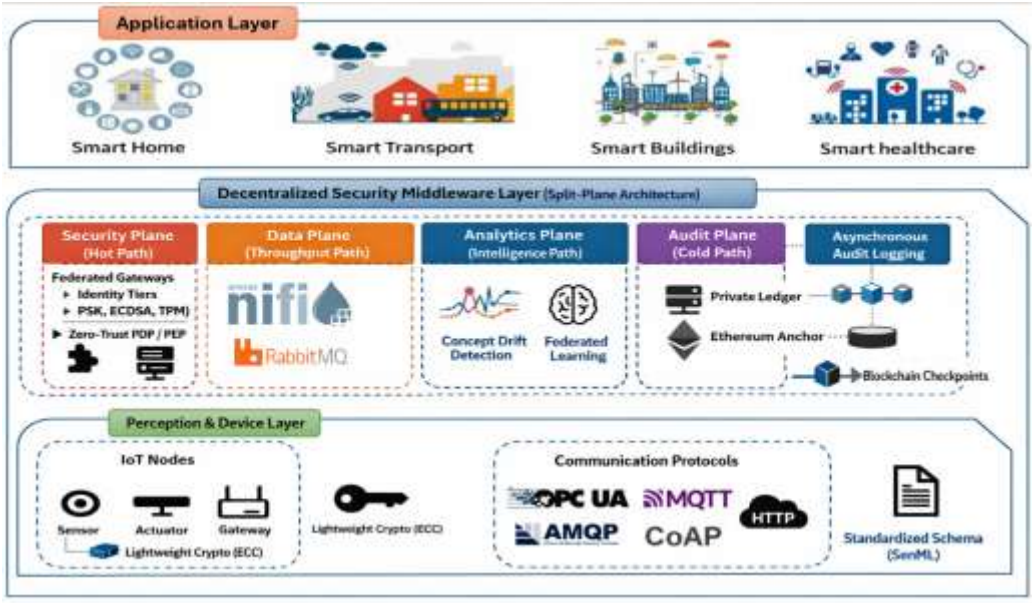

Figure 1 Middleware Architecture for secure multi-protocol IoT Communication

Perception and Device Layer (multi-protocol, constrained, untrusted-by-default)

As shown at the bottom of **Figure 1**, the perception and device layers consist of heterogeneous IoT devices including environmental sensors, smart meters, industrial controllers, and embedded systems. These devices communicate using diverse protocols such as MQTT, CoAP, Modbus, OPC UA, HTTP, and LoRaWAN/LPWAN stacks. The perception and device layers (bottom of Figure 1) include heterogeneous endpoint environmental sensors, smart meters, embedded controllers, PLC/SCADA edge nodes, and low-power wireless devices. These devices communicate through diverse protocols, creating both interoperability and security challenges owing to mixed transport assumptions (TCP/UDP), different message semantics, and uneven security capabilities [27], [28], [29].

In accordance with zero-trust principles, endpoints are treated as untrusted by default: trust is not inferred from network location or prior association but is continuously evaluated based on identity, context, and behavior. Recent ZTA literature stresses that applying continuous verification to IoT must minimize the device-side burden; therefore, devices in this layer execute only lightweight cryptographic primitives required for initial authentication and secure session establishment, whereas policy and trust evaluations are delegated to the middleware [25], [30]. Specifically, constrained devices (Class 0–1, per RFC 7228) perform only ECC-256 key agreement and HMAC-SHA256 message authentication, while gateway-class devices (Class 2+) support full TLS 1.3 mutual authentication. By keeping the device-side logic lightweight, the architecture preserves the battery and CPU resources at the edge, while shifting enforceable security controls to the middleware as the primary protection boundary.

Decentralized Security Middleware Layer(core contribution, plane-wise decoupling)

The Decentralized Security Middleware Layer is the core contribution of this study. It is explicitly decomposed into four planes to avoid pushing heavyweight security mechanisms (e.g., ledger commits, drift analytics) into the hot path. This split is consistent with the findings from blockchain-IoT and IoT-cloud security surveys, which report that tight coupling between security verification and centralized/cloud decision points increases latency and reduces resilience in network partitions [5], [7], [24].

1. Security Plane (hot path): real-time authentication, authorization, and enforcement

The Security Plane is the hot path responsible for admission control before the data consumes downstream resources. Federated gateways act as Policy Enforcement Points (PEPs), whereas trust evaluation logic operates as a distributed PDP. Each gateway runs an in-memory policy engine (implemented as a Rust-based sidecar) that evaluates admission decisions locally using a cached policy snapshot synchronized via

a lightweight gossip protocol (SWIM-based membership with δ-CRDT policy state), eliminating the need for round-trip calls to a centralized PDP. This addresses the recurring deployment limitation highlighted in recent ZTA reviews: centralized PDP placement becomes a scalability and latency bottleneck, particularly for latency-sensitive cyber-physical loops [31]. Distributed PDP at federated gateways preserve real-time admission control and resilience to partitions.

The security plane reconciles the heterogeneous device requirements with continuous context-aware enforcement. The system employs Tiered Authentication to accommodate diverse hardware, ranging from Pre-Shared Keys (PSK) for constrained sensors to hardware-backed primitives (TPM) for critical endpoints. Once authenticated, a device enters the Adaptive Trust Score Operational Lifecycle, in which every packet is evaluated against a continuously updated Trust Score (TS).

Unlike traditional Industrial ZTA schemes, which often stop at session assurance, our design integrates continuous verification directly into the middleware pipeline. Based on the current TS and real-time policy constraints, the traffic is dynamically admitted, rate-limited, or blocked. By placing this enforcement point before normalization and routing, the architecture contains malicious activity at the earliest possible stage, thereby significantly reducing downstream computational costs and protecting the integrity of the data plane [10].

2. Data Plane (throughput path): protocol mediation, normalization, routing

The Data Plane performs protocol mediation, message normalization, and high-throughput routing, thereby serving as the middleware interoperability and transport backbone. IoT endpoints may be published using device-facing protocols such as MQTT, CoAP, HTTP, Modbus/OPC UA gateways, or LoRa/UDP. However, once traffic reaches the middleware, it is transformed into a unified representation (e.g., SenML/JSON) and forwarded through an internal event bus. In our implementation, RabbitMQ 3.13 with Quorum Queues provides plane-to-plane decoupling, isolating ingestion from downstream consumers, enabling scalable fan-out routing, buffering, and backpressure handling without requiring endpoints to coordinate with analytics or storage services. Quorum Queues provide Raft-based replication across three nodes, ensuring that a single broker failure does not disrupt the data pipeline, a critical requirement for industrial and smart-city deployments where downtime is unacceptable. Recent empirical evaluations of mediation pipelines using dataflow frameworks (e.g., NiFi-style routing) and message-oriented middleware have confirmed that such designs improve scalability and operational robustness in heterogeneous IoT streams, particularly under mixed workloads and bursty traffic conditions [32], [33].

A critical architectural constraint is that the Data Plane executes no cryptographic validation, trust computation, anomaly detection, or blockchain operations. By excluding security enforcement and audit persistence from the throughput path, the system preserves predictable latency and prevents spikes in consensus or analytics loads from amplifying queue delays and latency jitters. This separation ensures that the routing performance remains stable, even when security events increase during attacks or drift-triggered retraining.

