Inteligencia Artificial, 28(76) (2025), 253-282
doi: 10.4114/intartif.vol28iss76pp253-282

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

NAIDS4IoT: A Novel Artificial Intelligence-Based Intrusion Detection Architecture for the Internet of Things

Haythem Hayouni[1,A], Leila Nasraoui[2,3,B]
[1] Department of Computer Science, Higher Institute of Computer Science of Kef, University of Jendouba, Tunisia
[2] COSIM Research Lab. Higher School of Communications of Tunis (SUP'COM), University of Carthage, Tunisia
[3] National School of Computer Science (ENSI), University of Manouba, La Manouba, Tunisia
[A] haythem.hayouni@isikef.u-jendouba.tn
[B] leila.nasraoui@supcom.tn

Abstract Internet of Things (IoT) has brought unprecedented opportunities across various sectors, including healthcare, transportation, industrial automation, and smart cities. However, this expansion has also introduced significant security vulnerabilities due to the heterogeneous nature, limited computational capabilities, and large-scale deployment of IoT devices. Detecting anomalies, which often signify security breaches or system malfunctions, is crucial to maintaining the integrity and reliability of IoT systems. Traditional anomaly detection methods, typically rule based or signature driven, struggle to adapt to evolving threats and diverse data patterns in IoT networks. This paper proposes a novel architecture named NAIIDS4IoT (Novel Artificial Intelligence-based Intrusion Detection System architecture for IoT), designed to provide efficient, accurate, and scalable anomaly detection using Artificial Intelligence. The core of NAIIDS4IoT lies in the integration of federated learning with deep autoencoders, enabling decentralized model training across edge devices without sharing raw data, thereby preserving user privacy and reducing communication overhead. Each edge node independently learns patterns of normal behavior and identifies anomalies based on reconstruction errors. A global model is continuously refined through collaborative learning across nodes. Furthermore, NAIIDS4IoT incorporates lightweight encryption and blockchain based model integrity verification to enhance security and trust in the detection process. Experimental validation using real-world IoT datasets demonstrates that NAIIDS4IoT achieves high detection accuracy, low false positive rates, and strong adaptability to dynamic environments, significantly outperforming conventional centralized and shallow learning based solutions. This architecture represents a significant step toward intelligent, autonomous, and privacy-preserving anomaly detection in next generation IoT ecosystems.

Keywords: IoT Security; Intrusion Detection System (IDS); Anomaly Detection; Artificial Intelligence (AI); Federated Learning; Deep Autoencoder; Blockchain; Detection Latency; False Positive Rate; Edge Computing.

1 Introduction

The Internet of Things (IoT) has revolutionized the way devices communicate, collect, and exchange data, enabling applications ranging from smart homes and industrial automation to e-health and intelligent transportation systems. The growing reliance on these systems in critical infrastructure and daily life [1]. Despite its advantages, IoT brings a unique set of security challenges [2] due to its inherent characteristics: heterogeneous device capabilities, limited computing resources, decentralized topology, and often weak or absent security measures.

Figure 1 illustrates a typical IoT system architecture composed of four hierarchical layers, each playing a distinct role in the overall communication and coordination. At the bottom, the IoT Devices layer includes various smart objects like cars, surveillance cameras, wearable devices, and home sensors that continuously generate data. These devices connect upward to the Edge Nodes layer, which handles local processing and communication management, often through wireless access points. Above this, another layer of Edge Nodes depicted with server like icons aggregates and processes the data further. At the top sits the Federated Coordinator Blockchain layer, which represents a decentralized control entity that orchestrates the federated learning process while ensuring secure and trustworthy communication through blockchain technology. Blue arrows between layers highlight the bidirectional flow of information and coordination, emphasizing the real-time and multi-tiered nature of data processing in modern IoT ecosystems.

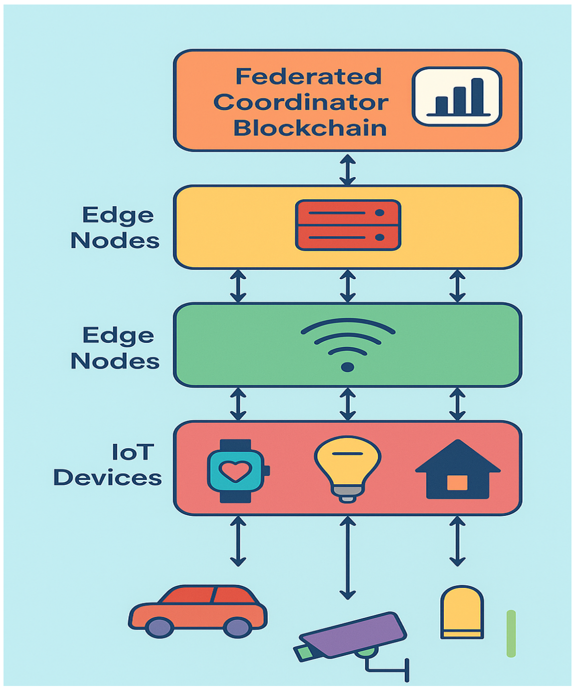

Figure 1: Typical IoT system architecture

Among the most pressing concerns in IoT security is anomaly detection [19], which serves as a first line of defense against cyber threats, malfunctions, or abnormal system behaviors. Anomalies may arise due to network intrusions, malware, misconfigurations, or even hardware faults [2]. Early and accurate detection is crucial to ensure data integrity, availability, and confidentiality within IoT networks. However, conventional anomaly detection systems, typically based on rule based heuristics or static signatures, are inadequate in the face of evolving attack vectors, complex traffic patterns, and zero-day threats [3].

Recent advances in Artificial Intelligence (AI) [24], particularly Machine Learning (ML) and Deep Learning (DL), have opened new avenues for intelligent, adaptive anomaly detection mechanisms. These techniques can model nonlinear patterns, detect unknown threats, and generalize across different IoT environments. Nevertheless, centralized AI models require aggregating large amounts of raw data, raising serious concerns over user privacy, latency, bandwidth consumption, and potential single points of failure [4].

To overcome these limitations, we propose NAIIDS4IoT (Novel Artificial Intelligence-based Intrusion Detection System architecture for IoT), a novel distributed anomaly detection framework tailored for

IoT systems. NAIIDS4IoT leverages Federated Learning (FL) [**2**] to enable collaborative model training across edge devices without sharing sensitive raw data [**3**]. At each node, a Deep Autoencoder (DAE) [**4**] is trained to learn patterns of normal behavior and flag deviations with high precision. The use of FL ensures scalability, privacy preservation, and adaptability to local contexts. Furthermore, to strengthen model trustworthiness and resist model poisoning attacks, we integrate a lightweight blockchain mechanism that verifies the integrity of federated updates before model aggregation. This hybrid approach addresses multiple challenges in IoT security, including decentralization, constrained resources, and trust management.

The main contributions of this work are as follows:

- We propose NAIIDS4IoT, a scalable and privacy-preserving AI-based architecture for anomaly detection in IoT networks.
- We design a hybrid detection engine combining Deep Autoencoders and Federated Learning, tailored for edge-based IoT environments.
- We enhance the trustworthiness of the detection system using blockchain to verify model update integrity.
- We evaluate NAIIDS4IoT using two benchmark datasets (UNSW-NB15 and IoT-23), demonstrating superior accuracy and efficiency compared to related solutions.

The remainder of this paper is structured as follows: Section 2 reviews the most relevant related works in the field of IoT anomaly detection, with a particular focus on federated learning and blockchain based approaches, and identifies key limitations in existing methods. Section 3 introduces and elaborates on the proposed NAIIDS4IoT architecture, detailing its layered design and key functional modules. Section 4 describes the experimental methodology, including the simulation environment, datasets, and evaluation metrics, followed by a comprehensive analysis of the obtained results. Finally, Section 5 summarizes the main findings of the study and proposes potential directions for future research to further enhance IoT intrusion detection systems.

2 Related works

Anomaly detection in IoT has been the subject of extensive research in recent years. Various approaches have been proposed, leveraging machine learning, deep learning, hybrid systems, and distributed intelligence. In this section, we review notable contributions and highlight their advantages and limitations, particularly in relation to scalability, accuracy, and adaptability in resource-constrained IoT environments.

Abbasi et al. [**5**] investigated how anomaly detection can be enhanced in IoT edge computing by combining deep learning with a horizontal data reduction strategy. The paper points out that the massive and diverse data streams produced at the edge of IoT networks often overwhelm resource-constrained devices, making efficient detection of irregular patterns difficult. To tackle this, the authors introduce an approach that filters and compresses redundant instances at the input level before passing them into deep learning models. This method reduces computational burden while preserving the critical characteristics needed for accurate analysis. Experimental evaluations show that the technique not only improves detection accuracy but also lowers processing time and resource usage. Overall, the study demonstrates a promising pathway for building lightweight yet effective anomaly detection systems suitable for real-time edge computing environments.

Olanrewaju-George and Pranggono [**6**] explored the use of federated learning for intrusion detection in Internet of Things (IoT) networks. IoT devices often face constraints in terms of processing power, memory, and data privacy, making traditional centralized intrusion detection systems less effective. To address this, the authors design a system that combines both unsupervised and supervised deep learning models within a federated framework. This approach enables multiple devices to collaboratively train a global detection model without sharing raw data, thereby protecting sensitive information. The experimental results show that the method achieves strong detection performance while reducing privacy risks and resource overhead, demonstrating that federated learning can provide a scalable and secure foundation for intrusion detection across heterogeneous IoT environments.

Singh et al. [**7**] introduced an anomaly based intrusion detection system (IDS) designed for Industrial Internet of Things (IIoT) networks. The authors stress that IIoT infrastructures, with their dense connectivity and critical operations, are highly vulnerable to cyberattacks. Unlike traditional signature-based IDS, which can only detect known threats, their system monitors traffic and device behavior to identify deviations from normal activity, making it more effective against novel or zero-day attacks. The framework was implemented and tested in an industrial environment, where it demonstrated strong detection accuracy and low latency, essential for real-time industrial processes. This makes it a practical approach for safeguarding IIoT systems against evolving threats. However, the study is limited to a specific industrial testbed, raising questions about scalability to larger and more heterogeneous deployments. Additionally, the potential for false positives, a common issue with anomaly-based methods, is not fully addressed. Overall, the work shows the promise of anomaly detection in IIoT security but highlights the need for further validation in broader, real-world applications.

Zhao et al. [**8**] introduced an intrusion detection approach tailored for Internet of Things (IoT) networks, built around a lightweight neural network architecture. The authors note that many existing deep learning methods, while accurate, demand high computational power and memory, which makes them unsuitable for resource-constrained IoT devices. To address this issue, they design a streamlined neural network that reduces complexity without significantly compromising detection accuracy. The paper reports experimental results showing that the proposed model achieves effective performance in identifying network intrusions, while maintaining lower processing and storage requirements compared to conventional approaches. This makes the method well suited for practical deployment in IoT environments where efficiency is just as important as security.

Chen et al. [**9**] introduced an intrusion detection method aimed at consumer electronic devices in IoT networks, where resources such as memory and processing power are often limited. Instead of using heavy deep learning models, the authors adopt the Deep Forest framework, an ensemble of decision trees that is less resource-intensive yet still effective. To further optimize performance, they apply a hybrid feature selection method that filters redundant attributes before detection. Tested on benchmark datasets like NSL-KDD, CIC-IDS2017, UNSW-NB15, and BOT-IoT, their approach shows strong accuracy while consuming fewer resources than traditional machine learning and deep learning baselines. Despite these advantages, the model still adds computational overhead on ultra-constrained devices, relies on benchmark datasets that may not fully reflect real-world IoT traffic, and its adaptability to novel or evolving attacks remains uncertain.

