# Toward Enhancing Privacy Preservation of a Federated Learning CNN Intrusion Detection System in IoT: Method and Empirical Study

DAMIANO TORRE, University of Washington Tacoma, Tacoma, WA, USA
ANITHA CHENNAMANENI, JAEYUN JO, GITIKA VYAS, and BRANDON SABRSULA,
Texas A&M University—Central Texas, Killeen, TX, USA

Enormous risks and hidden dangers of information security exist in the applications of Internet of Things (IoT) technologies. To secure IoT software systems, software engineers have to deploy advanced security software such as Intrusion Detection Systems (IDS) that are able to keep track of how the IoT devices behave within the network and detect any malicious activity that may be occurring. Considering that IoT devices generate large amounts of data, Artificial Intelligence (AI) is often regarded as the best method for implementing IDS, thanks to AI's high capability in processing large amounts of IoT data. To tackle these security concerns, specifically the ones tied to the privacy of data used in IoT systems, the software implementation of a Federated Learning (FL) method is often used to improve both privacy preservation (PP) and scalability in IoT networks. In this article, we present an FL IDS that leverages a 1-Dimensional Convolutional Neural Network (CNN) for efficient and accurate intrusion detection in IoT networks. To address the critical issue of PP in FL, we incorporate three techniques: Differential Privacy, Diffie–Hellman Key Exchange, and Homomorphic Encryption. To evaluate the effectiveness of our solution, we conduct experiments on seven publicly available IoT datasets: TON-IoT, IoT-23, BoT-IoT, CIC IoT 2023, CIC IoMT 2024, RT-IoT 2022, and EdgeIIoT. Our CNN-based approach achieves outstanding performance with an average accuracy, precision, recall, and F1-score of 97.31%, 95.59%, 92.43%, and 92.69%, respectively, across these datasets. These results demonstrate the effectiveness of our approach in accurately identifying and detecting intrusions in IoT networks. Furthermore, our experiments reveal that implementing all three PP techniques only incurs a minimal increase in computation time, with a 10% overhead compared to our solution without any PP mechanisms. This finding highlights the feasibility and efficiency of our solution in maintaining privacy while achieving high performance. Finally, we show the effectiveness of our solution through a comparison study with other recent IDS trained and tested on the same datasets we use.

CCS Concepts: • **Software and its engineering** → **Automatic programming**; • **Security and privacy** → **Software security engineering**;

Additional Key Words and Phrases: Privacy Preservation, Federated Learning, Convolutional Neural Network, Empirical Study

This research was supported in part by the Air Force Research Laboratory (AFRL) and Department of Homeland Security (DHS) Science and Technology (S&T) Directorate under award FA8750-19-C-0077.

Authors' Contact Information: Damiano Torre (corresponding author), University of Washington Tacoma, Tacoma, WA, USA; e-mail: dtorre@uw.edu; Anitha Chennamaneni, Texas A&M University—Central Texas, Killeen, TX, USA; e-mail: anitha.chennamaneni@tamuct.edu; JaeYun Jo, Texas A&M University—Central Texas, Killeen, TX, USA; e-mail: jjo@tamuct.edu; Gitika Vyas, Texas A&M University—Central Texas, Killeen, TX, USA; e-mail: gitikavyas@tamuct.edu; Brandon Sabrsula, Texas A&M University—Central Texas, Killeen, TX, USA; e-mail: brandon.sabrula@tamuct.edu.


ACM 1557-7392/2025/1-ART53
https://doi.org/10.1145/3695998

**ACM Reference format:**
Damiano Torre, Anitha Chennamaneni, JaeYun Jo, Gitika Vyas, and Brandon Sabrsula. 2025. Toward Enhancing Privacy Preservation of a Federated Learning CNN Intrusion Detection System in IoT: Method and Empirical Study. *ACM Trans. Softw. Eng. Methodol.* 34, 2, Article 53 (January 2025), 48 pages.
https://doi.org/10.1145/3695998

## 1 Introduction

With the recent deployment and wide acceptance of the 5G cellular network, numerous **Internet of Things (IoT)** applications have been developed in a variety of industries, including smart manufacturing, real-time entertainment, virtual reality, intelligent transportation, and smart healthcare [90]. IoT systems describe physical objects with sensors, processing ability, software, and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks. Examples of IoT software systems can be installed on devices such as connected appliances, smart home security systems, autonomous farming equipment, wearable health monitors, smart factory equipment, wireless inventory trackers, and so on. IoT technologies are currently widely deployed in various domains such as RFID (radio frequency identification) tags, smart cities, smart homes, smart agriculture, industrial automation, security, medical services, and digital consumer [31]. The real-time interaction capabilities of IoT systems not only require timely responses from smart IoT devices but also put forward high-quality requirements for remote cloud services, such as low response time, high throughput, high availability, and so on. [67]. IoT technologies come with various requirements, in terms of software security [63]. Enormous risks and hidden dangers of information security exist in the applications of IoT-based technologies [18]. Software security is critical because a malware attack can cause extreme damage to any piece of software while compromising integrity, authentication, and availability. If programmers take this into account in the programming stage and not afterward, damage can be stopped before it begins. Therefore, software security starts with the developers, making sure that the software is prepared for attacks or anything that tries to bring it down. This requires effort and commitment from programmers and engineers in the software development stage. This process is out of the hands of the IoT device end-user but should be an important part in deciding which software pieces to rely on. In IoT, software security problems could lead to privacy-sensitivity issues [35]. If attackers are able to infiltrate the IoT device's software, they could also access higher-level computational functions to conduct more sophisticated attacks. This is due to the fact that the characteristics of the IoT systems, which are lacking updates, have longer lifetimes, are delayed patched, and face consequences of compromise [36, 50]. To secure IoT software systems, software engineers have to deploy advanced security software such as **Intrusion Detection Systems (IDSs)** that are able to keep track of how the IoT devices behave within the network and detect any malicious activity that may be occurring [72]. When an incursion (attack) is found, the IDS alerts the device administrator, who then takes the appropriate steps to stop them by isolating the hostile devices. IDSs can help assure device security in an IoT environment by using appropriate security mechanisms and effective monitoring of system traffic. IDSs in IoT environments offer a line of defense against attacks on the Internet and on the devices connected to it. Considering that IoT devices generate large amounts of data from sensors, **Artificial Intelligence (AI)** is often regarded as the best method for implementing IDS, thanks to AI's high capability in processing large amounts of IoT data. AI techniques can considerably help detect intrusions in IoT software systems and can also be used to process the vast amount of information the users generate daily. Researchers have put forward many AI techniques to detect cybersecurity attacks using **Machine Learning (ML)** and **Deep Learning (DL)** methods. ML is a type of AI technique that can automatically discover useful information from massive

datasets [54]. DL is a subset of ML algorithms that are based on artificial neural networks that can achieve superior performances [47]. A very unbalanced dataset might cause an ML model to be biased in favor of attack types that are overrepresented. On complex datasets with numerous attributes, it becomes challenging for ML algorithms to categorize the various attack types, which motivates researchers to further investigate DL techniques. Different attack kinds are easier to differentiate because of the novel data representations that DL obtains from the input attributes [22]. DL approaches have demonstrated their capacity to handle diverse data. Additionally, they can discover data dependencies, gather information by analyzing dynamic and large-scale data, and recognize known and new attack patterns by studying historical attack patterns [8]. In recent years, DL algorithms have produced exciting results when used in support of IDS for IoT devices [66], thanks to algorithmic breakthroughs and physical parallel hardware applied to neural networks for processing massive amounts of data. In particular, centralized DL algorithms use IoT sensor data for various tasks such as demand optimization, production planning, and IDS [84]. Several DL techniques have been presented in the literature to detect cybersecurity attacks from IoT software systems, i.e., **Convolutional Neural Network (CNN)**, **Long Short-Term Memory (LSTM)**, **Recurrent Neural Network (RNN)**, **Auto-Encoders (AE)**, **Deep Neural Network (DNN)**, and so on. CNN is among the most used and best performing DL techniques to be used for intrusion detection [80]. In this article, we implement a CNN-based IDS to support the detection of cybersecurity attacks from IoT systems. In IoT systems, edge devices such as network gateways are the primary data collection points. A centralized CNN model training can pose a risk to the privacy of the training data. During the training process, attackers could potentially gain access to all data generated in the IoT system. Sensitive IoT data may give insight into the actual physical processes conducted in given sites. Moreover, attackers may use this information to launch cyber-physical attacks against critical infrastructure. Massive data collection, while vital for DL, raises the issue of privacy [66]. For instance, the broad DL-based applications in the IoT domain have also been hindered by emerging strict legal and ethical regulations to protect individual privacy [9, 60], i.e., the European General Data Protection Regulation [78]. Legally, privacy and confidentiality concerns may prevent organizations such as hospitals and research centers from sharing their medical/research datasets, barring engineers building these software systems from enjoying the advantage of large-scale DL over joint datasets. To tackle these privacy problems in IoT systems, the software implementation of a **Federated Learning (FL)** method is often used to improve both **Privacy Preservation (PP)** and scalability in IoT networks via the adoption of a global model [57]. A considerable amount of recent studies have been addressing the implementation of PP that implement technical measures, such as pseudonymization and data minimization for complying with PP principles. PP can be defined as a method of protecting information that can be sensitive to any individual [79]. In other words, the basic reason for PP is to prevent an intruder from learning more than the minimum required information regarding any specific individual either in the case of real-time or statistical data [33]. FL has been introduced to enable privacy-preserved learning without sharing private data with external entities [15, 43, 52]. Particularly, rather than sharing potentially privacy-sensitive raw data, the FL software installed on each device processes its local data to create a DL model ("local model"), which is then collectively combined to build an aggregated model ("global model") often by a coordinator. In this way, the privacy concern can be mitigated by creating a software system that is able to limit the sharing of raw data in the FL setting. The FL approach adds the capability of training DL models in a distributed fashion, reducing the risk of privacy loss because the data do not leave the local training nodes. Practical applications of FL identities are represented by large, multinational companies which have to

manage several heterogeneous systems at the same time.[1] With FL in place, the attackers would have to attack multiple distributed software systems at the edge, rather than a single centralized system, to gain control of the information. Decentralization introduced by FL increases the complexity of possible attacks and decreases the likelihood of their success. FL has become an important area of research as it provides an optimal solution for training a collaborative model when data are dispersed across different organizations. Another advantage of FL is that it allows statistical models to be trained over remote entities while keeping data localized. FL has shown great potential in many fields involving the use of mobile phones, wearable devices, and autonomous vehicles [44]. Despite its benefits, the traditional FL framework keeps the global model in an untrusted federated server, which imposes many communication congestion and privacy concerns. On one hand, FL reduces data privacy risks, on the other hand, privacy concerns still exist since it is possible to leak information about the training dataset from the trained model's parameters. Specifically, FL software systems are vulnerable to data poisoning attacks. These malicious attacks are initialized at the local side of the FL. The attacks can compromise the privacy of individuals in the training data by inferring details of a training dataset from the model's weights [88]. The attacks are conducted on the trained data handled by the local models to decrease the prediction accuracy of the DL algorithm [57]; here the adversary needs to implant a backdoor into the aggregated detection model to disrupt the data resulting from the classification [10]. The attacker then controls the data resulting from the local training procedure and the hyperparameters and may modify the weights of the resulting model before submitting it for aggregation. For this reason, it is important to develop an FL software system that implements PP techniques. To improve the security of FL systems, this article proposes an IDS leveraging an FL CNN that implements the following three PP algorithms: (a) **Differential Privacy (DP)**, which adds private noise to the training data created by the CNN models, (b) **Diffie–Hellman Key (DK)** *Exchange*, which add a secure exchange of cryptographic keys (key agreement protocol) over the FL network (insecure) in a way that overheard communication does not reveal the keys, and (c) **Homomorphic Encryption (HE)**, which performs computations on the encrypted training data without the need for the secret key to decrypt the ciphertext. As we explain more in detail in Section 7, we can find several recent studies presenting the implementation of an FL system. However, to the best of our knowledge, no previous works have implemented all three PP techniques together for FL systems' security, which motivated us in this direction. Our homomorphic cryptographic algorithm leverages a set of mathematical expressions. This solution is often called postquantum cryptography and refers to conventional cryptographic algorithms based on mathematical expressions other than factoring and discrete logarithms, and that we believe are secure against quantum attacks [56]. These solutions have the convenience of working on conventional hardware. With these solutions, we are still in the situation of computational security based on the hypothesized hardness of some problem [56]. The efficiency of our solution is tested over the following seven publicly available IoT datasets: (1) *TON-IoT*, which is a new generation of Industry 4.0/IoT and **Industrial IoT (IIoT)** dataset, (2) *IoT-23*, which is a labeled dataset with malicious and benign IoT network traffic, (3) *BoT-IoT*, which is a combination of normal and Botnet traffic, (4) *CIC IoT 2023*, which is a novel and extensive IoT attack dataset to foster the development of security analytics applications in real IoT operations, (5) *CIC* **Internet of Medical Things (IoMT)** *2024*, which offers data from the medical field, (6) *RT-IoT 2022*, which includes real-time IoT device data, and (7) *EdgeIIoT*, which provides data from Edge IoT and IIoT applications. Section 2 provides more details on the datasets used in this study.

[1] https://www.pingidentity.com/en/resources/identity-fundamentals/authentication/federated-identity-management.html

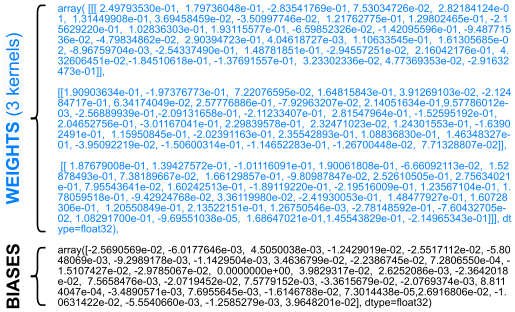


Fig. 1. Example of weights and biases of a CNN model.

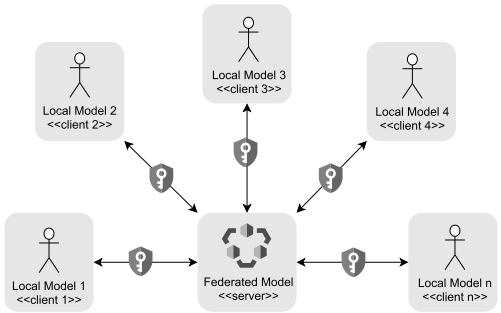


Fig. 2. Centralized FL scheme.

### 1.1 Practical Example

Figure 1 displays an example of a code snippet of weights and biases of a trained three kernels CNN-based IDS model that, in an FL environment, it could be sent from a client to the server, or vice versa, from the server to the client (see Figure 2). The CNN model in Figure 1 comprises the weights and biases resulting from the training phase of our CNN over an IoT dataset. When deployed, this IDS model is meant to be used by security software in order to detect specific cybersecurity attacks (for example, Denial of Service, Injection, Reconnaissance). If the reliability of the IDS is compromised, the security software using the model is also at risk. In a CNN, the weights and biases (see Figure 1) are the learnable parameters that the model uses to make predictions on new data. These parameters are learned during the training phase of the CNN using an optimization algorithm. The weights of a CNN refer to the values assigned to the filters in each convolutional layer. Each filter is a small matrix of values that is convolved with the input data to extract local features. During training, the weights are learned by updating the values of the filter matrices to minimize the loss function. The biases of a CNN refer to the values added to the output of each convolutional layer before applying the activation function. The bias terms allow the model to shift the activation function to the left or right, which can be helpful in fitting the training data. Like the weights, the biases are learned during training. The weights and biases are used to map the output of the convolutional layers to the final output classes. Overall, the weights and biases of a CNN are critical components of the model, and their values are learned during training to optimize the performance of the model on the given task. If the weights and biases of a CNN are compromised

by a cyber attack, it could have a significant impact on the model's performance and predictions. Here are a few potential consequences:

- —*Adversarial Attacks*: If an attacker is able to modify the weights and biases of a CNN, they may be able to craft adversarial examples that are misclassified by the model. Adversarial examples are inputs that are intentionally designed to cause the model to make a mistake, and they can be created by adding small perturbations to the input data. If an attacker has access to the model's weights and biases, they may be able to craft more effective adversarial examples that are harder to detect and defend against.
- —*Reduced Accuracy*: If the weights and biases of a CNN are compromised, it could lead to reduced accuracy on normal input data. For example, if an attacker modifies the weights and biases in a way that biases the model toward certain classes, it could cause the model to make incorrect predictions on normal input data.
- —*Model Stealing*: If an attacker is able to obtain the weights and biases of a CNN, they may be able to use that information to train a copy of the model on their own dataset. This could be particularly damaging if the original model was proprietary or used for sensitive applications. For instance, reverse engineering attacks are the act of querying information on an original targeted model, like the model architecture, test/input data, and so on, to create a surrogate model. The attacker can use the surrogate model to experiment on other attacks, like **Test-Time Evasion (TTE)**, to test if the attacks are effective on the original targeted model [55, 64, 81]. TTE attacks perturb a model's test data and cause the model to incorrectly predict a given input. The perturbations can be minimal and imperceptible to the human eye but can still change the model's decision on the input [55]. TTE attacks are easier to perform in a white-box setting due to the availability of the information on the test data but TTE attacks are also possible in a black-box setting through reverse engineering [14].
- —*Data Leaks*: In some cases, the weights and biases of a CNN may reveal sensitive information about the training data. For example, if the model was trained on medical data, the weights and biases may reveal information about individual patients or groups of patients. If an attacker is able to obtain this information, it could lead to privacy violations or other security breaches.

Overall, compromising the weights and biases of a CNN can have a wide range of consequences, depending on the nature of the attack and the application of the model. It is important to take steps to protect these parameters from potential cyber attacks, such as using secure encryption. For this reason, this article proposes an IDS leveraging an FL CNN that implements three PP techniques: (a) DP, (b) DK exchange, and (c) HE.

### 1.2 Research Questions (RQs)

The article investigates the following four RQs:

*RQ1: How can the privacy of the data exchanged between the clients and a server of a CNN-based FL system be improved?* To answer RQ1, we present a CNN-based FL approach that implements three layers of PP techniques: (i) DP, (ii) DK exchange, and (iii) an HE.

*RQ2: How accurate is our proposed approach in detecting intrusions from IoT datasets?* To answer RQ2, we examine the accuracy of our approach by testing it over seven IoT datasets: TON-IoT, IoT-23, BoT-IoT, CIC IoT 2023, CIC IoMT 2024, RT-IoT 2022, and EdgeIIoT. We ultimately compute precision, recall, accuracy, and F1-score for each dataset tested.

