This article has been accepted for publication in The ISC International Journal of Information Security*, but has not been fully edited. Content may change prior to final publication.*

The ISC Int'l Journal of Information Security

ISeCure

Manuscript template of ISeCure Journal (pp. 1–12)

http://www.isecure-journal.org

Collaboration-based Privacy-Preserving Firewall for Home Networks

Yasamin Rahimi Sadegh [1] Sajad Shirali-Shahreza [1,*]

[1] *Department of Computer Engineering, Amirkabir University of Technology, Tehran, Iran*

ARTICLE INFO.

Keywords:
Federated Learning, Firewall, Smart Home, IoT Security.

Abstract

The rise of IoT devices has increased security risks, requiring faster and more effective defenses. Although home gateways are the first line of defense, they struggle against advanced threats. Current learning-based detection methods offer speed, but face scalability and privacy issues in distributed IoT environments. To address this, our research proposes a collaborative firewall using federated learning. It enables attack detection at the gateway while maintaining raw training records on participating clients. The firewall employs federated tree models that exchange training summaries with an aggregator rather than centrally collecting raw gateway traffic. On Edge-IIoTSet dataset, our feature selection reduces the number of input features from 61 to 22, cutting training time by 75% and per-packet inference time by more than 65%, while the accuracy slightly reduced from 93.6% to 92.53%. When the federated training was used, the accuracy slightly increased to 93.8%. This shows that collaborative training did not reduce detection accuracy in our setting. Our framework combines local attack detection, collaborative training, and reduced feature and execution costs for smart-home security.

© 2026 ISC. All rights reserved.

1 Introduction

The widespread adoption of IoT devices has increased end users' attack surface [1]. A prominent example is the growing use of compromised IoT devices in DDoS attacks [2]. These threats require low-latency responses because they may propagate faster than traditional defenses can react, while sophisticated attacks against home users often require collaborative intelligence from multiple networks and detailed traffic logs [3, 4]. Accordingly, recent work increasingly applies AI-based methods to protect home IoT environments [2, 5].

Machine learning enables accurate traffic analysis, but transferring large volumes of user data to cloud servers increases communication overhead, response latency, and privacy risks. Local detection and mitigation at the network edge are therefore essential [4, 6]. IoT gateways are well positioned to intercept threats early [1, 4, 5, 7, 8]; however, scaling learning-based protection across many gateways remains challenging because centralized model training requires sharing local traffic data, potentially conflicting with privacy regulations [9].

Federated learning addresses this limitation by keeping raw data local and exchanging only model updates with a central server, thereby supporting collaborative training with lower communication cost and improved privacy [9]. We combine federated learning with programmable networking to build a distributed smart-home firewall that analyzes traffic, detects and blocks

* Corresponding author.
Email addresses: yasamin.rahimi@aut.ac.ir, shirali@aut.ac.ir

ISSN: 2008-2045 © 2026 ISC. All rights reserved.

attacks at the gateway, and benefits from knowledge collected across multiple homes. This approach follows recent privacy-preserving federated security solutions [7] while retaining the flexibility and traffic visibility of programmable networks.

Our method employs a programmable network-based firewall that operates as a distributed attack detection framework at the network edge. It uses federated tree models to keep raw training records at participating gateways while exchanging model-training summaries. The contributions of this work are as follows:

(1) **Collaboration without sharing raw traffic**: We demonstrate that federated training, in which raw traffic is not sent outside of the home, can match the same detection accuracy as centralized training.
(2) **Quantified cost of better feature selection**: We demonstrated the potential benefits of better feature selection. On Edge-IIoTSet dataset, our proposed feature selection reduces the number of input features from 61 to 22. This translated to 75% reduction in training time and 65% reduction in per-packet inference time, without increasing the model's memory footprint. This also reduces the statistics exchanged in each federated round, which is proportional to the number of features.
(3) **Federated-specific feature selection**: Federated learning-based intrusion detection systems typically reuse the same feature sets used for centralized training [9–11]. We demonstrate that a better set of features can be selected for federated model, which can compensate the usual drop in accuracy that is reported in prior work when they switch to federated training.
(4) **Model Efficiency in Attack Detection**: A single multi-class model can both distinguish benign from malicious traffic and classify attack types without sacrificing detection quality, providing a lighter and more efficient alternative to two-stage classification.

Evaluation on Edge-IIoTSet, UNSW-NB15, and CICIDS2017 datasets shows that our feature selection can achieve overall accuracies of 92.53%, 96.66%, and 97.73%, respectively. However, this also results in lower macro-F1 on the two flow-based datasets (0.48 and 0.78) because of weak performance on some of rare classes that only have few samples. Furthermore, our federated training achieved 93.8% accuracy (93.9% with federated-specific feature selection) on Edge-IIoTSet. This shows that collaborative training did not necessarily reduce accuracy.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 defines the problem and design requirements, and Section 4 presents the system design, the privacy scope, and the research questions and hypotheses. Section 5 reports the experimental evaluation, and Section 6 concludes the paper and outlines future work.

2 Background & Related Works

Software-defined networking (SDN) separates the control and data planes, enabling centralized and programmable network management. Programmable data planes further extend SDN by allowing customized packet processing using architectures such as PISA and languages such as P4 [6, 12]. Federated learning (FL) enables collaborative model training across multiple participants without sharing raw traffic, making it attractive for privacy-preserving IoT security [9, 13]. Firewalls remain the primary defense at network gateways by filtering traffic based on predefined or learned rules [4].

Existing IDS solutions typically address either binary classification or a single attack category. Recent studies have shown that multi-class learning can improve practical deployment by identifying different attack types using a single model [14]. To further improve scalability and privacy, collaborative approaches based on SDN or federated learning have been proposed. Most of these systems rely on deep learning, whereas selecting lightweight models and optimized feature subsets remains relatively unexplored for smart-home deployments [9–11, 15].

2.1 Attack Detection/Prevention

Existing intrusion detection systems employ behavior-based anomaly detection, SDN-assisted traffic analysis, and deep learning models [4, 6, 9, 12, 16–18]. For example, Shirali-Shahreza and Ganjali [4] uses their previously proposed sampling extension for statistics collection [19] to detect attacks centrally without forwarding all of the home traffic to the controller. Another example is how Wankhade and Khandare [17] uses Graph Attention Networks (GAT) for anomaly detection. Recent deep learning approaches achieve over 99.8% detection accuracy for specific attacks such as DDoS but generally operate as intrusion detection systems rather than preventive firewalls and often target individual attack categories [20, 21]. Most of these works, like ours, are supervised. There are also works such as [18] that uses unsupervised federated detection.