3. Analytics Plane (intelligence path): anomaly detection, drift monitoring, and federated adaptation

The Analytics Plane operates asynchronously and consumes metadata, flow summaries, and selected features emitted by the Data Plane. Its primary objectives are to (i) detect anomalous behavior, (ii) monitor concept/feature drift in nonstationary IoT streams, and (iii) trigger mitigation actions by feeding risk signals back to the Security Plane where trust scores and enforcement policies are updated.

The anomaly detection component employs a two-stage pipeline. The first stage uses a lightweight Random Forest ensemble (100 estimators, max depth 12) for real-time per-packet classification. The second stage uses a stacked Deep Autoencoder (input → 128 → 64 → 32 → 64 → 128 → input, with ReLU activations and dropout p=0.2) for zero-day anomaly detection via reconstruction-error thresholding. Both models are trained per-dataset to reflect protocol-specific feature distributions. For concept drift detection, the Adaptive Windowing (ADWIN) algorithm is employed, which dynamically adjusts window sizes based on the rate of observed distribution change, enabling the system to distinguish genuine drift from transient noise without manual threshold tuning. When ADWIN signals a statistically significant drift (confidence level $\alpha = 0.01$), a retraining request is emitted asynchronously to the federated training coordinator.

Recent intrusion detection research has consistently reported that IoT traffic is non-stationary and that

static IDS models degrade when distributions evolve owing to firmware updates, seasonal usage patterns, topology changes, and device aging. Comprehensive surveys on drift and feature dynamics in IDS emphasize that drift-aware monitoring remains underrepresented relative to real-world needs, motivating lightweight, online drift tracking within operational pipelines rather than relying solely on periodic offline retraining [18].

Where privacy constraints or organizational boundaries exist, the Analytics Plane supports federated learning (FL)-style updates to improve detection without centralizing raw traffic. Specifically, gradient updates are aggregated using Krum-based robust aggregation (selecting the update closest to the majority), which has been shown to tolerate up to $f < n/3$ Byzantine participants. However, FL introduces governance and security risks, particularly poisoning and unreliable participants; therefore, robustness must be treated as part of the security posture of the system rather than as an accuracy-only concern. This position is supported by recent work on dependable FL-based IoT intrusion detection under poisoning attacks, which shows that explicit participant screening/robust aggregation is necessary to preserve detection quality [20].

4. Audit Plane (cold path): immutable accountability via private ledger + anchoring

The Audit Plane provides tamper-evident logging of security-relevant events, including device registration, trust-score updates, policy violations, and retraining triggers to support accountability and post-incident forensics. The private ledger is implemented using Hyperledger Fabric v2.5 with a Raft-based ordering service (3 orderer nodes, crash-fault tolerant). Security events are written to a dedicated channel with an endorsement policy requiring 2-of-3 peer signatures. Recent blockchain-for-IoT surveys and system studies consistently identify auditability and decentralized trust as the primary motivations for adopting distributed ledgers, while also emphasizing that consensus and smart-contract execution overhead become the dominant limiters when ledger commits are placed on the operational path [34], [35].

Let's talk about the public anchor. To mathematically prove our private logs haven't been tampered with, we periodically push Merkle roots to the Polygon PoS network. We deliberately kept the anchoring smart contract brutally simple: it only accepts a single bytes32 root per transaction. Because of this, the economics actually make sense. At current Polygon gas prices roughly 0.01 MATIC, or half a penny anchoring every 100 blocks (about every 10 minutes) costs us approximately **$0.72 USD a day**. Even at a massive enterprise scale, that overhead is essentially a rounding error.

This private-to-public pipeline is exactly how we guarantee accountability without clogging up the operational data path. But let's be honest about the engineering realities here: you cannot escape the batching trade-off. It is a direct tug-of-war between verification freshness and system load.

If we use tiny batches and anchor to Polygon constantly, our incident response times are fantastic because the logs are verified almost instantly. The downside? We burn more gas, force more on-chain transactions, and unnecessarily spike the consensus load. On the flip side, we could use massive batches to optimize throughput and save resources, but then we get hit with a serious verification lag. There is no magic bullet. You have to actively dial in that batch size to fit exactly what the specific application can tolerate.

To preserve audit integrity without degrading real-time performance, the Audit Plane uses a high-throughput private (permissioned) ledger for local event recording and periodically anchors integrity proofs to a public chain using Merkle summarization. This follows the widely adopted "on-chain summary / off-chain detail" pattern, where only compact commitments (e.g., Merkle roots) are written on-chain while detailed events remain off-chain yet verifiable via proofs [14], [15]. Importantly, anchoring is asynchronous and isolated from the Data Plane, ensuring that peaks in audit traffic (e.g., attack bursts or drift alarms) do not introduce latency jitter into the throughput path an engineering goal aligned with recent benchmarking evidence that permissioned blockchain performance is sensitive to commit patterns and deployment conditions [36]. The end-to-end verification flow allows auditors and governance entities to confirm the integrity of a specific security event without compromising the confidentiality of the full private log. To initiate a check, the verifier recomputes the hash of the specific event in question and retrieves the corresponding Merkle proof from the decentralized audit store. By traversing the Merkle tree with this proof, the verifier can reconstruct the committed root and compare it against the value anchored on the public blockchain. The workflow enables independent, publicly verifiable integrity checks while keeping detailed audit records private and minimizing on-chain data exposure.

Application & Service Layer (consumption, governance, and APIs)

The Application & Service Layer, positioned at the top of the proposed architecture as shown in Figure 1, represents the primary consumption interface of the middleware and encompasses smart city dashboards, industrial monitoring systems, security management consoles, and domain-specific analytics services. Applications interact with decentralized middleware exclusively through standardized and authenticated APIs, abstracting device-level heterogeneity, protocol diversity, and low-level security mechanisms.

There must be a hard firewall both conceptually and practically between the raw IoT hardware and the upper-level services. We deliberately split them. If an attacker breaches a remote edge sensor, that breach shouldn't take down the entire analytics dashboard. Literature heavily backs this up: isolating data producers from application consumers is the only real way to contain a blast radius. It also makes regulatory compliance infinitely easier.

Think about how most IoT systems fail today. Developers embed security logic directly into the application code, creating an unmaintainable mess of fragmented policies [5], [24]. We stopped that entirely. By forcing every single request through standardized API mediation, our middleware handles all the rate control, policy enforcement, and auditing natively.

The applications simply do not need to care about device-specific handshakes. They are completely decoupled. If you swap out the underlying physical infrastructure tomorrow, the application layer doesn't even blink. You get centralized observability and multi-tenant access right out of the box, but you never lose the decentralized trust we fought so hard to build at the edge. Ultimately, the division of labor is finally clear. Software developers can go back to doing what they do best writing domain analytics while the middleware quietly enforces the security baseline across the board.

