electronics

MDPI

Article

RPFL: A Reliable and Privacy-Preserving Framework for Federated Learning-Based IoT Malware Detection

Mohammed Asiri *, Maher A. Khemakhem, Reemah M. Alhebshi, Bassma S. Alsulami and Fathy E. Eassa

Department of Computer Science, King Abdulaziz University, Jeddah 21589, Saudi Arabia; makhemakhem@kau.edu.sa (M.A.K.); ralhebshi@kau.edu.sa (R.M.A.); balsulami@kau.edu.sa (B.S.A.); feassa@kau.edu.sa (F.E.E.)

* Correspondence: masiri0215@stu.kau.edu.sa

Abstract: The proliferation of Internet of Things (IoT) devices and their vulnerability to malware infections pose critical security challenges in IoT networks and multi-access edge computing (MEC). Traditional federated learning-based IoT malware detection (FL-IMD) methods face limitations in privacy, reliability, and client authentication, necessitating innovative solutions. This study proposes a reliable and privacy-preserving federated learning framework (RPFL) that integrates elliptic curve digital signature algorithm (ECDSA), homomorphic encryption and blockchain technology to enhance privacy, reliability, and client verification in FL-IMD. To address challenges with fully homomorphic encryption (FHE), particularly its reliance on an external aggregator, we introduce two smart contract-based schemes: one to incentivize client participation and another to mitigate aggregator failures. Experimental results on the N-BaIoT dataset show that RPFL achieves IoT malware detection accuracy comparable to state-of-the-art methods, while significantly enhancing reliability and privacy in the aggregation process. Furthermore, our blockchain integration outperforms the prominent blockchain-based FL framework, BCFL, by reducing communication costs and latency. These findings highlight the potential of RPFL to advance privacy-preserving, reliable, and secure FL-based IMD in IoT networks and MEC environments.

Keywords: IoT malware detection; IoT security; federated learning; N-BaIoT dataset; blockchain; blockchain-based federated learning (BCF); smart contracts; privacy preserving; trustworthiness

check for updates

Academic Editor: Aryya Gangopadhyay

Received: 27 January 2025
Revised: 24 February 2025
Accepted: 4 March 2025
Published: 10 March 2025

Citation: Asiri, M.; Khemakhem, M.A.; Alhebshi, R.M.; Alsulami, B.S.; Eassa, F.E. RPFL: A Reliable and Privacy-Preserving Framework for Federated Learning-Based IoT Malware Detection. *Electronics* **2025**, *14*, 1089. https://doi.org/10.3390/electronics14061089

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The exponential growth of cyberattacks, coupled with the vulnerabilities of Internet of Things (IoT) devices connected to advanced networks such as 5G, 6G, and multi-access edge computing (MEC), necessitates robust security solutions [1,2]. For instance, distributed servers owned by different Home Service Providers (HSPs) and deployed in MEC networks are particularly vulnerable to attacks from malware-infected IoT devices, as shown in Figure 1. Such attacks can overload computational resources and disrupt service availability [3]. Federated learning-based IoT malware detection (FL-IMD) presents an advanced collaborative solution for securing these servers. However, designing and validating such a solution is highly complex due to the inherent limitations of traditional federated learning (FL) and the challenges involved in managing the collaborative FL process [4].

Monitoring network behavior has become a widely adopted approach for detecting malware-infected devices in IoT networks [5,6]. Recent studies [7–10] have increasingly leveraged FL, a privacy-preserving machine learning paradigm, to address IoT network

security challenges. FL facilitates collaborative learning while ensuring sensitive data remain decentralized and protected [11].

To evaluate FL-based malware detection, researchers frequently utilize the N-BaIoT dataset, which contains diverse benign and malicious IoT network traffic. Its heterogeneity, encompassing various IoT devices and malware types, has made it a widely recognized benchmark for evaluating FL-IMD approaches [8–10,12,13].

Despite significant progress, existing FL-IMD approaches primarily focus on leveraging federated learning's benefits while often overlooking privacy preservation and aggregation reliability concerns. Some methods enhance privacy through differential privacy (DP); however, this often degrades model accuracy due to noise injection. Others improve aggregation reliability yet fail to fully mitigate inference attacks or address potential failures at the aggregator level.

Our research aims to bridge these gaps by proposing RPFL: a novel, reliable and privacy-preserving framework for FL-IMD that integrates homomorphic encryption (HE), elliptic curve digital signatures (ECDSA), and blockchain technology. RPFL is designed as a reference architecture for detecting malware on IoT devices connected to MEC servers, ensuring robust privacy, reliability, and decentralization in FL-IMD without compromising model accuracy. To validate our research objectives, we investigate the following key research questions:

1. What are the most effective ways to integrate homomorphic encryption into FL-IMD to enhance privacy without compromising model performance?
2. How can blockchain and elliptic curve digital signatures improve the reliability and integrity of the FL-IMD aggregation process while mitigating single points of failure?
3. What are the computational and communication overheads introduced by homomorphic encryption and blockchain, and how do they compare to existing FL-IMD approaches?
4. What decentralized incentive mechanisms can be designed to encourage honest participation in FL-IMD while ensuring fairness and security?

Building on these questions, our study differentiates itself from existing work by introducing a comprehensive solution that incorporates mechanisms for reliability, privacy preservation, mitigation of single points of failure, and fair participant incentivization, all within a unified reference architecture. Our study proposes a novel integration of reliability and privacy preservation for FL-IMD while maintaining model accuracy. Meanwhile, it introduces a new scheme to address single points of failure and presents a novel evaluation mechanism for client contributions without compromising privacy.

Research Contributions:To address these challenges, we propose RPFL: a novel, reliable, and privacy-preserving framework for FL-IMD. RPFL integrates homomorphic encryption (HE), elliptic curve digital signatures (ECDSA), and blockchain technology to enhance security, privacy, reliability, and decentralization in FL-IMD.

Specifically, this study makes the following key contributions:

- Reliable Aggregation and Privacy Preservation: We leverage ECDSA to ensure that only verified clients participate in aggregation. Additionally, homomorphic encryption (HE) protects the privacy of local model weights while maintaining model accuracy.
- Blockchain-Based Decentralized Mechanisms: We develop two smart contract-based decentralized schemes to address key challenges in FL-IMD:
  - A performance-based client participation evaluation mechanism to ensure fair and incentivized collaboration.
  - A decentralized tracking and reporting system to detect and mitigate aggregator failures in real time.

- Performance Evaluation: We conduct extensive experiments to evaluate the effectiveness of the RPFL framework. The results demonstrate that our approach:
  - Achieves comparable model accuracy to state-of-the-art FL-IMD methods.
  - Enhances the aggregation process by protecting privacy and improving reliability. While this introduces computational overhead, it remains manageable.
  - Reduces communication costs and latency compared to Blockchain-based Federated Learning using the InterPlanetary File System (BCFL-IPFS).
- Cost and Scalability Analysis: We analyze cost considerations related to communication and discuss constraints on scalability and model evaluation in the context of deployment challenges, while also exploring potential solutions.

The remainder of this paper is structured as follows. Section 2 provides a brief background on the approaches leveraging FL with the N-BaIoT dataset for designing and evaluating IoT malware detectors. Section 3 outlines the key technical primitives used in developing this framework. Section 4 provides a detailed explanation of the proposed framework. Section 5 presents and discusses the experimental results and performance evaluation. Section 6 analyzes key considerations and constraints associated with the deployment challenges of the proposed approach. Finally, Section 7 concludes the paper.

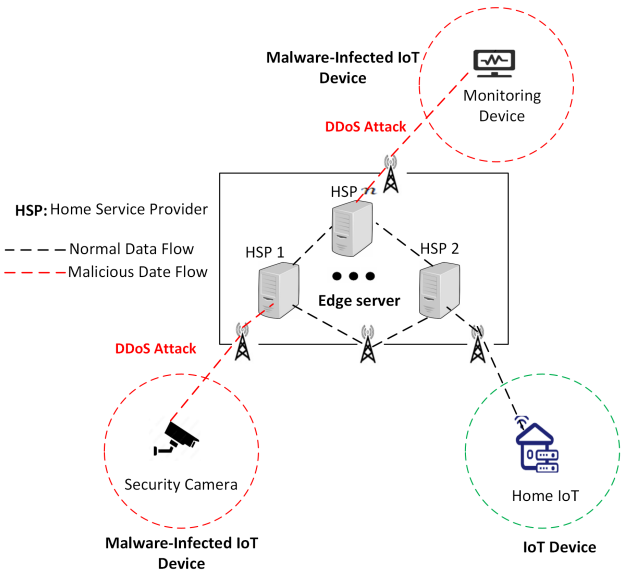

Figure 1. Security challenges in MEC network: malware-infected IoT devices connected to MEC servers.