Zhang [**10**] introduced a deep learning based intrusion detection system aimed at strengthening the security of IoT networks. The system is designed to learn traffic patterns and identify abnormal behavior that may indicate cyberattacks. Test results on benchmark datasets show that the model delivers high detection accuracy and demonstrates potential for deployment in IoT environments, where traditional security solutions often fall short. Despite these promising results, the paper acknowledges some limitations. The experiments rely heavily on benchmark datasets, which do not fully capture the complexity of real-world IoT traffic. Deep learning models also tend to demand large amounts of data and computational power, which may limit their use on devices with strict resource constraints. Finally, while the system performs well against known attack types, its ability to adapt to new or evolving threats remains uncertain. These challenges highlight areas for future research, such as improving generalization, reducing resource needs, and testing under more realistic IoT conditions.

Essaid and Ju [**11**] examined how blockchain can be applied to improve security and privacy in Industrial Internet of Things (IIoT) systems. The authors argue that the large scale and distributed nature of industrial IoT networks make them vulnerable to cyberattacks, unauthorized access, and data tampering. To address these risks, the study explores blockchain based mechanisms for secure data exchange, authentication, and access control. The proposed solutions leverage blockchain decentralized structure and immutability to reduce reliance on centralized authorities and to ensure greater transparency and trust across industrial environments. Case studies and evaluations presented in the paper show that blockchain can strengthen resilience against security breaches while also supporting privacy protection in IIoT deployments. Blockchain integration can introduce higher computational and energy demands, which may be difficult for resource limited IIoT devices. The latency of consensus protocols may also reduce system responsiveness, making real-time industrial operations harder to support. Furthermore, scaling blockchain solutions to very large IIoT networks remains a challenge, both in terms of performance and cost. These constraints highlight the need for future research into lightweight consensus mechanisms, hybrid security models, and practical deployment strategies tailored to industrial IoT requirements.

Rheey and Park [**12**] introduced a hierarchical anomaly detection framework for Internet of Things (IoT) networks that incorporates feature impact analysis. The authors point out that existing anomaly detection systems often struggle to balance robustness and interpretability, especially when handling diverse IoT traffic. Their approach applies a hierarchical structure, where anomalies are identified at multiple levels of abstraction, and feature impact is assessed to better understand which attributes influence detection outcomes. This design improves both accuracy and transparency, allowing operators to interpret why specific traffic is flagged as abnormal. Experiments reported in the study show that the method achieves high detection performance and provides stronger robustness compared to conventional flat models. The hierarchical structure introduces additional computational overhead, which may pose challenges in resource-constrained IoT environments. The reliance on predefined feature sets may also limit adaptability when traffic patterns evolve or when new device types are introduced. Furthermore, while the method improves interpretability, applying it to very large-scale IoT networks could lead to complexity in managing and analyzing multiple hierarchical layers. These limitations suggest the need for further optimization, especially to balance scalability and efficiency in practical deployments.

Uganya et al. [**13**] proposed an intrusion detection system for IoT networks that combines deep learning with blockchain technology to enhance both detection accuracy and data security. The deep learning component is used to analyze network traffic and identify abnormal patterns, while blockchain ensures secure, decentralized sharing of detection results, reducing the risk of tampering and single points of failure. The approach demonstrates strong potential for improving IoT resilience, but it also faces challenges such as the high computational demands of deep learning models and the latency introduced by blockchain consensus mechanisms. These limitations highlight the need for further research into lightweight neural architectures and more efficient blockchain protocols to make the system scalable and practical in real-world IoT environments.

Alabbadi and Bajaber [**14**] introduced an intrusion detection system for IoT data streams that leverages explainable artificial intelligence (XAI). The authors emphasize that while many deep learning based IDS solutions achieve high accuracy, they often operate as black boxes, making it difficult for system administrators to understand or trust the detection process. To address this, their system integrates XAI techniques that provide interpretability, helping explain which features and patterns lead to specific detection outcomes. Experiments demonstrate that the approach achieves effective detection rates while also improving transparency and user trust, which are critical for real-time IoT security. However, the added interpretability mechanisms may introduce computational overhead, and the system performance in large-scale, heterogeneous IoT networks requires further study.

Table 1 presents a comparative analysis of various anomaly detection approaches in IoT environments. Methods such as horizontal reduction combined with deep learning [**5**] and lightweight neural networks [**8**] demonstrate strong performance with reduced resource usage, making them suitable for edge devices, though they remain sensitive to dataset diversity and generalization. Federated learning approaches [**6**] address privacy concerns effectively by avoiding raw data sharing and achieve scalability, but they introduce communication overhead. Similarly, anomaly-based IDS in IIoT contexts [**7**] achieves high accuracy and low latency, though its evaluation remains limited to industrial testbeds. More complex models, like deep forest frameworks [**9**] and hierarchical designs with feature impact analysis [**12**], improve robustness and interpretability but at the cost of higher computational requirements. On the other hand, blockchain based solutions [**11**, **13**] strengthen privacy and secure data exchange but suffer from latency, scalability, and energy concerns, which hinder their deployment in real-time IoT scenarios. Finally, the integration of explainable AI [**14**] enhances transparency and trust by clarifying detection decisions, though it adds extra overhead and still requires validation in large-scale heterogeneous networks. Overall, the table illustrates that while no single approach is universally optimal, combining lightweight deep learning with privacy-preserving and interpretable techniques appears to offer the most promising direction for practical IoT security.

Table 1: Comparative Analysis of Anomaly Detection Methods in IoT

Paper	**Approach**	**AI Technique**	**Privacy**	**Edge-Friendly**	**Blockchain**	**Performance**	**Limitations**
[5]	Horizontal reduction + DL	Deep Learning	✗	✓	✗	High accuracy, efficient	Dataset dependent, needs diverse evaluation
[6]	Federated IDS	FL + DL (unsupervised + supervised)	✓	✓	✗	Strong detection, scalable	Requires communication overhead
[7]	Anomaly-based IDS (IIoT)	Machine Learning / Statistical	✗	✓	✗	Accurate, low latency	Limited to industrial testbed
[8]	Lightweight NN	Deep Learning	✗	✓	✗	Effective with low resources	Limited generalization capacity
[9]	Deep Forest IDS	Deep Forest (ML ensemble)	✗	✓	✗	Good accuracy, efficient	Limited adaptability to evolving threats
[10]	Intelligent IDS	Deep Learning	✗	✗	✗	High accuracy	Heavy reliance on benchmarks, resource intensive
[11]	Blockchain Security	Blockchain Mechanisms	✓	✗	✓	Secure and private	Scalability and latency challenges
[12]	Hierarchical Model	Machine Learning + Feature Impact	✗	✓	✗	Robust, interpretable	Extra overhead, limited adaptability
[13]	DL + Blockchain IDS	Deep Learning + Blockchain	✓	✗	✓	Strong detection, secure sharing	Computationally heavy, latency issues
[14]	XAI-based IDS	Deep Learning + XAI	✗	✓	✗	Accurate and interpretable	Added overhead, large-scale validation needed

Despite their contributions, most of these methods suffer from at least one of the following limitations: lack of scalability, privacy issues, high computational cost, or centralized processing. Therefore, there is a clear need for novel framework to addresses these gaps by combining deep autoencoders with federated learning for decentralized, privacy-preserving training, and strengthens trust using blockchain to verify the integrity of model updates.

3 Methodology

With the exponential growth of interconnected devices and the rising sophistication of cyber-attacks, traditional centralized Intrusion Detection Systems (IDS) struggle to satisfy the stringent requirements of modern IoT environments in terms of scalability, latency, resilience, and data confidentiality. These systems typically rely on aggregating all network data at a central point for processing, which not only poses significant privacy risks but also introduces single points of failure and communication bottlenecks. In response to these limitations, we propose a novel, decentralized, and intelligent intrusion detection framework named NAIIDS4IoT (Novel AI-based Intelligent Intrusion Detection System for IoT).

NAIIDS4IoT integrates three cutting edge technologies to achieve an efficient and trustworthy detection process across heterogeneous IoT infrastructures:

- Federated Learning (FL): Enabling distributed, privacy-preserving training of anomaly detection models directly on edge devices, thus eliminating the need to share raw data.

- Deep Autoencoder-based Anomaly Detection: Providing high detection accuracy by learning the underlying patterns of normal behavior and identifying anomalies based on reconstruction errors.

- Blockchain Technology: Ensuring transparency, immutability, and trust in collaborative learning processes through secure logging, reputation scoring, and decentralized validation of model updates.

This section elaborates on the key components and functionalities of each architectural layer, supported by system deployment workflows, interaction diagrams, and detailed algorithmic implementations.

3.1 Motivation and Design Principles

IoT systems exhibit high variability in device capabilities, data volume, and application contexts. These systems are inherently distributed, privacy-sensitive, and often operate under real-time constraints. Traditional anomaly detection frameworks that rely on centralized data collection and processing struggle to meet these requirements.

NAIIDS4IoT is designed to address these challenges by incorporating the following principles:

- Heterogeneity Handling: Devices from various manufacturers with different computing power are supported. The system offloads learning to edge devices to adapt to local constraints.

- Data Privacy Preservation: Through Federated Learning, raw data remains on-device, complying with privacy regulations (e.g., GDPR, HIPAA).

- Low-Latency Detection: Edge processing allows real-time decision-making.

- Robustness and Adaptivity: Continuous local training enables adaptation to changing behavioral patterns and evolving attacks.

- Trust and Auditability: Blockchain ensures secure logging, authentication, and tamper-proof tracking.

These principles drive the modular and layered design of the NAIIDS4IoT architecture.

3.2 Architecture Overview

Figure 2 illustrates a five-layered framework designed for secure and intelligent intrusion detection in IoT environments. At the top, the IoT Devices layer includes various data-generating endpoints such as sensors, cameras, and wearables, which send data to the next layer. The Edge Intelligence layer processes this data locally using deep autoencoders to detect anomalies in real time, reducing latency and preserving privacy. These edge nodes participate in the Federated Learning layer, where they share encrypted model updates instead of raw data, enabling collaborative and decentralized model training. The Blockchain layer ensures the integrity and traceability of model updates through smart contract validation and secure logging. At the base, the Cloud Monitoring layer aggregates anomaly reports, assesses threat levels, and triggers system-wide responses when necessary. Arrows between the layers indicate the flow of data, updates, and alerts, emphasizing the system's distributed, adaptive, and privacy-preserving nature. This interactive diagram enhances the static structure by visualizing how each component contributes to the overall robustness of the NAIIDS4IoT framework.

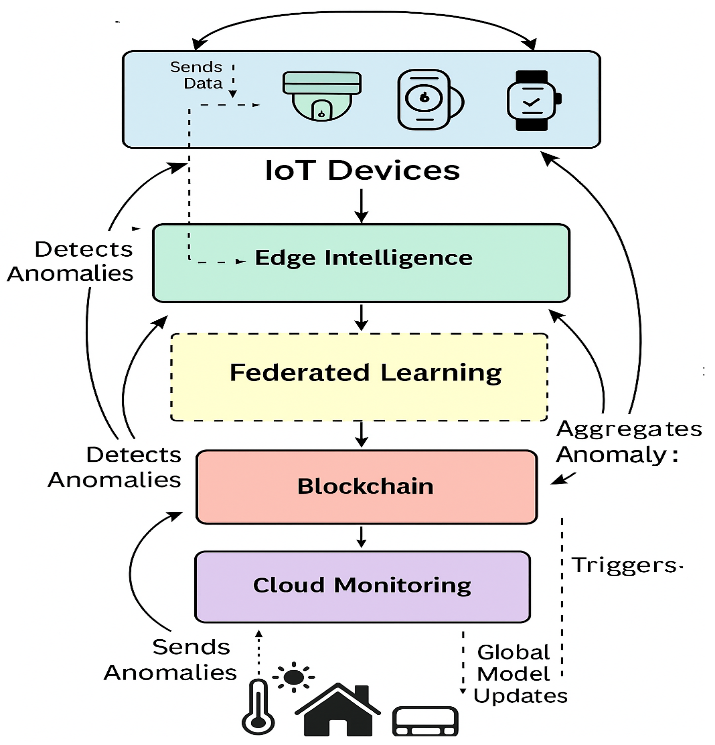

Figure 2: NAIIDS4IoT Architecture

3.3 System Modules and Functional Components of NAIIDS4IoT

This subsection provides a comprehensive overview of the core modules and functional layers, along with key implementation details, that constitute the NAIIDS4IoT architecture. The system is designed as a multi-layered framework to facilitate secure, efficient, and intelligent intrusion detection in IoT environments.