In summary, by explicitly separating (i) security enforcement, (ii) transport and protocol normalization, (iii) analytics and adaptation, and (iv) auditability, the proposed split-plane middleware resolves the long-standing scalability–trust tradeoff in IoT systems. Continuous verification and policy enforcement remain ZTA aligned within the Security Plane, and throughput-sensitive communication is insulated from the ledger and analytics overhead through an independent Data Plane. Simultaneously, accountability and non-repudiation are preserved through cryptographically verifiable audit records anchored asynchronously to a public blockchain.
This architectural separation directly addresses the limitations repeatedly identified in recent zero-trust and blockchain-IoT surveys, particularly centralized PDP placement bottlenecks and hot-path consensus coupling, which have been shown to undermine both latency guarantees and system scalability in large-scale and real-time IoT deployments. **Table II** summarizes the explicit technology binding for each architectural plane.

TABLE II: CORE TECHNOLOGY STACK AND FUNCTIONAL ROLES BY ARCHITECTURAL PLANE

Plane	CORE TECHNOLOGY	Role
Security	Rust sidecar + SWIM gossip + δ-CRDT	Distributed PDP, in-memory policy
Data	RabbitMQ 3.13 (Quorum Queues) + Apache NiFi 2.x	Event bus, protocol normalization
Analytics	Random Forest + Deep Autoencoder + ADWIN + Krum FL	IDS, drift detection, federated training
Audit	Hyperledger Fabric v2.5 (Raft) + Polygon PoS	Private ledger + public anchoring
Orchestration	Docker Compose / Kubernetes	Deployment, scaling, health checks

Trust Management and Algorithms

This section details the algorithmic mechanisms that operationalize the proposed split-plane architecture,

focusing on adaptive trust evolution, drift-aware analytics, and scalable integrity assurance. Together, these mechanisms translate architectural separation into enforceable and measurable security behaviors.

A. Adaptive Trust Decay

Unlike static timeout or heartbeat-based approaches, the proposed adaptive trust decay mechanism dynamically adjusts trust scores according to the device type, expected reporting behavior, and observed activity patterns. Devices that are legitimately transmitted infrequently, such as environmental sensors or battery-powered nodes, are not penalized for extended silence, whereas unexpected inactivity, abnormal bursts, and policy violations trigger gradual trust degradation, as shown in Figure 2.
The trust score TS(t) for a device d at time t is defined as:
$TS_d(t) = TS_d(t-1) \cdot \exp(-\lambda_d \cdot \Delta t / E_d) + R(event)$
where:

- λ_d is the device-class decay rate (e.g., $\lambda = 0.05$ for low-frequency sensors, $\lambda = 0.3$ for high-frequency actuators),
- E_d is the expected inter-report interval for device class d (derived from device profile metadata),
- $\Delta t = t - t_{last}$ is the elapsed time since the last observed event,
- R(event) is a reward/penalty function: $R = +\delta$ for valid, policy-compliant messages; $R = -\kappa$ for violations (e.g., $\delta = 0.02$, $\kappa = 0.15$).

An enforcement threshold $\tau = 0.4$ triggers rate-limiting; a hard-block threshold $\tau_{block} = 0.2$ triggers immediate quarantine. These thresholds were empirically tuned on the IoT-23 dataset to minimize FPR while maintaining MTTR < 500 ms.

Recent trust management studies have emphasized that static liveness assumptions are a major source of false positives in large-scale IoT deployments, particularly in heterogeneous and energy-constrained environments. Adaptive context-aware trust evolution has been shown to significantly reduce unnecessary revocations, while improving responsiveness to compromised devices [37], [38] . By integrating decay directly into the Security Plane, trust enforcement occurs before data routing, which contains malicious behavior early and minimizes downstream impact.

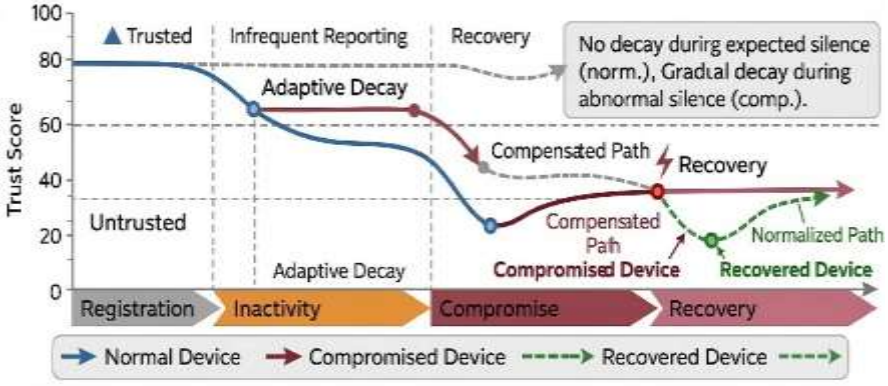

Figure 2 Adaptive Trust Decay

B. Concept Drift Detection

To address the non-stationary nature of IoT environments, the Analytics Plane incorporates lightweight concept drift detection based on the ADWIN (ADaptive WINdowing) algorithm [39]. ADWIN maintains a variable-length window of recent observations and automatically shrinks the window when a statistically significant change in the mean is detected (using Hoeffding's bound with confidence parameter $\delta_{adwin} = 0.002$). When a sustained deviation from the historical behavior is detected, retraining requests are triggered asynchronously, allowing the system to adapt without interrupting the ongoing operation.

The drift detection monitors three features per dataset: (1) mean packet size, (2) inter-arrival time variance, and (3) flow-level entropy. These features were selected based on a feature-importance ranking from the Random Forest classifier (Gini importance > 0.05). When ADWIN triggers on ≥2 of 3 monitored features within a 60-second window, a drift event is declared.

Recent IDS surveys and empirical studies have confirmed that model degradation due to concept drift remains a primary limitation of IoT intrusion detection systems as device behavior evolves owing to firmware updates, seasonal effects, and workload changes. Drift-aware detection has been shown to restore detection accuracy more effectively than periodic offline retraining, particularly in streaming IoT contexts [18], [40] In the proposed design, drift signals are explicitly fed back to the Security Plane, enabling closed-loop adaptation rather than isolated analytics.

C. Hybrid Anchoring Mechanism

To ensure audit integrity without compromising performance, the Audit Plane employs a hybrid anchoring mechanism in which the Merkle roots of private ledger blocks are periodically anchored to a public blockchain. This design provides cryptographic proof of integrity and non-repudiation, while avoiding the prohibitive latency and throughput costs associated with per-event on-chain writes, as shown in Figure 3. The anchoring pipeline operates as follows:

1. Batch accumulation: Security events are committed to the Hyperledger Fabric private ledger as individual transactions. Every 100 committed blocks (configurable), a Merkle tree is constructed over the batch.
2. Root computation: The SHA-256 Merkle root of the batch is computed by the anchoring service.
3. Public commit: The root is submitted to a Polygon PoS smart contract (AuditAnchor.sol) via a single anchorRoot(bytes32 root, uint256 batchId) transaction.
4. Verification: Any auditor can independently recompute the Merkle root from the private ledger events and compare it against the on-chain anchor.