2. Background and Related Work

It is challenging to evaluate FL-based IoT malware detection models using decentralized public IoT security network datasets, as most existing datasets are generated in centralized environments [8]. These centralized datasets require a more manual partitioning process before they can be effectively utilized in FL scenarios. However, the N-BaIoT dataset enables realistic evaluations of FL-based IMD due to its unique structure, which organizes network traffic data from each real IoT device into separate files.

2.1. FL-IMD Approaches

There is a range of approaches leveraging FL with the N-BaIoT dataset for designing and evaluating IoT malware detectors, as observed in our literature survey. These approaches are suggested for decentralized IoT environments, focusing on ensuring the privacy and security of data used for training while developing effective malware detectors.

The authors in [9] proposed federated deep learning for detecting botnet attacks in IoT networks. They discussed the limitations of traditional centralized models in handling zero-day attacks while preserving data privacy. Their proposed framework, validated using the N-BaIoT dataset, showed improved classification accuracy and data privacy. Federated Averaging (FedAvg) was used to aggregate local model updates without sharing sensitive data. Rey et al. [8] proposed an FL-based framework utilizing supervised and unsupervised learning to detect malware-infected IoT devices connected to networks. They compared their FL-based framework against centralized and native distributed approaches. The results demonstrated better performance for the proposed framework while maintaining data privacy. Regan et al. [12] proposed a federated deep autoencoder model to detect abnormal behaviors caused by malware-infected IoT devices. Secure gateways were proposed to train the model locally, while the server aggregated these local updates. The study reported up to 98% accuracy in anomaly detection using the N-BaIoT dataset. Wardana et al. [10] proposed a Federated Deep Learning (FL-DNN)-based collaborative hierarchical framework for detecting IoT botnets in heterogeneous IoT ecosystems. The framework integrates edge-fog-cloud computing and leverages FL to ensure data privacy. The authors used the N-BaIoT dataset to evaluate the framework, demonstrating high detection accuracy, precision, and recall. The study focuses on achieving scalability and fostering collaboration across different IoT device layers in FL environments.

2.2. Addressing Advanced Security Threats in FL-IMD

While the aforementioned studies have made significant progress in enhancing privacy and detection accuracy by using FL for IMD, they often overlook advanced security challenges specific to FL frameworks. Two critical challenges remain prevalent:

Privacy of Shared Model Parameters: FL systems require the sharing of model gradients and weights between participants and the central server, which can be vulnerable to inference attacks. Studies like [14] have used Differential Privacy (DP) to protect these updates, but the noise added by DP often impacts model accuracy.

Integrity of Model Updates: Ensuring the trustworthiness of model updates is essential for maintaining the robustness of the FL-IMD framework. Malicious clients may attempt to compromise the system by launching poisoning attacks, in which they send manipulated or corrupted updates to degrade the performance of the global model. The method proposed by Thein et al. [13] uses a server-side technique: cosine similarity between local models and a pre-computed global model to detect poisoned clients. While these approaches improve detection, they fail to address a critical concern: protecting the privacy of shared model weights at the server. This leaves the system vulnerable to potential privacy breaches and exploitation.

2.3. Blockchain Integration for Enhanced Security

Recently, Goh et al. [15] proposed a Blockchain-based Federated Learning (BCFL) architecture that integrates blockchain technology with FL to improve security, trust, and data privacy. While this architecture eliminates the need for a central server and introduces using the InterPlanetary File System (IPFS) for decentralized storage of updated models, it presents several challenges when integrated with FL-IMD, such as potential privacy breaches due to aggregators' ability to download all model updates, as well as increased communication costs and latency.

Table 1 provides a summarized comparison of existing literature on FL-IMD approaches. All the listed studies utilize the N-BaIoT dataset for evaluating their methods.

Table 1. A comparative summary of existing FL-IMD approaches.

Study	Methodology	Strengths and Limitations
[9]	FL with deep learning using FedAvg for botnet attack detection	Strengths: High classification accuracy, privacy-preserving framework. Limitations: Lacks security mechanisms against adversarial attacks; no protection for model integrity.
[8]	FL with supervised and unsupervised learning for IoT malware detection	Strengths: Better performance than centralized models, improved privacy. Limitations: Does not address advanced security threats related to the privacy of shared model weights at the server
[12]	FL-based deep autoencoder for anomaly detection	Strengths: High anomaly detection accuracy (98%), local training on edge devices. Limitations: Lacks protection against poisoning attacks; no privacy mechanisms for model updates.
[10]	Hierarchical FL-DNN with edge-fog-cloud computing for botnet detection	Strengths: High detection accuracy, scalable across IoT layers. Limitations: High communication overhead; lacks robustness against adversarial threats.
[13]	Personalized FL-based intrusion detection system with a server-side poisoned client detector using cosine similarity	Strengths: Detects poisoned clients. Limitations: Fails to protect the privacy of shared model weights at the server, leaving vulnerabilities for potential inference attacks.

2.4. Enhancing FL-IMD with Privacy and Security Mechanisms

Our proposed RPFL framework enhances existing FL-IMD approaches by integrating an elliptic curve digital signature algorithm, fully homomorphic encryption and blockchain technology to improve both privacy and reliability of FL-IMD. This enhanced FL-IMD approach ensures the privacy of the shared weights of the model while ensuring the reliability of the model aggregation process, mitigating the risk of malicious participation. Furthermore, we identify challenges associated with fully homomorphic encryption (FHE) integration, particularly those stemming from reliance on an external aggregator. To address these challenges, we propose two smart contract-based approaches: one for tracking and evaluating client participation and another for monitoring the aggregator's status, thereby mitigating the risks associated with its potential failure.

3. Preliminaries

In this section, we provide a concise overview of the key technical primitives used in developing this framework.

3.1. Homomorphic Encryption

Homomorphic encryption (HE) distinguishes itself from traditional encryption methods by enabling computations on encrypted data without requiring decryption. This capability ensures that sensitive information remains secure even if an untrusted entity gains access to the encrypted data during processing, as the underlying data cannot be deciphered or disclosed.

Formally, a homomorphic encryption scheme consists of an encryption algorithm E that is mathematically homomorphic with respect to an operation $\star$. Specifically, the scheme satisfies the equation

$$E(m_1) \star E(m_2) = E(m_1 \star m_2) \quad (1)$$

for all messages m_1, m_2 within the message space $\mathcal{M}$. This property enables direct computations on ciphertexts, preserving the privacy of the data throughout the processing pipeline [16]. HE schemes are generally categorized based on the types and extent of operations they support [17]:

- Partially Homomorphic Encryption (PHE): Supports a single type of operation.
- Somewhat Homomorphic Encryption (SWHE): Allows both addition and multiplication operations but only up to a limited extent, beyond which decryption becomes impractical.
- Fully Homomorphic Encryption (FHE): Permits unrestricted addition and multiplication operations on ciphertexts without accuracy loss. We focus on this FHE in this paper.

In this study, we employ the Cheon–Kim–Kim–Song (CKKS) homomorphic encryption scheme [18] from among various fully homomorphic encryption schemes to encrypt the model's layer weights. The CKKS scheme is particularly suitable for applications in machine learning and data analysis due to its ability to perform approximate computations on encrypted real or complex numbers [19]. Additionally, the integration of CKKS within the TenSEAL library enables seamless incorporation into the Flower framework, which we use to conduct our experiments [20–23].

Our approach to encrypting the model layer weights using the CKKS scheme involves the following key steps:

1. Encoding: This initial step includes converting $Z \in \mathbb{C}^{\frac{N}{2}}$ as a vector or tensor of real numbers into $m(X)$ as a plaintext polynomial within the ring

$$R = \mathbb{Z}[X]/(X^N + 1) \tag{2}$$

where R represents the set of cyclotomic polynomial rings [24], and N is chosen as a power of two, establishing the ring dimension. The polynomial approach offers a balance between security and computational efficiency compared to direct vector operations. This encoding step relies on three primary parameters:

   - Scaling Factor (Global Scale): Determines the precision of the encoding and is crucial for balancing precision and noise in the encrypted data.
   - Polynomial Modulus Degree (Poly Modulus Degree): Influences the number of coefficients in the plaintext polynomials, affecting both computational performance and the encryption's security level.
   - Coefficient Modulus Sizes: Represented as a list of binary values, these sizes determine the size of ciphertext elements and the overall security level. The length of this list indicates the number of supported encrypted multiplications.