*RQ3: What is the impact of adding three PP techniques in a CNN-based FL system in terms of the FL performance?* To answer RQ3 we compute the **client computation time (CCT)** and the **time to receive federated data (TFD)** of our CNN model tested on seven different IoT datasets.

*RQ4: Compared to our solution, how well other AI-based IDSs are able to detect intrusion from the seven IoT datasets considered in this article?* To answer RQ4, we compare the performance of other state-of-the-art AI-based IDS solutions that were tested on the seven datasets considered in this article to our CNN-based FL solution.

### 1.3 Contributions

This article makes the following four main contributions:

(1) We describe the design of our novel FL system for an IDS powered by a CNN and secured by three PP techniques. We believe that the details of the design of our solution will improve replicability, methodological clarity, modifiability, possible future integration, scalability assessment, and overall the potential for future research and development.
(2) We provide the pseudo-code implementation of the algorithms of our (i) **1-Dimensional (1D)** CNN, (ii) DP, (iii) DK exchange, and (iv) HE. Including the pseudo-code of our implemented algorithms enhances the clarity, replicability, adaptability, and comparability of our proposed algorithms and final solution.
(3) We present the empirical evaluation of our approach using seven recent publicly available IoT datasets: TON-IoT, IoT-23, BoT-IoT, CIC IoT 2023, CIC IoMT 2024, RT-IoT 2022, and EdgeIIoT. We extensively study the effectiveness of this solution in terms of CNN-based IDS (i.e., accuracy prediction) and FL complexity (i.e., execution time). Our solution achieves an excellent average accuracy, precision, recall, and F1-score (97.31%, 95.59%, 92.43%, and 92.69%, respectively) across the seven datasets. These datasets collectively contain 415 million records.
(4) We provide a comparison of previous IDSs with our approach. In particular, we discuss the performances of previous IDSs that were trained and tested on the seven datasets used in our work. Our CNN-based IDS obtained better performance in terms of accuracy than the majority of the previous works.

### 1.4 Structure

The rest of this article is structured as follows. Section 2 provides background information about CNN, FL, and the datasets used in this work. In Section 3 we describe our approach implementing a CNN-based FL scheme with three PP techniques. This is followed by the empirical evaluation of our solution in Section 4. A preliminary discussion of the main findings of the experiments carried out is provided in Section 5. Section 6 discusses threats to validity. Section 7 compares our contributions with related work. Finally, Section 8 draws the conclusions and provides directions for future works.

## 2 Background

In this section, we briefly summarize the necessary background related to our technical approach. We then introduce the seven datasets we use.

### 2.1 CNN

The CNN has had groundbreaking results over the past decade in a variety of fields related to pattern recognition (from image processing to voice recognition) [45]. The CNN is a kind of **Feed-Forward Neural Network (FNN)** that is able to automatically extract features from data with convolution structures. The architecture of CNN is inspired by visual perception [37] where a biological neuron corresponds to an artificial neuron, the CNN kernels represent different receptors that can respond to various features, and the activation functions simulate the function that only neural electric signals exceeding a certain threshold can be transmitted to the next neuron. Loss functions and

optimizers are something researchers introduced to teach the CNN system to learn what we expect [6]. CNN's ability to effectively use binary image files for network anomaly detection has made it a reliable technique for malware classification, and various types of cyber attacks, respectively [80]. With a proper structure, CNN-based solutions can allow the concatenation of features and utilize fewer parameters for both training and testing operations. In retrospect to classification, after feature extractions, each malware code is encoded into the binary format with equal length and then can convert simhash bit to a pixel value. Depending on the range, the pixels can be then arranged in n pixel matrix using the simhash to convert them into a grayscale image to detect malware variants from the same family [34, 62]. However, when CNN's accuracy algorithm falls below the 80th percentile or a preset threshold, the weights must be adjusted using gradient descent in order to improve accuracy—a tedious process that the developer may have to perform manually, after every epoch. In lieu of its adaptability and capability, CNN, as an DL technique, is a type of artificial neural network used for both image/object classification to detect anomalies and intrusions in the realm of cybersecurity. In this work we train CNN client (i.e., local) models on local data samples, then we exchange specific parameters (e.g., the weights and biases of the CNN model) between these local nodes and a server in order to create a federated (i.e., global) model. The weights and biases are neural network parameters that simplify ML data identification. In particular, the weights and biases develop how a neural network propels dataflow forward through the network; this is called forward propagation. Once forward propagation is completed, the neural network will then refine connections using the errors that emerged in forward propagation. Then, the flow will reverse to go through layers and identify nodes and connections that require adjusting; this is referred to as backward propagation.

### 2.2 FL

FL enables multiple parties collaboratively train an ML model without exchanging the raw local data [42]. FL can be seen as an ML scheme, aiming at tackling the problem of data island while preserving data privacy [41]. In particular, it refers to multiple clients (such as mobile phones, wearable devices, autonomous vehicles, institutions, organizations, etc.) coordinated with one or more central servers for decentralized ML settings. The first FL deployed was developed by Google in 2016 to predict user's text input within users of Android devices [51]. There are three types or settings of FL, namely: centralized, decentralized, and heterogeneous FLs. In this article, we focus on the implementation of a centralized FL. Centralized FL (thereafter referred only as "FL") is generally described as shown in Figure 2. Notice that in this study we refer to the FL training approach called FL average involving multiple clients and one server. To discuss the FL scheme, we use interchangeably (a) local model or client model, and (b) federated model or server model (Figure 2). Firstly, each client trains on a local model. Secondly, the server will be improved by multiple model updates with local data, which belong to different clients separately, and then upload related gradient information to the global model in an encryption mode. Thirdly, the averaged update of local models implemented in the global model will be dispatched to each client as a renewed model. Finally, the above procedures repeat until the local model achieves a certain desired performance (convergence between local and global models) or the final deadline arrives (i.e., the end of iterations of each client). As explained, an FL scheme implements the following constraint: Given an evaluation metric, in our case accuracy, the performance of the model learned by FL should be better than the model learned by local training with the same model architecture.

### 2.3 Datasets

*TON-IoT Botnet* is a collection of datasets[2] that are a new generation of IoT and IIoT datasets for evaluating the fidelity and efficiency of different cybersecurity applications based on AI [16]. They are aimed at applications, such as intrusion detection, **Threat Intelligence (TI)**, adversarial ML, and privacy-preserving models. The datasets have been called "ToN_IoT" as they include heterogeneous data sources collected from Telemetry datasets of IoT and IIoT sensors, Operating systems datasets of Windows 7 and 10 as well as Ubuntu 14 and 18 TLS and Network traffic datasets. The datasets were collected from a realistic and large-scale network designed at the Cyber Range and IoT Labs, the School of Engineering and Information technology, **University of New South Wales (UNSW)** Canberra at the Australian Defence Force Academy. A new testbed network was created for the industry 4.0 network that includes IoT and IIoT networks. The testbed was deployed using multiple virtual machines and hosts of windows, Linux, and Kali operating systems to manage the interconnection between the three layers of IoT, Cloud, and Edge/Fog systems. Table 1 presents the types of attacks involved in the TON-IoT Botnet dataset. In particular, there are various attacking techniques, such as DoS, DDoS, and ransomware, against Web applications, IoT gateways, and computer systems across the IoT/IIoT network, **Man in the Middle (MITM)**, and **Cross Site Scripting (XSS)** attacks. The datasets were gathered in a parallel processing to collect several normal and cyber-attack events from network traffic, Windows audit traces, Linux audit traces, and telemetry data of IoT services.

*IoT-23* is a dataset of network traffic from IoT devices. It has 20 scenarios of different captures executed in IoT devices, and 3 captures for benign IoT devices traffic. It was first published in January 2020, with captures ranging from 2018 to 2019 [29]. This IoT network traffic was captured in the Stratosphere Laboratory, AIC group, FEL, CTU University in Czech Republic. Its goal is to offer a large dataset of real and labeled IoT malware infections and IoT benign traffic for researchers to develop ML algorithms.[3] This dataset and its research is funded by Avast Software, Prague. Table 1 shows the attack types distribution (fourth column) in the IoT-23 dataset with the number of attacks for each type. The majority attack types are Normal, DDoS, Okiru, PortScan, and Attack.

*BoT-IoT* dataset[4] was developed in the UNSW Canberra Cyber Range Lab by designing a realistic network environment [39, 58]. The network environment consisted of a blend of normal and Botnet traffic. The BoT-IoT dataset's raw network packets (pcap files) were produced by using the tshark tool. Ostinato tools and Node-red were used to create simulated network traffic (for non-IoT and IoT, respectively). The source files for the dataset are offered in a variety of forms, such as the original pcap files, the produced argus files, and csv files. To help with labeling, the files were divided based on attack category and subcategory. More than 72.000.000 records can be found in the 69.3 GB-sized collected pcap files. The extracted flow traffic is 16.7 GB in size and is in csv format. The DDoS and DoS attacks are further categorized according to the protocol employed in the dataset, which also includes OS and Service Scan, Keylogging, and Data Exfiltration assaults. Certain MySQL queries were used to remove 5% of the original dataset to make managing the dataset easier. The 5% that was extracted consists of four files totaling roughly 1.07 GB in size and about 3 million records.

*CIC IoT 2023* dataset[5] was created by the Canadian Institute for Cybersecurity, located in Fredericton at the University of New Brunswick. This dataset is a comprehensive and enormous IoT attack dataset created to support the growth of security analytics applications in practical IoT

[2] https://research.unsw.edu.au/projects/toniot-datasets
[3] https://www.stratosphereips.org/datasets-iot23
[4] https://research.unsw.edu.au/projects/bot-iot-dataset
[5] https://www.unb.ca/cic/datasets/iotdataset-2023.html

Table 1. List of Attacks for Each IoT Dataset Used

| Type | Description | Dataset | Attacks |
|---|---|---|---|
| DoS | It aims to prevent access or availability to data by overloading a computer system's resources with traffic. | TON-IoT<br>BoT-IoT<br>CIC IoT<br>CIC IoMT | 20,000<br>33,005,194<br>8,090,738<br>2,222,205 |
| Backdoor | It bypasses security mechanisms by replying to specific constructed client applications. | TON-IoT | 20,000 |
| DDoS | Comparted to DoS, it uses multiple attack resources to flood a targeted IoT device. | TON-IoT<br>IoT-23<br>BoT-IoT<br>CIC IoT<br>CIC IoMT<br>RT-IoT<br>EdgeIIoT | 20,000<br>85,489<br>38,532,480<br>33,984,560<br>5,846,623<br>95,193<br>337,977 |
| Reconnaissance | It uses techniques to obtain knowledge of the targeted system in order to identify vulnerabilities. | TON-IoT<br>BoT-IoT<br>CIC IoT<br>CIC IoMT | 20,000<br>1,821,639<br>354,565<br>131,402 |
| Injection | It injects malicious code to alter the behavior of a system. | TON-IoT<br>EdgeIIoT | 20,000<br>104,752 |
| MITM | It intercepts traffic and communications between the victim and the host with which the victim is trying to communicate. | TON-IoT<br>EdgeIIoT | 1,043<br>1,214 |
| Password | It aims at retrieving victims' passwords. | TON-IoT | 20,000 |
| Ransomware | It takes control over the victim's systems then asks for compensation in exchange for restoring normal systems behavior. | TON-IoT | 20,000 |
| XSS | It targets Web applications or end-users through ports, e-mails, and scripts. | TON-IoT | 20,000 |
| Attack | It indicates that there was some type of attack from the infected device to another host. For example, a brute force to some Telnet login, a command injection in the header of a GET request, and so on. | IoT-23 | 1,716,778 |
| PortScan | It uses to scope out their target environment by sending packets to specific ports on a host and using the responses to find vulnerabilities and understand which services, and service versions, are running on a host. | IoT-23 | 98,346 |
| Torii | It is a worm that targets a wide range of IoT devices to enrol into a Botnet. Torii infects systems that have Telnet exposed and protected by weak credentials. | IoT-23 | 33,858 |
| C&C | It indicates that the infected device was connected to a Command-and-Control server. | IoT-23 | 23,984 |

(Continued)

Table 1. Continued

| Type | Description | Dataset | Attacks |
|---|---|---|---|
| HeartBeat | It refers to the packets that are sent on this connection are used to keep a track on the infected host by the C&C server. | IoT-23 | 12,895 |
| FileDownload | It refers to a file that is being downloaded to our infected device. | IoT-23 | 8,035 |
| Mirai/Okiru | It is a type of malware that targets consumer devices like smart cameras and home routers, turning them into a zombie network of remote controlled bots. Okiru is a Mirai variant. | IoT-23 (Mirai)<br>IoT-23 (Okiru)<br>CIC IoT (Mirai) | 756<br>100,055<br>2,634,124 |
| Theft | There are two types of theft attacks: (a) Keylogging: it compromises a remote host for an IoT device to record the administrator's keystrokes, potentially stealing sensitive information, and (b) Data exfiltration: it compromises an IoT device to gain unauthorized access to data to transfer it on a remote attacking machine. | BoT-IoT | 1,587 |
| Web-based | To execute this attack, Web services running on IoT devices are targeted in several ways. | CIC IoT | 24,829 |
| Brute force | It consists of the submission of data (e.g., passwords or passphrases) to eventually gain access to systems. | CIC IoT | 13,064 |
| Spoofing | It enables malicious actors to operate as a victim system and gain illegitimate access to the network traffic. | CIC IoT<br>RT-IoT | 486,504<br>7,750 |
| MQTT | An MQ Telemetry Transport (MQTT) attack refers to malicious activities exploiting vulnerabilities in the MQTT protocol to compromise communication, gain unauthorized access, or manipulate messages within IoT network. | CIC IoMT | 326,653 |
| Malware | A malicious software which infiltrate the system, harming its defenses and compromising sensitive data. | EdgeIIoT | 85,940 |
| Information Gathering | Obtaining intelligence about the targeted victim is the process in information gathering by which intruders can perform attacks. | EdgeIIoT | 73,675 |
| Normal | No malicious data. | TON-IoT<br>IoT-23<br>BoT-IoT<br>CIC IoT<br>CIC IoMT<br>RT-IoT<br>EdgeIIoT | 300,000<br>150,982<br>9,543<br>1,098,195<br>192,732<br>20,137<br>1615,64,3 |

C&C, Command and Control; MITM, Man in the Middle; XSS, Cross Site Script.

operations. During the dataset-building process, 33 attacks were carried out on 105 devices in an IoT topology. In addition to the 38 Zigbee and Z-Wave devices linked to 5 hubs, 67 IoT devices were involved in the attacks. The seven kinds of these attacks—DDoS, DoS, Recon, Web-based, Brute Force, Spoofing, and Mirai—were all carried out by malicious IoT devices with the intention of harming other IoT devices [61]. Additionally, this dataset contains a variety of attacks that are not included in other IoT datasets and enables IoT experts to create new security analytics solutions utilizing data in various forms. This dataset is meant to promote the development of IoT security research and to support numerous initiatives in various IoT security aspects. It can be used in a variety of areas, including the optimization of DL models, the analysis of features and how they affect various DL models, the interpretation of classifications, and the analysis of transferability based on the comparison to other datasets.

*CIC IoMT 2024* dataset[6] was also created by the Canadian Institute for Cybersecurity similarly to CIC IoT 2023. This dataset [19] provides a realistic benchmark dataset to enable the development and evaluation of IoMT security solutions. Eighteen attacks were executed on a testbed of 40 IoMT devices (25 real and 15 simulated), using protocols common in healthcare like Wi-Fi, MQTT, and Bluetooth. These attacks fall into five categories: DDoS, DoS, Recon, MQTT, and spoofing. This dataset establishes a baseline complementary to existing contributions, aiding researchers in creating new security solutions for healthcare systems through ML mechanisms. The dataset features 18 cyber attacks on 40 IoMT devices with diverse healthcare protocols such as Wi-Fi, MQTT, and Bluetooth. This dataset significantly contributes to the healthcare industry. Additionally, the study employs an innovative methodology to simulate and capture IoMT network traffic, considering various cyber-attack scenarios with real and simulated devices. Advanced tools and hardware ensure seamless data collection. Profiling the lifecycle of IoMT devices was conducted to understand behavioral patterns under different operational states like power, idle, active, and interaction. This profiling aids in identifying security vulnerabilities and developing better solutions.

*RT-IoT 2022* dataset[7] is a proprietary dataset derived from a real-time IoT infrastructure It is introduced as a comprehensive resource integrating a diverse range of IoT devices and sophisticated network attack methodologies. This dataset [73] encompasses both normal and adversarial network behaviors, providing a general representation of real-world scenarios, and incorporating data from IoT devices such as ThingSpeak-LED, Wipro-Bulb, and MQTT-Temp, as well as simulated attack scenarios involving Brute-Force SSH attacks, DDoS attacks using Hping and Slowloris, and Nmap patterns. The bidirectional attributes of network traffic are meticulously captured using the Zeek network monitoring tool and the Flowmeter plugin. RT-IoT2022 offers a detailed perspective on the complex nature of network traffic.