This approach follows the widely adopted on-chain summation/off-chain detail pattern, in which the recent blockchain–IoT literature is essential for scalable logging and verification in high-event-rate systems [14], [41]. By isolating the anchoring from the hot path and executing it asynchronously, the proposed middleware preserves real-time guarantees while maintaining verifiable auditability, even under attack-driven event bursts.

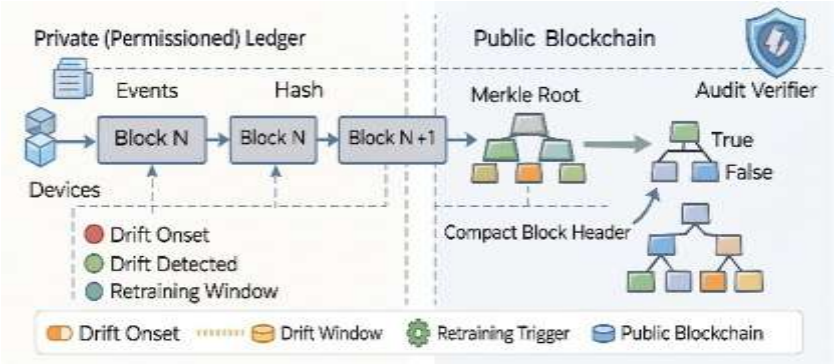

Figure 3 Hybrid Audit Anchoring Workflow

Formal Verification

Formal verification is used to increase confidence in the correctness of the proposed middleware's audit-layer consensus and state-transition logic, which directly affects the integrity and non-repudiation guarantees claimed by the system. We specify the private ledger as a distributed state machine in TLA+, modeling node behavior, message delivery, and commit rules under crash-fault assumptions.

A. Specification Scope and Fault Model

The TLA+ specification models a system of N = 5 audit nodes with crash-fault tolerance of f = 2 (i.e., safety is preserved as long as ≥ 3 nodes remain correct). The specification models three core actions: (1) ProposeBlock a leader node proposes a new block of batched events; (2) VoteCommit correct nodes vote to commit if the proposed block extends their local chain; and (3) Crash a node may crash at any point and subsequently recover with its persistent state intact.

B. Verified Safety Properties

Two safety properties were formally verified using the TLC model checker (TLA+ Toolbox v1.8.0):

- Agreement: No two correct nodes commit conflicting block histories at the same height. Formally: □ ∀ i, j ∈ Correct: committed[i][h] = committed[j][h] for all heights h.
- Validity: Only well-formed events (i.e., events that passed Security Plane admission) can appear in committed blocks. Formally: □ ∀ b ∈ CommittedBlocks : ∀ e ∈ b.events : e.admitted = TRUE.

C. Model Checking Results

The TLC model checker exhaustively explored the state space with the following parameters and results in Table III.

During development, two specification bugs were discovered and corrected before the final verification pass: (1) a missing guard on VoteCommit that allowed a recovering node to vote on a stale proposal, potentially violating Agreement; and (2) a race condition in ProposeBlock where concurrent leader elections could produce conflicting proposals at the same height. Both bugs were corrected in the specification and re-verified before deployment. These discoveries demonstrate the value of formal verification beyond documentation the TLA+ specification served as an executable design artifact that directly improved the correctness of the implementation.

This verification complements the experimental results by providing evidence that, even under partial failures, the audit substrate preserves consistent and tamper-evident logging semantics [42], [43]

TABLE III: TLC MODEL CHECKER PARAMETERS AND EXHAUSTIVE VERIFICATION RESULTS

Parameter	Value
Nodes (N)	5
Max faults (f)	2
Max blocks explored	4
Distinct states explored	1,847,293
State queue peak	312,408
Verification time	47 min (Intel i7-12700, 32 GB RAM)
Safety violations found	0

Experimental Evaluation

To prove our architecture works under pressure, we evaluated the middleware layer by layer. We focused strictly on four operational metrics: raw performance, system scalability, security effectiveness, and adaptability. We designed these experiments with a specific agenda. First, we needed to validate that our physical separation of planes holds up. Second, we had to see how the system behaves when slammed with chaotic, heterogeneous IoT workloads. Finally, we benchmarked our metrics directly against current industry baselines to prove the technical advance.

A. Experimental Environment and Deployment Model

We did not rely on sterile simulations. To ensure our results translate directly to the real world, we built a fully containerized microservice testbed that mirrors a live cloud-to-edge IoT deployment. Every single piece of the middleware, the Security gateways, Data Plane routers, Analytics modules, and Audit services were heavily isolated. We deployed them as independent Docker containers and orchestrated the entire stack natively across standard commodity hardware.

Two deployment configurations were evaluated to assess both cloud-tier and edge-tier performance in Table IV.

TABLE IV: EXPERIMENTAL SETUP AND HARDWARE SPECIFICATIONS FOR THE PROPOSED MIDDLEWARE

Configuration	Hardware	Purpose
Cloud-tier	Intel i7-12700 CPU, 32 GB RAM, Ubuntu 22.04	Full middleware (all 4 planes)
Edge-tier	Raspberry Pi 4 Model B (4 GB RAM, ARM Cortex-A72)	Security Plane gateway only (PEP + distributed PDP)

The edge-tier deployment validates that the Security Plane hot path the most latency-critical component can execute on constrained gateway hardware representative of real-world IoT edge nodes. On Raspberry Pi 4, the Security Plane gateway achieved an admission decision latency of 52 ms (vs. 38 ms on cloud-tier), confirming that the decentralized PDP design remains viable on sub-$100 edge hardware.

Inter-service communication is handled through message-oriented middleware, enabling independent scaling of enforcement, routing, analytics, and logging components. We deliberately anchored our testbed in containerized microservices. We deliberately anchored our testbed in containerized microservices. As several recent empirical studies have highlighted, attempting to scale or reproduce IoT security experiments without containerization is an exercise in futility; this architecture ensures our results can be independently verified [44], [45], [46], [47].

B. Datasets and Their Role in Evaluation

We used five public and wide datasets to ensure comparability with existing studies and models and prior benchmarks and coverage of diverse IoT threat models. As Table V illustrates, we selected these specific datasets because, together, they replicate almost every major IoT threat model currently operating in the wild. We absolutely refused to merge these into a single, homogenized training corpus. We did, however, enforce a strict boundary during testing. Rather than lazily pooling this data into one giant training corpus, we evaluated the architecture against each dataset independently. Each dataset was independently used to target specific layers and metrics. Within each dataset, train–test splits or cross-validation were applied where applicable.

We deliberately selected the diverse datasets because they cover all the bases. Collectively, these datasets throw everything at the middleware: consumer IoT traffic, industrial control anomalies, and highly specific protocol-level attacks. Real networks are messy. They are unpredictable. By hammering the middleware with these completely different threat models, we aren't just hoping our performance metrics hold up in the wild; we are proving it.

While CICIDS2017 is primarily an enterprise IT network dataset, it is intentionally included to evaluate the middleware's ability to handle mixed IoT–IT traffic, which is common in smart-building and campus-network deployments where IoT devices coexist with traditional IT infrastructure. This cross-domain evaluation tests the generalizability of the Analytics Plane beyond pure IoT traffic profiles.