2. Encryption:
   After encoding, the data are encrypted using public keys. This process involves adding noise to the data to enhance security and performing an intermediate rescaling step to manage the magnitude of plaintexts, thereby preventing noise accumulation.
3. Computation:
   Once encrypted, various operations such as addition and multiplication are performed on the ciphertexts. These operations enable the manipulation and processing of the encrypted information without compromising its security.
4. Decryption:
   Finally, the encrypted data are decrypted using private keys to retrieve the original information. This decryption process involves noise suppression techniques to ensure the recovered data are accurate and reliable.

3.2. Blockchain

An immutable and distributed ledger of blocks is maintained across untrusted participants in a peer-to-peer (P2P) network, eliminating the need for a centralized authority as a trusted third party [25]. Blockchain technology offers several significant benefits and features. It provides visibility and transparency, with all transactions recorded in a digital ledger that is append-only and auditable by the public. This ensures that the status of the ledger, transaction records, and function calls are stored in a secure, tamper-resistant, and decentralized manner accessible to all network participants.

Moreover, the blockchain ensures immutability through the structure of its digital blocks, each containing time stamps and cryptographic hashes. Additionally, each block includes the hash of the preceding block, forming a secure chain. Any alteration to a block would produce a different hash, thereby invalidating the block and preserving the integrity of the entire chain. The technology also facilitates traceability, as every user cryptographically signs each transaction and function call. This cryptographic signing process ensures that the records are verifiable and that no user can dispute or reverse transactions, reinforcing non-repudiation. Blockchain technology offers diverse applications by enhancing security and efficiency across multiple sectors. It enables secure financial transactions, ensures supply chain transparency, strengthens election integrity, manages digital identities, and protects healthcare records [26,27].

Integrating blockchain with artificial intelligence (AI) enhances security and transparency in various applications. Moreover, this integration can optimize blockchain processes, increase operational efficiency, and drive innovation across multiple industries [28].

3.3. Elliptic Curve Digital Signature Algorithm (ECDSA)

Elliptic Curve Cryptography (ECC) is a public-key cryptographic technique that utilizes the mathematics of elliptic curves to provide high levels of security with smaller key sizes compared to traditional cryptographic methods, such as RSA. These smaller key sizes enable faster computations in distributed systems while also reducing storage requirements and minimizing energy consumption. A specific application of ECC utilized in this study is ECDSA, which is used for generating and verifying digital signatures. ECDSA fulfills two primary functions: authenticating the signer to verify their identity and ensuring data integrity by detecting unauthorized modifications. The process involves two main operations. First, in the signing phase, a private key is employed to generate a unique signature for a given message, based on a combination of elliptic curve mathematics and a hash function, commonly SHA-256. Second, in the verification phase, the corresponding public key is used to validate the signature, confirming both that the message was signed by the private key's holder and that it has not been altered [29,30]. This robust mechanism enhances communication security and ensures data integrity, making it particularly valuable for federated learning processes, such as the proposed FL-IMD, as discussed later in this study.

4. Proposed Solution

In this section, we provide a comprehensive overview of the proposed RPFL framework, which integrates the Elliptic Curve Digital Signature Algorithm (ECDSA), homomorphic encryption (HE), and blockchain technology for IoT malware detection. We thoroughly explain each of its phases and stages, followed by a detailed discussion of the roles and operations of smart contracts within our framework. Finally, we describe our proposed schemes to highlight the innovative aspects of the RPFL architecture.

4.1. Workflow Overview of The RPFL for IMD

Figure 2 presents the RPFL system workflow, consisting of five stages divided into two key phases. The preliminary phase starts with Stage 1: Coordination, where the RPFL environment is initially set up. This is followed by Stage 2: Aggregator Selection, identifying the entity responsible for model aggregation.

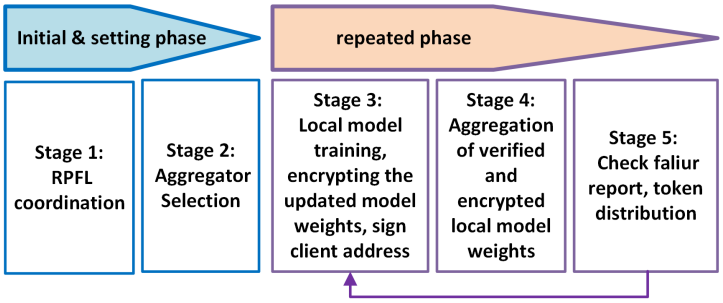

Figure 2. RPFL for IMD workflow, with two phases and five stages.

After these steps, the system enters the iterative phase, which occurs over multiple rounds. Stage 3 includes key operations such as aggregator failure tracking, model weight decryption, local model training, evaluation, model weight encryption, and signing the client address. In Stage 4, the aggregator verifies clients before model aggregation and evaluates client commitment. Finally, Stage 5 handles aggregator failure checking and token distribution based on client contributions.

4.2. RPFL System Components

Figure 3 provides a detailed view of the RPFL system, illustrating its components and their roles. Horizontally, the system comprises two key component types: entities and blockchain. Vertically, the first key component type is entities, which are classified into two categories: internal and external. Internal entities consist of clients, one of whom is designated as the coordinator. These internal entities collaborate to develop a malware detection system for IoT devices connected to their servers, utilizing the RPFL framework. Meanwhile, the external entity, known as the aggregator, is selected by the coordinator to facilitate the aggregation process.

The second component type is blockchain, which serves as the foundation for smart contracts. The RPFL contract addresses challenges associated with integrating homomorphic encryption, such as encouraging client participation and mitigating aggregator failure risks. Another smart contract, the Token contract, manages transactions involving network standard interfaces, such as ERC-20 tokens.

Now, we provide a stage-by-stage breakdown of the RPFL workflow.

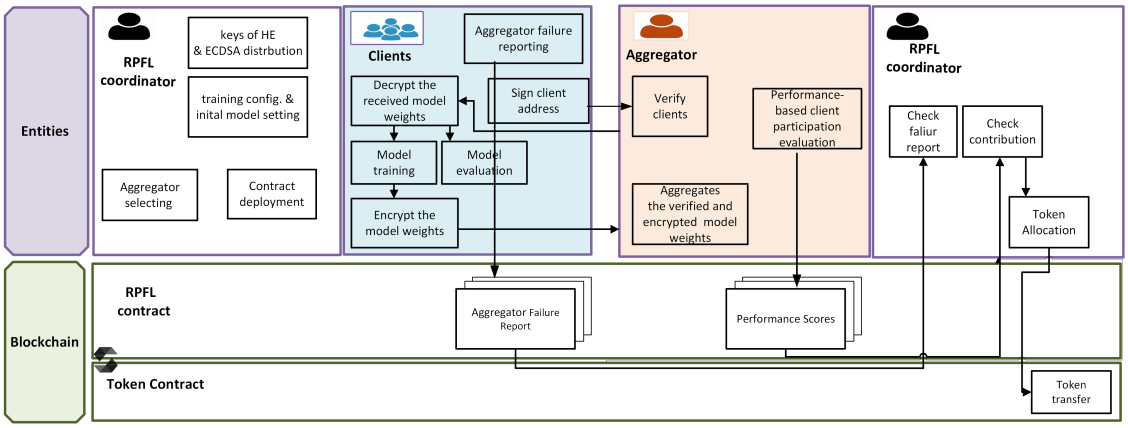

Figure 3. RPFL system architecture for detecting malware in IoT devices connected to IoT networks and MEC.

4.3. RPFL Workflow Stages

Stage 1: RPFL Coordinator Setup

- Clients select a coordinator, responsible for:
  - Generating and securely distributing Homomorphic Encryption (HE) keys.
  - Providing ECDSA private keys to clients for signature verification.
  - Distributing initial global model parameters and training settings.
  - Deploying the RPFL smart contract and Token contract on the blockchain.

Stage 2: Aggregator Selection

- The coordinator selects an external aggregator to ensure model privacy.
- The aggregator receives the public key for signature verification.
- This prevents unauthorized participants from contributing malicious updates.

Stage 3: Local Training and Security Measures

- Each client:
  - Receives and stores HE encryption keys securely.
  - Signs their Ethereum wallet address using their ECDSA private key.
  - Locally trains the malware detection model.
  - Encrypts the updated model weights before sharing them.
- If a client does not receive the updated global model from the aggregator within the specified time, it automatically reports the aggregator's failure indicator via a smart contract.