*EdgeIIoT* dataset[8] provides a realistic cybersecurity dataset of Edge IoT and IIoT applications. The dataset [25] is organized into seven layers, including, Cloud Computing Layer, Network Functions Virtualization Layer, Blockchain Network Layer, Fog Computing Layer, Software-Defined Networking Layer, Edge Computing Layer, and IoT and IIoT Perception Layer. In each layer, new emerging technologies are considered in order to satisfy the key requirements of IoT and IIoT applications, such as ThingsBoard IoT platform, OPNFV platform, Hyperledger Sawtooth, Digital twin, ONOS SDN controller, Mosquitto MQTT brokers, Modbus TCP/IP, and so on. The IoT data are generated from various IoT devices (more than 10 types) such as Low-cost digital sensors for sensing temperature and humidity, Ultrasonic sensor, Water level detection sensor, pH Sensor Meter, Soil Moisture sensor, Heart Rate Sensor, Flame Sensor, and so on. The 14 attacks related to

[6] https://www.unb.ca/cic/datasets/iomt-dataset-2024.html
[7] https://archive.ics.uci.edu/dataset/942/rt-iot2022
[8] http://ieee-dataport.org/8939

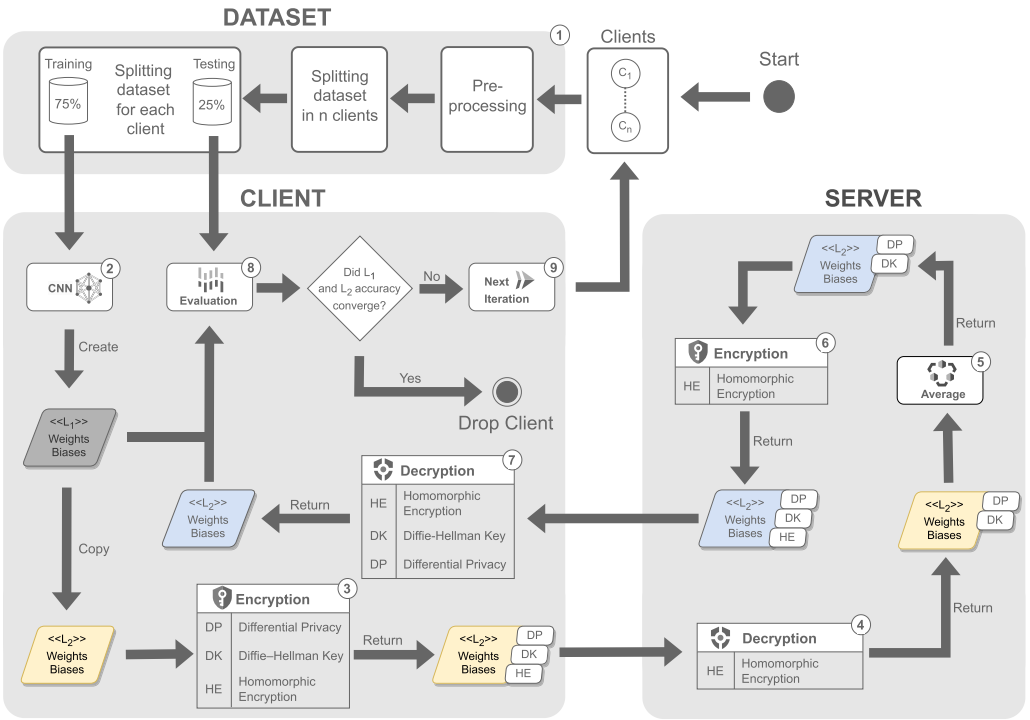


Fig. 3. Approach that implements three PP techniques for a CNN-based FL scheme.

IoT and IIoT connectivity protocols are categorized into DoS/DDoS attacks, Information gathering, MITM attacks, Injection attacks, and Malware attacks.

## 3 Approach (RQ1)

In this section, we describe our CNN-based FL scheme that implements the following three PP techniques to answer RQ1: (1) DP, (2) DK exchange, and (3) an HE. Our approach for implementing three types of PP techniques to enhance the security of CNN-based FL has nine steps, as depicted in Figure 3. In order to describe and evaluate our approach, we created a test set using four clients and one server. Each client can run up to a total of 10 iterations of the FL network in order to achieve convergence. Notice that the number of clients and the iterations are configurable for any number greater than or equal to 1. The server involved in the FL network must be exactly 1.

### 3.1 Step 1: Dataset

In our approach, we initially implemented 5% of each one of the seven considered datasets (hereafter referred to as the 5% dataset and which was cleansed by the dataset developer) for early design testing with our CNN. This dataset is a collection of .csv files containing information that one might find in a .pcap file. As follows we describe the three sub-steps that describe how each dataset was used.

Before explaining the functionality of the encryption methods, it is important to note that the original dataset is split into two sets for each client:

—$x$ is a **2-Dimensional (2D)** array of shape ($a$,$n$), where $a$ is the number of features selected for the current client's model to train against, and $n$ is the length of the dataset for the current iteration.
—$y$ is a 1D list of shape ($n_2$) where $n_2$ is the same length as $n$ of $x$.

In addition to these sets, there are also the sets of output generated by the client-side CNN models, which are:

—$Weights$ is a **3-Dimensional (3D)** array of floating point values that represent the importance of each feature of the original input; the shape of weights is ($depth$, $height$, $width$) where the $depth$ dimension corresponds to the number of channels or filters in a particular layer of the CNN, the $height$ dimension represents the spatial extent of the filter in the vertical direction, and the $width$ dimension represents the spatial extent of the filter in the horizontal direction.
—$Biases$ is a 1D array that is also commonly referred to as intercepts; this feature helps shift a feature in either the positive or negative direction; the shape of biases is ($depth_b$) where $depth_b$ is equal to $depth$ of weights.

*Pre-Processing*: We pre-processed each one of the seven datasets according to specific techniques. Please see Section 4.2 for the details of the pre-processing steps for each dataset.

*Splitting Dataset According to the n Clients*: Our test set environment considers 4 clients and 10 iterations for each client. Each dataset is split into 40 equal pieces (i.e., 10 iterations for each one of the 4 clients over 10 iterations). We use random sampling to split our datasets.

*Splitting Dataset of Each n Client*: Each part of the dataset split corresponds to an iteration of a client. Each piece of the dataset (i.e., 1/40) is then split into 75% for training, and the other 25% for testing.

### 3.2 Step 2: CNN Training

The 75% of the dataset is used to train a CNN model that produces a set of weights and biases values. The CNN model used for our purposes is TensorFlow's Keras Sequential model. Figure 4 shows the architecture of CNN comprising eight layers (i.e., L1 to L8). Algorithm 1 describes the details of the architecture, initialization, and training of our CNN. In particular, our CNN consists of an input layer, followed by one 1D convolutional layer followed by a max pooling layer. The output of the final max pooling layer is flattened and passed through a dense layer with 128 units and a ReLU activation function. The final output layer uses the softmax activation function to output the classification probabilities for each class. The model is compiled using the categorical cross-entropy loss function, the Adam optimizer, and accuracy as the evaluation metric. The fit method is used to train the model using the training data, and the evaluation method is used to calculate the test loss and accuracy of the test data. We instantiate our model with one dense layer, one dropout layer, and one batch normalization layer. Due to the fact that the CNN model uses multiprocessing and the simulation of clients is performed with multiprocessing as well, we have to instantiate a lock and barrier so that clients don't compete for resources and cause hanging processes. The lock makes it so that only one client can enter the CNN model at a time and the barrier ensures that no client may continue past the CNN model until all clients have completed their CNN training for the current iteration. Once each client has completed their CNN training for the current iteration, the CNN model will return the weights and biases for the current client's training iteration model, named $L_1$. Each Client Agent will then create a copy of the local client model, $L_2$, of the weights and biases and will then send it through the encryption process.

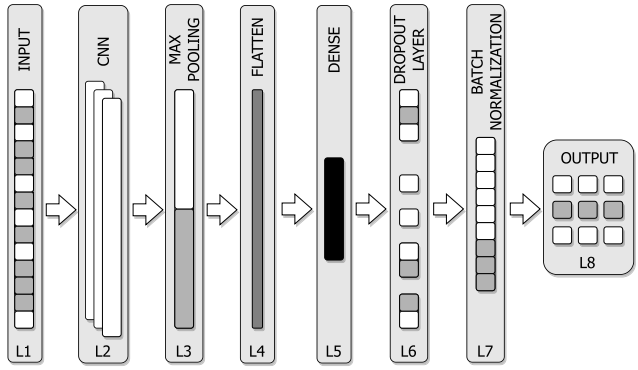


Fig. 4. CNN architecture.

**Algorithm 1:** CNN Architecture, Initialization, and Training

**Ensure:** Build CNN architecture
**Ensure:** Initialize CNN
**Ensure:** Train CNN
# Initialize the layers
*input_layer* = Input(*shape*=*input_shape*) **#L1**
*convolutional_layer*_1 = Conv1D(*filters*=32, *kernel_size*=3, activation=*relu*)(*input_layer*) **#L2**
*max_pooling_layer*_1 = MaxPooling1D(*pool_size*=2)(*convolutional_layer*_1) **#L3**
*flatten_layer* = Flatten()(*max_pooling_layer*_1) **#L4**
*dense_layer*_1 = Dense(128, activation=*relu*)(*flatten_layer*) **#L5**
*dropout_layer* = Dropout(0.5) **#L6**
*batch_normalization_layer* = BatchNormalization() **#L7**
*output_layer* = Dense(*output_shape*, activation=*softmax*)(*dense_layer*_1) **#L8**
# Compile the model
*model* = Model(inputs=*input_layer*, outputs=*output_layer*)
*model*.compile(loss=*categorical_crossentropy*, optimizer=*adam*, metrics=*accuracy*)
# Train the CNN
*model*.fit(*x_train*, *y_train*, *batch_size*= 512, *epochs* = 10, *verbose* = 1, *use_multiprocessing* = True)

### 3.3 Step 3: Encryption of the Weights and Biases (Client Side)

As follows, we present the three encryption layers we use in order to improve privacy in a CNN-based FL system.

*DP.* DP offers effective and statistical assurances against adversary learning. Adding noise to the data to disguise sensitive information so that the other party cannot differentiate the individual's information is a typical technique for using DP. As a result, it is difficult to recover the original data, making inference attacks ineffective [46]. Algorithm 2 describes the DP approach we use to add the first layer of encryption to $L_2$. In particular, the algorithm presents a method implementing a Laplace Gamma distribution that is an asymmetric method of DP that places differentially private, and floating point noise ($DN$) on our unencrypted dataset $L_2$ (see Figure 3, the incoming box $L_2$ (yellow color) incoming into Step 3 encryption). We perform this procedure using the method

**Algorithm 2:** Encryption Layer to Add Noise to $L_2$ (Differential Privacy)

**Require:** Unencrypted weights and biases $L_2$
**Ensure:** Noise-encrypted weights and biases $L_2$
1: Let $A$ be an array of 0
2: Let $A$.shape = $L$.shape
3: Let $DN$ be each element of $A$, the set of differential noise being added to $L_2$
4: $\forall(\text{DN} \in A \Rightarrow \text{DN} = 0)$
5: Let $GD_1$, $GD_2$ be floating point gamma distributions, determined using shape parameter $\alpha$ and scale parameter $\beta$
6: **for** $DN$ in $A$, **using** Laplace **do**
7: Let $DN$ = $GD_1$ - $GD_2$
8: **end for**
9: $L_2$ = ($L_2$ + $A$)

outlined by [59], in which our differentially private noise is obtained from the following equation:

$$L(\mu, \lambda) = \frac{\mu}{n} + \sum_{p=1}^{n} \gamma_p - \gamma'_p,$$

where $\mu$ and $\lambda$ are the mean and scale parameters of the Laplacian mechanism. $\gamma_p$ and $\gamma'_p$ are Gamma distributed random variables with probability density functions [59].

This method of DP creates an array of fractional noise between the values 0 and 1 and appends that array of values to $L_2$. Once that process is complete, $L_2$ is then sent through the implemented offset encryption method:

$$\lambda = \frac{\Delta}{x}$$
$$\Delta = \max_{i=1}^{n} |x_i - y_i|.$$

*DK.* DK describes the process of obtaining a symmetric key by two parties using the DK agreement protocol without any secret information being exchanged. The protocol primarily consists of three phases: initialization, creation of the private key and public key, and key agreement [20]. Algorithm 3 describes the DK exchange approach we use to add the second layer of encryption to $L_2$. This algorithm is incorporated in two major junctions within the program. First, we instantiate a key exchange, during which we exchange unique keys between each client. Each client creates unique public and secret key pairs and shares the public keys with each other client. This portion of the Diffie–Hellman incorporation occurs after the data are cleansed and split for training and testing cycles, but before the encryption of $L_2$ begins.

Once the differential noise has been added (see previous step DP), the next step is to add a security offset for each client. This offset is a cumulative addition or subtraction based on the deltas of all active clients that is applied to $L_2$. Deltas are a shared integer value representing the public key value for each client's communication with the current client. The deltas in this case are calculated by each recipient client, during the DK exchange and are stored in the directory of active clients until needed for the offset method, applied to $L_2$.

In the offset method, each client's deltas for the communication with the current client are cumulatively added to a storing variable $O_{tot}$ until the client that is running the method is also the client selected in the directory of active clients. The deltas of every client after this occurs are then subtracted from the current $O_{tot}$. Once all the deltas of every client have been added and

**Algorithm 3:** Encryption Layer to Add Offset to $L_2$ (Diffie-Hellman Key)

**Require:** Noise-encrypted weights and biases $L_2$
**Ensure:** Noise and offset-encrypted weights and biases $L_2$
Let $D$ be a directory of active clients and their offset value for the current iteration
Let $C$ be each active client in $D$
Let $CC$ be the current client executing this method
Let $O$ be the offset applied by this method, stored in $D$
Let $O_{tot}$ = 0
Let ADD = **true**
**for** $C \in D$ **do**
  **if** $C == CC$ **then**
    for all $C \in D$ after $CC$, ADD = **false**
    $O_{tot}$ -= $O$
  **else**
    $O_{tot}$+ = $O$
  **end if**
**end for**
$L_2$+ = $O_{tot}$
$CC$ updates $O$ of $D$ for all $C$ before allowing next $C$ to execute this method

subtracted, accordingly, the resultant value of $O_{tot}$ is then added to $L_2$. Without the variable $O_{tot}$ incorporated, we noted that there was a loss of noise data in $L_2$ caused by the recurring addition and subtraction to $L_2$ directly, so we removed this loss by instead summing the resultant offset first, then adding that offset total, $O_{tot}$, to $L_2$ upon completion of said summation.

Once the offset is applied to $L_2$, we then call a method to update the deltas, in which the current client's shared directory deltas are updated using a portion of the original delta as a seed value to generate a new delta. Once all deltas have been updated, the calling client moves on to the next phase of the encryption process.

*HE.* HE is a cryptographic method that enables computations to be executed on encrypted data without requiring decryption [68]. In other words, it allows operations to be conducted on encrypted data, producing an encrypted output that matches the outcome of the same operations on the unencrypted data once decrypted. There are two different types of HE schemes: (a) *partially HE* [69], which only works with specific types of computations that can be performed on the encrypted data. For example, an encryption scheme may support either addition or multiplication operations on encrypted values, but not both, and (b) *fully HE* [76], which allows for arbitrary computations to be performed on encrypted data, including both addition and multiplication operations, while preserving the confidentiality of the data. In this article, we introduce a new **fully homomorphic encryption (FHE)** technique. Algorithm 4 provides the HE method incorporated in our approach. This algorithm applies a set of flexible encryption techniques based both on information stored within the current $L$ list of the dataset, and information about each $E$ element of $L$.

The information pre-checks that are performed determine which calculations can be performed on the current dataset. The encryption method first determines if any $E$ results in a value of 0 without the previous layer's offset encryption $O_{tot}$. If any $E$ in the current list is 0, the encryption method selects $E_{opt}$, the operation to perform, from the list (add, subtract, multiply) to prevent zero-division errors from occurring. If no element in the list is 0, the encryption method instead selects from the list (add, subtract, multiply, divide), since no zero-division errors can occur, given this condition and the selection method for $B_l$.

Once the above conditions are decided, the method then proceeds, iteratively, through each $E$ of $L$ of $L_2$ and, if the selected $E_{opt}$ is not division, this function selects a single list-wise $B_l$ value for the current $L$ and apply that value and the current operation for each element of the current $L$. If the operation is division, then the value of $B_l$ instead be selected so that $B_l$ is a factor of the current $E$ in the current $L$ of $L_2$. The calculations performed on the dataset at this junction point are as follows, given that $E_{int} = (E + O_{tot})$, in the current method:

$$\textbf{addition} = E_{int} + B_l \tag{1}$$

$$\textbf{subtraction} = E_{int} - B_l \tag{2}$$

$$\textbf{multiplication} = [(E_{int} - O_{tot}) * B_l] + O_{tot} \tag{3}$$

$$\textbf{division} = \left(\frac{E_{int} - O_{tot}}{B_l}\right) + O_{tot}. \tag{4}$$

The value $O_{tot}$ is both removed and re-implemented in the same procedural step to prevent any encryption loss and allows the encryption to take place so that, in a simpler context, the following encryption can take place:

$$\textbf{addition} = E + B_l \tag{5}$$

$$\textbf{subtraction} = E - B_l \tag{6}$$

$$\textbf{multiplication} = E * B_l \tag{7}$$

$$\textbf{division} = \frac{E}{B_l}. \tag{8}$$

The randomly selected operation from above (Equations (1)–(4)) then re-appends the offset, $O_{tot}$, in the same variable value-setting step so that no value loses its integrity through the duration of the operation. Once the equation calculation has occurred, the method then checks to ensure that $E_{int}$ and $E_{flt}$ are both positive or negative, with respect to each other. If $E_{flt}$ and $E_{int}$ are not both one or the other individually, then the method inverts the sign of $E_{flt}$ by setting $E_{flt} = E_{flt} * -1$.

Once this condition has been checked, the method adds $E_{int} + E_{flt}$, sets the current $E$ of the current $L$ of $L_2$ to the resultant value, and stores the value of the current $E$ in the internal encryption directory, along with an integer representation of the operation that was performed, so that the same operational code can be used internally on $INVERSE_{E_{opt}}$ in the decryption algorithm's process.

Once Step 3 is completed, i.e., $L_2$ was encrypted with DP, DK exchange, and the HE, the encrypted $L_2$ is then sent to the server.

### 3.4 Step 4: Homomorphic Decryption of the Weights and Biases (Server Side)

The model $L_2$, containing both fully encrypted weights and biases, is accessed by the server from the client. The server performs the decryption in the inverse order that the encryption took the place of the HE.

*Homomorphic Decryption.* Algorithm 5 first performs a check to ensure that the called list $L$ of $L_2$ holds the same values within the corresponding decryption directory ($D_X$ = weights directory, $D_Y$ = biases directory). Once the appropriate list $L$ is identified, the method will give each value of that keyset a variable name so that $INVERSE_{E_{opt}}$ can perform the inverse operation of whatever was originally selected. If, for example, an addition was applied to the current $L$, the code for selecting an addition that was stored in the dataset during the encryption process would also be the code for selecting subtraction in the decryption process. This concept is the same for multiplication, subtraction, and division.