2.2 Privacy

Protecting user privacy remains a major challenge in collaborative IoT security. Existing solutions em-

Table 1. Comparison of Previous Works

Paper	Detection Method	Privacy	Collaboration	Centralized Control	Traffic Pattern Learning	Multi-home Data
[4]	Firewall	×	Indirect	✓	×	×
[6]	Firewall	×	Indirect	✓	✓	×
[9]	IPS	FL	Indirect	×	✓	✓
[12]	IPS	×	×	×	✓	×
[15]	Attack Mitigation	FL	Indirect	×	✓	✓
[10]	Firewall	FL	Indirect	×	✓	✓
[11]	Anomaly Detection	FL	Indirect	×	✓	✓
[16]	Anomaly Detection	Tor	Direct	×	×	✓
[22]	Anomaly Detection HSU & SDN		×	✓	✓	×
[23]	IDS	FL	Indirect	×	✓	✓
This Work	Firewall	FL	Indirect	✓	✓	✓

ploy flow sampling [4], statistical traffic features [22], anonymization through Tor [16], or federated learning [9–11, 15, 23]. However, most FL-based studies reuse feature sets designed for centralized learning and rarely investigate whether alternative feature subsets improve federated performance. Federated learning by itself does not guarantee privacy, because the shared model updates can leak information about local training data [24]. For this reason, secure aggregation [25] and privacy-preserving protocols for federated tree boosting [26] are often layered on top. Recent explainable FL frameworks address transparency but do not focus on feature optimization for smart-home gateways [13].

2.3 Attack Type

Most existing IDSs focus on binary classification or a specific attack type, since specialized models generally achieve higher accuracy. More recent studies adopt multi-class detection to distinguish multiple attack categories simultaneously, improving practical deployment while reducing dependence on attack-specific models [9, 12, 14, 21].

2.4 Collaboration

Collaboration improves detection by combining knowledge across multiple homes. Existing approaches implement collaboration either directly among household agents [16], through SDN controllers [6, 22], or via federated learning servers [10, 11, 15]. However, existing approaches generally do not combine programmable-network control, privacy-preserving federated learning, and feature optimization within a unified firewall architecture for smart-home environments.

Table 1 summarizes representative smart-home security approaches. Existing solutions typically satisfy only a subset of the desired properties, whereas the proposed firewall combines programmable-network control, federated learning, collaborative model training, and privacy preservation within a unified framework.

3 Research Problem

3.1 Problem Definition

IoT gateways are essential for defending against threats at the edge of the IoT. One effective method for protecting IoT devices at home is to use firewalls at the network edge [27]. Previous research classifies edge/cloud computing protection strategies into two categories based on the edge device's capabilities: simple (low power, cloud-dependent) and complex (high power, edge-dependent). While simple devices (such as Shirali-Shahreza and Ganjali's work [4]) often compromise privacy by sending significant data to the cloud, complex devices (such as FamilyGuard [22] and FLEAM [15]) allow for more local processing, enhancing privacy but increasing cost and complexity for users.

3.2 Design Requirements

Designing firewall policies for evolving IoT attacks is challenging. Machine learning enables effective attack detection, SDN provides a global network view for traffic analysis, and federated learning preserves user privacy during collaborative model training [6, 10, 11, 15, 22]. Based on the problem definition and limitations of existing approaches, the proposed programmable network-based firewall should satisfy the following requirements:

(1) Detect multiple attack classes using a single machine learning model to support practical deployment and accurate traffic classification [6].
(2) Apply feature selection to identify only the most informative packet features, reducing model complexity while maintaining detection accuracy [12].
(3) Support collaborative learning across multiple homes by exploiting the global visibility provided by programmable networks [10, 11, 15].
(4) Keep raw traffic at the gateway by employing federated learning, so that each home contributes to global model training without sharing its raw traffic records [9, 11, 15].
(5) Achieve detection accuracy comparable to centralized learning despite preserving privacy through federated training [9, 15].

4 System Design

4.1 Attack Model

This work addresses network-protocol attacks targeting gateway connections in both "neighbor" scenarios, where attackers share the local network, and "tenant" scenarios, where compromised IoT devices participate in botnets [28]. Attackers may exploit network vulner-

abilities through passive actions such as information gathering or active actions such as using end devices in botnets. We assume that other defenses are absent or have been bypassed, making the IoT gateway the primary line of defense [29].

4.2 Design Overview

Our primary goal is rapid, multi-class attack detection through continuously updated machine learning models, while avoiding central collection of raw traffic when aggregating knowledge across households. This research focuses on input feature selection to maintain detection accuracy in federated settings.

Figure 1 illustrates the high-level architecture in two contexts: SDN (Figure 1a) and programmable data plane networks (Figure 1b). Despite their different contexts, both architectures use the same tree-based model and a cloud-based aggregator. In SDN, a local controller extracts features via OpenFlow [4, 22]; in programmable data planes, a state machine parser using P4 handles this [12]. Allow/block decisions are enforced by SDN controllers or P4 match-action tables.

The home gateway operates as a software-defined or programmable switch, similar to [4, 9, 11, 12, 22], collecting traffic information and enforcing detection rules. Each gateway trains a local model through federated learning, keeping raw data at the gateway and not sharing it with anyone else while contributing to a global model. The updated model detects attacks and installs preventive rules directly at the gateway, thereby functioning as a firewall with the goal of achieving accuracy comparable to centralized raw-data training. Because the design relies on raw-packet feature selection rather than architecture-specific mechanisms, it remains applicable across different programmable-network platforms.

4.3 System Components

As shown in Figure 2, the framework processes traffic locally at the IoT gateway to preserve privacy while supporting low latency and high throughput [9]. Incoming packets undergo feature extraction and classification; benign traffic is forwarded, whereas detected attacks are dropped [12]. The framework contains two main components.

Attack detection uses learning-based methods for rapid threat identification. Features are first extracted from traffic, then a decision tree model classifies each packet, allowing benign traffic and immediately dropping detected threats.

Model update uses federated learning through three functional blocks: local logging, which records extracted features; local model update, which trains on local traffic; and global model training, which aggregates model parameters at a central server. In each iteration, local parameters are sent to the server, combined into a global model, and redistributed to the gateways. Feature selection is applied during initial model construction to determine the input features, enabling continuous learning and rapid identification of abnormal traffic. Although the current setup uses a central aggregation server, it could be replaced by a decentralized scheme such as HashLearner [30].