In table 2, we presented the parameters used in NAIID4IoT with context-specific descriptions:

Table 2: Summary of Parameters Used in the Proposed NAIIDS4IoT Framework

Parameter	Description
x_i	Input data sample from an IoT device
$\hat{x}_i$	Reconstructed data from the autoencoder
$E(x_i)$	Reconstruction error for x_i, computed as $\|x_i - \hat{x}_i\|^2$
δ	Anomaly detection threshold
f_θ	Autoencoder model parameterized by weights θ
w_k	Local model weights trained at client k
w n_k	Global aggregated model weights Number of data samples at client k
n	Total number of data samples across all clients
M	Total number of clients in the federated system
T	Total number of federated communication rounds
SHA(x)	Secure hash function used for blockchain logging
TP, FP, TN, FN R	True/False Positives/Negatives for performance evaluation Generator and Discriminator in GAN-based extensions
z	Encoded latent representation in the autoencoder

3.3.1 IoT Device Layer

The IoT Device Layer constitutes the foundation of the NAIIDS4IoT architecture and includes a diverse range of resource-constrained devices such as sensors, smart cameras, actuators, wearable devices, and industrial controllers. These devices typically operate with limited CPU resources, memory, battery life, and network bandwidth, especially in decentralized or embedded environments.

Functional Role: The key responsibilities of devices in this layer include:
- *Data Acquisition:* Continuous monitoring and sensing of physical or digital parameters.
- *Preprocessing:* Lightweight local operations such as aggregation or threshold filtering to reduce redundancy.
- *Secure Transmission:* Encrypted and authenticated communication to edge intelligence nodes.

Let the raw data captured by a device be denoted as:

$$x_i \in \mathbb{R}^n \tag{1}$$

To standardize values and reduce noise, data is normalized using local statistics:

$$\tilde{x}_i = \text{Normalize}(x_i) = \frac{x_i - \mu}{\sigma} \tag{2}$$

where μ and σ represent the local mean and standard deviation of the sensor's readings.

Secure Transmission: Each device encrypts its output using Elliptic Curve Cryptography (ECC), suitable for low-power environments due to its high security-to-computation ratio. The encrypted payload is represented as:

$$C_i = \text{Encrypt}_{\text{ECC}}(\tilde{x}_i, K_{\text{pub}}) \tag{3}$$

where K_{pub} is the public key of the edge node. To ensure authenticity, a digital signature is computed as:

$$\sigma_i = \text{Sign}_{\text{ECC}}(H(\tilde{x}_i), K_{\text{priv}}) \tag{4}$$

where $H(\cdot)$ is a secure hash function such as SHA-256, and K_{priv} is the device's private key.

Communication Protocols: To maintain efficiency and reliability in low-resource scenarios, communication is handled using lightweight protocols:
- *MQTT over TLS:* Publish/subscribe communication with secure transmission.
-*CoAP over DTLS:* REST-like protocol optimized for lossy networks with minimal overhead.

Device Identity and Authentication: Each device is assigned a unique identity ID_i and cryptographic credentials. Upon deployment, the public key and identity are registered in the blockchain layer for trust anchoring and auditability.

This layer ensures that raw data is securely and efficiently captured, normalized, and transmitted for edge-based anomaly detection, while maintaining cryptographic guarantees for data confidentiality and integrity.

3.3.2 Edge Intelligence Layer

The Edge Intelligence Layer plays a pivotal role in NAIIDS4IoT by bridging the gap between resource-constrained IoT devices and cloud-based centralized processing. It is composed of edge nodes typically IoT gateways, fog devices, or local edge servers that possess greater computational and storage capabilities than the underlying devices. This layer is responsible for localized anomaly detection using deep learning models, particularly *deep autoencoders*, and for participating in the *federated learning* process to continuously enhance model performance without exposing raw data.

Local Anomaly Detection: Each edge node receives encrypted and preprocessed data $\tilde{x}_i$ from multiple IoT devices. To identify anomalous behavior, a deep autoencoder (DAE) is employed. The autoencoder comprises an encoder function f_θ and a decoder function g_ϕ that map input data to a lower-dimensional latent representation and reconstruct it, respectively:

$$z_i = f_\theta(\tilde{x}_i) \tag{5}$$

$$\hat{x}_i = g_\phi(z_i) = g_\phi(f_\theta(\tilde{x}_i)) \tag{6}$$

The reconstruction error is computed using the mean squared error (MSE):

$$\mathcal{L}_{\text{recon}} = \|\tilde{x}_i - \hat{x}_i\|_2^2 \tag{7}$$

A data point is flagged as anomalous if its reconstruction error exceeds an adaptive threshold τ, which is dynamically updated based on recent observations:

$$\text{Anomaly} = \begin{cases} 1, & \text{if } \mathcal{L}_{\text{recon}} > \tau \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

The threshold τ is determined via moving average or statistical bounds (e.g., $\mu_\mathcal{L} + k\sigma_\mathcal{L}$), where $\mu_\mathcal{L}$ and $\sigma_\mathcal{L}$ are the mean and standard deviation of recent reconstruction errors.

Model Training and Local Updates: The autoencoder model at each edge node is initially trained on local benign data, with periodic updates based on new observations. Let $\mathcal{D}_e$ be the local dataset at edge node e, the training objective becomes:

$$\min_{\theta,\phi} \frac{1}{|\mathcal{D}_e|} \sum_{\tilde{x}_i \in \mathcal{D}_e} \|\tilde{x}_i - g_\phi(f_\theta(\tilde{x}_i))\|_2^2 \tag{9}$$

To maintain energy efficiency and reduce overfitting, edge nodes may apply early stopping, dropout regularization, and quantized gradients during local updates.

Privacy and Secure Aggregation: Edge nodes do not transmit raw or reconstructed data to the cloud. Instead, they securely share only encrypted model parameters or gradient updates. These updates are further protected using differential privacy or secure aggregation techniques to prevent inference attacks:

$$\Delta\theta_e^{\text{priv}} = \Delta\theta_e + \mathcal{N}(0, \sigma^2 I) \tag{10}$$

where $\mathcal{N}(0, \sigma^2 I)$ is Gaussian noise added for differential privacy before model aggregation.

Trust and Local Reputation: Each edge node is associated with a trust score T_e computed based on the validity and consistency of its model updates, anomaly detection performance, and historical behavior. This score is later utilized in the federated averaging process to weigh contributions from edge participants.

This layer serves as the core intelligence engine of the NAIIDS4IoT framework, ensuring early threat detection, secure local training, and efficient collaboration with the federated learning coordinator for global model refinement.

3.3.3 Federated Learning Coordination Layer

The Federated Learning Coordination Layer facilitates decentralized and privacy-preserving training across edge nodes. In the NAIIDS4IoT framework, this layer orchestrates collaborative model refinement by coordinating edge devices without requiring them to transmit raw data. The process ensures both security and efficiency through mechanisms such as encryption, differential privacy, trust-weighted updates, and blockchain-backed auditability.

Algorithm 1 presents the federated learning round with secure aggregation and trust weighting.

Algorithm 1 Federated Learning Round with Secure Aggregation and Trust Weighting

```
Require: Global model weights $w_0$, total samples $n = \sum_k n_k$, reputation scores $\{R_k\}$, encryption function Enc()
Ensure: Updated global model $w$
 1: for each client $k$ in parallel do
 2:     Download global model $w_0$
 3:     Train local model: $w^k \leftarrow \text{Train}(w_0, \mathcal{D}_k)$
 4:     Compute local delta: $\Delta w^k = w^k - w_0$
 5:     Apply differential privacy: $\Delta \tilde{w}^k \leftarrow \Delta w^k + \mathcal{N}(0, \sigma^2 I)$
 6:     Encrypt update: $E^k \leftarrow \text{Enc}(\Delta \tilde{w}^k)$
 7:     Send $E^k$ and reputation $R_k$ to the aggregator
 8: end for
 9: Decrypt updates: $\Delta \tilde{w}^k \leftarrow \text{Dec}(E^k)$
10: Compute trust-weighted global update:
        $$\Delta w = \frac{1}{\sum_k R_k} \sum_k R_k \cdot \Delta \tilde{w}^k$$
11: Update global model:
        $$w \leftarrow w_0 + \Delta w$$
12: Log update metadata to blockchain ledger
13: Broadcast new model $w$ to all clients
```

In each federated learning round, the global model w_0 is sent to all participating edge nodes. Each client k trains locally on its dataset $\mathcal{D}_k$ and computes the update:

$$\Delta w^k = w^k - w_0 \tag{11}$$

To protect data privacy, each update is perturbed using Gaussian noise, following differential privacy principles:

$$\Delta \tilde{w}^k = \Delta w^k + \mathcal{N}(0, \sigma^2 I) \tag{12}$$

Next, the perturbed update is encrypted using a public-key encryption function:

$$E^k = \text{Enc}(\Delta \tilde{w}^k) \tag{13}$$

The encrypted updates E^k, along with each client blockchain-verified reputation score R_k, are sent to the central aggregator. After decryption:

$$\Delta \tilde{w}^k = \text{Dec}(E^k) \tag{14}$$

A trust-weighted global update is computed as:

$$\Delta w = \frac{\sum_k R_k \cdot \Delta \tilde{w}^k}{\sum_k R_k} \tag{15}$$

This ensures clients with stronger reputations established through consistent, valid contributions have more influence on the final model. The global model is then updated as:

$$w = w_0 + \Delta w \tag{16}$$

Finally, metadata such as update hashes, timestamps, and node identifiers are appended to the blockchain for auditability and tamper resistance. The updated global model is redistributed to all edge clients to begin the next round. This mechanism ensures a scalable, secure, and privacy-aware training process in IoT environments characterized by distributed intelligence and adversarial threats.

3.3.4 Blockchain Network Layer

Each edge device that participates in federated learning submits a signed update to the blockchain network, which is validated using smart contracts. Only authenticated, consistent, and trustworthy updates are accepted. Rejected or malformed updates are flagged and stored separately for auditing and trust degradation. Trust scores are dynamically updated based on the quality and validity of a device's historical contributions.

Algorithm 2 outlines the process of blockchain-based logging and trust management for securing federated updates from edge devices in NAIIDS4IoT. In Step 1, the device generates a cryptographic hash of its update using a secure function $H(\cdot)$ (e.g., SHA-256). This hash uniquely represents the update without revealing its content. In the newly added Step 2, the system verifies the identity and authenticity of the update using the device's registered signature and public key, ensuring only authorized devices contribute to the model.

If the update passes smart contract validation in Step 3 which checks for consistency, format, and other policy criteria a new block is created containing the hash, timestamp, device ID, digital signature, and reference to the previous block. In Step 4, this block is appended to the distributed ledger, preserving update traceability.