Once $INVERSE_{E_{opt}}$ has been executed, the decryption will reuse the same methods outlined in the encryption method to reverse the process performed on $E_{int}$ and return the final $E_{int}$ value

**Algorithm 4:** Homomorphic Encryption Layer Added to $L_2$

**Require:** Noise and offset encrypted weights and biases $L_2$
**Ensure:** Noise, offset, and homomorphic-encrypted weights and biases $L_2$

$D_X \leftarrow \emptyset$
$D_Y \leftarrow \emptyset$
Let $E_{opt}$ be a randomly selected operation (add, subtract, multiply, divide)
Let $O_p$ be the integer offset of the previous encryption
Let $FN \in$ (**true**, **false**)
Let $B_l$ be the pseudo-randomly selected integer used by this method to encrypt the dataset
**if** FN **then**
  **for** $L$ in $L_2$ **do**
    **if** $E_{opt}$ is not divide **then**
      Set the value, $B_l$ for the current $L$
      **for** $E$ in $L$ **do**
        Perform $E_{opt}$ on $E_{int}$ using current $B_l$
        **if** adding $E_{int}$ to $E_{flt}$ changes $E_{flt}$ **then**
          $E_{flt}$ = -$E_{flt}$
        **end if**
        Encrypt the original $E$ in the dataset
        Add $E$ and corresponding $B_l$, $E_{opt}$ data to $D_X$ or $D_Y$
      **end for**
    **else if** $E_{opt}$ is divide then **then**
      **for** $E \in L$ **do**
        Select $B_l$ such that $B_l$ is a factor of $E_{int}$
        Perform $E_{opt}$ on $E_{int}$ using selected $B_l$
        **if** $E_{int}$ and $E_{opt}$ are not both positive or negative **then**
          $E_{flt}$ = -$E_{flt}$
        **end if**
        $E = E_{int} + E_{flt}$
        Add $E$ and corresponding $B_l$, $E_{opt}$ data to $D_X$ or $D_Y$
      **end for**
    **end if**
  **end for**
**end if**

without $B_l$ applied for the given calculation type. Once this process is complete for the current record, the record will be removed from the directory and the process will begin for the next record until the entirety of $L_2$ has been fully reverted to its offset and noise-encrypted version of $L_2$, after which it will return the model back to the server for the next step to start.

### 3.5 Step 5: Averaging of the Weights and Biases from All Active Clients

The server is not training its CNN but instead serves to run the FL algorithm. It is responsible for requesting weights and biases from the active clients, averaging them, and returning to the clients the federated version of $L_2$. In this work, we use the federated averaging algorithm [44]. The federated averaging algorithm is one of the most popular approaches for FL. It is a customized version of parallel Stochastic Gradient Descent. Upon receiving the weights and biases from each client, the server averages the weights and biases and returns the federated weights and biases (i.e.,

**Algorithm 5:** Homomorphic Decryption of $L_2$

**Require:** Noise, offset, and homomorphic-encrypted Client Model weights and biases dataset $L_2$
**Ensure:** Noise and offset encrypted Client Model weights and biases dataset $L_2$

```
Let D_X and D_Y be the directories of information about the homomorphic encryption applied to each weights and biases, respectively, of L_2
Let CN be the name of the current client accessing this method
Let INVERSE_{E_opt} be a directory that reuses E_opt from the pseudo-code for homomorphic encryption in its inverse order
for each list L of L_2 do
    for each element, E of L do
        find D_X, D_Y for CN for E
        Perform INVERSE_{E_opt} for the current E
        Set E = INVERSE_{E_opt} result
        Remove E from D_X, D_Y for current CN
    end for
end for
```

the server federated weights and biases (see the blue box output of Step 5 in Figure 3)) that are ready to be sent back to each client in parallel. In other words, our solution follows an asynchronous training approach in which each client trains a local model and sends it to a centralized server, which computes the average of all models. Then the server sends the average model back to the clients.

### 3.6 Step 6: HE of the Weights and Biases (Server Side)

The encryption on the server side occurs, procedurally, right before each client re-obtains the server federated version of $L_2$ (see blue box incoming into Step 6 of Figure 3). This step reuses the encryption algorithms outlined in Step 3. It should be noted that, throughout this process, the DP noise has not yet been removed and is not removed until the dataset $L_2$ is processed by its creating/recipient client. For the purpose of again maintaining the integrity of the client-server connection, we implement a fixed offset value and reinstate the same HE method outlined in Step 3. The fixed offset value is just to ensure that the homomorphic method is not calculated against unencrypted integer values. Once the HE from Step 3 has been reapplied, we send back $L_2$ to the client to begin the decryption of the three applied encryption layers.

### 3.7 Step 7: Decryption of the Weights and Biases (Client Side)

Here, we first re-implement the homomorphic decryption method outlined in Step 4 to receive $L_2$ with only the fixed integer offset. Once we have this version of the model, we then subtract the fixed offset (i.e., the DK) from the $L_2$ and proceed to remove the noise (i.e., the DP) for $L_2$ of the client currently accessing this method, by accessing a directory of differentially private noise stored in the client agent class and organized by client. We perform an array-wise subtraction of the noise from the client's dataset, then perform the decentralized testing using the $L_1$ (mentioned briefly at the end of Step 2) as client (i.e., local) weights and biases, and $L_2$, as the server (i.e., federated) weights and biases.

### 3.8 Step 8: Check Convergence

Upon receiving the federated weights and biases, each client computes the accuracy of the federated model (i.e., $L_2$ blue box incoming into Step 8 in Figure 3) vs. its local model (i.e., $L_1$ gray box

incoming into Step 8 in Figure 3) on the test set (25%). To compute the accuracies of $L_1$ and $L_2$, the corresponding weights and biases are converted into a TensorFlow array from a numpy array, and the CNN Evaluation Model then calculates the accuracy by pre-initializing itself with the weights and biases from the past model (i.e., $L_1$ and $L_2$) and uses TensorFlow's built-in method for calculating accuracy. In this step, we evaluate the model's ability to correctly determine whether or not an attack has been identified, and we compare the $L_1$ ability to do so against the ability of $L_2$. The results for each client's local and federated accuracies are then stored in a directory before determining if another iteration is needed. Each client then computes whether its weights and biases (i.e., $L_1$) have converged with the federated weights and biases (i.e., $L_2$) and tells the server whether it is dropping out after this iteration or not. At the end of each iteration, we compute (a) the local accuracy of $L_1$, the federated accuracy of $L_2$ to assess the performance of the CNN, and (b) the CCT for each iteration and the time to receive the federated weights for each client to assess the performance of the FL scheme. We provide the details of the evaluation of our experiment in Section 4. At the end of this step, we check the value of the weights and biases by determining if the maximum absolute difference between the federated and local weights and biases is within a pre-configured tolerance range. If this is the case, there is convergence and the client is terminated because reached convergence. If the absolute maximum difference of the weights and the absolute maximum difference of the biases are both within tolerance for all clients, then the model will print out the iteration of convergence for each client, then will print out the averaged local and federated accuracies for all clients on each iteration. In our experiment, the pre-configured tolerance range for convergence is 0.0005. We decided to set our experiment with this tolerance value to allow the maximum number of iterations among the clients and server and to try to improve the prediction accuracy of our CNN-based IDS.

### 3.9 Step 9: Next Iteration

If no convergence is achieved, i.e., if the differences of the weights and the biases are not both within the pre-configured tolerance expectations, then the federated system scheme will run for another iteration and initialize the new version of the CNN training model (i.e., Step 2 that creates the new version of $L_1$) with the current federated weights and biases from the previous iteration (i.e., the current $L_2$). The FL scheme will then restart as shown in Figure 3. Our approach will run for a maximum of 10 iterations. Once all clients have performed all 10 iterations (when needed), we provide a summary of the iteration when each client converged on, the average local accuracy per iteration (where the average is calculated by summing the accuracy for each client for a given iteration and dividing by the number of clients), and the average federated accuracy per iteration (where the average is calculated by summing each client's server-side test accuracy and dividing by the number of clients).

### 3.10 How Is Privacy Preserved

In this subsection, we elaborate on the PP techniques described above and provide formal discussion and theoretical justifications on how the privacy of the CNN model is guaranteed.

*3.10.1 DP with Laplace Gamma Distribution.* Our DP offers effective and statistical assurances against adversary learning by adding noise to the data to disguise sensitive information, making it difficult to recover the original data. This technique renders inference attacks ineffective. Our DP approach adds a first layer of encryption to $L_2$ using a Laplace Gamma distribution, an asymmetric DP method placing differentially private, floating point noise ($DN$) on our unencrypted dataset $L_2$.

The differentially private noise is obtained from the equation:

$$L(\mu, \lambda) = \frac{\mu}{n} + \sum_{p=1}^{n} \gamma_p - \gamma'_p,$$

where $\mu$ and $\lambda$ are the mean and scale parameters of the Laplacian mechanism. $\gamma_p$ and $\gamma'_p$ are Gamma-distributed random variables. The DP guarantee is formalized by ensuring that for any two neighboring datasets $D$ and $D'$ differing by at most one element, and any set of outputs $S$:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in S] + \delta.$$

By adding noise sampled from a Laplace distribution with parameters $\mu$ and $\lambda$, the mechanism $\mathcal{M}$ achieves $(\epsilon, \delta)$-DP. This can be proven using the Laplace mechanism proof [65].

Considering the weights and bias provided in the example from Figure 1, once we apply the DP technique using the Laplace Gamma Distribution, we add noise to the data to disguise sensitive information. For instance, if the original weights and bias are:

—Original Weights: [0.24979353, 0.17973605, −0.28354177, 0.07530347, 0.28218412, ...]
—Original Bias: [−0.02569057, −0.00601776, 0.00450500, −0.01242902, −0.02551711, ...]

By adding Laplace noise with a mean ($\mu$) of 0 and scale ($\lambda$) of 0.1 to each weight and bias value, we modify these values to include DP. For example, adding noise samples such as [0.045, −0.023, 0.067, −0.012, 0.054, ...], the updated weights and bias would be

—Weights after DP: [0.29479353, 0.15673605, −0.21654177, 0.06330347, 0.33618412, ...]
—Bias after DP: [0.01930943, −0.02901776, 0.07150500, −0.02442902, 0.02848289, ...]

This noise addition ensures that inference attacks on the data are ineffective, thereby preserving privacy.

*3.10.2 DK Exchange.* Our DK works by obtaining a symmetric key by two parties using the DK agreement protocol without exchanging any other information. The protocol consists of three phases: initialization, creation of the private key and public key, and key agreement. Each client creates unique public and secret key pairs and shares the public keys with other clients. The differential noise is added first, followed by a security offset for each client, calculated by each recipient client during the DK exchange. The security of the DK exchange relies on the computational Diffie–Hellman assumption [32], which states that given $g$, $g^a$, and $g^b$, it is computationally infeasible to compute $g^{ab}$ without knowing $a$ or $b$. This ensures that an eavesdropper cannot derive the shared secret key.

Considering the weights and bias provided in the example from Figure 1, once we apply the DK technique, we secure the data using a symmetric key obtained through the Diffie–Hellman protocol. Let's consider the original weights and bias shown above for DP. With a shared secret key $k = 0.05$ and differential noise such as [0.03, −0.02, 0.04, −0.01, 0.02, ...], the weights and bias are modified by adding a security offset. The resulting values will look like:

—Weights after DK: [0.32979353, 0.10973605, −0.24354177, 0.06530347, 0.30218412, ...]
—Bias after DK: [0.00430943, −0.02601776, 0.04450500, −0.02242902, −0.01551711, ...]

This technique ensures that the shared secret key remains secure and the data are protected during transmission.

*3.10.3 HE.* Our HE allows computations to be performed on encrypted data, producing an encrypted result that matches the output of operations on unencrypted data once decrypted. Our approach uses an FHE scheme, allowing for arbitrary computations on encrypted data. The

encryption method performs checks to prevent zero-division errors and selects operations (addition, subtraction, multiplication, division) based on the data properties. The security of our HE scheme is based on the hardness of problems like Learning with Errors [3] and Ring Learning with Errors [30]. These problems are computationally hard, meaning that it is infeasible for an attacker to decrypt the data without the secret key. The fully homomorphic property of the encryption scheme ensures that any sequence of additions and multiplications performed on the ciphertexts directly corresponds to the same sequence of operations on the plaintexts. This is formally represented as: let $E_1 = Enc(m_1)$ and $E_2 = Enc(m_2)$ be the encrypted values of plaintexts $m_1$ and $m_2$. The homomorphic property guarantees that: (1) Addition: $E_1 + E_2 = Enc(m_1 + m_2)$, (2) Multiplication: $E_1 \times E_2 = Enc(m_1 \times m_2)$, (3) Subtraction: $E_1 - E_2 = Enc(m_1 - m_2)$, and (4) Division: $E_1 \div E_2 = Enc(m_1 \div m_2)$ (assuming $m_2 \neq 0$ and is a factor of $m_1$).

These properties ensure that the encrypted computations are consistent with the plaintext computations, preserving the integrity of the data while maintaining confidentiality.

Again, let's consider the weights and bias provided in the example from Figure 1, and once we apply the HE technique, we enable computations to be performed on encrypted data. Let's consider the original weights and bias shown above for DP. Using an HE scheme where each weight and bias is encrypted with a key $k = 2$, the encrypted values would be

—Weights after HE: [0.49958706, 0.35947210, −0.56708354, 0.15060694, 0.56436824, ...]
—Bias after HE: [−0.05138114, −0.01203552, 0.00901000, −0.02485804, −0.05103422, ...]

This encryption allows for secure computations on the encrypted data, ensuring that any operations performed on the ciphertexts directly correspond to the same operations on the plaintexts, thus maintaining data integrity and confidentiality.

### 3.11 Approach Advantages and Practical Scenario

Our approach leverages a combination of three PP techniques: DP, DK exchange, and HE—to enhance the security of a CNN-based FL system. By combining these techniques, we achieve the following general advantages: (1) *Strong Data Privacy*: by using DP, we add random noise to the data, making it difficult for anyone to identify specific individuals from the dataset. This ensures that personal information remains confidential; (2) *Secure Communication*: the DK enables secure sharing of cryptographic keys between clients, ensuring that even if someone intercepts the communication, they cannot decipher the exchanged information; (3) *Protected Data Computation*: HE allows computations to be performed on encrypted data without decrypting it. This means that data remain encrypted throughout the processing stages, preventing unauthorized access to sensitive information; and (4) *Flexibility and Scalability:* the approach is flexible and scalable, capable of handling different numbers of clients and iterations without compromising security.

For example, let's imagine one is part of a network of hospitals that want to collaborate on developing an ML model to detect diseases from patient data. However, due to privacy concerns, the hospitals cannot share their patient data directly with each other or with a central server. Here's how our approach helps:

(1) *Data Preparation (Step 1)*: Each hospital has its own patient data. These data are pre-processed and split into smaller chunks for training and testing. The data include features (like patient symptoms) and labels (like diagnosis results).
(2) *Local Training (Step 2)*: Each hospital uses its data to train a local CNN model. This model learns to identify patterns in the data to make predictions (e.g., whether a patient has a certain disease).

(3) *Encryption (Step 3)*:
   —DP: adds noise to the data to protect individual patient information.
   —DK: securely exchanges cryptographic keys between hospitals to ensure secure communication.
   —HE: encrypts the model weights and biases so computations can be performed without decrypting the data.
(4) *Server Aggregation (Steps 4–6)*: The encrypted model parameters (weights and biases) are sent to a central server. The server decrypts the data, averages the model parameters from all hospitals, re-encrypts the aggregated model, and sends it back to each hospital.
(5) *Decryption and Evaluation (Steps 7–9)*: Each hospital decrypts the received aggregated model and evaluates its performance. They check if the model has converged (i.e., the differences in weights and biases are within a set tolerance). If not, the process repeats for more iterations.

The main motivation behind this approach is to enable collaborative learning while preserving the privacy of data. In scenarios like the example above of healthcare, patient data privacy is crucial, and sharing raw data is not an option. As follows, our approach provides the following advantages for the healthcare scenario:

(1) *Privacy*: by adding noise (DP), securely exchanging keys (DK), and performing computations on encrypted data (HE), we ensure that sensitive patient information remains confidential.
(2) *Security*: the use of multiple encryption layers makes it extremely difficult for attackers to access or infer any meaningful data.
(3) *Accuracy*: the FL process allows hospitals to build a robust and accurate model by leveraging data from multiple sources, without compromising privacy.

In sum, our approach would allow different hospitals to work together to build a powerful disease detection model without ever sharing their sensitive patient data, ensuring privacy and security throughout the process.

## 4 Empirical Evaluation

This section describes the empirical evaluation of our approach according to the remaining three RQs that focus on the performance of CNN-based IDS (i.e., RQ2), the FL scheme (i.e., RQ3), and the comparison of the performance of other state-of-the-art AI-based IDS solutions that were tested on the seven datasets we use (i.e., RQ4).

### 4.1 Implementation

We have implemented our approach using Python version 3.9.12 and Jupyter Notebook version. The implementation has 2,895 lines of code including comments and third-party libraries. See Table 2 showing the main third-party software libraries used in our implementation. The entire implementation is composed by 20 Python files. The entire development of our approach, i.e., the FL system and the CNN training and testing, on a Lambda machine that uses Windows 10 Pro, Intel i7-1987H Processor 2.3 GHz, 64 GB RAM memory, and NVIDIA GeForce RTX 2080.

### 4.2 Pre-Processing of the Datasets

As follows we present the pre-processing steps executed on each of the seven datasets considered in this article. In this article, to avoid the possibility of creating a poorly performing model that may lead to overfitting and poor generalization performance, we do not apply any under-sampling or over-sampling to any of the seven datasets we use.

Table 2. Software Libraries

| Library | Version | License | Developer | Language | Link |
|---|---|---|---|---|---|
| Numpy | 1.23.5 | BSD | T. Oliphant | Python, C | https://numpy.org/ |
| Pandas | 1.5.2 | New BSD | W. McKinney | Python, C, Cython | https://pandas.pydata.org/ |
| TensorFlow | 2.11.0 | Apache 2.0 | Google | Python, C++ | https://www.tensorflow.org/ |
| Keras | 2.11.0 | Apache 2.0 | F. Chollet | Python | https://keras.io/ |
| Scikit-learn | 1.0.2 | New BSD | D. Cournapeau | Python, C, C++, Cython | https://scikit-learn.org/ |

*BoT-IoT*: We used the complete IoT Botnet Dataset, which is a collective dataset of approximately 73 million records, and worked on the development of a data cleansing tool that pre-processes and omits any inaccurate data from evaluation, without altering the original files. By, "inaccurate data," we mean that there were some rows where the data did not match the rest of the data for that column in a meaningful manner. For example, the primary culprits for any value issue were the sport (sender port) column and the dport (destination port) column. The expected value was the integer value of a port; however, there were random string and hex values for a few thousand of these records (primarily hexadecimal though). In an effort to address these issues, we converted the hex values to their integer counterparts, and we changed string values to 0 in sport and dport columns. There were some missing values too, so we filled them with 0. We dropped smac, dmac, soui, sco, and dco columns since they were empty columns. After that, we changed string values to integers by using the method *LabelEncoder*() of the software library Scikit-learn.