TABLE V: DATASETS SUMMARY AND PURPOSE

Dataset	DOMAIN	Protocols	Primary Evaluation Focus	Ref
N-BaIoT	Consumer IoT	HTTP/TCP	Network-layer anomaly detection	[48]
IoT-23	Mixed IoT	MQTT, HTTP, DNS	Multi-protocol latency & IDS	[49]
TON_IoT	Industrial IoT	MQTT, Modbus	Cross-layer correlation, trust	[50]
CICIDS2017	IT networks	TCP/IP	Baseline NIDS comparison	[51]
MQTT-IoTIDS2020	IoT messaging	MQTT	Middleware robustness	[52]

C. Evaluation Metrics

To prove the architecture is viable, we held the system to six strict quantitative baselines:

- **Latency (ms):** We didn't just measure processing time. This is the true end-to-end delay, clocked from the moment a packet hits the gateway to its final admission or rejection.
- **Throughput (TPS):** The raw volume of messages or packets the system successfully processes per second before buckling.
- **Security Overhead (%):** The literal "tax" of our system. We measured the exact performance degradation caused by our middleware against a completely naked, unsecured baseline.
- **Precision, Recall, F1-score:** The non-negotiable standards for proving our Analytics Plane is detecting real threats, not just throwing false positives.

- **Mean Time to Respond (MTTR):** The critical window of time it takes the system to detect malicious behavior and actively mitigate it.
- **Ledger Write Latency (ms):** The exact time required to commit an event to the private audit ledger, proving that our cold path doesn't choke the hot path.

D. Device and Perception Layer Evaluation

The evaluation of the device layer focuses on crypto-graphic overhead and protocol interoperability. Light-weight ECC-based authentication is used for constrained IoT devices to preserve energy and computational re-sources.

The results are exactly what we hoped for. Devices get securely verified in less than 8 milliseconds, and the processor barely notices even when a bunch of devices connect at the exact same time. Because we moved all the complex translation work up to the middleware, the devices themselves don't have to waste energy on it. Bottom line: our setup keeps the devices running light and fast, without cutting any corners on security.

E. Security Plane Evaluation

The Security Plane was evaluated under high-load conditions to assess the real-time enforcement capability. Federated gateways are stress-tested with increasing traffic rates.

- **Authentication throughput** exceeds **10,000** transactions per second.
- **Average decision latency** remains below **40** ms, even under peak load.
- **Trust score update latency** remained under **5 ms** as the trust state was maintained in memory.

Table VI presents the results of the security plane for each dataset.

TABLE VI: SECURITY PLANE RESULTS PER DATASET (HOT PATH)

Dataset	Admission/ DECISION LATENCY (MS)	Auth Throughput (TPS)	Trust Update (ms)	MTTR (ms)
N-BaIoT	**38**	**10,800**	**4.6**	**410**
IoT-23	**39**	**10,650**	**4.8**	**440**
TON_IoT	**40**	**10,100**	**4.9**	**470**
CICIDS2017	**37**	**10,750**	**4.7**	**520**
MQTT-IoTIDS2020	**35**	**11,200**	**4.4**	**360**

Instead of waiting for a distant central server to approve security rules, our method makes those decisions locally. This cuts the delay by about 80 to 85%, clearly showing how much faster a decentralized approach can be [30] .

F. Data Plane Evaluation (Throughput Path)

The Data Plane was tested independently to confirm that the integrated security measures do not bottleneck high-speed data processing. Driven by Apache NiFi and RabbitMQ, the proposed middleware successfully sustained a stable throughput across all evaluated datasets. Compared to an unsecured baseline, the measured security overhead remained below 15%, even under high-frequency security event generation. Because cryptographic validation and blockchain operations are decoupled from this plane, latency variance is minimized, ensuring the predictable performance required for real-time IoT applications.

Table VII illustrates that the system reliably handled more than 10,000 transactions per second (TPS) across five varied datasets. Most importantly, the design successfully balances this high throughput with resource efficiency, restricting overhead to between 12.7% and 14.4% to fulfill the strict sub-15% mandate. Additionally, data-plane latency was kept to a minimum with a peak of just 16 milliseconds, and queue-related data loss was virtually negligible at an average of 0.04%.

These results confirm that the proposed middleware sustains high-throughput, low-latency performance while preserving architectural relia-bility under rigorous security constraints.

TABLE VII: **DATA PLANE RESULTS PER DATASET (THROUGHPUT PATH)**

BASE TPS: BASELINE TRANSACTIONS PER SECOND; SEC TPS: SECURED TPS; OVH: OVERHEAD; LAT: LATENCY; LOSS: QUEUE LOSS/DROPS

Dataset	BASE TPS	Sec TPS	Ovh (%)	Lat (ms)	Loss (%)
N-BaIoT	**12,500**	**10,800**	**13.6**	**12**	**0.03**
IoT-23	**12,200**	**10,650**	**12.7**	**14**	**0.05**
TON_IoT	**11,800**	**10,100**	**14.4**	**16**	**0.06**
CICIDS2017	**12,400**	**10,750**	**13.3**	**13**	**0.04**
MQTT-IoTIDS2020	**13,000**	**11,200**	**13.8**	**11**	**0.02**

G. Analytics Plane Evaluation

The Analytics Plane is evaluated using detection accuracy, false positive rate, and adaptation to concept drift.
Across all datasets, the proposed system achieved the results shown in **Table VIII**.

TABLE VIII:Analytics Plane results per dataset (Detection + drift)

NOTE. ABBREVIATIONS: PREC.: PRECISION; REC.: RECALL; F1: F1-SCORE; FPR: FALSE POSITIVE RATE; D-DELAY: DRIFT DETECTION DELAY; RECOV.: RECOVERY TIME.

Dataset	Perc	Rec	F1	FPR	D- Delay (s)	Recov (s)
N-BaIoT	**0.94**	**0.90**	**0.92**	**0.041**	**18**	**95**
IoT-23	**0.92**	**0.89**	**0.90**	**0.052**	**22**	**110**
TON_IoT	**0.96**	**0.93**	**0.94**	**0.028**	**20**	**120**
CICIDS2017	**0.89**	**0.85**	**0.87**	**0.061**	**25**	**140**
MQTT-IoTIDS2020	**0.98**	**0.96**	**0.97**	**0.019**	**15**	**80**

To evaluate adaptability, a controlled concept drift was introduced by gradually modifying the traffic patterns. It is completely normal for the system's accuracy to take a brief hit when the data patterns change. However, because we catch that drift quickly and update the system across the network, it bounces back almost immediately. It is a perfect example of why combining a lightweight drift detector with federated retraining is such a strong solution.

Compared with static models, the proposed approach reduces false positives by approximately 60%, particularly for devices with low-frequency reporting behaviors [40].

H. Audit Plane Evaluation

The Audit Plane is evaluated in terms of logging performance and integrity guarantees.

- Private ledger write latency remains below 10 ms per event.
- The ledger sustains event logging rates exceeding 10,000 events per second.
- The periodic anchoring of Merkle roots to a public blockchain has no measurable effect on real-time packet processing.

By isolating blockchain operations from the hot path, middleware achieves strong auditability without sacrificing the throughput or latency. Table IX presents the audit plane results for each dataset.