4.4 Privacy Scope and Threat Model

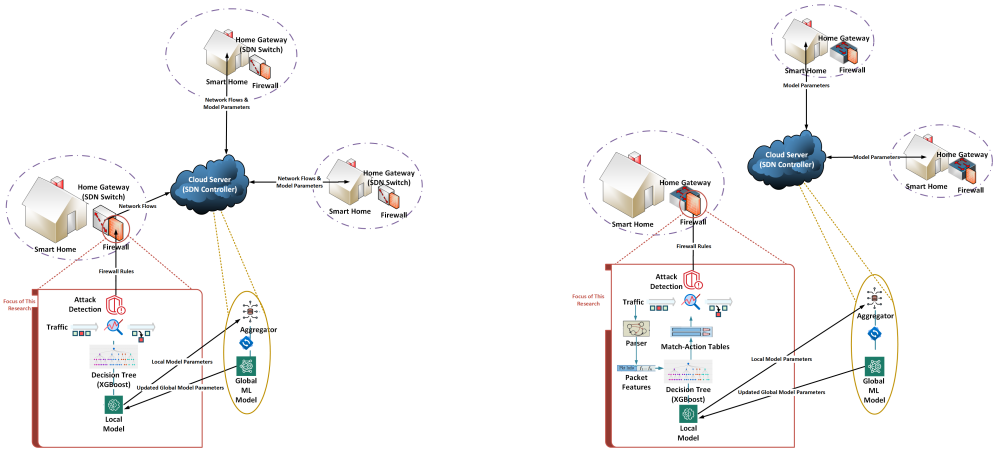
We assume that the gateway never transmits packets, payloads, or per-packet feature records. In each model update round (i.e., boosting round), the gateway sends only aggregate statistics: per-feature histograms of gradients and Hessians over quantile bins, that are used to choose tree splits [31]. However, these aggregates can still leak coarse properties of a household's traffic distribution such as frequent feature ranges. Furthermore, model updates in general can leak information about training data [24]. Our current implementation does not use secure aggregation [25] or differential privacy. Therefore, our privacy claim is limited to *keeping raw traffic at the gateway.* We left the problem of hardening the exchanged histograms for future work.

Within this scope, feature selection also acts as data minimisation. Histograms are exchanged per feature, so the statistics a gateway discloses per round, and the corresponding upload volume, grow linearly with the number of input features. Using 22 or 23 features instead of the full 61 on Edge-IIoTSet therefore reduces the shared statistics per round by about 62–64%, in addition to the computational savings reported in Section 5.

4.5 Research Questions and Hypotheses

We formulated four research questions (RQs), each paired with a hypothesis (H), to evaluate our proposed framework and test them in Section 5. All comparisons use the same held-out test set. We call two models *comparable* when their accuracy and macro-F1 are within 1% of each other. Considering that the test sets are large, even very small differences become statistically significant. Therefore, we base our conclusions on whether the overall accuracies are within 1% of each other.

- **RQ1**: Does two-step detection improve performance over a single multi-class model?
  H1: The two-step model achieves higher detection accuracy than the single model.



(a) (b)

Figure 1. Proposed System Architecture. (a) SDN. (b) Programmable Data Plane (PISA).

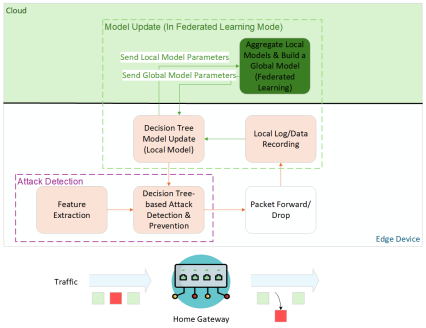


Figure 2. Proposed Design Framework.

- **RQ2**: Can feature selection maintain detection performance with lower cost?
  H2: Selected features retain comparable detection performance while reducing computational costs. Costs are measured as the number of input features, training time, per-packet inference time, model size, and memory.
- **RQ3**: Are federated training results close to centralized training condition?
  H3: The federated model's accuracy is comparable to the accuracy of centralized model.
- **RQ4**: Is it beneficial to repeat feature selection in federated training condition?
  H4: Repeating feature selection in the federated setting results in a different feature subset and improves accuracy over reusing the centralized features.

5 Evaluation Results

We replicated P4Pir [12] using the authors' publicly available code [1]. Our baseline results closely matched those reported in [12], achieving 0.999 accuracy for vulnerability scanning and DDoS TCP SYN attacks and above 0.9 for HTTP flooding.

Table 2. Evaluation Datasets

Dataset	Attack class	Feature Count	Used in
Edge-IIoTSet [33]	14	61	[12]
UNSW-NB15 [34]	9	47	[12, 15, 17, 37]
CICIDS 2017 [35]	11	78	[9, 12, 17, 23, 37]

5.1 Experimental Setup

Attack detection uses XGBoost, a tree-based ensemble model suited to tabular network traffic [31] and resource-constrained programmable switches and gateways because of its match-table-like structure [32]. Recursive feature elimination based on XGBoost feature-importance rankings is used for feature selection [33].

We evaluate the proposed method on Edge-IIoTSet [33], UNSW-NB15 [34], and CICIDS2017 [35], summarized in Table 2. These datasets are widely used to evaluate intrusion detection in the IoT domain [36]. Experiments run in Python 3 on an Ubuntu 22.04 virtual machine with 12 GB RAM and four CPU cores. All XGBoost models use `n_estimators=5` and `max_depth=5`; binary and multi-class models use logistic and softmax objectives, respectively. Class weights are applied to UNSW-NB15 and CICIDS2017 to address class imbalance [2]. Performance is measured using precision, recall, accuracy, and F1-score, with macro and weighted averages reported.

[1] `https://github.com/In-Network-Machine-Learning/P4Pir`

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html`

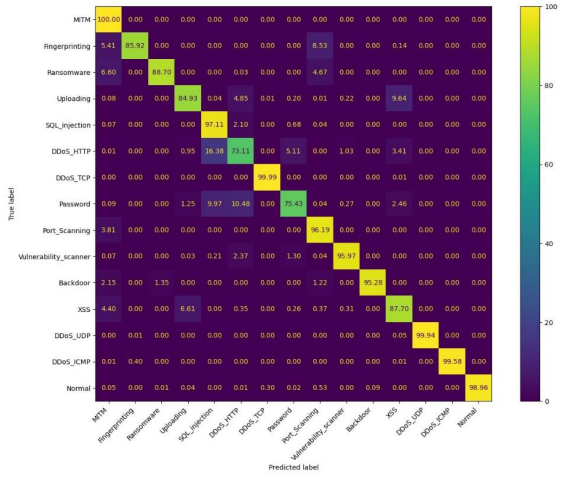


Figure 3. Two-Step Attack Detection Confusion Matrix

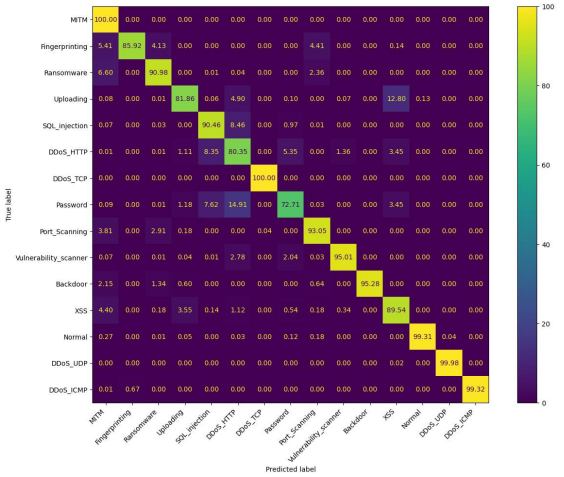


Figure 4. Single-Step Attack Detection Confusion Matrix

5.2 Centralized Learning

5.2.1 Model Selection

To evaluate H1, we compare two attack-detection strategies on Edge-IIoTSet [33]:

(1) **Two-step attack detection**: A binary model separates benign from attack traffic, after which a multi-class model identifies the attack type. Their outputs are combined in the confusion matrix shown in Figure 3.
(2) **Single-step attack detection**: One multi-class model classifies traffic as benign or as a specific attack type, as shown in Figure 4.

The difference between accuracy of two strategies was less than 1% (0.925 vs. 0.936) and there are similar class-level patterns in Figure 3 and Figure 4. The two-step pipeline is therefore not more accurate. Hence, H1 is not supported. Since a single classifier is simpler, we used it in all subsequent experiments.

The confusion matrices (Figures 3 and 4) show the same error structure for both strategies. Volumetric attacks (DDoS TCP, UDP, and ICMP) are detected almost perfectly. Most errors occur among application-layer attacks that share HTTP/TCP header patterns: in the single-step model, 16.4% of DDoS HTTP samples are labelled SQL injection, about 10% of Password samples each go to SQL injection and DDoS HTTP, and 9.6% of Uploading samples are labelled XSS. The MITM column also absorbs 2–7% of Fingerprinting, Ransomware, Port Scanning, Backdoor, and XSS samples. This explains why MITM has perfect recall but low precision (0.47 in Table 3).

5.2.2 Feature Selection

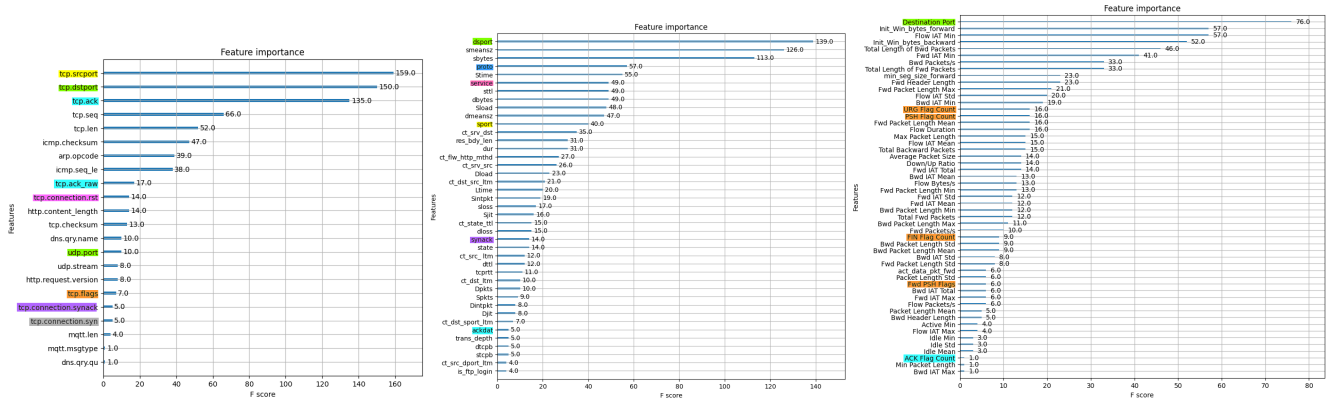
To evaluate H2, we apply recursive feature elimination to Edge-IIoTSet [33], UNSW-NB15 [34], and CICIDS2017 [35]. The method iteratively removes the weakest features according to XGBoost's `feature_importances_` ranking [33], reducing model complexity, training time, and inference time. Training parameters remain unchanged, and class weights are retained for UNSW-NB15 and CICIDS2017 [3].

Figure 5 presents the most important selected features according to the XGBoost weight metric, with equivalent features highlighted across datasets. Tables 3–6 compare the per-class and overall performance of the lightweight and full models. The lightweight models retained comparable accuracy while removing 64% of Edge-IIoTSet features, 13% of UNSW-NB15 features, and 32% of CICIDS2017 features. Performance decreased for a few sparsely represented classes, such as XSS, which may be addressed through upsampling or class merging.

Figure 5 shows that port numbers rank among the leading features in all three datasets, whereas the remaining rankings are dataset-specific: Header fields (TCP ports, sequence and acknowledgement numbers, flags) dominate on Edge-IIoTSet, while flow statistics dominate on UNSW-NB15 and CICIDS2017 datasets. Tables 3–6 show the accuracy of lightweight model in comparison to the full model on three different datasets. They show that the overall aggregate accuracy is largely retained after our feature selection.

However, our feature selection significantly reduces the costs of training and running the models. For example on the Edge-IIoTSet dataset, feature selection reduced the number of input features from 61 to 22 (64% reduction). This translates to a decrease in training time from 60 seconds to only 15 seconds (75% reduction in training time). The inference time of the

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html`



(a) (b) (c)

Figure 5. Feature Selection Results in the Evaluated Datasets. (a) Edge-IIoTSet. (b) UNSW-NB15. (c) CICIDS2017.

Table 3. Evaluation Results for Each Attack Class on Edge-IIoTSet

Attack Class	Precision Lightweight Model	Precision Full Model	Recall Lightweight Model	Recall Full Model	F1-score Lightweight Model	F1-score Full Model
MITM	0.39	**0.47**	1.00	1.00	0.56	**0.64**
Fingerprinting	0.90	**0.94**	0.86	0.86	0.88	**0.90**
Ransomware	0.95	**0.99**	**0.91**	0.89	0.93	0.93
Uploading	0.92	**0.94**	**0.82**	0.80	0.87	0.87
SQL injection	0.85	**0.87**	**0.90**	0.89	0.88	0.88
DDoS HTTP	0.72	**0.78**	0.80	**0.83**	0.76	**0.81**
DDoS TCP	1.00	1.00	1.00	1.00	1.00	1.00
Password	0.88	0.88	0.73	**0.80**	0.80	**0.84**
Port Scanning	**0.96**	0.93	0.93	**0.96**	0.94	**0.95**
Vuln. Scanner	0.98	0.98	0.95	**0.96**	0.97	0.97
Backdoor	1.00	1.00	0.95	0.95	0.98	0.98
XSS	0.82	**0.83**	0.90	**0.96**	0.85	**0.89**
Normal	1.00	1.00	0.99	**1.00**	1.00	1.00
DDoS UDP	1.00	1.00	1.00	1.00	1.00	1.00
DDoS ICMP	1.00	1.00	0.99	**1.00**	1.00	1.00

trained model also reduced from the original 1-2 µs to 0.5-0.7 µs per packet (more than 65% reduction in inference time). However, the lightweight model had similar size (less than 150 KB) and used similar RAM (0.849 MB). This is expected: with 5 boosting rounds of depth-5 trees, model size is bounded by the number of tree nodes, not by the number of available inputs. For an edge gateway, the relevant savings are therefore in *per-packet processing*: fewer header fields to parse and fewer features to evaluate per decision, which in a P4 target translates into fewer parser states and match-action keys. The number of inputs (i.e., features), also sets the volume of statistics exchanged in each federated round. Together with the accuracy results, these measurements support the cost component of H2.