*TON-IoT*: This dataset was filtered to generate standard features and their label. The authors processed and filtered the entire dataset in the format of CSV files to be used by any platform. In our case, we used TON IoT Train_Test_Network dataset. The dataset comprises 461,040 records that were selected for evaluation by AI and ML algorithms. Since the dataset was already pre-processed, we did not need to fix any missing value or mixed data type. Similarly to what we did with BoT-IoT, we converted columns with string type to an integer by using the method *LabelEncoder*() of the software library Scikit-learn.

*IoT-23*: We used the entire IoT-23 Dataset, which is a collective dataset of approximately 325.3 million records. As follows there are two adjustments done to this dataset: (a) we changed string values in duration, orig-bytes, and resp-bytes columns to 0, and (b) we dropped local-orig, local-resp, missed-bytes, and tunnel-parents columns since they are empty. We do not consider Heartbeat, FileDownload, and Mirai attacks because of the low number of samples available. Similarly to what we did with BoT-IoT and TON-IoT, we converted columns with string type to an integer by using the method *LabelEncoder*() of the software library Scikit-learn.

*CIC IoT 2023*: We used a complete CIC IoT Dataset, which is a collective dataset of approximately 46.7 million records. This dataset is already pre-processed, so there was no missing value or mixed data type that needed to be fixed. In particular, the creators of this dataset [61] split the dataset into smaller chunks to perform the conversion from pcap to csv. Subsequently, they extracted several features using the DPKT package[9] and store the data in separate csv files. Once the features were identified, the authors grouped the values captured in window sizes of 10 and 100 packets

[9]https://pypi.org/project/dpkt/

to mitigate data size discrepancy (e.g., DDoS and CommandInjection) and calculate their mean values using Pandas and Numpy. Finally, they combined all subfiles into a processed csv dataset using Pandas. Notice that we do not consider Web-based and Brute-force attacks because of the low number of samples available.

*CIC IoMT 2024*: We utilized the complete dataset, comprising approximately 9 million records. The dataset required minimal pre-processing. We focused on further refining the data by ensuring consistency in column values. Specifically, we converted any string values in the columns to integers using the method $LabelEncoder()$. Additionally, we addressed missing values by filling them with 0 to maintain data integrity, removed empty columns, and normalized the dataset using $MinMaxScaler()$. This last method is used to normalize features by scaling them to a range between 0 and 1, ensuring equal contribution of all features to the analysis.

*RT-IoT 2022*: This dataset, consisting of about 1.5 million records, was processed to ensure uniformity across all data points. We identified and addressed any inconsistent data entries by converting hexadecimal values to their integer equivalents and replacing any string values with 0 in relevant columns. We also removed any completely empty columns. Similarly to the previous dataset, this dataset was then converted into a format suitable for ML algorithms, with string values in columns converted to integers using the method $LabelEncoder()$ and finally normalized with $MinMaxScaler()$.

*EdgeIIoT*: For this dataset, which includes approximately 2.5 million records, we performed comprehensive data cleansing to rectify inconsistencies. Any string values in numerical columns were replaced with appropriate integer values or 0s where necessary. We filled missing values with 0 and eliminated columns that were entirely empty, and renamed multiple columns to a better understanding of the dataset. $LabelEncoder()$ and $MinMaxScaler()$ were used also to pre-process this dataset.

### 4.3 CNN-Based IDS Performance (RQ2)

We answer RQ2 by discussing the metrics pertaining to the accuracy performance of our CNN model. For our study, we compute four evaluation metrics: precision, recall, accuracy, and F1-score. As follows, we discuss how those evaluation metrics are calculated. Assuming a classifier that predicts whether a given intrusion attack (e.g., DDoS) belongs to a certain class or not (returns true for positive prediction and false otherwise), we refer to a classification of an item as:

— *True Positive (TP)* when the item actually belongs to the class and the prediction is true,
— *False Positive (FP)* when the item actually does not belong to the class and the prediction is true,
— *True Negative (TN)* when the item actually does not belong to the class and prediction is false, or
— *False Negative (FN)* when the item actually belongs to the class and prediction is false TP and TN refer to correct classification cases (ground truth and classifications are identical), whereas FP and FN refer to incorrect classifications.

According to the above definitions, as follows, we present the definition of precision, recall, accuracy, and F1-score metrics:

— *Precision*: the proportion of the number of correctly predicted cases as positive to the number of predicted cases as positive, high precision relates to a low false-positive rate.
$Precision = \frac{TP}{TP+FP}$
— *Recall*: the proportion of the number of correctly predicted as positive to the number of cases labeled as positive.

Table 3. Results: TON-IoT

| CA | A (%) | P (%) | R (%) | F1 (%) |
|---|---|---|---|---|
| Normal | 98.88 | 99.10 | 99.17 | 99.14 |
| DDoS | 99.64 | 95.47 | 96.59 | 96.03 |
| Reconnaissance | 99.67 | 96.29 | 95.93 | 96.11 |
| Backdoor | 99.60 | 94.24 | 96.68 | 95.44 |
| Injection | 99.58 | 96.96 | 93.03 | 94.96 |
| Password | 99.73 | 97.84 | 95.85 | 96.84 |
| Ransomware | 99.68 | 93.91 | 98.99 | 96.38 |
| XSS | 99.63 | 96.75 | 94.86 | 95.79 |
| DoS | 99.97 | 99.74 | 99.69 | 99.72 |
| MITM | 99.90 | 94.46 | 61.00 | 74.09 |

Table 4. Results: IoT-23

| CA | A (%) | P (%) | R (%) | F1 (%) |
|---|---|---|---|---|
| Normal | 99.98 | 99.90 | 98.97 | 99.43 |
| PortScan | 99.98 | 99.97 | 99.99 | 99.98 |
| C&C | 99.99 | 97.06 | 64.11 | 71.40 |
| Okiru | 99.99 | 99.99 | 99.99 | 99.99 |
| DDoS | 100.00 | 99.99 | 99.99 | 99.99 |
| Attack | 99.99 | 99.26 | 97.12 | 98.18 |

Table 5. Results: BoT-IoT

| CA | A (%) | P (%) | R (%) | F1 (%) |
|---|---|---|---|---|
| DoS | 100.00 | 99.40 | 77.32 | 86.42 |
| Theft | 100.00 | 99.94 | 100.00 | 99.97 |
| DDoS | 100.00 | 94.76 | 98.65 | 96.64 |
| Reconnaissance | 100.00 | 100.00 | 100.00 | 100.00 |
| Normal | 100.00 | 100.00 | 100.00 | 100.00 |

$Recall = \frac{TP}{TP+FN}$

—*Accuracy*: the proportion of the number of correctly predicted cases to the labeled total of records.
$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

—*F1-Score*: the weighted average of precision and recall; this score considers both false positives and false negatives.
$F1 = \frac{2 \cdot precision \cdot recall}{precision+recall}$

The performance overview of our CNN-based model for IDS on all seven datasets that we have used can be seen in Tables 3–9. For the TON-IoT dataset as shown in Table 3, the percentages of all the evaluation measures show promising results with above 93% throughout and specifically accuracy above 98%. Out of the 10 attacks in this dataset, only the MITM attack has dropped in percentages to 61% and 74.09% for recall and F1-score, respectively. The reason of this drop on both

Table 6. Results: CIC IoT 2023

| CA | A (%) | P (%) | R (%) | F1 (%) |
|---|---|---|---|---|
| DDoS | 85.93 | 88.31 | 61.52 | 72.52 |
| DoS | 85.81 | 69.21 | 91.45 | 78.79 |
| Mirai | 99.85 | 99.83 | 99.52 | 99.68 |
| Normal | 96.24 | 73.17 | 97.24 | 83.49 |
| Spoofing | 97.77 | 90.03 | 54.42 | 67.82 |
| Reconnaissance | 98.16 | 84.59 | 51.12 | 63.49 |

Table 7. Results: CIC IoMT 2024

| CA | A (%) | P (%) | R (%) | F1 (%) |
|---|---|---|---|---|
| DDoS | 89.31 | 87.13 | 98.29 | 86. 43 |
| DoS | 89.10 | 93.84 | 93.31 | 96.48 |
| MQTT | 99.95 | 99.86 | 98.70 | 99.27 |
| Reconnaissance | 99.90 | 99.04 | 94.56 | 96.75 |
| Normal | 99.86 | 94.52 | 99.25 | 96.83 |

Table 8. Results: RT-IoT 2022

| CA | A (%) | P (%) | R (%) | F1 (%) |
|---|---|---|---|---|
| DDoS | 99.91 | 100.00 | 99.88 | 99.94 |
| Spoofing | 99.30 | 95.03 | 93.76 | 94.39 |
| Normal | 99.21 | 97.07 | 98.13 | 97.60 |

Table 9. Results: EdgeIIoT

| CA | A (%) | P (%) | R (%) | F1 (%) |
|---|---|---|---|---|
| DDoS | 91.50 | 97.79 | 93.83 | 92.66 |
| Information Gathering | 95.16 | 98.79 | 89.67 | 92.33 |
| Injection | 91.29 | 96.47 | 90.75 | 95.60 |
| Malware | 87.52 | 97.13 | 94.26 | 92.60 |
| MITM | 89.28 | 99.96 | 100.00 | 85.29 |
| Normal | 99.97 | 97.85 | 97.16 | 98.54 |

recall and F1-score is related to the low number of samples available for MITM (i.e., only 1,043 samples). In other words our CNN model has not enough data to be able to recognize MITM attacks as well as they do with other attacks of the TON-IoT dataset. Table 4 depicts the performance of the model on the IoT-23 dataset which indicates the reduction in recall and F1-score percentages to 64.11% and 71.40%, respectively, for **Command and Control (C&C)** Attack and the rest of the performance is above 97% with accuracy being above 99%. Similarly to what happened with MITM for the TON-IoT dataset, due to the scarcity of instances from the C&C attack (i.e., 23,984), the CNN may have limited exposure to its patterns and characteristics during training. With the BoT-IoT dataset in Table 5, we got exceptional results with accuracy being 100% with all the attack types and

average results for Denial-of-Service Attacks with moderate recall and F1-scores percentages of 77.32% and 86.42%, respectively. By analyzing this dataset we identified issues with data similarity between DDoS and DoS. This led to misclassification and subsequently refers to situations where our CNN model struggled to differentiate between instances that share similar characteristics. This similarity led to incorrect predictions and lower performance in terms of recall and F1-score. We were able to detect these data similarly by using a heat map that allows to visually inspect and identify misclassification patterns caused by class similarity in ML and DL models. Table 6 shows the results obtained for the CIC IoT 2023 dataset, which is the most recently published dataset by the University of New Brunswick, Canada. Our CNN model returns high-accuracy performance throughout all the attacks involved in this dataset. In particular, the highest accuracy that the CNN model got with this dataset is 99.85% with the least being 85.81%. For the CIC IoMT 2024 dataset, as shown in Table 7, the accuracy exceeds 99% for most attack types, such as MQTT and Reconnaissance, which achieved F1-scores of 99.27% and 96.75%, respectively. The Normal class also performed well with an accuracy of 99.86% and an F1-score of 96.83%. However, the model struggled with DDoS and DoS attacks, where accuracy dropped to around 89%. The DDoS attack, in particular, showed a significant variance with a recall of 98.29% and an F1-score of 86.43%, indicating possible issues in balancing precision and recall for this attack type. This discrepancy suggests that while the model is good at detecting DDoS attacks, it might also be prone to some false positives, leading to lower precision. The DoS attack had better overall performance with an F1-score of 96.48%, but the lower accuracy compared to other classes indicates room for improvement in handling these types of attacks. Table 8 depicts the results for the RT-IoT 2022 dataset with the highest accuracy recorded at 99.91% for DDoS attacks and perfect precision. Normal and Spoofing attacks also showed robust results with accuracy rates of 99.21% and 99.30%, respectively. Spoofing had a slightly lower F1-score of 94.39%, mainly due to a recall of 93.76%. This indicates that while the model is highly accurate, it might occasionally miss some spoofing instances. In Table 9, the results for the EdgeIIoT dataset are presented. Our model maintained high accuracy across most attack types, with the Normal class achieving a near-perfect accuracy of 99.97% and an F1-score of 98.54%. However, MITM attacks presented a significant challenge, with a slight drop in the F1-score to 85.29% and accuracy to 89.28%, despite a high precision of 99.96%. Other attack types such as DDoS, Information Gathering, Injection, and Malware demonstrated robust performance, with accuracy rates above 87%, underscoring the model's efficacy in detecting a wide range of cyber threats within the EdgeIIoT dataset.

The overall comparative evaluation of our CNN model based on multiple IoT datasets is depicted in Table 10. For this study, we prioritized the accuracy of our CNN model's predictions to accurately classify the different attack types across the seven datasets. Our CNN model obtained an impressive average accuracy rating of 99.63% for the TON-IoT dataset, approximately verging on 461,000 records, a 99.99% accuracy rating for the IoT-23 dataset, approximately verging on 325.3 million records, and a 100.00% accuracy rating for the BoT-IoT dataset, approximately verging on 73 million records. Despite the high accuracy, the CIC IoT 2023 dataset showed a lower accuracy of 93.96% over approximately 46.7 million records. This lower accuracy can be attributed to the complexity and diversity of the attacks included in this dataset. For the CIC IoMT 2024 dataset, the model achieved an accuracy of 95.63%, demonstrating strong performance. The RT-IoT 2022 dataset showed very high accuracy rates of 99.47%. The EdgeIIoT dataset presented mixed results, with an accuracy of 89.12%. For this dataset, MITM attacks presented a significant challenge, with a substantial drop in the F1-score to 75.29%, despite a high precision of 99.96%. This might be caused by the small sample size of this type of attack (i.e., 1,214.) In sum, Table 10 indicates that our model performed satisfactorily altogether and the BoT-IoT dataset outperformed all the datasets in terms of evaluation measures with accuracy being perfect 100%. The overall average performance of our

Table 10. Overall Results for the Seven IoT Datasets

| CA | A (%) | P (%) | R (%) | F1 (%) |
|---|---|---|---|---|
| TON-IoT | 99.63 | 96.48 | 93.18 | 94.45 |
| IoT-23 | 99.99 | 99.36 | 93.37 | 94.83 |
| BoT-IoT | 100.00 | 98.82 | 95.19 | 96.60 |
| CIC IoT 2023 | 93.96 | 84.19 | 75.88 | 77.63 |
| CIC IoMT 2024 | 95.63 | 94.88 | 96.82 | 95.16 |
| RT-IoT 2022 | 99.47 | 97.37 | 97.26 | 97.31 |
| EdgeIIoT | 92.45 | 98.00 | 94.28 | 92.84 |
| Average | 97.31 | 95.59 | 92.43 | 92.69 |

A, Accuracy; P, Precision; R Recall; F1, F1-Score.

CNN model across all evaluated datasets resulted in an accuracy of 96.83%, precision of 95.59%, recall of 92.43%, and F1-score of 92.45%. This comprehensive evaluation underscores the efficacy of our CNN model in handling diverse IoT datasets. By prioritizing accuracy and refining our model, we aim to enhance its capability to assess network traffic data accurately and effectively identify malicious activities.

### 4.4 FL Performance (RQ3)

In this section, we answer RQ3 by assessing the performance of our FL system by running two sets of experiments. In our experiments, we assumed that the clients were in four different geographic locations. In particular, we set the communication latencies for each one of the four clients as follows: $C1$: 0.6 s, $C2$: 3 s, $C3$: 5 s, and $C4$: 6 s. The communication latencies were selected randomly to simulate that the clients are located in four different geographical locations. For both experiments, we analyze and compare the performance of our FL system with two implementation settings: (a) our solution without using any one of the three PP techniques (i.e., Table 11), and our solution using the three PP techniques (i.e., see Table 12).