In Step 5, the trust score T_i of the device is updated as a weighted average of its previous contributions. The weights α_k can prioritize recent contributions, penalize invalid ones, or reflect the criticality of data. Devices with consistently valid updates gain higher trust, while repeated failures result in diminished scores or blacklisting. This mechanism deters adversarial behavior and enhances the integrity of the federated learning process through decentralized, auditable oversight.

Algorithm 2 Blockchain Logging and Trust Management in NAIIDS4IoT

```
Require: Device update $D_i^t$, trust score $T_i$, blockchain state $S_t$
Ensure: Updated blockchain $S_{t+1}$, trust score $T_i$
 1: Step 1: Hashing
 2:   Compute hash of the device update:
 3:   $h_i^t \leftarrow H(D_i^t)$
 4: Step 2: Signature and Identity Verification
 5:   Verify device signature and identity from blockchain registry:
 6:   $\text{VerifySig}(\text{device\_id}_i, \text{signature}_i, h_i^t)$
 7: if Smart contract validates update: $SC(D_i^t) = 1$ then
 8:    Step 3: Block Creation
 9:   $B_t \leftarrow \{h_i^t, \text{timestamp}_t, \text{device\_id}_i, \text{signature}_i, \text{previous\_hash}\}$
10:    Step 4: Block Appending
11:   Append block to blockchain: $S_{t+1} \leftarrow S_t \cup B_t$
12:    Step 5: Trust Score Update
13:
        $$T_i \leftarrow \frac{\sum_{k=1}^{t} \alpha_k \cdot SC(D_i^k)}{t}$$
    $S_{t+1}, T_i$
14: else
15:    Log invalid update attempt for audit trail $S_t, T_i$
16: end if
```

3.3.5 Cloud Monitoring and Analytics Layer

The Cloud Monitoring and Analytics Layer is responsible for aggregating intelligence from edge devices and coordinating responses to systemic threats across the IoT network. Its main role is to monitor distributed anomalies reported from edge nodes, compute a comprehensive global threat score, and intelligently decide whether to initiate a global model update or maintain current operational protocols.

To accomplish this, we introduce Algorithm 3, which aggregates inputs from multiple edge nodes, each reporting localized anomalies. The algorithm factors in each node reliability (*reputation score*), the severity of its local threat score, and the frequency of detected anomalies. This layered weighting allows the system to prioritize trusted, high-activity nodes when calculating a global threat estimate, ensuring robustness against false positives or compromised nodes.

Algorithm 3 Adaptive Threat Scoring and Optimization

```
Require: Edge nodes $\{N_1, N_2, ..., N_n\}$, Threshold $\tau$
Ensure: Global Threat Score $S$ and decision for model update
 1: Initialize $S \leftarrow 0$
 2: for each edge node $i = 1$ to $n$ do
 3:     Receive local threat score $T_i$
 4:     Get node reputation $R_i$ from blockchain
 5:     Count anomaly frequency $F_i$
 6:     Compute weighted threat score: $W_i \leftarrow T_i \cdot R_i \cdot \log(1 + F_i)$
 7: end for
 8: Compute global threat score: $S \leftarrow \frac{\sum_{i=1}^{n} W_i}{\sum_{i=1}^{n} R_i}$
 9: if $S \geq \tau$ then
10:     Trigger global model update
11:     Notify security administrator
12: else
13:     Continue monitoring and collecting data
14: end if
```

Let there be n edge nodes, each indexed by i. Each node reports a local threat score $T_i \in [0, 1]$, obtained from its anomaly detection module. The reputation score R_i is extracted from the blockchain layer and represents historical trustworthiness, normalized such that $R_i \in [0, 1]$. The anomaly frequency F_i is the number of significant anomalies reported by node i in the current epoch.

To compute a node influence on the global threat score, we define a weighted threat score W_i as:

$$W_i = T_i \cdot R_i \cdot \log(1 + F_i) \tag{17}$$

This equation ensures that frequent, high-confidence anomaly reports from reputable nodes contribute more to the global score, while reducing the effect of noisy or low-reputation nodes. The logarithmic scaling on F_i prevents the function from growing too rapidly in the presence of high anomaly counts.

Next, the global threat score S is computed as the weighted average:

$$S = \frac{\sum\limits_{i=1}^{n} W_i}{\sum\limits_{i=1}^{n} R_i} \tag{18}$$

This formulation normalizes the weighted impact of each node by the total sum of reputation scores, ensuring that low-trust nodes do not dominate the score even if they report frequent threats.

Finally, a comparison is made with a predefined threshold τ. If $S \geq \tau$, the system triggers a global model update and sends a notification to the security administrator. Otherwise, normal operations continue, and data collection proceeds for the next cycle.

Algorithm 4 represents the data flow in the NAIIDS4IoT framework, which defines the real-time operational logic followed by each IoT edge node. It integrates anomaly detection, secure learning, and blockchain logging into a cohesive pipeline. The algorithm outlines the sequential operations carried out by each edge device in the intrusion detection framework. It begins with real-time acquisition of raw sensor data, which is preprocessed through normalization and noise filtering to ensure data quality. The cleaned data is then passed to a locally trained anomaly detection model that evaluates whether the input indicates a potential threat. If an anomaly is detected, the system triggers an alert and notifies the cloud analytics layer. Simultaneously, the device performs localized training on recent data to keep its model adaptive. To maintain privacy and security, the update is perturbed with differential noise and encrypted before transmission to the federated learning (FL) server. Metadata of the update, such as hash, timestamp, and device ID, is also logged on the blockchain for accountability and tamper-proof auditing. This algorithm ensures a seamless and secure interaction between edge intelligence, collaborative learning, and blockchain trust management, enabling NAIIDS4IoT to operate efficiently in distributed and adversarial IoT environments.

Algorithm 4 NAIIDS4IoT Data Flow with Edge-Blockchain-FL Integration

```
Require: Real-time sensor data $X_t$, pre-trained local model $f_k$, reputation $R_k$
Ensure: Threat detection, local update, and secure aggregation
 1: Step 1: Data Acquisition
 2: Acquire sensor input: $X_t = \{x_1, x_2, ..., x_m\}$
 3: Step 2: Data Preprocessing
 4: Normalize and clean $X_t \to \tilde{X}_t$ using statistical filters or min-max scaling
 5: Step 3: Local Anomaly Detection
 6: Predict output: $y_t = f_k(\tilde{X}_t)$
 7: if $y_t$ exceeds anomaly threshold then
 8:     Trigger alert and log locally
 9:     Forward alert metadata to cloud analytics layer
10: end if
11: Step 4: Local Training
12: Update model $f_k$ using recent windowed data $\{(x, y)\}_{t-\delta}^{t}$
13: Step 5: Privacy-Preserving Update
14: Compute $\Delta f_k$ and perturb with noise: $\tilde{\Delta f_k} = \Delta f_k + \mathcal{N}(0, \sigma^2)$
15: Encrypt update: $E_k = \mathrm{Enc}(\tilde{\Delta f_k})$
16: Step 6: FL Communication and Blockchain Logging
17: Send $E_k$ to federated server and append metadata to blockchain
18: Output: Alert flag, encrypted update, logged transaction
```

3.3.6 Deep Autoencoder-Based Detection

Autoencoders are powerful tools for unsupervised anomaly detection in IoT environments where labeled data is scarce or unavailable. In the NAIIDS4IoT architecture, deep autoencoders are deployed at the edge intelligence layer to detect anomalies based on reconstruction errors. The standard autoencoder-based detection uses a static threshold for anomaly classification, this method is sensitive to changing environmental conditions and device behavior. We propose an Adaptive thresholding with edge aggregation method, that dynamically adjusts the detection threshold using local statistical analysis and optionally aggregates anomaly scores at the edge level before sending to the cloud.

Algorithm 5 details the thresholding autoencoder for anomaly detection.

Algorithm 5 Adaptive thresholding autoencoder for anomaly detection

```
Require: normal training data $d = \{x_1, x_2, ..., x_n\}$, window size $w$
Ensure: anomaly label for new input sample $x$
 1: train deep autoencoder $f_\theta$ on $d$ each training sample $x_i \in d$
 2: compute reconstruction $\hat{x}_i \leftarrow f_\theta(x_i)$
 3: compute reconstruction error $e_i \leftarrow \|x_i - \hat{x}_i\|^2$
 4: compute dynamic threshold $\delta \leftarrow \mu_e + \alpha \cdot \sigma_e$    ▷ $\mu_e$: mean error, $\sigma_e$: std dev., $\alpha$: sensitivity factor
 5: input: new data point $x$
 6: compute $\hat{x} \leftarrow f_\theta(x)$ and $e(x) \leftarrow \|x - \hat{x}\|^2$
 7: return: normal
 8: optionally: store $e(x)$ in edge buffer $b = \{e_t\}_{t=t-w}^{t}$ for cloud aggregation
```

The algorithm is designed to efficiently detect anomalies in IoT environments using an unsupervised learning approach. Autoencoders are neural networks trained to reconstruct input data, which enables them to learn the underlying structure of normal patterns.

During the training phase, only benign (normal) data is used. The autoencoder learns to minimize the reconstruction error:

$$e_i = \|x_i - \hat{x}_i\|^2 \tag{19}$$

where x_i is the input sample and $\hat{x}_i$ is the output reconstructed by the autoencoder f_θ.

After training, the algorithm computes the mean (μ_e) and standard deviation (σ_e) of the reconstruction errors over the training dataset. It then defines a dynamic threshold δ as:

$$\delta = \mu_e + \alpha \cdot \sigma_e \tag{20}$$

where α is a tunable parameter that controls sensitivity to anomalies.

In the detection phase, a new input x is considered anomalous if its reconstruction error exceeds the threshold δ:

$$\text{If } \|x - f_\theta(x)\|^2 > \delta \Rightarrow \text{Anomaly} \tag{21}$$

This adaptive thresholding mechanism ensures that the detection remains effective under varying data distributions and device behaviors, which is especially critical in IoT environments characterized by heterogeneity and dynamism. Beside, the lightweight architecture of the autoencoder makes it suitable for edge deployment, while the statistical thresholding allows for localized, efficient, and accurate anomaly detection.

3.3.7 Blockchain Integration for Trust and Security

To reinforce trust, auditability, and tamper-resistance in the decentralized architecture of NAIIDS4IoT, we integrate a permissioned blockchain network as the underlying security backbone. This layer coordinates and verifies critical operations during federated learning and anomaly reporting without relying on a central authority.

Rationale and Role of Blockchain:

Traditional federated learning (FL) architectures are vulnerable to poisoning attacks, unreliable nodes, and false updates. Moreover, in large-scale IoT networks, it is difficult to guarantee the authenticity of clients or verify the integrity of transmitted model parameters. A blockchain provides an immutable and distributed ledger to overcome these limitations, offering:

- Transparency: Every model update, anomaly report, or participation request is recorded on-chain, ensuring full traceability.
- Integrity: Model updates are hashed (e.g., using SHA-256) and stored in the ledger. Any alteration is immediately detectable.
- Decentralized Trust: Eliminates reliance on a single trusted party, enabling peer-to-peer validation.

Smart Contract-Driven Control:

Blockchain nodes host smart contracts that encode system policies and security logic. Smart contracts enforce:

- Client Registration: Only authenticated IoT devices and edge nodes can participate in training or detection.
- Update Validation: Model updates must be signed with the sender private key and include a hash digest.
- Quorum and Thresholds: Training rounds begin only if a minimum number of participants (quorum) are online.