Table 4. Evaluation Results for Each Attack Class on UNSW-NB15

Attack Class	Precision Lightweight Model	Precision Full Model	Recall Lightweight Model	Recall Full Model	F1-score Lightweight Model	F1-score Full Model
Normal	1.00	1.00	0.99	0.99	0.99	0.99
Exploits	0.82	0.82	0.45	0.45	0.58	0.58
Reconnaissance	0.78	0.78	0.84	0.84	0.81	0.81
DoS	0.30	0.30	0.10	0.10	0.15	0.15
Generic	1.00	1.00	0.98	0.98	0.99	0.99
Shellcode	0.43	0.43	0.99	0.99	0.60	0.60
Fuzzers	0.43	0.43	0.85	0.85	0.57	0.57
Worms	0.05	0.05	0.85	0.85	0.10	0.10
Backdoors	0.13	0.13	0.93	0.93	0.23	0.23
Analysis	0.08	0.08	0.30	0.30	0.13	0.13

Table 5. Evaluation Results for Each Attack Class on CICIDS2017

Attack Class	Precision Lightweight Model	Precision Full Model	Recall Lightweight Model	Recall Full Model	F1-score Lightweight Model	F1-score Full Model
DDoS	1.00	1.00	1.00	1.00	1.00	1.00
BENIGN	1.00	1.00	0.97	0.97	0.99	0.99
Web Bruteforce	0.06	0.06	0.64	0.64	0.11	0.11
PortScan	0.99	0.99	1.00	1.00	0.99	0.99
DoS Hulk	0.94	0.94	1.00	1.00	0.97	0.97
SSH-Patator	0.74	0.74	1.00	1.00	0.85	0.85
DoS GoldenEye	0.97	0.97	1.00	1.00	0.98	0.98
DoS slowloris	0.55	0.55	0.98	0.98	0.70	0.70
FTP-Patator	0.97	0.97	1.00	1.00	0.98	0.98
Slowhttptest	0.69	0.69	0.99	0.99	0.81	0.81
Web XSS	0.06	0.06	0.73	0.73	0.12	0.12
Infiltration	0.83	0.83	0.88	0.88	0.86	0.86

5.2.3 Discussion

UNSW-NB15 and CICIDS2017 rely on flow-based features derived from multiple packets, making real-time feature extraction more complex than the packet-based features of Edge-IIoTSet, which can be ob-

Table 6. Model Evaluation Results for Edge-IIoTSet, UNSW-NB15 and CICIDS2017 Datasets

	Edge-IIoTSet		UNSW-NB15		CICIDS2017	
Metric	Lightweight	Full	Lightweight	Full	Lightweight	Full
Accuracy	0.93	0.94	0.97	0.97	0.98	0.98
Balanced Acc.	0.92	0.93	0.74	0.74	0.93	0.93
Macro F1	0.89	0.91	0.48	0.48	0.78	0.78
Micro F1	0.93	0.94	0.97	0.97	0.98	0.98

tained directly from protocol headers. On UNSW-NB15 dataset, accuracy of 0.967 contrasts with the balanced accuracy of 0.743 and macro-F1 of 0.477, indicating that the aggregate accuracy masks weak performance in several classes. Despite using only selected features, the lightweight models achieved accuracy comparable to full-feature models across most attack classes. The destination port consistently emerged as the most important feature, followed by protocol, service, and TCP flags (SYN, ACK, FIN), demonstrating that using all features is unnecessary for acceptable detection performance. On Edge-IIoTSet dataset, the lightweight model is 1.1% (accuracy) and 1.6% (macro-F1) below the full model, while training about 4× faster and classifying 2–3× faster. On UNSW-NB15 and CICIDS2017 the accuracy of two models are identical. H2 is therefore supported.

The per-class results (Tables 3–5) also show where the models struggle. On Edge-IIoTSet, the weakest classes are MITM (precision 0.39–0.47 despite perfect recall, because other attacks are absorbed into this class; see Figure 4), DDoS HTTP, and Password. On UNSW-NB15, several rare classes (Worms, Backdoors, Analysis, Shellcode, Fuzzers) have high recall but very low precision (0.05–0.43). This is the expected effect of the class weights, which push the model to predict rare classes more often at the cost of false positives. DoS is the weakest class overall (F1 0.15). On CICIDS2017, the two web-attack classes (Brute Force and XSS) have precision of only 0.06, largely because they are hard to separate from each other using flow features. For a firewall that drops traffic, such low-precision classes would block benign or other traffic. In deployment they could be merged into a single "web attack" class, or handled with a confidence threshold before a drop rule is installed.

5.3 Federated Learning

We used the MC2 federated XGBoost implementation [4] for collaborative training across two clients and an aggregator. Raw training traffic records remained at the clients and they only exchanged training summaries with the aggregator to construct the model. Each participating machine constructs a local data histogram, sends it to an aggregator, and receives the aggregated histograms to update its model, analogous to distributed XGBoost training [31]. The evaluated configuration did not use secure aggregation or differential privacy.

Our experiments used Edge-IIoTSet with the same training parameters and train/test split as the centralized setup. Two client machines represented gateways or switches, while a third machine acted as the cloud aggregator. The data were shuffled and distributed independently and identically across clients, and training ran for 5 rounds of model updates. We evaluated two configurations: reusing the features selected in centralized mode and providing the complete feature set so that feature selection could be repeated in federated mode.

5.3.1 Same Feature Set as Centralized Mode

To evaluate H3, the federated model was first trained using the same 22 features selected in centralized mode. The federated model retained 19 of these features and removed the other three. It also assigned different importance rankings to the selected features, as shown in Figure 6a. Kendall's tau and Spearman tests were used to compare the order of centralized and federated feature-importance ranks. The results ($\tau = 0.845$ and $p = 4.29 \times 10^{-08}$ for Kendall's tau and $\rho = 0.930$ and $p = 3.86 \times 10^{-10}$ for Spearman) indicate a significant correlation between the ranking. As Figure 6 shows, the five most important features (source and destination ports, TCP acknowledgement, TCP sequence number, and TCP length) keep the same order, while several mid- and low-ranked features change position slightly.