*Experiment 1.* In the first experiment, we compute two evaluation metrics: the *CCT*, which indicates how long the training took for a given client iteration, and the *TFD*, which takes into account client-defined communication latencies between the clients and the global FL. In Tables 11 and 12, fourth and fifth columns from the left, we present the CCT and the TFD as the time when each client and server accuracies converged. Here the I# indicates the number of iterations at which convergence happened. The above section of Table 12 indicates the FL scheme evaluation with PP in place and Table 11 shows the results without PP. It can be observed that throughout the two tables the average CCT and TFD time measures a little higher for the FL scheme with the PP techniques involved rather than without PP. For instance, for the TON-IoT dataset, the average CCT is 7.9 s without PP and 10.0 s with PP which indicates that convergence occurred 2.1 s faster (27%) when PP was not implemented and similarly average TFD without PP was 3.9 s faster (12%) than with PP. For the IoT-23 dataset, the average CCT is 15:45.6 s without PP and 16:20.5 s with PP which indicates that convergence occurred 34.9 s faster (4%) when PP was not implemented and similarly average TFD without PP was 35.6 s faster (4%) than with PP. For the BoT-IoT dataset, the average CCT is 9:45.2 s without PP and 10:30.2 s with PP which indicates that convergence occurred 45 s faster (8%) when PP was not implemented and similarly average TFD without PP was 46.8 s faster (8%) than with PP. For the CIC IoT 2023 dataset, the average CCT is 1:43.0 s without PP and 1:54.8 s with PP which indicates that convergence occurred 11.8 s faster (11%) when PP was not implemented and similarly average TFD without PP was 13.8 s faster (11%) than with PP. The

Table 11. Performance without PP Techniques

| | | | FL | | CNN | |
|---|---|---|---|---|---|---|
| **FL#** | **Dataset** | **I#** | **CCT** | **TFD** | **L-A (%)** | **F-A (%)** |
| $C_1$ | TON-IoT | 9 | 7.2 s | 27.8 s | 98.24 | 98.34 |
| $C_2$ | | 9 | 7.2 s | 30.2 s | 98.15 | 98.34 |
| $C_3$ | | 9 | 7.2 s | 37.2 s | 98.03 | 98.34 |
| $C_4$ | | 10 | 10.2 s | 35.0 s | 98.23 | 98.21 |
| Average | | 9.25 | 7.9 s | 32.6 s | 98.16 | 98.31 |
| $C_1$ | IoT-23 | 4 | 20:03.0 s | 20:23.6 s | 99.98 | 99.99 |
| $C_2$ | | 7 | 17:06.3 s | 17:29.3 s | 99.99 | 99.99 |
| $C_3$ | | 7 | 17:06.3 s | 17:36.3 s | 99.99 | 99.99 |
| $C_4$ | | 3 | 08:46.5 s | 09:11.5 s | 99.99 | 99.98 |
| Average | | 5.25 | 15:45.6 s | 16:10.2 s | 99.99 | 99.99 |
| $C_1$ | BoT-IoT | 2 | 9:45.2 s | 10:05.8 s | 99.99 | 99.99 |
| $C_2$ | | 2 | 9:45.2 s | 10:08.2 s | 99.99 | 99.99 |
| $C_3$ | | 2 | 9:45.2 s | 10:15.2 s | 99.99 | 99.99 |
| $C_4$ | | 2 | 9:45.2 s | 10:10.2 s | 99.99 | 99.99 |
| Average | | 2 | 9:45.2 s | 10:09.8 s | 99.99 | 99.99 |
| $C_1$ | CIC IoT 2023 | 4 | 1:16.1 s | 1:36.7 s | 82.60 | 81.23 |
| $C_2$ | | 2 | 2:09.8 s | 2:32.8 s | 81.12 | 80.01 |
| $C_3$ | | 4 | 1:16.1 s | 1:46.1 s | 82.40 | 81.23 |
| $C_4$ | | 2 | 2:09.8 s | 2:34.8 s | 80.04 | 80.01 |
| Average | | 3 | 1:43 s | 2:07.6 s | 81.54 | 80.62 |
| $C_1$ | CIC IoMT 2024 | 3 | 3.57 s | 3.18 s | 87.77 | 87.57 |
| $C_2$ | | 2 | 4.74 s | 4.57 s | 77.07 | 76.87 |
| $C_3$ | | 4 | 5.13 s | 5.13 s | 88.12 | 88.14 |
| $C_4$ | | 4 | 7.43 s | 8.08 s | 88.33 | 88.14 |
| Average | | 3.25 | 5.22 s | 5.24 s | 87.82 | 87.68 |
| $C_1$ | RT-IoT 2022 | 7 | 22.32 s | 53.43 s | 98.43 | 98.18 |
| $C_2$ | | 6 | 23.88 s | 55.3 s | 98.73 | 98.23 |
| $C_3$ | | 7 | 22.35 s | 55.92 s | 98.74 | 98.24 |
| $C_4$ | | 8 | 26.87 s | 57.76 s | 98.16 | 98.22 |
| Average | | 7 | 23.85 s | 55.59 s | 98.5 | 98.22 |
| $C_1$ | EdgeIIoT | 6 | 44.91 s | 55.03 s | 99.79 | 99.84 |
| $C_2$ | | 2 | 1:06 s | 1:29.3 s | 99.67 | 99.27 |
| $C_3$ | | 3 | 59.43 s | 1:29.12 s | 99.77 | 99.67 |
| $C_4$ | | 6 | 54.43 s | 59.98 s | 99.87 | 99.84 |
| Average | | 4.25 | 54.70 s | 1:13.36 s | 99.77 | 99.66 |

FL, Federated Learning; CNN, Convolutional Neural Network; CCT, client computation time; TFD, time to receive federated data; L-A, local accuracy; F-A, federated accuracy; I, iteration.

final CCT average increase of CCT and TFD for the FL implementing the three PP techniques is of 12% and 8%, respectively.

Table 12. Performance with PP Techniques

| | | | FL | | CNN | |
|---|---|---|---|---|---|---|
| **FL#** | **Dataset** | **I#** | **CCT** | **TFD** | **L-A (%)** | **F-A (%)** |
| $C_1$ | TON-IoT | 9 | 9.3 s | 31.8 s | 98.06 | 98.26 |
| $C_2$ | | 8 | 12.5 s | 36.8 s | 98.09 | 98.03 |
| $C_3$ | | 9 | 9.6 s | 41.2 s | 98.02 | 98.26 |
| $C_4$ | | 9 | 8.6 s | 36.2 s | 98.2 | 98.26 |
| Average | | 8.75 | 10.0 s | 36.5 s | 98.09 | 98.20 |
| $C_1$ | IoT-23 | 7 | 24:33.4 s | 24:54.7 s | 99.99 | 99.99 |
| $C_2$ | | 2 | 8:07.0 s | 08:31.2 s | 99.98 | 99.97 |
| $C_3$ | | 7 | 24:33.7 s | 25:04.1 s | 99.99 | 99.99 |
| $C_4$ | | 2 | 8:07.8 s | 8:33.2 s | 99.98 | 99.97 |
| Average | | 4.5 | 16:20.5 s | 16:45.8 s | 99.98 | 99.98 |
| $C_1$ | BoT-IoT | 3 | 10:30.1 | 10:52.6 s | 99.99 | 99.99 |
| $C_2$ | | 3 | 10:30.4 s | 10:55.0 s | 99.99 | 99.99 |
| $C_3$ | | 3 | 10:30.6 s | 11:02.0 s | 99.99 | 99.99 |
| $C_4$ | | 3 | 10:29.9 s | 10:57.0 s | 99.99 | 99.99 |
| Average | | 3 | 10:30.2 s | 10:56.6 s | 99.99 | 99.99 |
| $C_1$ | CIC IoT 2023 | 4 | 1:47.1 s | 2:09.9 s | 81.97 | 81.82 |
| $C_2$ | | 3 | 2:17.6 s | 2:42.2 s | 81.49 | 81.44 |
| $C_3$ | | 4 | 1:47.4 s | 2:19.3 s | 81.37 | 81.82 |
| $C_4$ | | 4 | 1:47.0 s | 2:14.3 s | 82.81 | 81.82 |
| Average | | 3.75 | 1:54.8 s | 2:21.4 s | 81.91 | 81.72 |
| $C_1$ | CIC IoMT 2024 | 4 | 3.34 s | 3.58 s | 86.4 | 87.44 |
| $C_2$ | | 2 | 4.40 s | 4.11 s | 76.96 | 75.74 |
| $C_3$ | | 4 | 5.34 s | 6.07 s | 87.14 | 87.44 |
| $C_4$ | | 3 | 7.42 s | 8.11 s | 87.6 | 87.58 |
| Average | | 3.25 | 5.87 s | 5.46 s | 87.72 | 87.05 |
| $C_1$ | RT-IoT 2022 | 5 | 25.76 s | 56.43 s | 98.07 | 98.20 |
| $C_2$ | | 8 | 27.2 s | 55.98 s | 98.84 | 98.34 |
| $C_3$ | | 8 | 22.54 s | 1:02.91 s | 98.28 | 98.31 |
| $C_4$ | | 7 | 28.98 s | 57.89 s | 98.44 | 98.03 |
| Average | | 7 | 26.12 s | 56 s | 98.41 | 98.22 |
| $C_1$ | EdgeIIoT | 3 | 42.12 s | 59.12 s | 99.73 | 99.87 |
| $C_2$ | | 2 | 1:07.41 s | 1:39.52 s | 99.6 | 99.31 |
| $C_3$ | | 2 | 1:10.61 s | 1:36.98 s | 99.46 | 99.34 |
| $C_4$ | | 3 | 44.55 s | 1:03 s | 99.77 | 99.8 |
| Average | | 2.5 | 58.3 s | 1:16.41 s | 99.64 | 99.55 |

FL, Federated Learning; CNN, Convolutional Neural Network; CCT, client computation time; TFD, time to receive federated data; L-A, local accuracy; F-A, federated accuracy; I, iteration.

For the CIC IoMT 2024 dataset, the average CCT without PP is 5.22 s, whereas with PP it is 5.87 s. This shows a notable increase of 0.65 s (11%) when PP techniques are applied. The average TFD without PP is 5.24 s, and with PP, it is 5.46 s. This indicates a reduction in TFD with PP by 0.22

Table 13. Comparing AI-Based Solutions Tested on the TON-IoT Dataset

| Reference | Year | Model | Sampling | Classification | A (%) |
|---|---|---|---|---|---|
| Al-Hawawreh et al. [5] | 2020 | DSAE, GRNN | None | Multi | 99.85 |
| Lo et al. [48] | 2022 | GNN | None | Binary, Multi | 97.87 |
| Sangeetha et al. [71] | 2022 | DT | None | Binary | 98.10 |
| Baz [11] | 2022 | MLP, ReNN | None | Binary, Multi | 99.90 |
| Gad et al. [28] | 2022 | XGBoost | SMOTE, Chi-square | Binary, Multi | 100.00 |
| This article | 2024 | CNN 1D | None | Multi | 99.63 |

A, Accuracy.

s (4%). In the case of the RT-IoT 2022 dataset, the average CCT without PP is 23.85 s, and with PP, it is 26.12 s, which indicates that convergence occurred 2.27 s faster (9%) when PP was not implemented. The average TFD without PP is 55.59 s and with PP, it is 58.3 s, showing a 2.71 s (5%) faster convergence without PP. For the EdgeIIoT dataset, the average CCT without PP is 54.70 s, while with PP it is 56.17 s, indicating that convergence happened 1.47 s faster (3%) without PP. The average TFD without PP is 1:13.36 s, and with PP, it is 1:16.41 s, showing a 3.05 s (4%) faster convergence without PP.

In sum, we can say that there is an increase of 10% of the computation time when the FL involves the three PP techniques. The results show that for all seven datasets, the clients C1, C2, C3, and C4 encounter convergence quickly in FL schemes without PP when compared on the basis of CCT and TFD.

*Experiment 2.* In the second experiment, we calculate the local accuracy (L-A), which refers to the accuracy of our CNN model trained on a local dataset, and federated accuracy (F-A), which refers to the overall accuracy of the FL system, taking into account the contributions of all clients. Tables 11 and 12 show the local accuracy L-A (%) and the federated accuracy F-A (%) (see sixth and seventh columns from the left) for the four different clients with different datasets. The change in the accuracies throughout the tables is negligible within the two experiments (with and without PP techniques). For the BoT-IoT dataset, the accuracies are exactly the same and for the CIC IoT 2023 dataset, the accuracies differ with not more than around a 1% of the gap.

Overall, our FL solution is comparatively 10% faster without the PP techniques involved and almost unnoticeable changes appeared in the local and federated accuracies when PP was applied.

### 4.5 Comparison with Other IDSs (RQ4)

In this section, we answer RQ4 by comparing a set of previously published AI-based IDSs that were tested on the seven datasets used in this article against our solution. When possible, for each dataset, we selected the 5 best studies (in terms of intrusion detection accuracy performance) that we were able to find, with a total of 23 studies to carry out our comparison study. Each one of these studies was selected because they met all the following five criteria: (1) it is tested on one of the seven datasets involved in this article, (2) it implements a DL technique, (3) the description of the solution is published in a peer-reviewed full journal or conference article, (4) it provides a complete empirical evaluation of the presented solution, and (5) it uses 100% of the dataset.

*TON-IoT Dataset.* Table 13 shows the five studies [5, 11, 28, 48, 71] using the TON-IoT dataset that was compared to our solution. Al-Hawawreh et al. [5] suggested a TI system that can identify cyber threats from Space, Air, Ground, and Sea networks. To identify malicious events' patterns the **Deep Sparse Autoencoder (DSAE)** was used and **Gated Recurrent Neural Network (GRNN)** was used for the identification of attack type of malicious patterns. This work implemented a multiclassification algorithm and no sampling techniques were applied to the dataset. Their solution ultimately

Table 14. Comparing AI-Based Solutions Tested on the IoT-23 Dataset

| Reference | Year | Model | Sampling | Classification | A (%) |
|---|---|---|---|---|---|
| Dutta et al. [21] | 2020 | DNN, LSTM, LR | SMOTE, ENN | Binary | 99.70 |
| Ullah and Mahmoud [82] | 2021 | CNN 2D | RFE | Multi | 99.90 |
| Abdalgawad et al. [1] | 2021 | BiGAN | Undersample, SMOTE | Multi | 97.00 |
| Ullah and Mahmoud [83] | 2022 | FFN | None | Multi | 99.37 |
| Bowen et al. [17] | 2023 | CNN 1D, BLSTM | None | Multi | 100 |
| This article | 2024 | CNN 1D | None | Multi | 99.99 |

LR, Logistic regression; A, Accuracy.

returned an accuracy of 99.85%. Lo et al. [48] proposed and constructed an approach named E-GraphSAGE, a **Graph Neural Networks (GNN)**-based Network Intrusion Detection System that provided attack flow detection for IoT network intrusion detection. This system incorporated binary and multiple class classifications. This solution did not use any specific sampling technique and achieved an accuracy of 97.87%. Sangeetha et al. [71] proposed a multilayered neural network security method based on a **Decision Tree (DT)** classifier to safeguard the transport layer of IoT networks. A binary classification algorithm was applied and did not use any sampling technique. The accuracy attained was 98.10%. Baz [11] suggested a **Self Evolving Host-Based Intrusion Detection System (SEHIDS)**. Binary and multi-classification was implemented to separate benign from malicious traffic using **Replicator Neural Network (ReNN)** and to categorize various types of attacks using **Multilayer Perceptron (MLP)**, respectively, and achieved an accuracy of 99.99%. In order to identify attacks on the IoT and prevent malicious activity, Gad et al. [28] suggested a novel distributed detection system based on ML techniques. The most crucial variables were chosen using the Chi-square and correlation matrix. The **Synthetic Sensors Minority Over-Sampling Technique (SMOTE)** sampling method was used to resolve the issue of class imbalance. In order to find intrusions, the architecture used a multiclass classification method in addition to a binary classification strategy. The results showed that **Extreme Gradient Boosting (XGBoost)** beats all other ML techniques with accuracy being 100%.

*IoT-23 Dataset.* Table 14 shows the five studies [1, 17, 21, 82, 83] using the IoT-23 dataset that were compared to our solution. Dutta et al. [21] present an ensemble method that leverages deep models such as the DNN and LSTM and a meta-classifier (i.e., **Logistic Regression (LR)**) following the principle of stacked generalization. The authors use SMOTE and **Edited Nearest Neighbors (ENN)** to balance the dataset. Ullah and Mahmoud [82] discuss a DL-based model for anomaly detection in IoT networks that implements a CNN in 1D, 2D, and 3D. Table 14 reports only CNN 2D because it is the model that provides the best accuracy. The authors used a feature selection technique called **Recursive Feature Elimination (RFE)** to extract relevant features from the dataset. Abdalgawad et al. [1] use generative DL methods like **Bidirectional Generative Adversarial Networks (BiGAN)** to detect intruders based on an analysis of the network data. The authors used a combination of Random Under-Sampling and SMOTE to under-sample the majority class while oversampling the minority classes. The authors implement binary classification: normal and malware (zero, one) for their approach. Ullah and Mahmoud [83] present the design and development of an anomalous activity detection system for IoT networks based on flow and control flags features using an FNN. Bowen et al. [17] propose a hybrid DL model that combines

Table 15. Comparing AI-Based Solutions Tested on the BoT-IoT Dataset

| Reference | Year | Model | Sampling | Classification | A (%) |
|---|---|---|---|---|---|
| Ullah and Mahmoud [82] | 2021 | CNN 1D | RFE | Multi | 99.97 |
| Baz [11] | 2022 | MLP, ReNN | None | Binary, Multi | 100.00 |
| Sangeetha et al. [71] | 2022 | DT | None | Binary | 98.10 |
| Ullah and Mahmoud [83] | 2022 | FFN | None | Multi | 99.86 |
| Sabitha et al. [70] | 2023 | HNN | None | Binary | 99.80 |
| This article | 2024 | CNN 1D | None | Multi | 100.00 |

A, Accuracy.

CNN and **Bidirectional Long Short-Term Memory (BLSTM)** layers. CNN allows the IDS to recognize patterns in the features of the network data in a fast computation time. The results are sent to two BLSTM layers, which capitalize on the forward and backward propagation of data to identify malicious traffic.

*BoT-IoT Dataset.* Table 15 shows the five studies [11, 70, 71, 82, 83] using the BoT-IoT dataset that were compared to our solution. The CNN (1D, 2D, and 3D) presented by Ullah and Mahmoud [82] was also tested on the BoT-IoT dataset. Table 14 reports only the CNN 1D because it is the model that provides the best accuracy. The FNN-based IDS proposed by Ullah and Mahmoud [83] was evaluated also on the BoT-IoT dataset. The effectiveness of SEHIDS was thoroughly evaluated in Baz [11] using three recent datasets (one of which was BoT-IoT) that included a wide range of cyber attacks directed at IoT networks. These evaluations were made with two goals—first, determining how well SEHIDS can identify various sorts of attacks, and second, determining how well SEHIDS can operate on devices with limited resources. Sangeetha et al. [71] established a baseline with intrusion detection datasets like BoT-IoT, and DL methods were utilized to monitor the IoT network in order to label activities as "normal" or "malware" for each tier of the architecture. The adaptive and intelligent **Artificial Immune System (AIS)**, a novel system model, mimics human function. Therefore, a system with the highest accuracy and lowest false alert rate was needed. In the study by Sabitha et al. [70], the **Hopfield Neural Network (HNN)** was used by the AIS model for classification. Since the HNN classifier's back-propagation feature prevents it from being altered, its weight is fixed. The author used Fast-Particle Swarm Optimization to determine the fixed weight in the best possible way, improving the HNN classifier's accuracy in the process. After that, their classifier model distinguished between IoT attacks with a high detection rate and normal signal.

*CIC IoT 2023 Dataset.* To the best of our knowledge, the study by Neto et al. [61] is the only research paper available involving ML/DL IDS trained and tested over the CIC IoT 2023 dataset. This study describes the ML evaluation process utilized and provided examples to show how the CIC IoT 2023 dataset may be used to train ML-based attack detection and classification techniques. In order to categorize and identify malicious or benign IoT network traffic, the authors tested the effectiveness of ML and DL algorithms using the CIC IoT 2023 dataset. They merged all of the generated datasets first. In this way, malicious and benign traffic was combined and shuffled into a single dataset (i.e., Blended Dataset). After integrating the data, they assessed the performance of the ML system from three different angles: (i) Multiclass classification, which focused on categorizing 33 individual attacks; (ii) Grouped classification, which took into account 7 attack groups (such as DDoS and DoS); and (iii) Binary classification, which distinguished between malicious and benign attacks. Five ML techniques—LR, Perceptron, Adaboost, Random Forest, and DNN—that have been effectively used in a variety of applications, including cybersecurity, were used in the evaluation process. The findings demonstrated that all techniques exhibited great performance for binary

Table 16. Comparing AI-Based Solutions Tested on the EdgeIIoT Dataset

| Reference | Year | Model | Sampling | Classification | A (%) |
|---|---|---|---|---|---|
| Bhandari et al. [12] | 2023 | DNN | None | Multi | 93 |
| Friha et al. [26] | 2023 | DNN | None | Multi | 84.64 |
| Li et al. [40] | 2023 | CNN, LSTM, TCN | None | Multi | 93.56 |
| Abdulkareem et al. [2] | 2024 | SEL | None | Multi | 89.05 |
| Jeffrey et al. [38] | 2024 | LR, NB, SVM, MLP | None | Binary, Multi | 93.19 |
| This article | 2024 | CNN 1D | None | Multi | 92.45 |

SVM, Support vector machine; A, Accuracy.

classification with all techniques achieving accuracy levels of over 98%. But only Random Forest and DNNs sustained high accuracy above 98% in the classification of attack groups and multiclass classification of individual attacks.