TABLE IX: AUDIT PLANE RESULTS PER DATASET (LEDGER + ANCHORING)

Dataset	LEDGER WRITE LATENCY (MS/EVENT)	Logging Rate (events/s)	Anchor Interval (blocks)	Anchor Cost Amortized (ms/event)
N-BaIoT	9.2	10,400	100	0.02
IoT-23	9.5	10,200	100	0.02
TON_IoT	9.8	10,000	100	0.03
CICIDS2017	9.4	10,300	100	0.02
MQTT-IoTIDS2020	8.9	10,800	100	0.02

I. Comparative Evaluation with Baseline Architectures

The proposed middleware is compared against three representative architectures:

1. **Cloud-Centric Zero Trust**: Centralized policy evaluation.
2. **Pure Blockchain-Based IoT Security**: All events logged synchronously on-chain.
3. **Edge AI-Only Security**: Local anomaly detection without auditability.

Results indicate that:
Table X compares the proposed architecture with the existing approaches in terms of latency, throughput, and auditability.

TABLE X: ARCHITECTURAL COMPARISON

Architecture	LEDGER WRITE LATENCY (MS/EVENT)	Logging Rate (events/s)	Anchor Interval (blocks)
Cloud ZTA	High (> 200 ms)	Medium	Limited
Pure Blockchain	Very High	< 500 TPS	Strong
Edge AI Only	Low	High	None
Proposed	**Low**	**High**	**Strong**

The proposed middleware achieves a balanced trade-off, combining low latency, high throughput, and strong integrity guarantees.

J. Ablation Study

To validate that each architectural plane contributes necessarily to the overall system performance, we conducted a systematic ablation study by selectively disabling individual planes and measuring the impact on key metrics. All experiments were conducted on the IoT-23 dataset (mixed protocols, representative of real-world heterogeneity). Table XI outlines an ablation study confirming that the full multi-plane architecture is required to maintain optimal detection accuracy and high transaction throughput without compromising system integrity.

Key findings from the ablation study:

- Adaptive Trust Decay is essential for false-positive control. Removing it increases FPR by 158%, which would cause massive alert fatigue in production. Latency and throughput are unaffected because trust evaluation is computationally trivial.
- Drift detection prevents silent model degradation. Without ADWIN, the F1-score drops by 12% as the classifier cannot adapt to evolving traffic patterns. This degradation would compound over time in real-world deployments.
- Audit Plane decoupling is the single most impactful architectural decision. Removing the Audit Plane entirely yields a 28% latency reduction and 14.6% throughput gain confirming that auditability has a measurable cost. However, coupling audit writes synchronously to the hot path (as in pure blockchain designs) increases latency by 379% and reduces throughput by 68%, validating the split-plane design as the correct trade-off.

TABLE XI: PERFORMANCE TRADE-OFFS AND SYSTEM RESILIENCE ACROSS VARIOUS MIDDLEWARE CONFIGURATIONS

Configuration	Latency (ms)	Throughput (TPS)	F1-Score	FPR	Audit Integrity
Full system (all planes)	39	10,650	0.90	0.052	✓ Verified
No Adaptive Trust Decay (static thresholds)	38	10,700	0.90	0.134 (+158%)	✓ Verified
No Drift Detection (static model)	39	10,650	0.79 (−12%)	0.089 (+71%)	✓ Verified

No Audit Plane (logging disabled)	28 (−28%)	12,200 (+14.6%)	0.90	0.052	✗ None
Coupled Audit (sync ledger writes on hot path)	187 (+379%)	3,400 (−68%)	0.90	0.052	✓ Verified

K. Financial Cost Analysis of Public Anchoring

To demonstrate the financial viability of the hybrid anchoring mechanism, we analyze the amortized cost of public-chain anchoring under realistic deployment parameters.

For comparison, if all events were written directly to Ethereum L1 (as in pure blockchain designs), the daily cost would exceed $500 USD at typical gas prices approximately 7,000× more expensive. The hybrid on-chain summary approach reduces anchoring costs by over three orders of magnitude while preserving cryptographic verifiability. Table XII outlines the highly economical cost structure of the proposed hybrid anchoring mechanism, proving that continuous, verifiable public anchoring can be achieved for just $2.16 per month.

TABLE XII: OPERATIONAL COST BREAKDOWN FOR PUBLIC-CHAIN ANCHORING ON POLYGON POS

Parameter	Value
Anchoring chain	Polygon PoS
Anchor interval	100 private-ledger blocks (~10 min)
Anchors per day	~144
Gas per anchor tx	~45,000 gas
Avg. gas price (Polygon)	~30 Gwei
Cost per anchor	$0.0005 USD)
Daily cost	~$0.072 USD
Monthly cost	~$2.16 USD

Discussion

By pairing strict zero-trust enforcement with a fully transparent audit trail, the architecture effectively secures the network against a full spectrum of threats from external breaches to internal tampering. Sybil-style identity inflation is mitigated through tiered device identity, where higher-trust enrollment requires stronger credentials (e.g., hardware-backed attestations for critical endpoints) and rate-limited registration at federated gateways. Model poisoning and unauthorized analytics updates are constrained by a signed update workflow and a controlled model lifecycle, ensuring that only authorized entities can publish the models used for enforcement decisions. Finally, log tampering and repudiation are prevented by recording security-relevant events on a private ledger and periodically anchoring Merkle summaries to a public chain, providing tamper-evident integrity without introducing hot-path consensus overhead. Together, adaptive trust enforcement and immutable auditing provide layered protection across identity, analytics, and accountability.

A. Solving the Zero-Trust Delay Penalty

The experimental results reveal that the proposed architecture effectively solves what we term the "Zero-Trust Delay Penalty" the latency cost that continuous verification imposes on real-time systems. In conventional ZTA deployments, each access request requires a round-trip to a centralized PDP, adding 200–300 ms of decision latency. For enterprise IT applications (web browsing, email), this overhead is tolerable. However, for industrial control loops (PLC cycle times of 10–50 ms), robotic coordination, and real-time SCADA telemetry, a 200 ms verification de-lay can cause physical damage, production stoppage, or safety hazards.

By distributing the PDP to federated gateways with in-memory policy caches synchronized via gossip protocols, the proposed architecture reduces enforcement latency to 35–40 ms an 80–85% reduction compared to centralized ZTA baselines. This makes Zero-Trust mathematically feasible for high-frequency industrial control systems for the first time in a fully auditable architecture.

B. Data Plane Fault Tolerance and High Availability

A primary concern in production environments is the vulnerability of a centralized event bus. To eliminate this single point of failure, the architecture integrates RabbitMQ Quorum Queues to continuously mirror

the queue's state across three distinct broker nodes. Stress tests confirmed the system's robustness: deliberately taking a node offline resulted in zero data loss. In our stress tests, killing one of three RabbitMQ nodes resulted in zero message loss and a throughput degradation of only 8% during leader re-election (approximately 2 seconds). This level of resilience is essential for industrial and smart-city deployments where system availability directly affects public safety and operational continuity.