Stage 4: Aggregation and Verification

- The aggregator follows these steps:
  - Timeout-based synchronization: Clients must submit updates within a specified time frame. If a client fails to send its model within the timeout period, it is excluded from that aggregation round.
  - Client authentication: The Ethereum wallet address of each client is verified via ECDSA.
  - Privacy-preserving aggregation: The encrypted model weights are aggregated using federated averaging.
  - Client commitment evaluation: The aggregator can record client scores via the smart contract based on their performance (e.g., whether a client provides an update in this round, how quickly the client completes training tasks), and at the

end, the system retrieves these results and, based on them, can incentivize clients to participate actively and efficiently.

Stage 5: Failure Handling and Token Incentives

- Check Failure Report
  - The coordinator examines aggregator failures using smart contract feedback.
  - Based on the severity, corrective actions are taken.
- Token Allocation
  - Clients are rewarded proportionally based on their performance.
  - The coordinator manages token distribution using the pre-deployed Token contract.

4.4. Smart Contract

The smart contract in our work is designed to oversee the federated learning process for IoT malware detection. For instance, the aggregator utilizes the smart contract to record the evaluation of each client's commitment to the FL process. Additionally, the smart contract allows clients to report the status of the aggregator, mitigating potential risks of failure. By leveraging blockchain technology, the smart contract ensures that the federated learning process remains transparent, secure, and efficiently managed. The functions required for the contract, as designed for our proposed architecture, are outlined in Table 2.

Table 2. Function categorization and description within the smart contract tailored for the proposed RPFL architecture.

Function	Main Actions	Description
Aggregator Selection	setAggregator	Assigns the AGGREGATOR_ROLE to an address and returns confirmation of the role assignment.
	revokeAggregator	Revokes the AGGREGATOR_ROLE from an address and returns confirmation that the role has been removed.
Round Management	currentRound	Returns the current training round number.
	secondsRemaining	Returns the number of seconds remaining in the current training round.
Evaluation and Scoring	setClientScore	Save a performance-based score for a client.
	getClientScore	Return a performance-based score for a client.
Aggregator Failure Tracking	reportAggregatorStatus	Record a binary value indicating the aggregator's status (success or failure).
	getAggregatorStatus	Retrieve the binary value representing the aggregator's status.
Token Distribution	countTokens	Returns the total number of tokens distributed in a given round.
	countTotalTokens	Returns total tokens distributed in all rounds.
	setTokens	Records total tokens utilized in a given round.
	distributeTokens	Distributes tokens to a specified wallet address.

4.4.1. Access Control and Security

To enhance security, we integrate OpenZeppelin's Access Control [31] into the smart contract deployed on the Ethereum blockchain. This integration provides a standardized, flexible, and well-audited framework for managing roles and permissions. By defining specific roles, COORDINATOR_ROLE, AGGREGATOR_ROLE, and CLIENT_ROLE, and restricting access to critical functions using the onlyRole modifier, we ensure that only

authorized entities can perform sensitive operations, thereby safeguarding our proposed smart contract-based mechanism.

4.4.2. Core Functions of the Smart Contract

Each function of our smart contract is designed to ensure the smooth execution and security of federated learning. These functions are categorized as follows:

1. Role and Access Management: Defines and manages roles (Coordinator, Aggregator, Client) to ensure that only authorized entities can perform specific actions within the contract.
2. Round Management: Initiates and controls training rounds, including setting durations, tracking progress, and managing transitions between rounds based on client contributions.
3. Evaluation and Scoring: Assigns scores to clients based on their level of commitment and contribution, serving as the basis for token-based rewards.
4. Aggregator Failure Tracking: Allows clients to record a binary value in the smart contract, indicating the aggregator's status. The coordinator can monitor these records for reliability assessment.
5. Token Distribution: Manages the distribution of ERC-20 tokens as rewards for client contributions. The smart contract enables the coordinator to set or update the ERC-20 token used, providing flexibility in the reward mechanism.

4.5. Detailed Description of Proposed Schemes

This section explains the innovative schemes integrated into the RPFL architecture for IoT malware detection, focusing on privacy preservation, reliability, and decentralization.

4.5.1. Reliable and Privacy-Preserving Aggregation Process for FL-IMD

To ensure a reliable aggregation process in FL-IMD, we employ the Elliptic Curve Digital Signature Algorithm (ECDSA) for secure authentication. This method guarantees that only verified participants contribute to the aggregation of local model updates.

Reliable Aggregation using ECDSA: Each client in the RPFL framework signs its Ethereum address using its ECDSA private key, providing cryptographic proof of identity. The client then compiles encrypted local model weights, signed Ethereum address and original Ethereum address. These are then transmitted to the aggregator, which verifies the signature using the client's ECDSA public key. Only clients with valid ECDSA credentials are authorized to participate in federated learning.

This process ensures:

- Authentication and integrity: Preventing unauthorized clients from contributing to the global model.

Privacy-Preserving Aggregation via Homomorphic Encryption: To protect client data, the homomorphic encryption technique is applied. Here, all local model weights remain encrypted before being sent to the aggregator, ensuring:

- Data confidentiality: The aggregator cannot access the raw model updates.
- Privacy preservation: Clients' training data and model parameters remain secure throughout the learning process.

Pseudo-code in Algorithms 1 and 2 formulate the mechanism of the reliable and privacy-preserving aggregation process at the client and aggregator, respectively.

Algorithm 1 Reliable and privacy-preserving aggregation process (client side)

Input: Initial model weights W_{initial} or encrypted global model weights $W_{\text{global_encrypted}}$, Homomorphic encryption keys $(HE_{\text{public_key}}, HE_{\text{private_key}})$, Elliptic Curve Digital Signature Algorithm (ECDSA) private key: $ECDSA_{\text{private_key}}$, Local training data D_{local}
Output: Encrypted model weights $W_{\text{local_encrypted}}$, Ethereum_address and signed Ethereum address $ECDSA_{\text{signature}}$

Start
Initialization:
Securely receive $HE_{\text{public_key}}$, $HE_{\text{private_key}}$, and $ECDSA_{\text{private_key}}$
Wait for the smart contract to announce the selected aggregator.
Train the model locally: $W_{\text{local}} \leftarrow \text{TrainModel}(W_{\text{initial}}, D_{\text{local}})$
Encrypt the updated weights using $HE_{\text{public_key}}$:
    $W_{\text{local_encrypted}} \leftarrow \text{Encrypt}(W_{\text{local}}, HE_{\text{public_key}})$
Generate ECDSA signature for the client's Ethereum address:
    $ECDSA_{\text{signature}} \leftarrow \text{Sign}(\text{Ethereum_address}, ECDSA_{\text{private_key}})$
Transmit $W_{\text{local_encrypted}}$, Ethereum_address and $ECDSA_{\text{signature}}$ to the aggregator.
Await the smart contract's signal and updated global weights from the aggregator.
Decrypt the global weights using $HE_{\text{private_key}}$:
    $W_{\text{global}} \leftarrow \text{Decrypt}(W_{\text{global_encrypted}}, HE_{\text{private_key}})$
Apply the same operations from Operation 5 (but with W_{global}) to Operation 8
End

Algorithm 2 Reliable and privacy-preserving aggregation process (aggregator side)

Input: *ClientUpdate*: Encrypted local model weights $W_{\text{local_encrypted}}$, Ethereum_address, $ECDSA_{\text{signature}}$, $ECDSA_{\text{public_key}}$
Output: Encrypted global model weights $W_{\text{global_encrypted}}$

Start
Initialize an empty list: $VerifiedUpdates \leftarrow []$
for all *ClientUpdate* received from a client **do**
    Extract the Ethereum_address and $ECDSA_{\text{signature}}$ from *ClientUpdate*
    `/* Verify the signature of the client's address */`
    $IsVerified \leftarrow \text{VerifySignature}(\text{Ethereum_address}, ECDSA_{\text{signature}}, ECDSA_{\text{public_key}})$
    if *IsVerified* **then**
        Append $W_{\text{local_encrypted}}$ to *VerifiedUpdates*
        Log the successful verification
    else
        Reject: Deny the update and ignore the client's contribution
        Log the failure for auditing
$W_{\text{global_encrypted}} \leftarrow \text{Aggregate}(VerifiedUpdates)$ `/* Using FedAvg method */`
Broadcast the updated $W_{\text{global_encrypted}}$ to all participating clients
End

4.5.2. Decentralized Schemes for Addressing HE Integration Challenges

Integrating homomorphic encryption into federated learning introduces challenges, particularly regarding client participation evaluation and aggregator reliability. To address these issues, we propose two decentralized schemes based on blockchain smart contracts.