*CIC IoMT 2024 Dataset.* To the best of our knowledge, no research papers have been published in any peer-reviewed venue—including workshops, conferences, or journals—that utilize the CIC IoMT 2024 dataset.

*RT-IoT 2022 Dataset.* To the best of our knowledge, the research papers by Sharmila and Nagapadma [73] and by Airlangga [4] are the only research papers available involving ML/DL IDS trained and tested over the RT-IoT 2022 dataset. The study by Sharmila and Nagapadma [73] presents an AE algorithm for attack detection and classification in resource-constrained IoT devices. In particular, **Quantized Auto-Encoder (QAE)** models, specifically QAE-u8 and QAE-f16, were developed to optimize the deployment of computationally expensive AI models in IoT edge devices. The performance was evaluated using several metrics: accuracy at 98.40%, precision at 98.40%, recall at 98.59%, and an F1-score of 99.19%. The work by Airlangga [4] proposes a novel IDS framework that leverages ML techniques specifically designed to address the unique challenges of IoT environments. It includes strategies for handling high-dimensional data, overcoming data imbalance, and ensuring model scalability and real-time detection capabilities. This study provides empirical evidence of the proposed IDS's effectiveness, utilizing a comprehensive set of performance metrics such as accuracy, precision, recall, and F1-score, achieving 96% across all metrics using LR.

*EdgeIIoT Dataset.* Table 16 shows the five studies [12, 26, 40] using the EdgeIIoT dataset. that were compared to our solution. Bhandari et al. [12] introduce a new approach to discover attacks and malware on the IoT devices using an AI-enabled approach. This approach is meant to facilitate live tracking of the streamed network traffic for the detection of malware and attacks and to help locating the security issues and the concerned affected devices that assist in reducing the maintenance effort. Li et al. [40] propose a **Federated Sequential Learning (FSL)** framework for detection of cyber attacks under real-world IIoT, where FSL can extract time-series features from sensory signals from real industrial application, while reducing the degradation of model accuracy due to the node heterogeneity. Friha et al. [26] present a secure, decentralized, and DP FL-based IDS, for securing smart industrial facilities. The proposed 2DF-IDS comprises three building blocks, namely: a key exchange protocol (for securing the communicated weights among all peers in the system), a differentially private gradient exchange scheme (achieve improved privacy of the FL approach), and a decentralized FL approach (that mitigates the single point of failure/attack risk associated with the aggregation server in the conventional FL approach). Abdulkareem et al. [2] introduce FI-SEL, a lightweight ensemble learning framework for attack detection in IoT/IIoT

networks. FI-SEL utilizes Feature Importance for feature dimension reduction, selecting the most crucial features from high-volume IoT traffic data. The SEL classifier then learns from these reduced feature sets to differentiate between attack and normal network traffic. The DT within SEL is analyzed, showing how it improves performance and enhances transparency and explainability in decision-making over single learners. Jeffrey et al. [38] propose a novel ensemble learning-based hybrid anomaly detection method comprised of signature-based detection for known threats, threshold-based metrics for the immutable physical characteristics of a CPS, combined with an ensemble-based learning model for behavior-based anomaly detection.

In this section, we have conducted an extensive comparative evaluations against the existing literature, utilizing as benchmark the seven datasets studied in this article and the accuracy as our evaluation metric. In particular, we discuss the performance of our CNN-based solution, which demonstrates superior or comparable accuracy performance compared to the existing literature in the field. This improvement is a result of innovative architectural design choices, optimization techniques, and advancements in training methodologies, thanks to our FL scheme.

## 5 Discussion

This research work proposes an FL IDS that leverages a 1D CNN for enhanced PP. To address privacy concerns, we implemented three techniques: DP, DK exchange, and HE. Our solution was evaluated on seven publicly available IoT datasets: TON-IoT, IoT-23, BoT-IoT, CIC IoT 2023, CIC IoMT 2024, RT-IoT 2022, and EdgeIIoT. The results showcase excellent average accuracy, precision, recall, and F1-score (97.31%, 95.59%, 92.43%, and 92.69%, respectively) across these datasets. Additionally, we observed that incorporating all three PP techniques incurred only a minimal 10% increase in computation time compared to the FL system without any privacy-preserving mechanisms.

The integration of a 1D CNN in our FL IDS yielded remarkable results, outperforming previous the majority of approaches in the field (see Section 4). The use of CNNs allows for effective detection of intrusion patterns by capturing relevant features and leveraging the temporal information in the 1D input data. The achieved high accuracy, precision, recall, and F1-score signify the efficacy of our system in identifying intrusions accurately while minimizing false positives and false negatives. These results demonstrate the potential of our FL-based approach for addressing intrusion detection challenges in IoT environments. We believe that the premise behind this shared training model has applications in government bodies, medical facilities, and other places where sensitive personal information has a regular need to be shared with external sources, but regulations or general practice prohibit sharing insecurely. In particular, our solution could be implemented for smart-healthcare networks to protect confidential data, Electronic Health Record Management, Medical Image Processing, Remote Health Monitoring, and other multiple healthcare uses [7], agriculture IoT systems [27], identify smart grid energy theft [85], determine cyber attacks in digital forensics [49] and for various other areas where real-time privacy of data is required; this research work can be a motivation.

The implementation of the three PP techniques is crucial in FL to protect the sensitive data of individual participants while still enabling collaborative model training. We employed DP to introduce private noise to the training data, ensuring that no single participant's data can be easily distinguished. This technique adds an additional layer of privacy protection to mitigate the risk of data reconstruction attacks. By incorporating DK exchange, we securely exchanged cryptographic keys over the FL network, ensuring that the keys remain concealed from potential eavesdroppers. This safeguarded key exchange protocol enhances the security of the FL system, preventing unauthorized access to sensitive information. Additionally, HE was employed to perform

computations on the encrypted training data. This technique enables secure and privacy-preserving computation while maintaining the confidentiality of the data.

Our experiments demonstrated that the implementation of all three PP techniques incurred only a minimal 10% increase in computation time compared to the FL system without any privacy-preserving mechanisms. This efficiency is crucial for real-world deployment, as it ensures that the overhead introduced by the privacy techniques does not hinder the practicality and scalability of the system. The modest increase in computation time indicates the feasibility and viability of our approach, enabling PP without sacrificing system performance.

The use of publicly available IoT datasets in our evaluation ensures the relevance and applicability of our FL IDS. By testing our solution on TON-IoT, IoT-23, BoT-IoT, CIC IoT 2023, CIC IoMT 2024, RT-IoT 2022, and EdgeIIoT datasets, which represent diverse IoT environments and intrusion scenarios, we validate the robustness and generalization capabilities of our system. The achieved high average accuracy, precision, recall, and F1-score across these datasets demonstrate the effectiveness of our approach in detecting intrusions across various IoT domains.

### 5.1 Effort Needed for Deployment and Maintenance

Deploying and maintaining IDS in real-world IoT networks, particularly on resource-limited devices, is a significant challenge [77]. Our CNN-based FL IDS has been designed with flexibility and scalability to address these practical concerns. The architecture allows for the configuration of the number of clients and iterations according to specific requirements and capabilities of the deployment environment. This configurability ensures that the system can adapt to various scales of IoT networks, thereby not overburdening any single device. Efficient resource management strategies are crucial for the successful deployment of our approach. Techniques such as edge computing [67] can be employed to offload heavy computational tasks from resource-constrained IoT devices to more capable edge servers. This approach can help manage the computational load without compromising the performance of the IoT devices. Additionally, our method of multiprocessing, coupled with the use of locks and barriers [75], can potentially minimize resource contention, ensuring system stability and performance. In addition, automating the deployment and maintenance processes can significantly reduce the effort required to manage the IDS. The use of tools like containerization (e.g., Docker[10]) and orchestration (e.g., Kubernetes[11]) can facilitate automated deployments, updates, and scaling of the system. This can make it easier to manage even in complex IoT environments. Finally, continuous monitoring and updating of security protocols are necessary to ensure that the IDS remains robust against emerging threats. Finally, regular updates to the privacy-preserving mechanisms are essential to adapt to evolving attack vectors.

### 5.2 Complexity and Computational Overhead

The integration of multiple PP techniques adds a layer of complexity to the system. However, we believe that this complexity is justified by the enhanced security and privacy it provides. Each component in our system is modular, allowing for individual components to be updated, maintained, or replaced without disrupting the entire system. While there is a 10% increase in computational overhead, this is a tradeoff for the significant privacy and security benefits offered. We have optimized our implementation to minimize this overhead, ensuring the system's viability for real-time applications. In scenarios where real-time performance is critical, additional optimizations can be applied. These include parallel processing, hardware acceleration (e.g., using GPUs), and efficient algorithmic implementations to further reduce latency. Our current focus is on IoT networks, but the principles and techniques applied in our solution can be extended to other domains, such as

[10] https://www.docker.com/
[11] https://kubernetes.io/

industrial control systems and autonomous vehicles. Future work will explore these applications, tailoring the system to meet the specific requirements and constraints of each domain. For instance, in industrial control systems, where reliability and low latency are crucial, additional optimizations and customizations will be implemented to ensure the IDS meets stringent performance criteria. In conclusion, we recognize the necessity of balancing security, privacy, and system performance. Our approach, while introducing some complexity and computational overhead, is designed with scalability, flexibility, and efficiency in mind to ensure its feasibility in real-world IoT deployments.

### 5.3 Future Work Directions

As follows, we present a set of potential future work directions for the research work presented in this article:

- *Performance Evaluation on Additional IoT Datasets.* While the proposed solution has been trained and tested on seven publicly available IoT datasets, further evaluation on additional datasets from diverse IoT environments would enhance the generalizability and robustness of the system. Including datasets with different network topologies, attack scenarios, and data distributions would provide a more comprehensive evaluation of the system's ability to handle diverse IoT environments, thereby improving its overall reliability and applicability in real-world scenarios.
- *Scalability Analysis.* We plan to investigate the scalability of the proposed system by exploring its performance with larger datasets, increasing the number of participating clients, or handling more complex intrusion detection models. Assessing the computational and communication overhead under various scalability scenarios will be crucial for determining the system's feasibility and practicality in real-world deployments.
- *Optimization of Privacy-Preserving Techniques.* We will explore techniques to optimize the PP techniques, in terms of computational efficiency and resource utilization, and investigate alternative privacy techniques or algorithmic improvements that can maintain privacy while minimizing the impact on system performance.
- *Extend Solution to Other Domains.* We have plans to extend our solution to other domains such as **Supervisory Control and Data Acquisition (SCADA)** systems and Autonomous vehicles. In light of our recent study on SCADA systems [53], we believe this work can significantly enhance SCADA security and intrusion detection. Given the vulnerability of SCADA systems to cyber threats, detecting unauthorized access and malicious activities is crucial. Autonomous vehicles rely on advanced perception systems to interpret their surroundings and make informed decisions. FL is particularly relevant in the context of autonomous vehicles, as it allows models to be trained collaboratively without sharing sensitive data.
- *Identify Real-World Deployment and Evaluation.* We will conduct experiments in real-world environments to assess the effectiveness and practicality of the proposed system, and deploy the FL IDS in IoT networks, SCADA systems, or autonomous vehicles to evaluate its performance, robustness against sophisticated attacks, and adherence to privacy requirements in live settings.

Addressing these future work directions would contribute to the advancement and practical applicability of our FL IDS solution, ensuring its effectiveness, PP, scalability, and suitability for real-world deployment in diverse domains such as IoT, SCADA, and autonomous vehicles.

In sum, our research work presents a robust FL IDS that leverages a 1D CNN and incorporates DP, DK exchange, and HE for PP. The achieved excellent accuracy, precision, recall, and F1-score indicate the superiority of our system in accurately identifying intrusions. Moreover, the minimal increase in computation time demonstrates the feasibility of implementing all three privacy-preserving

techniques in the FL framework. Our findings highlight the potential of our approach for secure and efficient intrusion detection in IoT environments while ensuring privacy and data confidentiality.

## 6 Threats to the Validity

Below, we discuss threats to the validity of our empirical results and what we did to mitigate these threats.

*External Validity.* As for our evaluation of accuracy of our IDS approach (see Section 4), whether the accuracy levels observed would generalize beyond the IoT domain, we noted that certain cybersecurity attacks were rare in the seven datasets used from the IoT domain. Furthermore, we have not yet conducted a multidomain evaluation of our approach. For these reasons, it would be premature to make claims about how our accuracy results would carry over to other datasets from other domains. That said, we believe that the core components of our approach, notably, our CNN-based FL system implementing three PP techniques, provide a versatile basis for the future development of a more broadly secure solution to apply FL. Additionally, while the focus of this article is on IoT networks, we make no claims on the applicability of our solution in other domains such as industrial control systems and autonomous cars to assess its generalizability and effectiveness across different environments.

*Internal Validity.* Bias was an important concern in relation to internal validity. In particular, datasets are critical for the study that we present because they can mislead the results obtained and ultimately our findings. Datasets bias occurs when the training data are not representative of the test data, leading to poor generalization performance. This can happen if the training data are collected from a biased sample or if it does not cover the full range of variation in the test data. In addition, CNNs require large amounts of data to train, and if the dataset is small or unrepresentative, the CNN may not be able to learn the relevant features and patterns in the data. To mitigate all these biases, we decided to train and test our approach over seven well-known IoT datasets. All seven datasets are based on real-world projects. Hence, they are good representatives of real cybersecurity attacks in the IoT domain. We also carefully performed random sampling to avoid possible bias when training our clients' models. Another potential threat to internal validity is that we could have carried out a heavy pre-processing of the datasets that would have reduced the generalizability of the results. This could lead to an overfitting problem when a model is too complex and performs well on the training data but poorly on new, unseen data. This can also happen if the CNN has too many layers, too many filters, or too many neurons in the fully connected layers, or if the training dataset is too small or unrepresentative of the test dataset. To minimize these threats posed by a subjective normalization of the datasets and an over complex model, we carefully explained in detail (a) how we pre-processed each dataset in Section 4, and (b) the design and implementation of our CNN, including how we chose the appropriate model architecture. Another internal validity can be related to possible errors in the code implementation. Implementation errors, such as bugs in the code or incorrect hyperparameter settings, can lead to poor performance or incorrect results. To mitigate this bias, we had the code carefully checked by all the authors.

*Conclusion Validity.* One conclusion validity bias is related to the computational overhead of our solution. Although the research indicates that implementing all three PP techniques incurs only a 10% increase in computation time, the scalability and efficiency of the system should be further explored. The impact of larger datasets, more complex intrusion detection models, or increased numbers of participants on the computational overhead needs to be investigated to ensure the practicality and feasibility of the proposed solution. Furthermore, integrating various privacy-preserving approaches adds complexity, which may make deployment and maintenance in real-world IoT networks difficult. This 10% overhead may still affect real-time application performance for resource-constrained IoT devices. Therefore, a thorough cost-benefit analysis is necessary

Table 17. Related Work

| Ref | Year | Model | Dataset | DP | DK | H | Main Goal |
|---|---|---|---|---|---|---|---|
| [13] | 2006 | N/A | N/A | × | × | × | Discusses the theoretical implementation of a cryptographic technique for FL systems. |
| [66] | 2017 | SGD | MNIST, SVHN | × | × | ✓ | Presents a PP DL system for an FL by using asynchronous stochastic gradient descent and additively HE. |
| [90] | 2020 | N/A | QoS | ✓ | × | × | Proposes a PP QoS prediction approach by leveraging FL techniques and developing reputation mechanisms for mobile edge environments. |
| [24] | 2020 | N/A | Synthetic | × | × | ✓ | Presents a PP tensor approach leveraging HE and federated cloud. |
| [59] | 2020 | LR | MNIST | ✓ | ✓ | × | Proposes a LR-based FL that implement DP and DK PP techniques. |
| [87] | 2021 | DNN | RF-based | × | × | ✓ | Describes a federated ML model based on drones' Radio Frequency features in IoT networks that implement a combination of HEs. |
| [84] | 2021 | NN | Speech, age, face detection | × | × | × | Discuses a blockchain-assisted secure framework for FL in Big Data analytics services for IoT. |
| [60] | 2021 | DT, SVM, LSTM | Hand detection | ✓ | × | ✓ | Proposes a PP hierarchical IoT solution that addresses the utilization of AI-driven IoT and data protection. |
| [18] | 2021 | GRU | Synthetic | ✓ | × | × | Discusses a method based on differentially private FL to predict the probability of subsequent APT attacks occurring in IoT systems. |
| [86] | 2021 | DDQN | Private | ✓ | × | × | Presents an algorithm DDQN based on fully decentralized federated framework (FDFF) with an HE. |
| [89] | 2022 | N/A | T-Drive, Tokyo | ✓ | × | × | Presents an approach to task allocation under geo-indistinguishability via group-based noise addition. |
| [23] | 2024 | RNN | KTH, UCF50 | ✓ | × | × | Presents a differentially private tensor-based RNN (DPTRNN) that can be applied in many DL sequence tasks for IoT systems. |
| This article | 2024 | CNN | Botnet, IoT-23, BoT-IoT, CIC IoT, CIC IoMT, RT-IoT, EdgeIIoT | ✓ | ✓ | ✓ | Proposes a CNN-based FL that implements DP, DK PP, and FHE techniques. |

DDQN, Double deep Q-network; DP, Differential Privacy; DK, Diffie-Hellman Key (DK) Exchange; HE, Homomorphic Encryption.

when considering the implementation of our solution in environments with limited computational resources.

## 7 Related Work

Several research papers presenting new solutions to preserve the privacy of FL environments have been published in recent years. Table 17 provides the comparison of 12 research papers that discuss various FL solutions implementing PP techniques. The first column (*"Reference"*) of the table provides a reference to each study. The second column, *"Year"* indicates the year when the study was published. The third column, *"Model"* shows the AI model used to implement the FL system. The fourth column, *"Dataset"* indicates the name of the dataset used to train the FL system. The fifth, sixth, and seventh columns show whether the three PP techniques implemented in this article are covered in the related work studies, respectively, *"DP"*, *"DK"*, and Homomorphic (*"H"*).