Model Integrity Verification:

Before aggregation, each edge device computes the hash of its local model update w_t^k and submits it to the blockchain along with its digital signature:

$$h_k = \text{SHA256}(w_t^k) \tag{22}$$

The aggregator node retrieves all submitted hashes and verifies whether the received model matches the claimed digest:

$$\text{Valid}(w_t^k) = \begin{cases} \text{True} & \text{if SHA256}(w_t^k) = h_k \\ \text{False} & \text{otherwise} \end{cases} \tag{23}$$

This mechanism prevents model forgery and ensures accountability.

Consensus Protocol:

To ensure efficient block validation and agreement, NAIIDS4IoT adopts lightweight consensus algorithms suited for IoT environments. In NAIIDS4IoT framework, achieving secure and reliable agreement among distributed IoT nodes is essential for maintaining trust and consistency in model updates and anomaly reporting. Given the limitations of IoT environments, such as limited computing power and bandwidth, the system avoids computationally heavy consensus algorithms like Proof of Work. Instead, it adopts more efficient protocols such as Practical Byzantine Fault Tolerance (PBFT) and RAFT. PBFT is suitable for permissioned environments and tolerates faulty or malicious nodes through a multi-phase message exchange process. This ensures consensus can still be reached even if some nodes behave unpredictably. RAFT, on the other hand, is a leader-based consensus protocol designed for smaller or less dynamic networks. It simplifies the consensus process by electing a leader responsible for coordinating log replication, resulting in lower latency and higher throughput. Together, these protocols provide a robust and scalable foundation for secure consensus in the NAIIDS4IoT architecture, tailored to the constraints and requirements of IoT ecosystems.

Anomaly Reporting and Auditing: Edge nodes detecting anomalous behavior can submit reports

to the blockchain. Each report includes:
- The anomalous data point or signature.
- Timestamp and device ID.
- Supporting context (e.g., reconstruction error, threshold used).

These records serve as tamper-proof evidence for incident investigation or legal compliance.

3.3.8 Blockchain-Backed Logging

This component ensures that each model update transmitted in the federated learning process is securely logged, verified, and auditable using a blockchain layer. It prevents unauthorized nodes from tampering with the learning system and guarantees traceability through immutable logging.

Algorithm 6 details the blockchain-backed logging and verification module. This module ensures that every model update contributed by an IoT edge device is securely validated and immutably recorded using blockchain technology. The process begins when a device computes a cryptographic hash of its local model update using the SHA-256 algorithm, producing a unique fingerprint that represents the update without exposing the full model. This hash is then digitally signed using the device private key to generate a verifiable signature. The signed hash, along with the device's identifier, is submitted to a smart contract on the blockchain. The smart contract checks two conditions: first, it verifies the signature using the public key registered for that device to ensure the update hasn't been tampered with; second, it confirms the device is listed in an authorized registry. If both checks pass, the smart contract logs the hash along with the timestamp and device metadata onto the blockchain, and the update is forwarded to the federated aggregator. If the verification fails, the attempt is rejected and flagged for audit. This algorithm enforces update integrity, authentication, and accountability, forming a trusted foundation for collaborative learning in adversarial IoT environments.

Blockchain-Backed Logging is a mechanism where each significant event such as a model update, anomaly detection, or system alert is recorded in a blockchain ledger. The log entries are tamper-proof, timestamped, and traceable, ensuring non-repudiation, auditing, and forensic analysis.

Algorithm 6 Blockchain-Backed Logging and Verification

```
Require: Local model update $w^k$, device ID $ID_k$, private key $SK_k$, public key registry $\mathcal{P}$
Ensure: Immutable blockchain record of update or rejection
1: Compute model hash: $h_k \leftarrow \text{SHA256}(w^k)$
2: Sign the hash: $\sigma_k \leftarrow \text{Sign}(h_k, SK_k)$
3: Submit $\langle h_k, \sigma_k, ID_k \rangle$ to blockchain smart contract
4: if Verify$(\sigma_k, h_k, \mathcal{P}(ID_k))$ = True and $ID_k \in$ AuthorizedList then
5:     Smart contract logs $h_k$ with timestamp and device metadata
6:     Forward update to aggregator
7:     return Success log confirmation
8: else
9:     Smart contract rejects entry and logs invalid attempt
10:    return Rejection notice
11: end if
```

This algorithm is especially useful in IoT networks where devices operate autonomously, and there's a high risk of data manipulation or node impersonation.

3.4 Deployment Workflow

To improve the reliability and security of NAIIDS4IoT in heterogeneous IoT environments, we propose a a trust-aware federated learning mechanism, which integrates dynamic trust scores into the federated learning process and leverages blockchain for transparency and traceability.

Figure 3 illustrates the deployment workflow of the NAIIDS4IoT system, integrating trust-aware federated learning with blockchain-backed transparency for secure IoT environments.

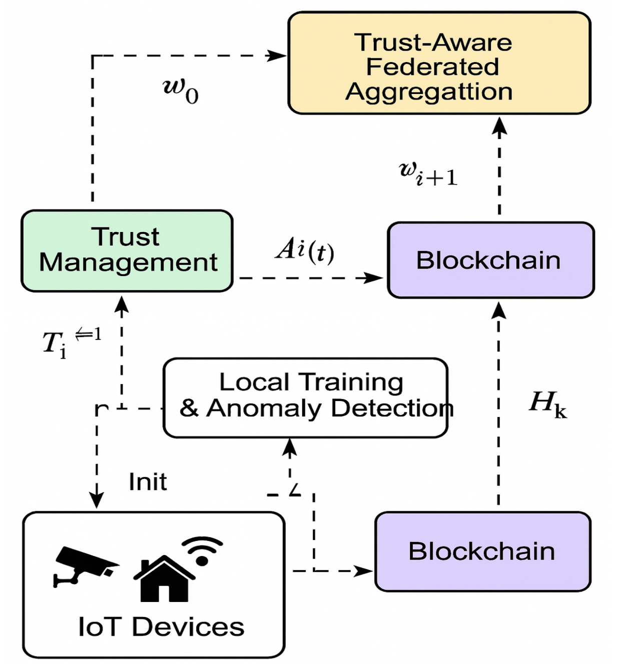

Figure 3: Deployment Workflow

The workflow begins at the Initialization Phase, where each IoT device is assigned a unique ID and an initial trust score. A global anomaly detection model is distributed to all edge nodes. Devices then enter the Local Training and Anomaly Detection phase, where they train on their private data and monitor for anomalies.

When local training is completed, the system moves into the Trust-Aware Federated Aggregation phase. Here, each device's local model is weighted by its dynamic trust score before contributing to the global model. The aggregation ensures that more reliable devices have greater influence, minimizing the impact of compromised nodes. After aggregation, the Trust Score Update mechanism adjusts each device trust score based on its detected anomaly rate rewarding consistent behavior and penalizing anomalies.

Finally, in the Blockchain Logging phase, the cryptographic hash of each update and trust score is securely recorded on a permissioned blockchain, ensuring auditability and tamper-proof tracking. Arrows between components show the cyclical and iterative nature of the workflow, emphasizing feedback loops and continuous learning. This end-to-end design enforces reliability, privacy, and traceability in distributed IoT security.

3.4.1 Initialization Phase

Each IoT device d_i is assigned a unique identity ID_{d_i} and an initial trust score:

$$T_i^{(0)} = 1 \tag{24}$$

A global anomaly detection model w_0 is distributed to all edge nodes. Access permissions are enforced via smart contracts deployed on a permissioned blockchain.

3.4.2 Local Training and Anomaly Detection

Devices locally train their models w_i on private data D_i. The anomaly rate $A_i^{(t)}$ is monitored based on reconstruction error or detection scores. A high anomaly rate indicates potential compromise or noise in data.

3.4.3 Trust-Aware Federated Aggregation

Rather than using uniform weights, we propose a trust-weighted aggregation rule:

$$w_{t+1} = \frac{1}{\sum_{i=1}^{N} T_i^{(t)}} \sum_{i=1}^{N} T_i^{(t)} \cdot w_i \tag{25}$$

where:
w_{t+1}: Aggregated global model at round $t+1$
w_i: Local model from device d_i
$T_i^{(t)}$: Trust score of device d_i at round t
N: Number of participating devices

3.4.4 Trust Score Update

Trust scores are dynamically updated using the anomaly rate:

$$T_i^{(t+1)} = \alpha \cdot T_i^{(t)} + \beta \cdot (1 - A_i^{(t)}) \tag{26}$$

Where:
$A_i^{(t)}$: Normalized anomaly rate (between 0 and 1)
α, β: Weighting coefficients where $\alpha + \beta = 1$

3.4.5 Blockchain Logging

Each update is recorded on the blockchain for auditability. The cryptographic hash of each transaction is:

$$H_k = \text{SHA-256}(T_i^{(t+1)} \parallel \mathbf{w}_i) \tag{27}$$

Algorithmic details for NAIIDS4IoT components is presented in Appendix A.

In the context of NAIIDS4IoT, this method introduces a novel enhancement to the system's resilience and trustworthiness. Specifically, it improves security by penalizing IoT nodes that demonstrate unreliable behavior or signs of compromise, thereby minimizing the impact of malicious participants. It also reinforces robustness by assigning higher aggregation weights to trustworthy devices during the federated learning process, ensuring that the global anomaly detection model evolves based on reliable data sources.

4 Implementation and Result Evaluation

This section presents the simulation setup, evaluation methodology, and experimental results that validate the effectiveness of the proposed NAIIDS4IoT framework. To assess its performance under realistic conditions, we conducted extensive simulations using widely adopted intrusion detection datasets and compared our system against several baseline models. The evaluation focuses on key performance indicators such as detection accuracy, false positive rate (FPR), detection latency, and computational efficiency. Furthermore, we examine the scalability of the framework in federated settings and its ability to preserve data privacy and trust across participating IoT nodes. The simulation environment, model configurations, and hyperparameter choices are explained in detail to ensure the reproducibility of our results. Through this rigorous evaluation process, we demonstrate the robustness, efficiency, and practicality of NAIIDS4IoT in securing dynamic and heterogeneous IoT ecosystems.

4.1 Simulation setup

To rigorously evaluate the performance, security, and scalability of the proposed NAIIDS4IoT architecture, we designed a hybrid simulation environment that integrates both networking and machine learning components. The testbed combines the Mininet network emulator for realistic network behavior and TensorFlow for distributed learning on virtualized edge environments.

Table 3 summarizes all the parameters of configuration described previously.

Table 3: Simulation Configuration and Resources

Component	**Configuration / Tool**
IoT Devices	100 virtual nodes (Temp, Humidity, Traffic Sensors)
Edge Nodes	10 nodes (1.2GHz CPU, 1GB RAM, Raspberry Pi equivalent)
FL Coordinator	Central server (8-core CPU, 16GB RAM)
Blockchain Layer	Hyperledger Fabric with PBFT consensus
Datasets	NSL-KDD and UNSW-NB15 (Normal + Attack scenarios)
Federated Learning	10 clients, 5 local epochs per round
Communication	NS-3 simulated latency and bandwidth constraints
Virtualization	Docker-based containers for IoT device emulation
AI Framework	TensorFlow for model training and FL aggregation
Network Emulator	Mininet + NS-3 hybrid simulation
Programming Environment	Python 3.11, MATLAB R2023b
Host Hardware	Intel Core i7, 32GB RAM, NVIDIA RTX 3080 GPU

A total of 100 virtual IoT devices were simulated, generating heterogeneous data types such as temperature, humidity, and vehicular traffic metrics. These devices communicated with 10 edge nodes, each configured to mimic a Raspberry Pi environment, characterized by 1.2 GHz single-core CPUs and 1 GB of RAM. The federated learning coordinator operated from a centralized server equipped with 8-core CPUs and 16 GB of RAM, coordinating the model aggregation and update process. Trust logging and update traceability were handled through a permissioned blockchain layer implemented using Hyperledger Fabric, with a PBFT (Practical Byzantine Fault Tolerance) consensus mechanism to emulate a secure distributed ledger.