Scheme 1: Performance-Based Client Participation Evaluation

Client participation is crucial for accurate federated learning models. However, in traditional FL systems without HE, client models are evaluated using pre-prepared datasets and unencrypted model weights. While effective, this approach introduces privacy risks, potentially discouraging clients from participating.

To resolve this, we propose a decentralized, privacy-preserving evaluation scheme using smart contracts. This mechanism:

- Records and evaluates contributions securely without exposing model weights.
- Incentivizes active clients while ensuring privacy compliance.

Figure 4 illustrates the proposed evaluation scheme, and its detailed operation is outlined in Algorithm 3.

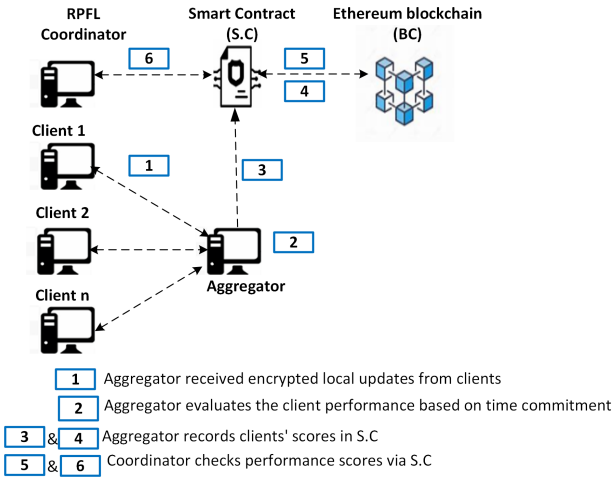

Figure 4. Performance-based client participation evaluation.

Algorithm 3 Performance-based client participation evaluation (aggregator side)

Input:
ClientUpdates: A list of encrypted model updates received from clients.
Timeout: Maximum wait time for client updates.
Output:
PerformanceScores: Recorded performance-based scores for clients.
Start
Aggregator starts waiting for clients to connect and sends requests for encrypted model updates.
Record the start time as *RequestTime*.
for each *ClientUpdate* received from a client **do**
    Record the time of response as *ResponseTime*.
    Calculate $TimeTaken \leftarrow ResponseTime - RequestTime$.
    if *Client* is verified and trust **then**
        `/* Calculate the performance score for the client */`
        $Score \leftarrow 1/TimeTaken$
        `/* Record the score in the smart contract */`
        $SmartContract.SetClientScore(ClientAddress, Score)$
        Log the action for audit purposes.
    else
        Log the unauthorized client attempt for auditing.
End

Scheme 2: Aggregator Failure Mitigation

A key reliability challenge in federated learning is aggregator failure, which can disrupt the model training process. The RPFL framework integrates a proactive aggregator monitoring mechanism to detect failures early and mitigate risks.

This scheme operates as follows:

1. Continuous monitoring: The aggregator's response time and activity levels are tracked.
2. Failure detection triggers: If the aggregator exhibits slow response times or becomes unresponsive, an alert is raised.

3. Proactive intervention: The RPFL coordinator takes corrective actions before a complete system failure occurs.
4. Performance evaluation: The aggregator's reliability is assessed over time, helping to optimize network efficiency.

Figure 5 illustrates the proposed monitoring scheme, and Algorithm 4 provides a step-by-step breakdown.

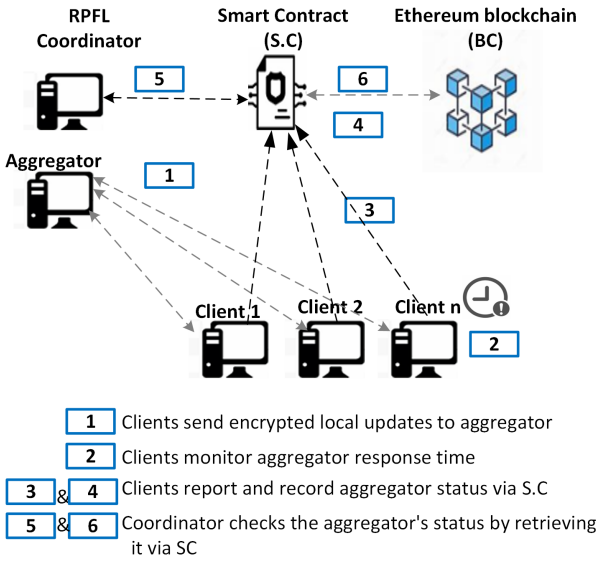

Figure 5. Decentralized monitoring for aggregator's status to mitigate its failure risks.

Algorithm 4 Aggregator failure mitigation (client-side perspective)

Input:
TimeoutThreshold: Predefined timeout duration.
UpdatedGlobalModel: Global model weights from the aggregator.
Output:
FailureReport: Indicator of aggregator status.
Start
`/* Initialize Parameters: */`
Start a timer after sending local model weights to the aggregator.
`/* Monitor Aggregator Response: */`
if *UpdatedGlobalModel* is received within *TimeoutThreshold* **then**
`/* Report a value of 1 for success: */`
$SmartContract.reportAggregatorStatus(ClientAddress, 1)$
Log the action for audit purposes.
else
`/* Report a value of 0 for failure: */`
$SmartContract.reportAggregatorStatus(ClientAddress, 0)$
Log the action for audit purposes.
End

5. Experiments

5.1. Dataset Setting and Model Design for FL-IMD

Overview of the N-BaIoT Dataset: The N-BaIoT dataset is a publicly available dataset containing network traffic logs from IoT devices under benign conditions and when infected with malware. It includes traffic from nine different IoT devices operating in a smart home environment. The dataset captures attacks from two major malware families:

- Mirai-based attacks: Includes attacks such as UDP flood, ACK flood, SYN flood, and scan attack.

- BASHLITE-based attacks: Includes TCP flooding, UDP flooding, and command injection attacks.

Each IoT device generates labeled traffic data, categorized as benign or malicious.

Dataset Partitioning and Data Preprocessing: Following the approach in [8], the dataset is divided into three primary subsets:

- 79% for training;
- 1% unused;
- 20% for testing.

Federated Data Distribution Across Clients:

The federated learning setup of our RPFL framework comprises nine clients, with each client assigned the data from a distinct IoT device within the N-BaIoT dataset. Each client trains exclusively on its own device's network traffic, ensuring a realistic non-IID (non-independent and identically distributed) FL scenario. This setup mimics real-world IoT environments, where devices operate independently and generate unique traffic patterns.

To ensure fairness in training, each client is allocated 100,000 samples (50,000 benign, 50,000 malicious), maintaining class balance across devices. However, since the original dataset contains varying class distributions across devices, we apply:

- Upsampling (replicating minority samples) when fewer than 50,000 samples exist.
- Downsampling (random selection) when more than 50,000 samples exist.

This partitioning strategy ensures that each FL client receives a representative dataset while preserving the natural data distribution of IoT devices. Furthermore, dataset partitioning occurs before class balancing, ensuring no data leakage between training and testing subsets.

Data Preprocessing: Normalization is performed using Min–Max feature scaling, where each sample value (denoted as x) is transformed into a new value (denoted as x') within a predefined range. Specifically, the formula

$$x' = \frac{x - x^{\min}}{x^{\max} - x^{\min}} \tag{3}$$

is applied, where $x^{\min}$and $x^{\max}$ represent the minimum and maximum values within the range, respectively. This operation is performed element-wise on each of the 115 features in the dataset.

Notably, the normalization parameters ($x^{\min}$ and $x^{\max}$) are calculated using only the training set data available to each client, ensuring that each client computes normalization values specific to its own dataset [8].

Training Model (Model Architecture):

For binary classification, we employ a Multi-Layer Perceptron (MLP) with:

- 115 input neurons (one per feature);
- Two hidden layers (115 and 58 neurons) with ELU activation;
- Sigmoid activation for binary classification.

The selection of MLP over alternative architectures (such as CNNs or RNNs) is based on the following considerations:

- Structured Data Compatibility: Unlike CNNs and RNNs, MLPs are well suited for tabular network traffic data.
- Baseline Consistency: Prior works [8] also utilized MLPs, ensuring a valid performance comparison.

5.2. Simulation Tools and Parameters

Federated Learning Framework: We utilize Flower [22], which provides:

- Seamless integration with PyTorch 2.3.1;
- Support for federated averaging (FedAvg) aggregation;
- Simulated FL with heterogeneous clients.