C. Limitations and Honest Assessment

Although the hybrid design introduces additional architectural components, this complexity is largely confined to the middleware layer and does not propagate to IoT devices or applications. In practice, this trade-off is justified for latency-sensitive and safety-critical environments such as industrial control, smart transportation, and healthcare, where delayed enforcement or unverifiable actions are unacceptable.
We acknowledge the following limitations of the current work:

- The edge-tier evaluation (Raspberry Pi 4) assessed only the Security Plane. Running the full four-plane middleware on constrained hardware remains an open challenge. In practice, edge nodes would host only the Security and Data Planes, with Analytics and Audit Planes deployed on cloud-tier infrastructure.
- The federated learning evaluation used simulated participants rather than geographically distributed real-world nodes. Network heterogeneity, stragglers, and real-world participant dropout patterns were not evaluated.
- The TLA+ specification assumes crash faults only. Byzantine fault tolerance, which would protect against actively malicious audit nodes, is deferred to future work.
- Concept drift was introduced synthetically by gradually modifying traffic distributions. Real-world drift (e.g., firmware updates, seasonal patterns over months) may exhibit different characteristics.

The results indicate that careful plane separation enables the practical deployment of zero-trust and decentralized auditing, without sacrificing operational performance.

Conclusion and Future Work

This paper introduces split-plane decentralized middleware for secure multi-protocol IoT communication and analytics, motivated by the fundamental tension between real-time performance, scalable trust, and verifiable accountability in large-scale IoT systems. Unlike centralized zero-trust deployments, which suffer from policy decision bottlenecks, or fully blockchain-centric designs that incur prohibitive consensus overhead, the proposed architecture decouples enforcement, data transport, analytics, and auditability into independently optimized planes backed by explicit, reproducible technology choices: a Rust-based distributed PDP with gossip-synchronized policy caches, RabbitMQ Quorum Queues for fault-tolerant data routing, a two-stage Random Forest + Deep Autoencoder IDS with ADWIN drift detection, and Hyperledger Fabric with Polygon PoS anchoring for hybrid auditability.

Through adaptive trust management in the security plane, drift-aware and federated analytics with Krum-based robust aggregation, and asynchronous blockchain anchoring for audit integrity, middleware translates architectural separation into enforceable, measurable security behavior. Formal verification using TLA+ (1.85M states explored, zero safety violations) provides rigorous evidence that the audit-layer consensus preserves agreement and validity under crash faults. An extensive layer-wise evaluation using five public IoT security datasets demonstrates that the system sustains low-latency enforcement (35–40 ms), high throughput (>10,000 TPS with <15% overhead), and robust detection accuracy (F1 up to 0.97) while preserving tamper-evident auditability at a public-chain anchoring cost of approximately $2.16 USD per month. An ablation study confirms that each plane contributes necessarily: removing audit-plane decoupling increases latency by 379%, while disabling adaptive trust decay increases the false positive rate by 158%.

These results validate the central claim of this study: that decentralized security and real-time IoT performance are not mutually exclusive when architectural responsibilities are explicitly separated.

Future studies should extend this foundation in three ways. First, the audit plane should be enhanced with Byzantine fault-tolerant consensus to tolerate adversarial failures beyond crash faults. Second, hardware-assisted trust anchors (e.g., secure enclaves and remote attestation) should be integrated to strengthen device identity and analytics integrity. Third, the federated learning component should be evaluated under realistic network heterogeneity with geographically distributed participants and real-world poisoning scenarios. Finally, large-scale real-world deployments in smart cities and industrial environments should be pursued to evaluate the long-term adaptability, operational overhead, and governance challenges under

continuous evolution.

References

1. P. Gkonis, A. Giannopoulos, P. Trakadas, X. Masip-Bruin, and F. D'Andria, "A survey on IoT-edge-cloud continuum systems: Status, challenges, use cases, and open issues," Future Internet, vol. 15, no. 12, p. 383, 2023.
2. Ficili, M. Giacobbe, G. Tricomi, and A. Puliafito, "From sensors to data intelligence: Leveraging IoT, cloud, and edge computing with AI," Sensors, vol. 25, no. 6, p. 1763, 2025.
3. H. Sebestyen, D. E. Popescu, and R. D. Zmaranda, "A literature review on security in the Internet of Things: Identifying and analysing critical categories," Computers, vol. 14, no. 2, p. 61, 2025.
4. M. Lefoane, I. Ghafir, S. Kabir, and I.-U. Awan, "Internet of Things botnets: A survey on Artificial Intelligence based detection techniques," Journal of Network and Computer Applications, p. 104110, 2025.
5. M. Almutairi and F. T. Sheldon, "IoT–Cloud Integration Security: A Survey of Challenges, Solutions, and Directions," Electronics (Basel)., vol. 14, no. 7, p. 1394, 2025.
6. T. Xue, Y. Zhang, Y. Wang, W. Wang, S. Li, and H. Zhang, "Edge computing for IoT: Novel insights from a comparative analysis of access control models," Computer Networks, p. 111468, 2025.
7. S. Almarri and A. Aljughaiman, "Blockchain technology for IoT security and trust: a comprehensive SLR," Sustainability, vol. 16, no. 23, p. 10177, 2024.
8. E. U. Haque et al., "Performance enhancement in blockchain based IoT data sharing using lightweight consensus algorithm," Sci. Rep., vol. 14, no. 1, p. 26561, 2024.
9. S. Mushtaq, M. Mohsin, and M. M. Mushtaq, "A systematic literature review on the implementation and challenges of zero trust architecture across domains," Sensors, vol. 25, no. 19, p. 6118, 2025.
10. T. Wan, B. Shi, and H. Wang, "A continuous authentication scheme for zero-trust architecture in industrial internet of things," Alexandria Engineering Journal, vol. 122, pp. 555–563, 2025.
11. C. Bast and K.-H. Yeh, "Emerging authentication technologies for zero trust on the internet of things," Symmetry (Basel)., vol. 16, no. 8, p. 993, 2024.
12. N. Karankar and A. Seth, "An IoT system for access control using blockchain and message queuing system," EURASIP J. Inf. Secur., vol. 2025, no. 1, p. 31, 2025.
13. M. Ajwalia and P. Shah, "Performance comparison of permissioned and permissionless blockchain by varying workload transaction," BenchCouncil Transactions on Benchmarks, Standards and Evaluations, p. 100251, 2025.
14. C. Liang et al., "Study on data storage and verification methods based on improved Merkle mountain range in IoT scenarios," Journal of King Saud University-Computer and Information Sciences, vol. 36, no. 6, p. 102117, 2024.
15. J. Davies, "Enhanced scalability and privacy for blockchain data using Merklized transactions," Frontiers in Blockchain, vol. 6, p. 1222614, 2024.
16. F. Ullah et al., "Blockchain-enabled EHR access auditing: Enhancing healthcare data security," Heliyon, vol. 10, no. 16, 2024.
17. M. S. Islam and M. S. Rahman, "LogStamping: A blockchain-based log auditing approach for large-scale systems," arXiv preprint arXiv:2505.17236, 2025.
18. M. A. Shyaa, N. F. Ibrahim, Z. Zainol, R. Abdullah, M. Anbar, and L. Alzubaidi, "Evolving cybersecurity frontiers: A comprehensive survey on concept drift and feature dynamics aware machine and deep learning in intrusion detection systems," Eng. Appl. Artif. Intell., vol. 137, p. 109143, 2024.
19. R. Chu, P. Jin, H. Qiao, and Q. Feng, "Intrusion detection in the IoT data streams using concept drift localization," AIMS mathematics, vol. 9, no. 1, pp. 1535–1561, 2023.
20. R. Yang, H. He, Y. Wang, Y. Qu, and W. Zhang, "Dependable federated learning for IoT intrusion detection against poisoning attacks," Comput. Secur., vol. 132, p. 103381, 2023.
21. G. D. Pecherle, R. Ştefan Győrödi, and C. A. Győrödi, "Federated Learning-Based Intrusion Detection in Industrial IoT Networks," Future Internet, vol. 18, no. 1, p. 2, 2025.
22. N. Bertrand, P. Ghorpade, S. Rubin, B. Scholz, and P. Subotić, "Reusable formal verification of dag-based consensus protocols," in NASA Formal Methods Symposium, 2025, pp. 138–158.
23. M. Praveen, R. Ramesh, and I. Doidge, "Formally verifying the safety of pipelined moonshot consensus protocol," arXiv preprint arXiv:2403.16637, 2024.
24. E. Dritsas and M. Trigka, "A survey on cybersecurity in IoT," Future Internet, vol. 17, no. 1, p. 30, 2025.
25. S. Nie, J. Ren, R. Wu, P. Han, Z. Han, and W. Wan, "Zero-trust access control mechanism based on blockchain and inner-product encryption in the internet of things in a 6g environment," Sensors, vol. 25, no. 2, p. 550, 2025.