Despite using the same input feature set, federated learning slightly improved accuracy. Figure 7 presents the corresponding confusion-matrix differences.

With the same 22 input features, the federated model reached 0.938 accuracy and 0.908 macro-F1, compared with 0.93 and 0.89 for the centralized model (Tables 6 and 7) (an increase of +1%). It also slightly exceeds the centralized model with all 61 features (0.936). The federated model is thus at least as accurate as its centralized counterpart, and H3 is supported. Figure 7 shows where the predictions changed: the largest gain is for Password (+8.0% correctly classified, mostly samples that the centralized model had labelled DDoS HTTP, −5.8%), with smaller gains for DDoS HTTP, Uploading, Port Scanning, and Vulnerability Scanner. Some errors also move between classes; for example, Fingerprinting errors shift from

[4] https://github.com/mc2-project/federated-xgboost

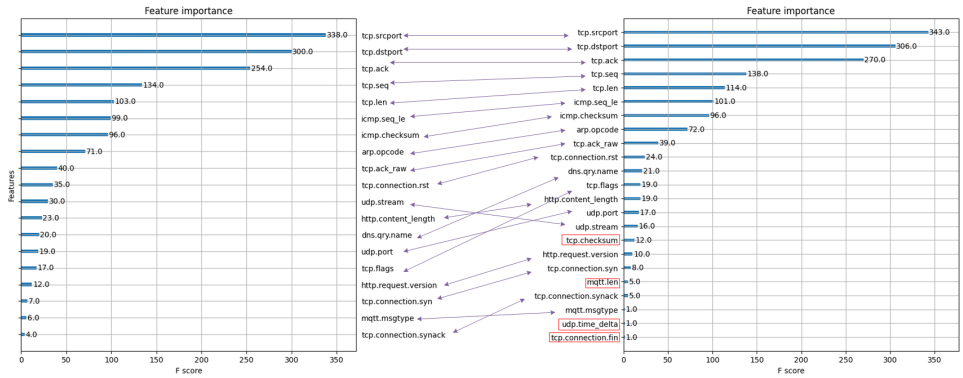


Figure 6. Most Important Features for the Federated Learning Model. (a) on the left shows the same feature set from centralized learning, while (b) on the right shows the renewed feature selection in federated learning mode.

Table 7. Evaluation Results for Federated Learning

	Precision		Recall		F1-score	
Attack Class	Same Features	Fed. Features	Same Features	Fed. Features	Same Features	Fed. Features
MITM	0.39	0.39	1.00	1.00	0.56	0.56
Fingerprinting	0.96	0.96	0.86	0.86	0.91	0.91
Ransomware	0.99	0.99	0.89	0.89	0.93	0.93
Uploading	0.98	0.98	0.83	0.83	0.90	0.90
SQL injection	0.88	0.88	0.89	**0.90**	0.88	**0.89**
DDoS HTTP	0.78	**0.79**	0.86	0.86	0.82	0.82
DDoS TCP	1.00	1.00	1.00	1.00	1.00	1.00
Password	0.86	**0.87**	0.80	**0.81**	0.83	**0.84**
Port Scanning	0.92	0.92	0.96	0.96	0.94	0.94
Vuln. scanner	**0.98**	0.97	0.97	**0.98**	0.97	0.97
Backdoor	1.00	1.00	0.95	0.95	0.98	0.98
XSS	0.87	0.87	0.91	0.91	0.89	0.89
Normal	1.00	1.00	0.99	0.99	1.00	1.00
DDoS UDP	1.00	1.00	1.00	1.00	1.00	1.00
DDoS ICMP	1.00	1.00	1.00	1.00	1.00	1.00

Metric	Same Features	Fed. Features
Accuracy	0.938	**0.939**
Bal. Accuracy	0.928	**0.929**
Macro F1	0.908	**0.909**
Micro F1	0.938	**0.939**

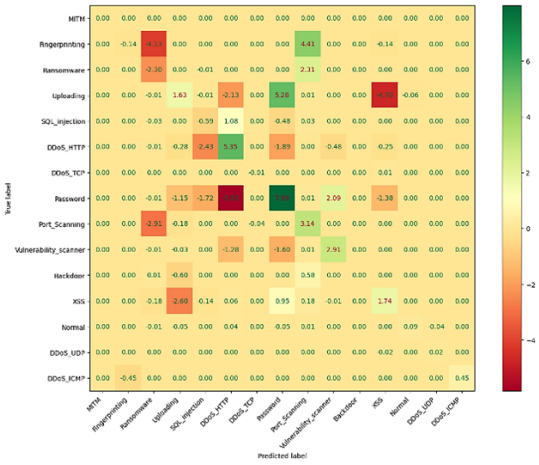


Figure 7. Difference Confusion Matrix Between Federated and Centralized Models (Same Features). Entries are federated minus centralized row-normalised percentages; positive values on the diagonal (green) mean more correct predictions under federated training.

Ransomware to Port Scanning.

5.3.2 Feature Selection in Federated Mode

To evaluate H4, the complete feature set was provided to the federated model so that feature selection could be repeated specifically for this setting. As shown in Figure 6b, the model selected 23 features, compared with the 19 retained when the centralized feature subset was reused. The highest-ranked features and their order remained unchanged. The four additional features (`tcp.checksum`, `mqtt.len`, `udp.time_delta`, and `tcp.connection.fin`) all have low importance.

The feature ranking remains strongly correlated with the centralized one (Kendall's $\tau = 0.889$, $p = 1.20 \times 10^{-9}$; Spearman's $\rho = 0.970$, $p = 4.88 \times 10^{-15}$), but the selected subset changes. Accuracy and macro-F1 each change by only 0.001 (0.938→0.939 and 0.908→0.909; Table 7), and the per-class values differ by at most 0.01. H4 is therefore only partly supported: federated re-selection yields a different subset, but in this IID two-client setting it brings no practically meaningful accuracy gain. It remains a cheap step worth including, because federated and centralized rankings need not coincide when client data are heterogeneous.