Hyperparameter Selection: Table 3 summarizes the selected hyperparameters. FedAvg is employed as the aggregation algorithm.

Table 3. Hyperparameter settings for classification tasks.

Parameter	Value
Optimizer	Stochastic Gradient Descent (SGD)
L2-regularization (weight decay)	$0,\ 10^{-5}$
Learning rate (lr)	0.5
Batch size (BS)	64
Training epochs (E)	4
Number of rounds (T)	30
Number of experiment repetitions	5

Each hyperparameter was selected based on fine-tuning experiments to optimize accuracy while balancing training efficiency.

Homomorphic Encryption (HE) Configuration: We integrate TenSEAL for preserving the privacy of the model updates of FL-IMD, using:

- CKKS Scheme: Supports encrypted operations on real numbers;
- Polynomial Modulus Degree: 8192, balancing security and efficiency;
- Coefficient Modulus Bit Sizes: [60, 40, 40, 60] for precision and encryption depth.

Unlike encrypting entire datasets, we encrypt only model weights to reduce computational overhead.

Regarding Differential Privacy (DP), we employ Local Differential Privacy with a privacy budget (ϵ) set to 4.

Blockchain Integration: A simulation environment developed by [15] was utilized and customized to validate our proposed architecture. Moreover, we deploy Ethereum smart contracts using:

- Ganache CLI: Creates a reproducible local Ethereum blockchain for smart contract deployment and testing,
- Truffle Framework: Develops Solidity-based contracts for functions,
- Web3.py: Integrates federated learning with blockchain functionalities.

5.3. Performance Metrics and Methodology, Baselines

To evaluate the effectiveness of our proposed RPFL framework, we use the following metrics:

- True Positive Rate (TPR), or recall, which measures the model's ability to correctly detect malicious traffic:

$$TPR = \frac{TP}{TP + FN} \tag{4}$$

- True Negative Rate (TNR), which evaluates how accurately the model classifies benign traffic:

$$TNR = \frac{TN}{TN + FP} \tag{5}$$

- Accuracy, which reflects overall classification performance across both benign and malicious traffic:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{6}$$

where True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) represent the classification outcomes.

Our experiments compare RPFL against the following baseline methods:

- Classical FL-IMD (without privacy mechanisms).
- FL-IMD with Differential Privacy (DP) to assess privacy impacts.
- State-of-the-art methods: Rey et al. [8] and Wardana et al. [10].

Computational Cost: Quantifies the efficiency of our approach in terms of training time per round and total execution time. This evaluation provides insights into the trade-off between enhanced security and computational efficiency in real-world deployments.

In comparing blockchain integration in RPFL with the BCFL-IPFS framework proposed by Goh et al. [15], as discussed in Section 2, the focus was on the average simulation time of FL processes in both frameworks, emphasizing the efficiency achieved in reducing latency.

5.4. Results and Analysis

Model Performance: Figure 6 demonstrates that our scheme consistently outperforms FL-IMD with DP across all evaluated performance metrics. Specifically, HE maintains model accuracy while excelling in the identification of both positive and negative instances, highlighting its robustness. This superiority arises from HE's ability to secure exchanged model updates without introducing noise into the training data, thereby preserving data integrity and utility. Conversely, DP enhances individual-level privacy by injecting noise into the training data. While this approach strengthens privacy protections, it adversely impacts the final model's performance, particularly in distinguishing negative instances. This limitation often results in a higher rate of false positives, posing challenges in scenarios that demand precise classification. Overall, the findings indicate that HE provides a more favorable balance between security and performance compared to DP, making it particularly well suited for applications such as FL-IMD.

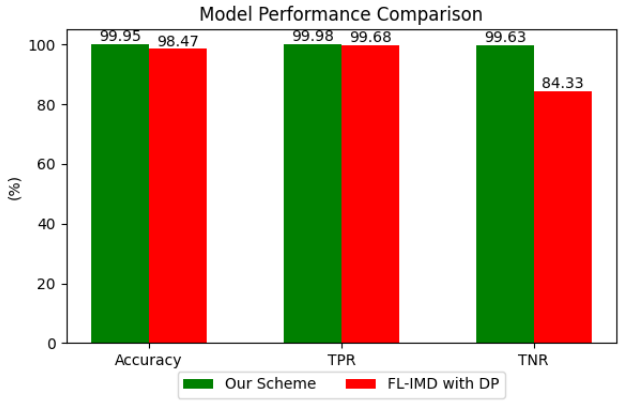

Figure 6. Performance comparison of our scheme and FL-IMD with DP using MLP model.

Figure 7 presents a comparative analysis of our scheme and classical FL-IMD using the MLP model. The results demonstrate that both approaches consistently achieve high performance across all evaluated metrics, with a slight, almost negligible difference. This indicates that integrating HE does not compromise the system's practical utility. These findings underscore the robust privacy protection provided by HE in our scheme, while maintaining system efficiency and accuracy.

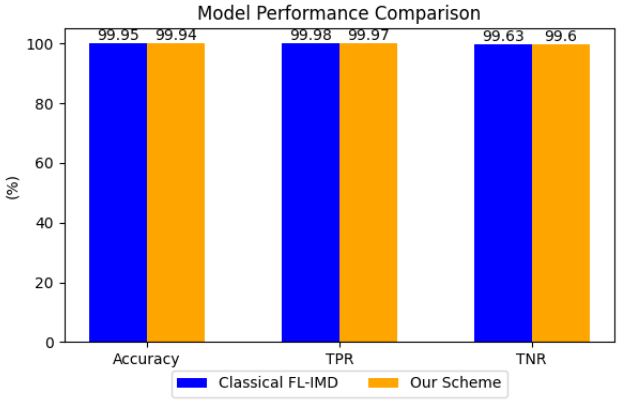

Figure 7. Performance comparison between classical FL-IMD and our scheme using MLP model.

Figure 8 illustrates a comparison of IoT malware detection accuracy between our proposed scheme and baseline methods. The results show a high similarity in accuracy between our approach and that of [8], attributed to the use of the same model and closely aligned parameters. However, our scheme enhances the security of the aggregation process in terms of reliability and privacy, without compromising the accuracy of IoT malware detection. Moreover, the approach presented in [10] shows a slight decrease in accuracy compared to our approach and [8].

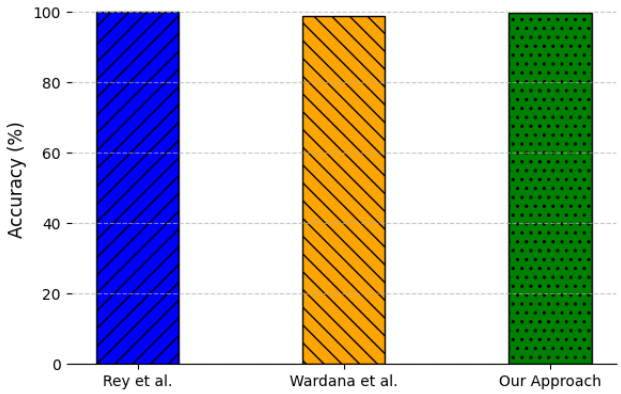

Figure 8. Comparison of IoT malware detection accuracy across proposed and baseline methods [8,10].

Computational Cost Analysis: To assess the efficiency of our approach, we compare the total training time of classical FL-IMD and our approach over 30 rounds. The results are summarized in Table 4.

Table 4. Comparison of training time per round and total simulation time for 30 rounds.

Method	Total Training Time (30 Rounds) (Minutes)	Training Time per Round (Seconds)
Classical FL-IMD	16.21 min	32.42 s
Our approach with HE	18.66 min	37.32 s

The integration of homomorphic encryption (HE) introduces a 15.1% increase in training time per round due to encryption and decryption overhead. However, since HE is applied only once per round, its impact remains bounded and does not scale exponentially with additional local epochs. Figure 9 illustrates the linear increase in training time as the number of rounds progresses. This confirms that, despite the added encryption steps, the overall computational overhead remains manageable for practical deployment.