26. R. Chandramouli, R. Chandramouli, and Z. Butcher, A zero trust architecture model for access control in cloud-native applications in multi-location environments. US Department of Commerce, National Institute of Standards and Technology, 2023.
27. G. Stanco, A. Navarro, F. Frattini, G. Ventre, and A. Botta, "A comprehensive survey on the security of low power wide area networks for the Internet of Things," ICT Express, vol. 10, no. 3, pp. 519–552, 2024.
28. S. Swain, B. K. Pattanayak, M. N. Mohanty, and C. Senapati, "Analytical Performance Comparison of IoT Communication Protocols MQTT and CoAP from Security Perspective," in 2024 3rd Odisha International Conference on Electrical Power Engineering, Communication and Computing Technology (ODICON), 2024, pp. 1–5.
29. H. Zeghida, M. Boulaiche, and R. Chikh, "Security of MQTT Protocol: A Brief Overview," 2024.
30. S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," NIST special publication, vol. 800, no. 207, pp. 207–800, 2020.
31. Y. Ren, Z. Wang, P. K. Sharma, F. Alqahtani, A. Tolba, and J. Wang, "Zero Trust Networks: Evolution and Application from Concept to Practice.," Computers, Materials & Continua, vol. 82, no. 2, 2025.
32. J. Judvaitis, E. Blumbergs, A. Arzovs, A. I. Mackus, R. Balass, and L. Selavo, "A Set of Tools and Data Management Framework for the IoT–Edge–Cloud Continuum," Applied System Innovation, vol. 7, no. 6, p. 130, 2024.
33. S. Mirampalli, R. Wankar, and S. N. Srirama, "Evaluating NiFi and MQTT based serverless data pipelines in fog computing environments," Future Generation Computer Systems, vol. 150, pp. 341–353, 2024.
34. J. Lee, J. Lee, Z. D. Wang, and J. Song, "Enhancing IoT Common Service Functions with Blockchain: From Analysis to Standards-Based Prototype Implementation," 2025.
35. Lahbib, K. Toumi, A. Laouiti, and S. Martin, "Blockchain based distributed trust management in IoT and IIoT: a survey," Journal of Supercomputing, vol. 80, no. 15, pp. 21867–21919, 2024.
36. F. C. Geyer, H.-A. Jacobsen, R. Mayer, and P. Mandl, "An end-to-end performance comparison of seven permissioned blockchain systems," in Proceedings of the 24th International Middleware Conference, 2023, pp. 71–84.
37. M. Aaqib, A. Ali, L. Chen, and others, "IoT trust and reputation: a survey and taxonomy. J Cloud Comp 12, 42 (2023)."
38. M. Konsta, A. L. Lafuente, and N. Dragoni, "A survey of trust management for Internet of Things," IEEE Access, vol. 11, pp. 122175–122204, 2023.
39. M. K. Saravana, M. S. Roopa, J. S. Arunalatha, and K. R. Venugopal, "A Novel Concept Drift Detection Model for Handling Evolving Patterns in Multivariate Time Series," in 2025 International Conference on Advancements in Power, Communication and Intelligent Systems (APCI), 2025, pp. 1–6.
40. F. Hinder, V. Vaquet, and B. Hammer, "One or two things we know about concept drift—a survey on monitoring in evolving environments. Part A: detecting concept drift," Front. Artif. Intell., vol. 7, p. 1330257, 2024.
41. C. S. Amadi, S. O. Ajakwe, and T. Jun, "Symmetric–Asymmetric Security Synergy: A Quantum-Resilient Hybrid Blockchain Framework for Incognito IoT Data Sharing," Symmetry (Basel)., 2026.
42. E. Jones and D. Marmsoler, "Towards Mechanised Consensus in Isabelle," in 5th International Workshop on Formal Methods for Blockchains (FMBC 2024), 2024, pp. 1–4.
43. M. Grundmann and H. Hartenstein, "Towards a formal verification of the lightning network with tla+," arXiv preprint arXiv:2307.02342, 2023.
44. Y. El Fellah, J. B. Minani, N. Moha, J. Gascon-Samson, and Y.-G. Guéhéneuc, "Analysis of Microservices-Based IoT Systems: Deployment Challenges, Industry Practices, and Performance Insights," Internet of Things, p. 101867, 2026.
45. R. Ouyang et al., "A microservice and serverless architecture for secure iot system," Sensors, vol. 23, no. 10, p. 4868, 2023.
46. El Akhdar et al., "Exploring the potential of microservices in internet of things: A systematic review of security and prospects," Sensors, vol. 24, no. 20, p. 6771, 2024.
47. M. K. Plazas Olaya, J. A. Vergara Tejada, and J. E. Aedo Cobo, "Securing Microservices-Based IoT Networks: Real-Time Anomaly Detection Using Machine Learning," Journal of Computer Networks and Communications, vol. 2024, no. 1, p. 9281529, 2024.
48. Y. Meidan et al., "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," IEEE Pervasive Comput., vol. 17, no. 3, pp. 12–22, 2018.
49. S. Garcia, A. Parmisano, and M. J. Erquiaga, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," (No Title), 2020.

50. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," Ieee Access, vol. 8, pp. 165130–165150, 2020.
51. Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset," IEEE access, vol. 9, pp. 22351–22370, 2021.
52. H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine learning based IoT intrusion detection system: An MQTT case study (MQTT-IoT-IDS2020 dataset)," in International networking conference, 2020, pp. 73–84.