5.3.3 Discussion

On the shared Edge-IIoTSet test set, federated training did not cost accuracy. With the same 22 features it reached 0.938 versus 0.93 for centralized training, and 0.939 with federated-specific feature selection. We do not claim that federation itself improves accuracy. The advantage is small, it was observed with two clients holding IID shares of one dataset and a single train/test split, and it may reflect differences in training configuration rather than collaboration. Non-IID clients, which are typical of real homes with

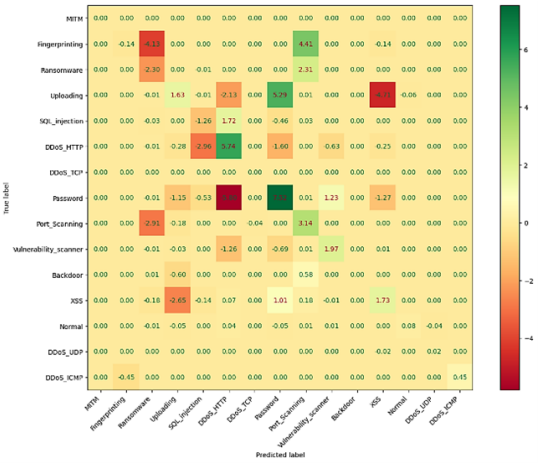


Figure 8. Difference Confusion Matrix Between Federated and Centralized Models (Renewed Features)

different devices, are the setting in which federated performance usually degrades and in which federated-specific feature selection is most likely to matter. This can be explored in future works (Section 6).

5.4 Comparison with Previous Works

Table 8 compares the precision per-class of our centralized and federated models with the results reported for P4Pir [12], which uses only five features: source and destination IP addresses, source and destination ports, and protocol. P4Pir trains a separate binary classifier for each attack type and must add a classifier when a new attack emerges, whereas our single multi-class model covers 14 attack classes at once. The two are therefore not strictly comparable. Where both report results, P4Pir is higher for vulnerability scanning (0.999 vs. 0.98) and HTTP flooding (0.902 vs. 0.72–0.79), the two are equal for DDoS TCP SYN, and our model is higher for UDP flooding (1.00 vs. 0.903). On CICIDS2017 (Table 9), our model detects port scanning (0.99 vs. 0.989) and DDoS LOIT (1.00 vs. 0.942) more reliably than P4Pir, while P4Pir additionally covers the Botnet ARES class, which is not among the classes we evaluate. High recall alone does not guarantee a usable firewall rule: MITM on Edge-IIoTSet and the web attacks on CICIDS2017 have high recall but low precision (Section 5, Tables 3 and 5).

Table 10 provides a broader comparison with related studies. The only like-for-like comparison is with GAT-AID [17], which also classifies all classes. Our model accuracy matches it on CICIDS2017 (0.98) and is slightly lower on UNSW-NB15 (0.97 vs. 0.98), while using a small tree ensemble instead of a graph neural network. FLIP4 [9] reports 1.0 for three attack classes only. Their single-attack or binary models solve an easier task than our all-class model. Our results should therefore be read as *comparable* accuracy at lower model complexity and in a single model, not as higher accuracy than prior work.

Table 8. Precision of our Model vs. P4Pir [12] under Edge-IIoTSet

Attack Class	P4Pir [12]	This Work Federated	This Work Centralized
DDoS TCP SYN	0.999	1.00	1.00
Vulnerability Scanning	0.999	0.97	0.98
HTTP Flooding	0.902	0.79	0.72
UDP Flooding	0.903	1.00	1.00
MITM	-	0.39	0.39
Fingerprinting	-	0.96	0.90
Ransomware	-	0.99	0.95
Uploading	-	0.98	0.92
SQL injection	-	0.88	0.85
Password	-	0.87	0.88
Port Scanning	-	0.92	0.96
Backdoor	-	1.00	1.00
XSS	-	0.87	0.82
DDoS ICMP	-	1.00	1.00

Table 9. Precision of our Model vs. P4Pir [12] under CICIDS2017

Attack Class	P4Pir [12]	This Work Centralized
Port Scanning	0.989	0.99
DDoS LOIT	0.942	1.00
Botnet ARES	0.989	-
Web Attack Brute Force	-	0.06
DoS Hulk	-	0.94
SSH-Patator	-	0.74
DoS GoldenEye	-	0.97
DoS slowloris	-	0.55
FTP-Patator	-	0.97
DoS Slowhttptest	-	0.69
Web Attack XSS	-	0.06
Infiltration	-	0.83

Table 10. Our Model vs. Related Works

Model	Dataset	Attack Classes	Centralized	Federated
This Work	CICIDS2017	All Classes	0.98	–
GAT-AID [17]	CICIDS2017	All Classes	0.98	–
FLIP4 [9]	CICIDS2017	Scanning, DDoS, Botnet	1.00	1.00
This Work	UNSW-NB15	All Classes	0.97	–
GAT-AID [17]	UNSW-NB15	All Classes	0.98	–

6 Conclusion

This paper presented a machine learning-based collaborative firewall that trains across smart-home gateways without centrally collecting their raw traffic records. Keeping raw records at the clients avoids their central collection, while protecting information in exchanged training summaries remains future work. On Edge-IIoTSet, the selected-feature centralized model achieved 92.53% accuracy. With the same features, federated training achieved 93.8% on the same test set (93.9% with federated-specific feature selection), so collaborative training did not reduce accuracy.

The experimental evaluation provides several practical design insights:

- **RQ1:** a two-step pipeline was not more accurate than a single multi-class model (difference below one point), so **H1** is **not supported** and one classifier suffices.
- **RQ2:** feature selection reduced the inputs by 64% on Edge-IIoTSet and training and inference time by 75% and more than 65%, with an accuracy loss of 1.1%. Therefore, **H2** is **supported**. On UNSW-NB15 and CICIDS2017 the eliminated features were unused, so the lightweight and full models are identical.
- **RQ3:** with the same features, federated training was at least as accurate as centralized training, so **H3** is **supported**.
- **RQ4:** repeating feature selection in the federated setting changed the selected subset but not accuracy in a meaningful way (only 0.1%), so **H4** is only **partly supported**.

Limitations. The federated experiments use one dataset, two clients with IID shares, and a single train/test split. The exchanged histograms are not protected by secure aggregation or differential privacy. Several top-ranked features, such as raw port numbers, TCP sequence numbers, TCP acknowledgement numbers, and capture timestamps in UNSW-NB15, may partly reflect how the datasets were recorded, which can inflate accuracy on the same dataset. Timings were measured on a general-purpose virtual machine rather than on gateway hardware.

Our proposed architecture can be implemented on different programmable-network platforms, making it suitable for flexible edge deployment. Future work includes evaluating more clients with non-IID data, where federated-specific feature selection is expected to matter most; protecting the exchanged histograms with secure aggregation or differential privacy; cross-dataset testing without potentially dataset-specific features such as raw sequence numbers and timestamps; and measuring latency, table usage, and communication cost on real gateway and P4 hardware.

References

[1] N. Hari et al. Standards-Compliant Zero Trust Framework-Based Software-Defined Perimeter (SDP) for Securing Residential IoT Networks. *IEEE Trans. Consum. Electron.*, 72(1):1312–1327, 2026.