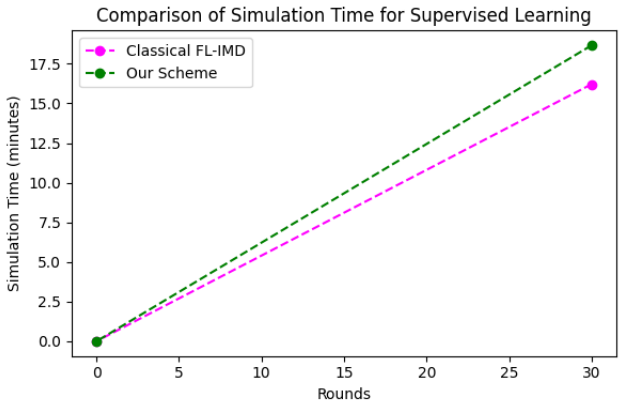

Figure 9. Impact of HE integration on total training time over 30 rounds.

Validation of Model Integrity: Figure 10 demonstrates the validated results of the proposed framework, where the aggregator ensures the reliability of the aggregation process by including updates only from authorized clients verified using the cryptographic method of elliptic curve digital signatures. Unauthorized or malicious clients are explicitly denied participation, safeguarding the accuracy and integrity of the global model.

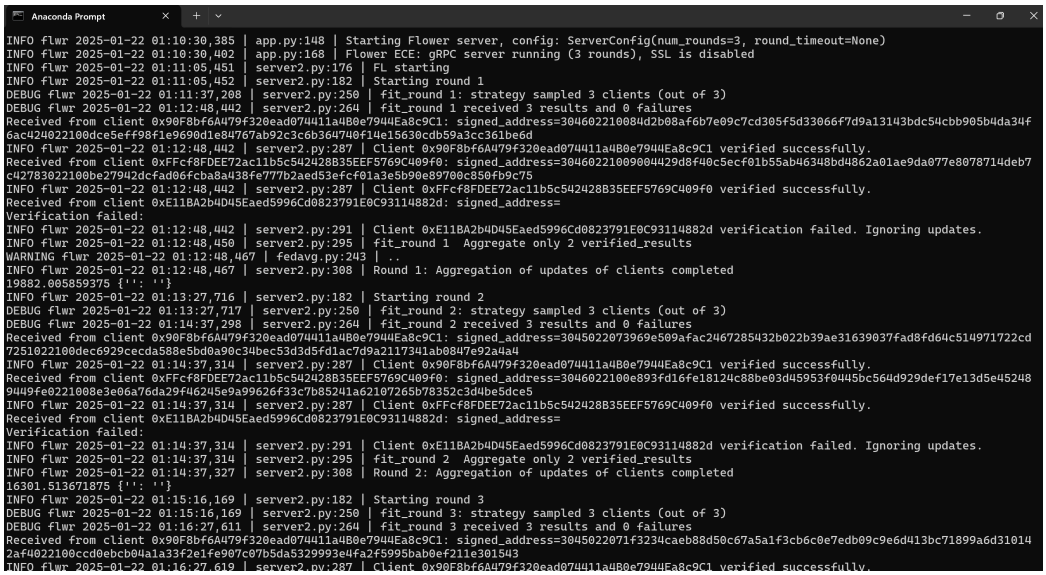

Figure 10. Validated results of the proposed framework, demonstrating its ability to protect the reliability and integrity of the aggregation process.

Impact of IPFS on Simulation Time: Figure 11 compares the simulation time of our proposed architecture with the architecture presented in [15], highlighting the notable impact of IPFS on latency. The BCFL-IPFS framework [15] exhibits a moderate increase in overall simulation time, primarily due to:

- Additional blockchain-IPFS interactions: Each model update must be uploaded to and retrieved from IPFS, introducing significant processing delays.
- Network latency overhead: The transmission of models through IPFS requires multiple communication cycles, further compounding delay.

In contrast, our proposed RPFL architecture eliminates IPFS dependency, resulting in faster execution. The reduction in latency makes our approach better suited for IoT malware detection, where real-time processing is crucial. The latency difference is expected to increase further when homomorphic encryption is incorporated, as previously noted.

However, by avoiding IPFS-related delays, our framework maintains higher efficiency without compromising security.

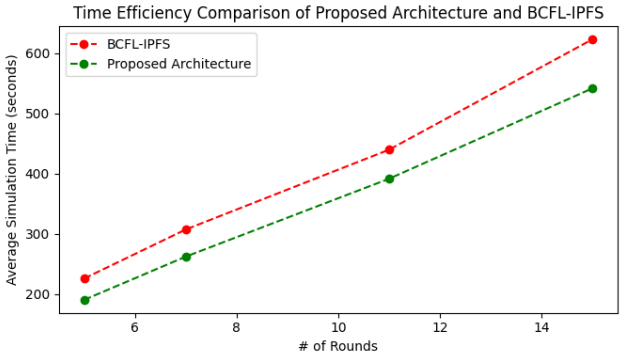

Figure 11. Comparison of simulation time between our proposed architecture and BCFL-IPFS, highlighting the impact of IPFS-induced latency.

Furthermore, the next section analyzes communication costs from the client's perspective, as well as scalability and evaluation constraints.

6. Discussion

In this section, we discuss key considerations regarding performance and the design of components, focusing on their interactions for deployment in real-world IoT networks and environments. While the experimental results demonstrated promising outcomes, additional factors such as communication costs, scalability, and model evaluation constraints require further analysis to assess the practical feasibility of our approach.

6.1. Communication Cost Analysis

Here, we compare our proposed architecture against BCFL-IPFS using the FedAvg aggregation method, focusing on communication costs. The two configurations differ in how they handle model sharing and interact with blockchain or IPFS, which directly impacts the observed costs.

In terms of communication costs, our proposed architecture involves direct model exchanges between clients and the aggregator, with blockchain interaction limited to logging transactions. The communication cost per client for this configuration is given by

$$\text{Communication Cost of Our architecture} = T \cdot S_m + \text{Blockchain Overhead}, \tag{7}$$

where T is the number of federation rounds, and S_m is the model size. In contrast, BCFL-IPFS requires models to be uploaded to and downloaded from IPFS, with additional blockchain interaction for content identifier registration. The corresponding communication cost is calculated as

$$\text{Communication Cost of BCFL-IPFS} = 2 \cdot T \cdot (S_m + \text{IPFS Overhead}) + \text{Blockchain Overhead}. \tag{8}$$

Due to the additional overhead introduced by IPFS, BCFL-IPFS incurs higher communication costs. For the model learning, the communication cost increases from 2.83 MB to 5.95 MB, as summarized in Table 5. Note that we neglect the communication overhead resulting from the integration of HE and ECDSA in both architectures, as the overhead will be identical for both.

Table 5. Comparison of communication costs per client, accounting for bidirectional data transfer (upload and download).

Metric	The Proposed Architecture	BCFL-IPFSS
Number of model transmissions	$T = 30$	$2 \cdot T = 60$
Communication cost (Assume S_m = 94 kB)	2.83 MB	5.95 MB

6.2. Scalability Discussion

Scalability is a key consideration in assessing the feasibility of federated learning frameworks, especially in multi-access edge computing (MEC) networks, where thousands or even millions of IoT devices may be connected. While our results confirm that the proposed RPFL framework scales efficiently in simulated IoT environments, real-world MEC deployments present additional challenges that cannot be fully evaluated using current datasets.

Need for Large-Scale and Diverse Datasets: To properly assess federated learning scalability in MEC-enabled IoT networks, future research should focus on the development of larger and more diverse datasets that better represent real-world conditions.

6.3. Model Evaluation Constraints

Evaluating the updated global model for malware detection on the aggregator side, particularly with data from a new IoT device not involved in the training phase, is crucial. This approach is expected to highlight significant differences from the training data, thereby providing insights into the model's adaptability to novel and unobserved scenarios. However, the updated global model cannot be evaluated on the aggregator side due to the lack of access to private keys of HE. Consequently, each client must independently assess the global model using a portion of data that was not included in the training process but originates from the same IoT devices owned by the clients.

7. Conclusions and Future Work

In this paper, we proposed RPFL, a reliable and privacy-preserving framework designed to address the security and privacy vulnerabilities of federated learning-based malware detection in IoT devices connected to the edge servers of multi-access edge computing (MEC). The architecture leverages elliptic curve digital signature algorithm (ECDSA) to ensure reliable global model aggregation, while homomorphic encryption safeguards the privacy of submitted local model weights without affecting the accuracy of the malware detector. Additionally, we identified and discussed new challenges arising from integrating homomorphic encryption into the architecture, particularly those related to relying on an external aggregator. To address these challenges, we introduced two smart contract-supported schemes: an incentive mechanism and a mitigation strategy to address potential aggregator failures. The proposed RPFL architecture was validated and demonstrated robust security by ensuring that only local models submitted by trusted clients are incorporated into the global model aggregation, effectively preventing malicious client participation. Furthermore, the integration of homomorphic encryption proved effective in preserving the privacy of exchanged local models, while experimental results show that RPFL maintains model accuracy without compromise. Moreover, results on communication cost and latency from the blockchain integration in RPFL indicate that our RPFL framework outperforms the prominent BCFL-IPFS architecture.