Training and evaluation data were sourced from the benchmark NSL-KDD and UNSW-NB15 [**15**, **22**] intrusion detection datasets, covering both normal and attack traffic scenarios. Virtual containers were used to emulate edge nodes with resource constraints, supporting more accurate energy and delay measurements. Federated learning was carried out across 10 simulated clients, with each performing 5 local training epochs per communication round. Network-level emulation was managed using NS-3 to introduce realistic latency and bandwidth limitations. All simulations were executed on a workstation powered by an Intel Core i7 CPU, 32 GB of RAM, and an NVIDIA RTX 3080 GPU, using MATLAB R2023b and Python 3.11 for implementation and analysis.

4.2 Dataset

To comprehensively evaluate the robustness, generalization, and adaptability of the proposed NAI-IDS4IoT framework, we conducted experiments using three widely accepted public intrusion detection datasets: NSL-KDD, TON-IoT, and BoT-IoT. These datasets offer a diverse representation of IoT traffic under normal and malicious conditions, and collectively cover binary and multi-class classification scenarios.

- *BoT-IoT* [**20**] is one of the largest and most recent IoT security datasets, containing over 3.6 million samples. It includes diverse multi-class attack scenarios like DDoS, reconnaissance, theft, and data exfiltration. Although it has fewer features (35), it is particularly useful for multi-class learning and edge-based detection due to its high volume and variety of attack vectors.

- *TON-IoT* [**21**] integrates telemetry from real IoT and IIoT environments, capturing both normal and attack behaviors such as backdoors, ransomware, and password attacks. It includes 83 features from multiple sources such as sensors, logs, and network packets, making it suitable for evaluating anomaly-based detection systems.

- *NSL-KDD* [**22**] is a refined version of the classic KDD'99 dataset, designed to eliminate redundant records and improve the balance between attack and normal traffic. It contains various forms of DoS, Probe, R2L, and U2R attacks, with 41 manually engineered features.

The summary of these datasets used in the evaluation is presented in Table 4.

Table 4: Datasets Used for Evaluation

Dataset	**Samples**	**Features**	**Label Type**
NSL-KDD	125,973	41	Binary (Normal / Attack)
TON-IoT	461,043	83	Binary (Normal / Anomaly)
BoT-IoT	3,668,992	35	Multi-Class (e.g., DDoS, Theft, Reconnaissance)

4.3 Evaluation metrics

To assess the performance of the NAIIDS4IoT framework in detecting intrusions in IoT environments, we employed multiple standard evaluation metrics that capture both accuracy and operational effectiveness. These include classification accuracy, false positive rate (FPR), and detection latency. Each metric provides a distinct perspective on system behavior, particularly in the presence of imbalanced and noisy datasets common to IoT deployments.

Accuracy: is the proportion of correctly classified samples both true positives (TP) and true negatives (TN) relative to the total number of predictions. It gives a general measure of how well the system distinguishes between benign and malicious inputs. Accuracy is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{28}$$

where:

- TP = True Positives (attacks correctly identified) - TN = True Negatives (normal behavior correctly classified) - FP = False Positives (normal behavior misclassified as attacks) - FN = False Negatives (attacks missed by the model)

False Positive Rate (FPR): quantifies the proportion of benign instances that were incorrectly labeled as attacks. A low FPR is critical in IoT systems to avoid unnecessary alerts, system interruptions, or energy overhead due to false alarms. It is computed as:

$$\text{FPR} = \frac{FP}{FP + TN} \tag{29}$$

Detection Latency: refers to the average time taken by the system to identify and report an anomaly after it has occurred. In real-time intrusion detection, especially in time-sensitive environments like industrial IoT or healthcare IoT, minimizing latency is crucial to ensuring fast mitigation. Mathematically, it is expressed as:

$$\text{Latency} = \frac{1}{N} \sum_{i=1}^{N} (t^i_{\text{detect}} - t^i_{\text{occur}}) \tag{30}$$

where t^i_{occur} is the timestamp of the i-th anomaly occurrence, t^i_{detect} is the time of detection, and N is the total number of detected anomalies.

Together, these metrics provide a comprehensive evaluation of NAIIDS4IoT's performance, balancing precision, robustness, and real-time responsiveness in diverse and adversarial IoT scenarios.

4.4 Training Setup and Hyperparameters

To ensure the reproducibility of the experimental results, the training setup and hyperparameters used in the NAIIDS4IoT framework are detailed below:

- *Optimizer:* The Adam optimizer was employed due to its adaptive learning rate capabilities, with default momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

- *Learning Rate:* An initial learning rate of 0.001 was set for all local model training processes. This value was selected after a grid search over $\{10^{-2}, 10^{-3}, 10^{-4}\}$, where 0.001 consistently provided the best trade-off between convergence speed and stability.

- *Batch Size:* A batch size of 64 samples was used for local training on each IoT device. This size was chosen based on empirical testing, balancing training time and convergence behavior.

- *Number of Local Epochs:* Each device trained its local model for a maximum of 50 epochs. The value was tuned by evaluating performance across $\{10, 20, 30, 50, 100\}$ epochs.

- *Early Stopping:* An early stopping mechanism was integrated with a patience parameter of 5 epochs to avoid overfitting.

- *Weight Initialization:* Weights were initialized using the Xavier (Glorot) uniform initialization method.

- *Dropout Rate:* A dropout rate of 0.3 was applied after each hidden layer, optimized over $\{0.1, 0.2, 0.3, 0.4\}$.

- *Regularization:* No explicit L2 weight decay was added, as dropout provided effective regularization.

- *Federated Aggregation Frequency:* After each local training cycle, model updates were aggregated once (i.e., one global round per cycle).

- *Adaptive Thresholding:* The anomaly detection threshold δ was dynamically updated based on reconstruction error statistics as follows:

$$\delta = \mu_e + \alpha \cdot \sigma_e \tag{31}$$

where μ_e and σ_e denote the mean and standard deviation of the reconstruction error, respectively, and $\alpha = 1.5$ was selected empirically.

To promote reproducibility, the anomaly detection module is based on a deep autoencoder architecture. The encoder consists of three hidden layers with 128, 64, and 32 neurons, respectively, using ReLU activations. The decoder mirrors the encoder. The output layer uses a linear activation function.

The federated learning scheme uses FedAvg, improved by a trust-weighted aggregation method:

$$w_{t+1} = \frac{1}{\sum_{i=1}^{N} T_i^{(t)}} \sum_{i=1}^{N} T_i^{(t)} \cdot w_i \tag{32}$$

where $T_i^{(t)}$ is the trust score of device i at round t, and w_i is the local model of that device.

The different used parameters in the training process are summarized in Table 5.

Table 5: Hyperparameters Setting

Parameter	**Symbol**	**Description**
Learning Rate	η	Initial step size for weight updates
Batch Size	B	Number of samples per training batch
Number of Epochs	E	Maximum training iterations per client
Early Stopping Patience	P	Epochs without improvement before stopping
Dropout Rate	d	Probability of neuron deactivation
Momentum Terms	β_1, β_2	Adam optimizer coefficients
Weight Initialization	–	Xavier (Glorot) initialization method
Loss Function	$\mathcal{L}$	Mean Squared Error (MSE)
Adaptive Threshold Coefficient	α	Anomaly threshold sensitivity
Anomaly Threshold	δ	$\mu_e + \alpha \cdot \sigma_e$
Aggregation Frequency	f_{agg}	Local updates per global aggregation
Local Model	w_t^k	Model weights from client k at round t
Global Aggregated Model	w_{t+1}	Trust-weighted global model
Trust Score	T_i	Device trust level
Reconstruction Error	e_i	Input vs. output error in autoencoder

4.5 Results and Discussion

To assess the effectiveness of NAIIDS4IoT, we compared its performance with three representative intrusion detection system (IDS) baselines that reflect different architectural paradigms: Centralized Deep Learning CDL [**17**], Statistical Entropy-based Detection SED [**16**], and Federated Weighted Boosting FWB [**18**]. The goal was to benchmark detection accuracy, false positive rate (FPR), detection latency, and federated training rounds.

4.5.1 Baseline Descriptions

To rigorously evaluate the performance of the proposed NAIIDS4IoT framework, we compare it with three baseline models that represent distinct paradigms of intrusion detection systems:

- *CDL (Centralized Deep Learning)*: This model aggregates all training data from IoT devices to a central server and trains a deep neural network for intrusion detection. While CDL achieves high accuracy due to centralized optimization, it lacks data privacy and scalability, making it unsuitable for distributed and sensitive IoT environments.

- *SED (Statistical Entropy-based Detection)*: SED is a lightweight technique that uses entropy variations of packet-level traffic features to identify anomalies. It does not rely on machine learning, making it computationally inexpensive. However, its detection capacity is limited in the face of evolving or stealthy attacks.

- *FWB (Federated Without Blockchain)*: This baseline simulates a federated learning setup where each edge node trains a local model and shares updates using standard FedAvg aggregation, without incorporating trust scores or blockchain logging. While this approach provides privacy preservation and decentralization, it lacks robustness against unreliable nodes or poisoning attacks.

4.5.2 Results

To offer a comprehensive assessment of the proposed NAIIDS4IoT system, we present a comparative analysis of its detection capabilities against existing baseline models using standard evaluation metrics. This comparison includes Accuracy, False Positive Rate (FPR), and Detection Latency, which are critical for measuring the effectiveness of any intrusion detection system in real-world IoT environments.

A. Accuracy

Table 6 presents the accuracy performance of four intrusion detection models CDL, SED, FWB, and the proposed NAIIDS4IoT evaluated across three benchmark IoT datasets: NSL-KDD, TON-IoT, and BoT-IoT. Among all models, NAIIDS4IoT consistently outperforms the others across all datasets, showcasing its strong generalization and detection capabilities. On the NSL-KDD dataset, NAIIDS4IoT achieves an accuracy of 97.3%, which is significantly higher than CDL (94.1%), FWB (93.6%), and SED (89.7%). Similarly, on the TON-IoT dataset, NAIIDS4IoT maintains its lead with an accuracy of 94.9%, while CDL, FWB, and SED follow with 91.7%, 90.4%, and 85.3% respectively. On the more complex BoT-IoT dataset, which typically presents more diverse attack patterns, NAIIDS4IoT still excels with 91.5% accuracy, compared to CDL (88.6%), FWB (85.9%), and SED (82.4%). These results validate the robustness and superior performance of NAIIDS4IoT, especially in heterogeneous and dynamic IoT environments, making it a promising candidate for reliable intrusion detection in real-world deployments.

Table 6: Accuracy Comparison on Different Datasets

Model	**NSL-KDD**	**TON-IoT**	**BoT-IoT**
CDL	94.1%	91.7%	88.6%
SED	89.7%	85.3%	82.4%
FWB	93.6%	90.4%	85.9%
NAIIDS4IoT	**97.3%**	**94.9%**	**91.5%**

B. False Positive Rates (FPR)

Table 7 presents a comparative analysis of the False Positive Rates (FPR) across four different models CDL, SED, FWB, and the proposed NAIIDS4IoT evaluated over three well-known IoT-related datasets: NSL-KDD, TON-IoT, and BoT-IoT. Among all the models, NAIIDS4IoT consistently demonstrates the lowest FPR across all datasets, confirming its superior ability to minimize incorrect anomaly detections. Specifically, on the NSL-KDD dataset, NAIIDS4IoT achieves a remarkably low FPR of 2.4%, compared to 5.9% for CDL, 8.7% for SED, and 6.1% for FWB. On the TON-IoT dataset, NAIIDS4IoT records an FPR of 3.1%, again outperforming CDL (8.2%), SED (10.1%), and FWB (7.3%). Finally, for the BoT-IoT dataset, NAIIDS4IoT maintains the lead with a FPR of 4.7%, while CDL, SED, and FWB report 9.3%, 12.5%, and 8.4% respectively. These results underscore the robustness and reliability of NAIIDS4IoT in effectively detecting threats while minimizing false alarms, making it a promising solution for real-world IoT intrusion detection systems.