[2] J. Ali et al. DDoS Intrusions Detection in Low Power SD-IoT Devices Leveraging Effective Machine Learning. *IEEE Trans. Consum. Electron.*, 71(1):343–351, 2025.

[3] K. Xu et al. Characterizing home network traffic: An inside view. *Pers. Ubiquitous Comput.*, 18: 967–975, 2014.

[4] S. Shirali-Shahreza and Y. Ganjali. Protecting home user devices with an SDN-based firewall. *IEEE Trans. Consum. Electron.*, 64:92–100, 2018.

[5] X. Pei et al. Edge Intelligence-Enabled Network Intrusion Detection in Compute First Networking. *IEEE Trans. Consum. Electron.*, 71(3):8178–8191, 2025.

[6] S. Prabakaran et al. Predicting attack pattern via machine learning by exploiting stateful firewall as virtual network function in an sdn network. *Sensors*, 22:709, 2022.

[7] M. Dey et al. Secure Federated Maximum Entropy Classifier for Intrusion Detection in IoT Consumer Electronics. *IEEE Trans. Consum. Electron.*, pages 1–1, 2026.

[8] M. Fu et al. FIR-GNN: A Graph Neural Network Using Flow Interaction Relationships for Intrusion Detection of Consumer Electronics in Smart Home Network. *IEEE Trans. Consum. Electron.*, 71(2):4892–4902, 2025.

[9] M. Zang et al. Federated learning-based in-network traffic analysis on IoT edge. In *IFIP Netw. Conf.*, pages 1–6, 2023.

[10] M. Dimolianis et al. DDos attack detection via privacy-aware federated learning and collaborative mitigation in multi-domain cyber infrastructures. In *11th Int. Conf. on Cloud Networking (CloudNet'11)*, pages 118–125. IEEE, 2022.

[11] T.D. Nguyen et al. DÏoT: A federated self-learning anomaly detection system for IoT. In *39th Int. Conf. on distributed computing systems (ICDCS'19)*, pages 756–767. IEEE, 2019.

[12] M. Zang et al. Toward continuous threat defense: in-network traffic analysis for IoT gateways. *IEEE Internet Things J.*, 11(6):9244–9257, 2023.

[13] M. Amiri-Zarandi et al. A federated and explainable approach for insider threat detection in IoT. *Internet of Things*, 24:100965, 2023.

[14] S. Cherfi et al. Exploring the ALNS method for improved cybersecurity: A deep learning approach for attack detection in IoT and IIoT environments. *Internet of Things*, 28:101421, 2024.

[15] J. Li et al. FLEAM: A federated learning empowered architecture to mitigate DDoS in ind. IoT. *IEEE Trans. Ind. Inform.*, 18:4059–4068, 2022.

[16] M. Yamauchi et al. Platform utilizing similar users' data to detect anomalous operation of home iot without sharing private information. *IEEE Access*, 9:130615–130626, 2021.

[17] N. Wankhade and A. Khandare. Gat-aid: A graph attention-based dual-branch framework for scalable anomaly and intrusion detection. *The ISC International Journal of Information Security*, 18 (2):1–14, 2026.

[18] H. R. Dashtabadi and S. Ahmadi. A federated framework for unsupervised intrusion detection on the modbus protocol in cyber-physical systems. *The ISC International Journal of Information Security*, 18(3):167–178, 2026.

[19] S. Shirali-Shahreza and Y. Ganjali. Empowering software defined network controller with packet-level information. In *IEEE Int. Conf. on Communications Workshops (ICC)*, pages 1335–1339, 2013.

[20] S. Hizal et al. A novel deep learning-based intrusion detection system for IoT DDoS security. *Internet of Things*, 28:101336, 2024.

[21] S.A. Bakhsh et al. Enhancing IoT network security through deep learning-powered intrusion detection system. *Internet of Things*, 24:100936, 2023.

[22] P.H. DeMelo et al. FamilyGuard: a security architecture for anomaly detection in home networks. *Sensors*, 22(8):2895, 2022.

[23] U.M. Aivodji et al. IOTFLA: A secured and privacy-preserving smart home architecture implementing federated learning. In *2019 IEEE security and privacy workshops (SPW)*, pages 175–180. IEEE, 2019.

[24] L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[25] K. Bonawitz et al. Practical secure aggregation for privacy-preserving machine learning. In *Proc. ACM SIGSAC Conf. on Computer and Communications Security*, pages 1175–1191, 2017.

[26] K. Cheng et al. Secureboost: A lossless federated learning framework. *IEEE Intell. Syst.*, 36(6): 87–98, 2021.

[27] J.-H. Lee and H. Kim. Security and privacy challenges in the internet of things. *IEEE Consum. Electron. Mag.*, 6:134–136, 2017.

[28] Q. Wang et al. MPInspector: A systematic and automatic approach for evaluating the security of IoT messaging protocols. In *USENIX Secur. Symp.*, pages 4205–4222, 2021.

[29] T. Yu et al. Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the internet-of-things. In *Proc. of the 14th ACM Workshop on hot topics in networks*, pages 1–7, 2015.

[30] K. Mohammadi et al. Hashlearner: A secure decentralized learning framework based on hashgraph. *The ISC International Journal of Information Security*, 18(3):191–205, 2026.

[31] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 785–794, 2016.

[32] C. Zheng et al. Automating in-network machine learning. *arXiv preprint arXiv:2205.08824*, 2022.

[33] M.A. Ferrag et al. Edge-IIoTset: A new comprehensive realistic cybersecurity dataset of IoT and IIoT applications for centralized and federated learning. *IEEE Access*, 10:40281–40306, 2022.

[34] N. Moustafa and J. Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conf. (MilCIS)*, pages 1–6. IEEE, 2015.

[35] I. Sharafaldin et al. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp*, 1(2018):108–116, 2018.

[36] A. Waqas. Performance evaluation of deep learning models on diverse iot datasets for intrusion detection. *The ISC International Journal of Information Security*, 18(1):19–33, 2026.

[37] C. Zheng et al. Planter: Rapid prototyping of in-network machine learning inference. *Comput. Commun. Rev.*, 54:2–21, 2024.



Yasamin Rahimi Sadegh received the B.Sc. degree in Computer Engineering from the University of Tehran in 2022 and the M.Sc. degree in Computer Engineering (Computer Networks) from Amirkabir University of Technology in 2024. She has been a Data Scientist at JobVision, Tehran, Iran, since 2021. Her research interests include artificial intelligence, machine learning, and network security.



Sajad Shirali-Shahreza is an assistant professor of computer engineering at Amirkabir University of Technology. He received his Ph.D. in computer science from the University of Toronto, where he also worked as a postdoctoral researcher. He previously worked at Google for four years. His research interests include human-computer interaction and computer networks.