In future work, we aim to explore a fully decentralized implementation of the RPFL framework and investigate its scalability across large-scale IoT networks and extended

communication infrastructures. Additionally, enhancing aggregator selection strategies presents a promising avenue for further improvement. Finally, addressing the secure sharing of keys for HE and ECDS schemes will be critical for strengthening the security and robustness of the RPFL framework.

Author Contributions: M.A.: Conceptualization, Methodology, Software, Simulation, Investigation, Validation, Writing—original draft. M.A.K. and R.M.A.: Supervision, Review and Editing. B.S.A. and F.E.E.: Review and Editing. All authors have read and agreed to the published version of the manuscript.

Funding: No external funding for this study.

Data Availability Statement: The dataset used in this study is publicly available and can be accessed at the following link: N-BaIoT Dataset (UCI Machine Learning Repository) (accessed on 23 February 2025).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Sun, P.; Shen, S.; Wan, Y.; Wu, Z.; Fang, Z.; Gao, X.Z. A survey of iot privacy security: Architecture, technology, challenges, and trends. *IEEE Internet Things J.* **2024**, *11*, 34567–34591. [CrossRef]
2. Sánchez, B.B.; Alcarria, R.; Robles, T. A Probabilistic Trust Model and Control Algorithm to Protect 6G Networks against Malicious Data Injection Attacks in Edge Computing Environments. *CMES Comput. Model. Eng. Sci.* **2024**, *141*, 631–654. [CrossRef]
3. Ma, Y.; Liu, L.; Liu, Z.; Li, F.; Xie, Q.; Chen, K.; Lv, C.; He, Y.; Li, F. A Survey of DDoS Attack and Defense Technologies in Multi-Access Edge Computing. *IEEE Internet Things J.* **2024**, *12*, 1428–1452. [CrossRef]
4. Chen, J.; Yan, H.; Liu, Z.; Zhang, M.; Xiong, H.; Yu, S. When federated learning meets privacy-preserving computation. *ACM Comput. Surv.* **2024**, *56*, 1–36. [CrossRef]
5. Heidari, A.; Jabraeil Jamali, M.A. Internet of Things intrusion detection systems: A comprehensive review and future directions. *Clust. Comput.* **2023**, *26*, 3753–3780. [CrossRef]
6. Alsoufi, M.A.; Siraj, M.M.; Ghaleb, F.A.; Al-Razgan, M.; Al-Asaly, M.S.; Alfakih, T.; Saeed, F. Anomaly-Based Intrusion Detection Model Using Deep Learning for IoT Networks. *Comput. Model. Eng. Sci.* **2024**, *141*, 823–845. [CrossRef]
7. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-baiot—Network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
8. Rey, V.; Sánchez, P.M.S.; Celdrán, A.H.; Bovet, G. Federated learning for malware detection in IoT devices. *Comput. Netw.* **2022**, *204*, 108693. [CrossRef]
9. Popoola, S.I.; Ande, R.; Adebisi, B.; Gui, G.; Hammoudeh, M.; Jogunola, O. Federated deep learning for zero-day botnet attack detection in IoT-edge devices. *IEEE Internet Things J.* **2021**, *9*, 3930–3944. [CrossRef]
10. Wardana, A.A.; Sukarno, P.; Salman, M. Collaborative Botnet Detection in Heterogeneous Devices of Internet of Things using Federated Deep Learning. In Proceedings of the 2024 13th International Conference on Software and Computer Applications, Bali Island, Indonesia, 1–3 February 2024; pp. 287–291.
11. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
12. Regan, C.; Nasajpour, M.; Parizi, R.M.; Pouriyeh, S.; Dehghantanha, A.; Choo, K.K.R. Federated IoT attack detection using decentralized edge data. *Mach. Learn. Appl.* **2022**, *8*, 100263. [CrossRef]
13. Thein, T.T.; Shiraishi, Y.; Morii, M. Personalized federated learning-based intrusion detection system: Poisoning attack and defense. *Future Gener. Comput. Syst.* **2024**, *153*, 182–192. [CrossRef]
14. Sánchez, P.M.S.; Celdrán, A.H.; Xie, N.; Bovet, G.; Pérez, G.M.; Stiller, B. Federatedtrust: A solution for trustworthy federated learning. *Future Gener. Comput. Syst.* **2024**, *152*, 83–98. [CrossRef]
15. Goh, E.; Kim, D.Y.; Lee, K.; Oh, S.; Chae, J.E.; Kim, D.Y. Blockchain-Enabled Federated Learning: A Reference Architecture Design, Implementation, and Verification. *IEEE Access* **2023**, *11*, 145747–145762. [CrossRef]
16. Doan, T.V.T.; Messai, M.L.; Gavin, G.; Darmont, J. A survey on implementations of homomorphic encryption schemes. *J. Supercomput.* **2023**, *79*, 15098–15139. [CrossRef]
17. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.

18. Cheon, J.H.; Kim, A.; Kim, M.; Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In Proceedings of the Advances in Cryptology—ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, 3–7 December 2017; Proceedings, Part I 23; Springer: Berlin/Heidelberg, Germany, 2017; pp. 409–437.
19. Bezuglova, E.; Kucherov, N. An Overview of Modern Fully Homomorphic Encryption Schemes. In Proceedings of the International Conference on Actual Problems of Applied Mathematics and Computer Science, Stavropol, Russia, 3–7 October 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 300–311.
20. Benaissa, A.; Retiat, B.; Cebere, B.; Belfedhal, A.E. Tenseal: A library for encrypted tensor operations using homomorphic encryption. *arXiv* **2021**, arXiv:2104.03152.
21. OpenMined. TenSEAL: A Library for Homomorphic Encryption Operations on Tensors. Available online: https://github.com/OpenMined/TenSEAL/tree/main (accessed on 16 November 2024).
22. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K.H.; Parcollet, T.; de Gusmão, P.P.B.; et al. Flower: A friendly federated learning framework. *arXiv* **2022**, arXiv:2007.14390.
23. Adap. Flower: A Friendly Federated Learning Research Framework. 2024. Available online: https://github.com/adap/flower (accessed on 23 February 2025).
24. Lang, S. *Algebraic Number Theory*, 2nd ed.; Graduate Texts in Mathematics; Springer: Berlin/Heidelberg, Germany, 1994; Volume 110.
25. Deepa, N.; Pham, Q.V.; Nguyen, D.C.; Bhattacharya, S.; Prabadevi, B.; Gadekallu, T.R.; Maddikunta, P.K.R.; Fang, F.; Pathirana, P.N. A survey on blockchain for big data: Approaches, opportunities, and future directions. *Future Gener. Comput. Syst.* **2022**, *131*, 209–226. [CrossRef]
26. Al-Nbhany, W.A.; Zahary, A.T.; Al-Shargabi, A.A. Blockchain-IoT healthcare applications and trends: A review. *IEEE Access* **2024**, *12*, 4178–4212. [CrossRef]
27. Zheng, P.; Jiang, Z.; Wu, J.; Zheng, Z. Blockchain-based decentralized application: A survey. *IEEE Open J. Comput. Soc.* **2023**, *4*, 121–133. [CrossRef]
28. Ressi, D.; Romanello, R.; Piazza, C.; Rossi, S. AI-enhanced blockchain technology: A review of advancements and opportunities. *J. Netw. Comput. Appl.* **2024**, *225*, 103858. [CrossRef]
29. Ullah, S.; Zheng, J.; Din, N.; Hussain, M.T.; Ullah, F.; Yousaf, M. Elliptic Curve Cryptography; Applications, challenges, recent advances, and future trends: A comprehensive survey. *Comput. Sci. Rev.* **2023**, *47*, 100530. [CrossRef]
30. Prakash, V.; Keerthi, K.; Jagadish, S.; Alkhayyat, A.; Soni, M. An Elliptic Curve Digital Signature Algorithm for Securing the Healthcare Data Using Blockchain Based IoT Architecture. In Proceedings of the 2024 International Conference on Data Science and Network Security (ICDSNS), Tiptur, India, 26–27 July 2024; pp. 1–5.
31. OpenZeppelin. Access Control Documentation. 2024. Available online: https://docs.openzeppelin.com/contracts/3.x/access-control (accessed on 8 December 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.