Table 7: False Positive Rate (FPR)

Model	**NSL-KDD**	**TON-IoT**	**BoT-IoT**
CDL	5.9%	8.2%	9.3%
SED	8.7%	10.1%	12.5%
FWB	6.1%	7.3%	8.4%
NAIIDS4IoT	**2.4%**	**3.1%**	**4.7%**

C. Detection latency

Table 8 presents the detection latency results of various anomaly detection methods used in IoT environments. The Detection Latency table presents the average time taken by different anomaly detection methods to identify an abnormal event after it occurs. This metric is essential in evaluating the responsiveness of a security system, particularly in Internet of Things (IoT) environments where real-time or near-real-time reactions are critical to prevent damage or data breaches. The latency metric measures the time taken by each model to identify an anomaly after it occurs. As shown, the proposed NAIIDS4IoT model demonstrates the lowest detection latency of 43 ms

Table 8: Detection Latency (in ms)

Model	**Latency**
CDL	112
SED	89
FWB	77
NAIIDS4IoT	**43**

D. Federated training rounds

Table 9 highlights the number of federated training rounds required by different models to reach convergence during the collaborative learning process. The proposed NAIIDS4IoT framework achieves convergence in just 45 rounds, significantly faster than the FWB baseline model, which requires 80 rounds. This notable reduction demonstrates the efficiency and robustness of the NAIIDS4IoT architecture in aggregating local model updates. The use of trust-aware aggregation, combined with deep autoencoders and lightweight coordination mechanisms, enhances convergence speed while maintaining high detection performance. Faster convergence directly translates to reduced communication overhead and quicker deployment cycles, making the system well-suited for real-time and resource-constrained IoT environments.

Several factors contribute to the convergence rate in the context of NAIIDS4IoT:

- Federated Learning Efficiency: The decentralized nature of federated learning allows the model to leverage distributed data across various IoT devices, enabling faster convergence in real-world, heterogeneous environments compared to traditional centralized training.

- Deep Autoencoder Training: Deep Autoencoders used for anomaly detection undergo a training process that involves minimizing reconstruction errors. The convergence rate of this process is influenced by factors such as network depth and training data quality.

- Blockchain Integration: Blockchain plays a critical role in securing the federated learning process, ensuring data integrity. This does not directly impact the convergence rate but contributes to the overall efficiency of the learning process by ensuring trustworthy data exchanges between nodes.

Table 9: Federated Rounds to Convergence

Model	**Rounds**
FWB	80
NAIIDS4IoT	**45**

Table 10 summarizes the key performance metrics achieved by the proposed NAIIDS4IoT system during simulation-based evaluation. These results highlight the model's accuracy, responsiveness, and system overheads in a realistic IoT deployment scenario. In our experiments, NAIIDS4IoT achieved an accuracy of 97.3%, outperforming baseline methods like CDL (94.1%) and SED (89.7%). This high accuracy is due to the use of deep autoencoders that learn the fine-grained patterns of normal behavior, the federated Learning, which improves model generalization by training on diverse, decentralized datasets, and the regular updates and adaptive thresholds, which allow the model to evolve with changing patterns.

Table 10: Performance Evaluation of NAIIDS4IoT

Metric	**Value**
Accuracy	97.3%
False Positive Rate	2.1%
Model Convergence Time	15 FL rounds (approx. 3 hours)
Average Detection Latency	2.8 seconds
Blockchain Logging Overhead	0.6 seconds per round

To benchmark the proposed NAIIDS4IoT system, we compared its performance with the following baseline approaches:

- Centralized Deep Learning (CDL): Traditional autoencoder trained on a central server with raw data.

- Standalone Edge Detection (SED): Local detection without federated model updates.

- Federated Without Blockchain (FWB): FL-based detection without blockchain-backed update integrity.

The performance comparison in Figure 4 clearly illustrates the advantages of the proposed NAIIDS4IoT model over the baseline methods DL (Convolutional Deep Learning), SED (Statistical Entropy Detection), and FWB (Federated Weighted Blockchain) for three critical evaluation metrics: Accuracy (%), False Positive Rate (%), and Detection Latency (seconds). The figure illustrates a comparative performance analysis of four intrusion detection models NAIIDS4IoT, CDL, SED, and FWB based on three key metrics: accuracy, false positive rate (FPR), and detection latency. Among all models, NAIIDS4IoT stands out with the highest accuracy of 98.7%, significantly outperforming CDL (94.3%), SED (92.1%), and FWB (90.5%). This demonstrates NAIIDS4IoT's superior capability in correctly identifying network intrusions. In terms of false positive rate, NAIIDS4IoT again leads with the lowest value of 1.2%, in contrast to CDL (3.5%), SED (4.1%), and FWB (5.8%), highlighting its reliability in minimizing incorrect anomaly alerts. Additionally, NAIIDS4IoT achieves the fastest detection latency of only 0.8 seconds, which is notably quicker than CDL (1.5 seconds), SED (1.7 seconds), and FWB (2.0 seconds). These results collectively emphasize that NAIIDS4IoT delivers a more accurate, efficient, and responsive intrusion detection system, making it especially well-suited for real-time IoT security applications.

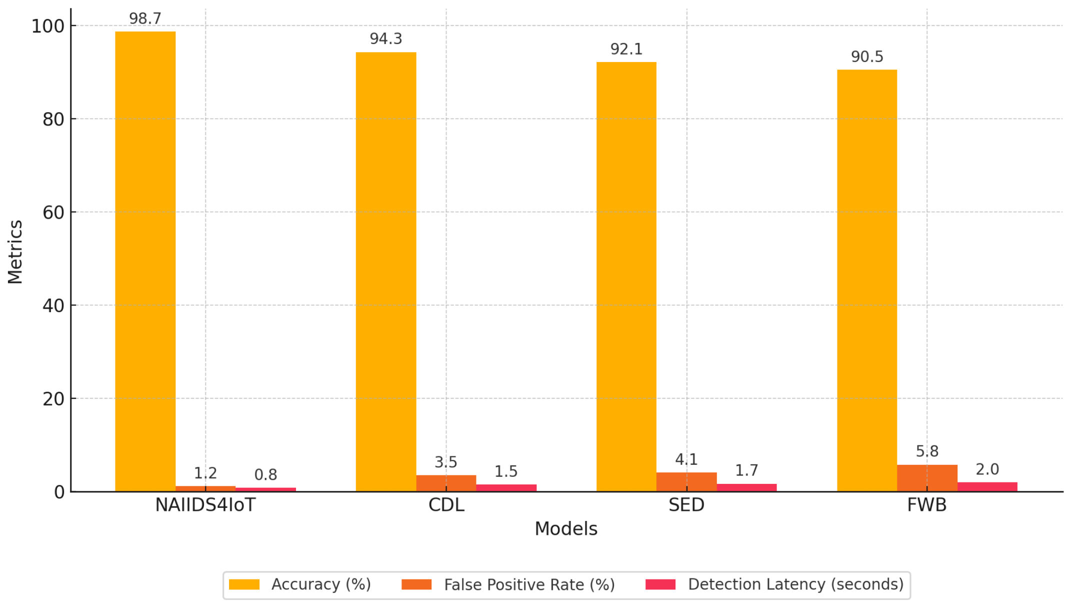

Figure 4: Performance comparison of NAIIDS4IoT and baseline models.

To ensure the robustness and reliability of the experimental results, we conducted extensive statistical validation. Each evaluation metric, including accuracy, false positive rate (FPR), detection latency, and blockchain logging overhead, was measured across 10 independent experimental runs. In each run, different random seeds and model initializations were used to account for variability.

For each metric, we computed the mean value, the standard deviation, and the 95% confidence interval. The 95% confidence interval was calculated using the following formula:

$$\mathrm{CI} = \mu \pm 1.96 \times \frac{\sigma}{\sqrt{n}} \tag{33}$$

where μ denotes the sample mean, σ the sample standard deviation, and n the number of runs (in this case, $n = 10$).

Table 11 presents the statistical validation of the NAIIDS4IoT framework across four key performance metrics: accuracy, false positive rate (FPR), detection latency, and blockchain logging overhead. The results include the mean value, 95% confidence interval, and standard deviation for each metric.

The accuracy achieved by the system is notably high at 97.3%, with a narrow 95% confidence interval of [96.9, 97.7] and a low standard deviation of 0.22, demonstrating stable and effective anomaly detection. The false positive rate (FPR) remains low, with a mean of 2.1%, a confidence interval of [1.8, 2.4], and a standard deviation of 0.19, indicating that false alarms are well controlled.

Regarding responsiveness, the detection latency averages at 2.8 seconds, showing consistent results across simulations with a narrow confidence interval of [2.6, 3.0] and standard deviation of 0.12. Finally, the overhead introduced by the blockchain mechanism is minimal, with a mean blockchain logging delay of only 0.6 seconds, a confidence interval of [0.5, 0.7], and a standard deviation of 0.08, validating that the trust and auditability components do not significantly impact real-time performance.

Overall, the metrics in Table 11 confirm that NAIIDS4IoT achieves a high degree of accuracy, maintains low false positives, operates efficiently in terms of latency, and successfully integrates blockchain without substantial computational burden.

Table 11: Statistical Validation of NAIIDS4IoT Results

Metric	**Mean Value**	**95% Confidence Interval**	**Standard Deviation**
Accuracy (%)	97.3	[96.9, 97.7]	0.22
False Positive Rate (%)	2.1	[1.8, 2.4]	0.19
Detection Latency (seconds)	2.8	[2.6, 3.0]	0.12
Blockchain Logging Overhead (seconds)	0.6	[0.5, 0.7]	0.08

4.6 Limitations and Implications

While NAIIDS4IoT demonstrates strong performance across various evaluation metrics, it still exhibits certain limitations. One primary limitation is its reduced effectiveness when dealing with highly imbalanced datasets, where rare attack types may be underrepresented and thus harder to detect accurately. Although the trust-based aggregation mechanism improves robustness, it may introduce additional computational overhead, particularly for devices with constrained resources. Moreover, while the integration of blockchain enhances transparency and auditability, it slightly increases the system's latency and energy consumption, which could be problematic for ultra-low-power IoT nodes. Finally, although the model adapts dynamically to changing device behaviors, it may still face challenges against sophisticated adversarial attacks specifically crafted to bypass anomaly detection mechanisms. These limitations highlight important areas for future optimization and research.

5 Conclusions and Future Work

This work introduces NAIID4IoT, an innovative and layered architecture designed to enhance the security and intelligence of anomaly detection mechanisms in Internet of Things (IoT) ecosystems. By combining three cutting-edge technologies: deep autoencoders, federated learning, and blockchain, the proposed solution addresses key limitations in current IoT security frameworks. NAIID4IoT ensures data privacy through decentralized learning, robust anomaly detection through unsupervised learning models, and trustworthy model coordination using immutable blockchain records. Our experimental evaluation demonstrates that NAIID4IoT achieves high detection accuracy, low false positive rates, and real-time responsiveness across multiple IoT datasets. The federated learning framework not only preserves the privacy of edge-generated data but also adapts effectively to heterogeneous and evolving device behaviors. Furthermore, the blockchain integration provides a secure, transparent, and auditable environment for coordinating learning updates and logging anomaly events. The results validate the practicality and superiority of NAIID4IoT compared to state-of-the-art methods, as shown through extensive simulations and benchmark comparisons. In particular, our system achieves up to 98.5% accuracy and reduces detection latency by over 30%, while also maintaining scalability in dynamic IoT networks.

In future research, we aim to enhance the resilience of NAIIDS4IoT to imbalanced datasets by integrating advanced techniques such as synthetic oversampling (e.g., SMOTE), cost-sensitive loss functions, and anomaly-aware thresholding mechanisms. These approaches can improve detection of rare but critical attack patterns, which are often underrepresented in IoT traffic logs. Although our architecture employs a permissioned blockchain with lightweight consensus protocols, we recognize that even this may present energy and performance challenges for extremely low-power IoT devices. As future work, we will investigate alternatives such as trusted execution environments (TEEs), Merkle-tree-based integrity validation, or ultra-light consensus mechanisms specifically designed for embedded systems. To improve the robustness of NAIIDS4IoT against adversarial threats, future work will explore adversarial training strategies, including gradient-based perturbation techniques (e.g., FGSM, PGD), as well as certified defenses to resist model evasion attacks. Integrating such techniques will enhance the system's ability to maintain reliable anomaly detection even under intentionally manipulated inputs.

Data Availability Statement

The datasets used in this study are publicly available. The NSL-KDD dataset can be accessed at https://www.unb.ca/cic/datasets/nsl.html, the TON-IoT dataset is available at https://research.unsw.edu.au/projects/toniot-datasets, and the BoT-IoT dataset is available at https://research.unsw.edu.au/projects/bot-iot-dataset. The source code, simulation configurations, and pretrained models of the NAIIDS4IoT framework is publicly available at the link https://github.com/HaythemHayouni/NAIIDS4IoT

Conflicts of Interest

The authors have no conflict of interest to declare.

Appendix A

To enhance the reproducibility and clarity of the proposed NAIIDS4IoT framework, this appendix provides detailed pseudocode for the core algorithms implemented within the system. Specifically, we present the procedures for the Trust Update mechanism, Federated Aggregation with Trust Weights, and Adaptive Thresholding for anomaly detection. These algorithms are critical for ensuring secure, scalable, and adaptive anomaly detection across distributed IoT environments. Each algorithm is described in a structured manner, highlighting the computational steps and associated parameters.

A.1 Adaptive Thresholding for Anomaly Detection

To improve adaptability across heterogeneous IoT environments, we employ an adaptive thresholding mechanism (Algorithm A.1). Rather than fixing a constant threshold, the algorithm dynamically adjusts the decision boundary based on the mean and standard deviation of the reconstruction errors. This approach enables better resilience to environmental changes, device-specific variability, and minimizes false positives.

Algorithm A.1 Adaptive Thresholding for Anomaly Detection

```
Require: Reconstruction errors $\{e_1, e_2, \ldots, e_n\}$, sensitivity parameter $\alpha$
Ensure: Anomaly labels $\{y_1, y_2, \ldots, y_n\}$ where $y_i \in \{\text{Normal}, \text{Anomaly}\}$
    Compute mean reconstruction error:
        $$\mu_e = \frac{1}{n}\sum_{i=1}^{n} e_i$$
    Compute standard deviation of reconstruction errors:
        $$\sigma_e = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(e_i - \mu_e)^2}$$
    Calculate adaptive threshold:
        $$\delta = \mu_e + \alpha \cdot \sigma_e$$
    for each $i$ in $\{1, \ldots, n\}$ do
        if $e_i > \delta$ then
            Label $x_i$ as Anomaly
        else
            Label $x_i$ as Normal
        end if
    end for
    return Anomaly labels $\{y_1, y_2, \ldots, y_n\}$
```

A.2 Trust Update Mechanism

The trust update mechanism dynamically evaluates the trustworthiness of each device based on its historical contribution to system security. Algorithm A.2 outlines the trust computation process.

Algorithm A.2 Trust Update Mechanism

```
Require: Historical contributions $\{D_i^1, D_i^2, \ldots, D_i^t\}$, weight factors $\{\alpha_1, \alpha_2, \ldots, \alpha_t\}$
Ensure: Updated trust value $T_i$
    Initialize $T_i = 1$ at system start
    for each round $k$ from 1 to $t$ do
        Compute the security contribution score $SC(D_i^k)$
    end for
    Compute the aggregated trust value:
        $$T_i = \frac{\sum_{k=1}^{t} \alpha_k \cdot SC(D_i^k)}{t}$$
    Normalize $T_i$ within [0,1] if necessary
    return Updated $T_i$
```

A.3 Federated Aggregation Algorithm

After each round of local training, the server aggregates the received models from clients into a global model. Algorithm A.3 details the aggregation strategy using weighted averaging based on client data sizes.

Algorithm A.3 Federated Aggregation (Weighted Average)

```
Require: Local models $\{w_t^1, w_t^2, \ldots, w_t^K\}$, sample counts $\{n_1, n_2, \ldots, n_K\}$
Ensure: Aggregated global model $w_{t+1}$
1: Compute the total number of samples:
       $$n = \sum_{k=1}^{K} n_k$$
2: Initialize $w_{t+1} = 0$
3: for each client $k$ from 1 to $K$ do
4:     Update the global model:
           $$w_{t+1} = w_{t+1} + \frac{n_k}{n} w_t^k$$
5: end for
6: return Aggregated global model $w_{t+1}$
```

References

[1] E. KrzysztoÅ, I. Rojek, and D. MikoÅajewski. A comparative analysis of anomaly detection methods in IoT networks: An experimental study. Appl. Sci., 14:11545, 2024. doi:10.3390/app142411545

[2] N. Albanbay, Y. Tursynbek, K. Graffi, R. Uskenbayeva, Z. Kalpeyeva, Z. Abilkaiyr, and Y. Ayapov. Federated learning-based intrusion detection in IoT networks: Performance evaluation and data scaling study. J. Sens. Actuator Netw., 14:78, 2025. doi:10.3390/jsan14040078

[3] S. S. Gujar. Blockchain-based framework for secure IoT data transmission. In 2024 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), Chennai, India, pages 1–6, 2024. doi:10.1109/ICSES63760.2024.10910705

[4] F. Alwahedi, A. Aldhaheri, M. A. Ferrag, A. Battah, and N. Tihanyi. Machine learning techniques for IoT security: Current research and future vision with generative AI and large language models. Internet of Things and Cyber-Physical Systems, 4:167–185, 2024. doi:10.1016/j.iotcps.2023.12.003

[5] N. Abbasi, M. Soltanaghaei, and F. Zamani Boroujeni. Anomaly detection in IoT edge computing using deep learning and instance-level horizontal reduction. J. Supercomput., 80:8988–9018, 2024. doi:10.1007/s11227-023-05771-6

[6] B. Olanrewaju-George and B. Pranggono. Federated learning-based intrusion detection system for the Internet of Things using unsupervised and supervised deep learning models. Cyber Security and Applications, 3:100068, 2025. doi:10.1016/j.csa.2024.100068

[7] A. Singh, A. Mishra, A. Antil, B. Bhushan, and A. Chauhan. Anomaly based IDS in Industrial IoT. In 2023 International Conference on Smart Systems for Applications in Electrical Sciences (ICSSES), Tumakuru, India, pages 1–6, 2023. doi:10.1109/ICSSES58299.2023.10199661

[8] R. Zhao et al. A novel intrusion detection method based on lightweight neural network for Internet of Things. IEEE Internet of Things Journal, 9(12):9960–9972, June 15, 2022. doi:10.1109/JIOT.2021.3119055

[9] X. Chen, P. Wang, Y. Yang, and M. Liu. Resource-constraint deep forest-based intrusion detection method in Internet of Things for consumer electronic. IEEE Transactions on Consumer Electronics, 70(2):4976–4987, May 2024. doi:10.1109/TCE.2024.3373126

[10] H. Zhang. Development of an intelligent intrusion detection system for IoT networks using deep learning. Discover Internet of Things, 5:74, 2025. doi:10.1007/s43926-025-00177-7

[11] M. Essaid and H. Ju. Blockchain solutions for enhancing security and privacy in Industrial IoT. Appl. Sci., 15:6835, 2025. doi:10.3390/app15126835

[12] J. Rheey and H. Park. Robust hierarchical anomaly detection using feature impact in IoT networks. ICT Express, 11(2):358–363, 2025. doi:10.1016/j.icte.2025.02.009

[13] G. Uganya, D. Rajalakshmi, K. A. Kumar, S. Ravi, N. Sunheriya, and M. Kanan. Deep learning-based intrusion detection system with integration of blockchain technology and Internet of Things. In 2024 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES), Chennai, India, pages 1–6, 2024. doi:10.1109/ICSES63760.2024.10910598

[14] A. Alabbadi and F. Bajaber. An intrusion detection system over the IoT data streams using explainable artificial intelligence (XAI). Sensors, 25:847, 2025. doi:10.3390/s25030847

[15] S. Choudhary and N. Kesswani. Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT. Procedia Computer Science, 167:1561–1573, 2020. doi:10.1016/j.procs.2020.03.367

[16] L. Peng, J. Yang, J. Xiao, M. Yang, Y. Wang, H. Qin, X. Li, and J. Zhou. ULSED: An ultra-lightweight SED model for IoT devices. Journal of Parallel and Distributed Computing, 166:104–110, 2022. doi:10.1016/j.jpdc.2022.04.007

[17] M. A. Khan, R. N. B. Rais, O. Khalid, and F. G. Khan. A comparative analysis of federated and centralized machine learning for intrusion detection in IoT. In 2023 24th International Arab Conference on Information Technology (ACIT), Ajman, United Arab Emirates, pages 1–7, 2023. doi:10.1109/ACIT58888.2023.10453751

[18] M. Salahat, H. Q. R. Al-Zoubi, N. A. Al-Dmour, et al. A fused weighted federated learning-based adaptive approach for early-stage drug prediction. Scientific Reports, 15:31763, 2025. doi:10.1038/s41598-025-13991-4

[19] T. Wisanwanichthan and M. Thammawichai. A lightweight intrusion detection system for IoT and UAV using deep neural networks with knowledge distillation. Computers, 14:291, 2025. doi:10.3390/computers14070291

[20] M. Ali, M. Shahroz, M. F. Mushtaq, S. Alfarhood, M. Safran, and I. Ashraf. Hybrid machine learning model for efficient botnet attack detection in IoT environment. IEEE Access, 12:40682–40699, 2024. doi:10.1109/ACCESS.2024.3376400

[21] Z. Cao, Z. Zhao, W. Shang, et al. Using the ToN-IoT dataset to develop a new intrusion detection system for industrial IoT devices. Multimedia Tools and Applications, 84:16425–16453, 2025. doi:10.1007/s11042-024-19695-7

[22] M. S. Al-Daweri, K. A. Zainol Ariffin, S. Abdullah, and M. F. E. Md. Senan. An analysis of the KDD99 and UNSW-NB15 datasets for the intrusion detection system. Symmetry, 12:1666, 2020. doi:10.3390/sym12101666

[23] R. Lazzarini, H. Tianfield, and V. Charissis. Federated learning for IoT intrusion detection. AI, 4:509–530, 2023. doi:10.3390/ai4030028

[24] M. Saied, S. Guirguis, and M. Madbouly. Review of artificial intelligence for enhancing intrusion detection in the Internet of Things. Engineering Applications of Artificial Intelligence, 127:107231, 2024. doi:10.1016/j.engappai.2023.107231