The eighth column provides a summarization of the *"Main Goal"* of each paper. Notice that N/A indicates that given data were not provided in the study. We discuss the selected studies next.

Zhang et al. [90] propose a privacy-preserving QoS prediction approach by leveraging FL techniques and developing reputation mechanisms for mobile edge environments. This method is evaluated on a large-scale real world QoS dataset and the results are shown to be more effective than similar techniques. **Mean Absolute Error (MAE)** is used as an evaluation metric to compare accuracy between methods. Authors evaluate the reliability, accuracy, and efficiency of the privacy-preserving QoS prediction approach through MAE with a large-scale real-world dataset. The evaluation has proven effective and reliable by producing high accuracy while successfully preserving user privacy and tolerating dishonest individuals in the MEC environment.

Xue et al. [86] present a novel algorithm **Double Deep Q-Network (DDQN)** based on a **Fully Decentralized Federated Framework (FDFF)** enabled by an integrated system named SMEC. It provides a stable way to infer real-time treatment policy from large amount of distributed observational **Electronic Medical Records (EMRs)**, and it guarantees the privacy of EMRs with the help of additively HE. This scheme focuses on PP for IoT eHealth and medical records. This scheme is evaluated and proven to be novel and effective. Authors evaluate the time cost of additively HE when applied to FDFF, and give some quantitative analysis on DDQN clinical treatment policy with experiments on large data sets based on FDFF. The evaluation turned out to be very promising for recommending real-time sequential clinical treatment policy for patients when implementing Clinical Decision Support Systems.

Bhargav-Spantzel et al. [13] introduce a cryptographic technique for Federated systems. This technique sets out to preserve identity privacy, implement access control, and to authenticate without jeopardizing security. This technique is both novel and privacy-preserving. Authors evaluate the dependability and complexity of the SIT protocols to assess theft protection in the existence of hostile parties and the communication costs. The evaluation proved to be highly efficient in terms of identity theft protection.

Yazdinejad et al. [87] describe a federated drone authentication model based on drone Radio Frequency features in IoT networks. This model provides a privacy-preserving advantage by applying the FL approach to train data locally. This technique uses a combination of federated ML, HE, and aggregation. This article includes an evaluation (that uses accuracy, precision, recall, and F1-measure) that validates this technique and proves the effectiveness compared to other models. The authors evaluate the federated drone authentication model on an RF-based dataset that was detected using a radio frequency classification toolbox. The evaluation is promising for making use of decentralized data generated by drones while maintaining privacy by using the FL approach to train data locally.

Unal et al. [84] detail a blockchain-assisted secure framework for FL in Big Data analytics services for IoT. This technique is focused on the PP of secure Big Data analytics services. This technique is both novel and privacy preserving and is validated through evaluation. Authors evaluate the use of fuzzy to detect variations and anomalies in FL training models against poisoning attacks by simulating attack modes in a pseudo-simulated environment. This evaluation is promising in ensuring the security of user data and trained models.

Nadian-Ghomshe et al. [60] present a distributed access control system based on blockchain technology to secure IoT data. Using the ABAC as access control technology, the edge node can assign the appropriate properties to the IoT data file, and only resource requesters with matching property requirements can access the IoT data file, setting the first barrier for protecting the IoT data file. Next, by using the hyperledger fabric blockchain technology, the file access model policies with timestamps are stored in blocks that are pre-defined by the edge node. The proposed mechanism is based on Fog computing and the concept of the alliance chain. This method uses mixed linear and

nonlinear spatiotemporal chaotic systems and the **Least Significant Bit (LSB)** to encrypt the IoT data on an edge node and then uploads the encrypted data to the cloud. Authors evaluate the LSB method algorithms and the mixture of linear and nonlinear environments to develop a secure data sharing mechanism for IoT data using realistic IoT scenarios. The evaluation proved promising in the solution for a single point of failure of access control by providing dynamic and fine-grained access control to create a controlled and secure environment for IoT data—DTs, **Support Vector Machines (SVM)**, and LSTM.

Cheng et al. [18] proposed an **APT Prediction Method Based on Differentially Private Federated Learning (APTPMFL)** to predict the probability of subsequent APT attacks occurring in an IoT system. The paper also presents a 5G-enabled edge computing-based framework to train and deploy the model, which can alleviate the computing and communication overhead of a typical IoT system. This technique is validated through evaluation and foes include F1-scores. This scheme is both novel and privacy preserving. Authors evaluate the APTPMFL's prediction for the probability of subsequent APT activities through the federal learning model varying to aggregating suspicious activities in the IoT system. The evaluation proved promising in predicting subsequent APT behaviors with a high accuracy rate while protecting the data.

Phong et al. [66] described a privacy-preserving DL system that works with multiple learning participants performing neural network-based DL over a combined dataset of MNIST and SVHN, without revealing the participants' local data to a central server. The authors revisited the previous work of Shokri and Shmatikov [74] and show that, with their method, local data information may be leaked to an honest-but-curious server. The author claims to fix that problem by building an enhanced system that avoids leaking information to the server with no deterioration of the accuracy. This solution implements asynchronous stochastic gradient descent as applied to neural networks, in combination with additively HE.

Mugunthan et al. [59] introduced a simulator for FL environments that includes latency simulation, robustness to client departure/failure, support for both centralized (with one or more servers) and decentralized (serverless) learning, and configurable privacy and security mechanisms based on DP and secure multiparty computation. To evaluate their simulator, the authors developed a secure and differentially private FL environment, based on an ML LR algorithm trained on the MNIST dataset for eight iterations.

Feng et al. [24] present a novel privacy-preserving tensor decomposition approach over semantically secure encrypted big data. The proposed approach leverages properties of HE and employs a federated cloud to securely decompose an encrypted tensor for multiple users, without the clouds collecting any knowledge about user's data. Authors evaluate the implementation of the protocol using both synthetic and real-world datasets using RMSE and MAE to assess the accuracy of the predicted ratings. The evaluation proved promising in implementing most privacy-preserving computations for Big Data and utilized in a cloud environment.

Zhang et al. [89] introduced a novel approach to task allocation under geo-indistinguishability via group-based noise addition, which employs group-based noise addition to preserve user location privacy during FL processes. This approach ensures that the noise added to the location data maintains a balance between privacy and utility, allowing effective task allocation without compromising user privacy.

Feng et al. [23] proposed a **differentially private tensor-based RNN (DPTRNN)** that can be applied in many challenging DL sequence tasks for IoT systems. Specifically, to process heterogeneous sequential data, the authors propose a tensor-based RNN model. Then, to guarantee privacy, they develop a tensor-based back-propagation through time algorithm with perturbation to avoid exposing the sensitive information for training the tensor-based RNN model within the framework of DP. The authors conduct empirical evaluations on two datasets: The KTH dataset and the UCF50 dataset.

To summarize, to the best of our knowledge, none of the 12 previous works implemented an FL IDS leveraging a CNN model trained on a set of seven real-world IoT datasets that involved DP, DK, and homomorphic PP techniques as we do in this article. In particular, our work differs from previous work as follows: (a) we implement a 1D CNN FL system. This is the first time that a CNN was implemented in an FL scheme as IDS; (b) we trained and tested our solution on seven IoT datasets for the first time in this domain; (c) we implemented three PP techniques in one solution for the first time; and (d) we provide the pseudo-code of our main algorithms and the empirical evaluation of our solution (none of the previous work did the same).

## 8 Conclusion

In this article, we proposed an FL IDS that leverages a 1D CNN. To enhance PP of the data, the system implements three key techniques: (a) DP, which adds private noise to the training data created by the CNN models, (b) DK exchange, which add a secure exchange of cryptographic keys (key agreement protocol) over the FL network (insecure) in a way that overheard communication does not reveal the keys, and (c) HE, which performs computations on the encrypted training data without the need for the secret key to decrypt the ciphertext.

Our solution is trained and tested on seven recently released publicly available IoT datasets: TON-IoT, IoT-23, BoT-IoT, CIC IoT 2023, CIC IoMT 2024, RT-IoT 2022, and EdgeIIot. We extensively study the effectiveness of this solution in terms of CNN-based IDS (i.e., accuracy prediction) and FL complexity (i.e., execution time). Our solution achieves an excellent average accuracy, precision, recall, and F1-score (97.31%, 95.59%, 92.43%, and 92.69%, respectively) across the seven datasets. Experiment results indicate that the version of our FL implementing all three PP techniques has an increase in computation time of only 10% compared to the same FL without any PP in place. Moreover, we show the effectiveness of our design choices through a comparison study with other recent IDS trained and tested on the same datasets we use.

## References

[1] Nada Abdalgawad, A. Sajun, Y. Kaddoura, Imran A. Zualkernan, and F. Aloul. 2021. Generative deep learning to detect cyberattacks for the IoT-23 dataset. *IEEE Access* 10 (2021), 6430–6441.

[2] Sulyman Abdulkareem, Chuan Foh, François Carrez, and Klaus Moessner. 2024. A lightweight SEL for attack detection in IoT/IIoT networks. *IIot Networks* 230 (2024), 103980.

[3] Shweta Agrawal, Shafi Goldwasser, and Saleet Mossel. 2021. Deniable fully homomorphic encryption from learning with errors. In *Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO '21,* Part II 41. Springer, 641–670.

[4] Gregorius Airlangga. 2024. Comparative analysis of machine learning models for intrusion detection in Internet of Things networks using the RT-IoT2022 dataset. *MALCOM: Indonesian Journal of Machine Learning and Computer Science* 4, 2 (2024), 656–662.

[5] Muna Al-Hawawreh, Nour Moustafa, Sahil Garg, and M. Shamim Hossain. 2020. Deep learning-enabled threat intelligence scheme in the internet of things networks. *IEEE Transactions on Network Science and Engineering* 8, 4 (2020), 2968–2981.

[6] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. 2017. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, 1–6. DOI: https://doi.org/10.1109/ICEngTechnol.2017.8308186

[7] Mansoor Ali, Faisal Naeem, Muhammad Tariq, and Geroges Kaddoum. 2022. Federated learning for privacy preservation in smart healthcare systems: A comprehensive survey. *IEEE Journal of Biomedical and Health Informatics* 27, 2 (2022), 778–789.

[8] Abdullah Aljumah. 2021. IoT-based intrusion detection system using convolution neural networks. *PeerJ Computer Science* 7 (2021), e721.

[9] Orlando Amaral, Sallam Abualhaija, Damiano Torre, Mehrdad Sabetzadeh, and Lionel C. Briand. 2022. AI-enabled automation for completeness checking of privacy policies. *IEEE Transactions on Software Engineering* 48, 11 (2022), 4647–4674. DOI: https://doi.org/10.1109/TSE.2021.3124332

[10] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *the 23rd International Conference on Artificial Intelligence and Statistics, AISTATS '20*. Silvia Chiappa and Roberto Calandra (Eds.), Proceedings of Machine Learning Research, Vol. 108, PMLR, 2938–2948. DOI: http://proceedings.mlr.press/v108/bagdasaryan20a.html

[11] Mohammed Baz. 2022. SEHIDS: Self evolving host-based intrusion detection system for IoT networks. *Sensors* 22, 17 (2022), 6505.

[12] Guru Bhandari, Andreas Lyth, Andrii Shalaginov, and Tor-Morten Grønli. 2023. Distributed deep neural-network-based middleware for cyber-attacks detection in smart IoT ecosystem: A novel framework and performance evaluation approach. *Electronics* 12, 2 (2023), 298.

[13] Abhilasha Bhargav-Spantzel, Anna Cinzia Squicciarini, and Elisa Bertino. 2006. Establishing and protecting digital identity in federation systems. *The Journal of Computer Security* 14, 3 (2006), 269–300. DOI: https://doi.org/10.3233/jcs-2006-14303

[14] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD '13*, Proceedings, Part III 13. Springer, 387–402.

[15] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards federated learning at scale: System design. DOI: https://doi.org/10.48550/ARXIV.1902.01046

[16] Tim M. Booij, Irina Chiscop, Erik Meeuwissen, Nour Moustafa, and Frank T. H. den Hartog. 2021. ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets. *IEEE Internet of Things Journal* 9, 1 (2021), 485–496.

[17] Brandon Bowen, Anitha Chennamaneni, Ana Goulart, and Daisy Lin. 2023. BLoCNet: A hybrid, dataset-independent intrusion detection system using deep learning. *International Journal of Information Security* 22, 4 (2023), 893–917.

[18] Xiang Cheng, Qian Luo, Ye Pan, Zitong Li, Jiale Zhang, and Bing Chen. 2021. Predicting the APT for cyber situation comprehension in 5G-enabled IoT scenarios based on differentially private federated learning. *Security and Communication Networks* (2021), 8814068:1–8814068:14. DOI: https://doi.org/10.1155/2021/8814068

[19] Sajjad Dadkhah, Euclides Carlos Pinto Neto, Raphael Ferreira, Reginald Chukwuka Molokwu, Somayeh Sadeghi, and Ali Ghorbani. 2024. CICIoMT2024: Attack vectors in healthcare devices-A multi-protocol dataset for assessing IoMT device security.

[20] Whitfield Diffie and Martin E. Hellman. 2022. New directions in cryptography. In *Democratizing Cryptography: The Work of Whitfield Diffie and Martin Hellman*, 365–390.

[21] Vibekananda Dutta, Michał ChoraŚ, Marek Pawlicki, and Rafał Kozik. 2020. A deep learning ensemble for network anomaly and cyber-attack detection. *Sensors* 20, 16 (2020), 4583.

[22] Omar Elghalhoud, Kshirasagar Naik, Marzia Zaman, and Ricardo Manzano. 2023. Data balancing and CNN based network intrusion detection system. In *2023 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 1–6.

[23] Jun Feng, Laurence T. Yang, Bocheng Ren, Deqing Zou, Mianxiong Dong, and Shunli Zhang. 2024. Tensor recurrent neural network with differential privacy. *IEEE Transactions on Computers* 73, 3 (2024), 683–693.

[24] Jun Feng, Laurence T. Yang, Qing Zhu, and Kim-Kwang Raymond Choo. 2020. Privacy-preserving tensor decomposition over encrypted data in a federated cloud environment. *IEEE Transactions on Dependable and Secure Computing* 17, 4 (2020), 857–868. DOI: https://doi.org/10.1109/TDSC.2018.2881452

[25] Mohamed Amine Ferrag, Othmane Friha, Djallel Hamouda, Leandros Maglaras, and Helge Janicke. 2022. Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. *IEEE Access* 10 (2022), 40281–40306.

[26] Othmane Friha, Mohamed Amine Ferrag, Mohamed Benbouzid, Tarek Berghout, Burak Kantarci, and Kim-Kwang Raymond Choo. 2023. 2DF-IDS: Decentralized and differentially private federated learning-based intrusion detection system for industrial IoT. *Computers & Security* 127 (2023), 103097.

[27] Othmane Friha, Mohamed Amine Ferrag, Lei Shu, Leandros Maglaras, Kim-Kwang Raymond Choo, and Mehdi Nafaa. 2022. FELIDS: Federated learning-based intrusion detection system for agricultural Internet of Things. *Journal of Parallel and Distributed Computing* 165 (2022), 17–31.

[28] Abdallah R. Gad, Mohamed Haggag, Ahmed A. Nashat, and Tamer M. Barakat. 2022. A distributed intrusion detection system using machine learning for IoT based on ToN-IoT dataset. *International Journal of Advanced Computer Science and Applications* 13, 6 (2022), 548–563.

[29] Sebastian Garcia, Agustin Parmisano, and Maria J. Erquiaga. 2020. IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) data set. In *Zenodo*. DOI: https://doi.org/10.5281/zenodo.4743746

[30] Craig Gentry, Amit Sahai, and Brent Waters. 2013. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference*, Proceedings, Part I. Springer, 75–92.

[31] Ana Goulart, Anitha Chennamaneni, Damiano Torre, Byul Hur, and Fadhil Y. Al-Aboosi. 2022. On wide-area IoT networks, lightweight security and their applications – A practical review. *Electronics* 11, 11 (2022). DOI: https://doi.org/10.3390/electronics11111762

[32] Goichiro Hanaoka and Kaoru Kurosawa. 2008. Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 308–325.

[33] Muneeb Ul Hassan, Mubashir Husain Rehmani, and Jinjun Chen. 2020. Differential privacy techniques for cyber physical systems: A survey. *IEEE Communications Surveys and Tutorials* 22, 1 (2020), 746–789. DOI: https://doi.org/10.1109/COMST.2019.2944748

[34] J. Hemalatha, S. Abijah Roseline, Subbiah Geetha, Seifedine Nimer Kadry, and Robertas Damasevicius. 2021. An efficient DenseNet-based deep learning model for malware detection. *Entropy* 23, 3 (2021), 344. DOI: https://doi.org/10.3390/e23030344

[35] Hossein Hosseini, Sungrack Yun, Hyunsin Park, Christos Louizos, Joseph Soriaga, and Max Welling. 2020. Federated learning of user authentication models. arXiv:2007.04618. Retrieved from https://arxiv.org/abs/2007.04618

[36] Pengfei Hu, Hongxing Li, Hao Fu, Derya Cansever, and Prasant Mohapatra. 2015. Dynamic defense strategy against advanced persistent threat with insiders. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, 747–755. DOI: https://doi.org/10.1109/INFOCOM.2015.7218444

[37] David H. Hubel and Torsten N. Wiesel. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology* 160, 1 (1962), 106.

[38] Nicholas Jeffrey, Qing Tan, and José R. Villar. 2024. Using ensemble learning for anomaly detection in cyber–physical systems. *Electronics* 13, 7 (2024), 1391.

[39] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. 2019. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems* 100 (2019), 779–796.

[40] Fangyu Li, Junnuo Lin, and Honggui Han. 2023. FSL: Federated sequential learning-based cyberattack detection for Industrial Internet of Things. *Industrial Artificial Intelligence* 1, 1 (2023), 4.

[41] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. 2020. A review of applications in federated learning. *Computers & Industrial Engineering* 149 (2020), 106854.

[42] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A survey on federated learning systems: Vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (2021), 3347–3366.

[43] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (May 2020), 50–60. DOI: https://doi.org/10.1109/msp.2020.2975749

[44] Yiwei Li, Tsung-Hui Chang, and Chong-Yung Chi. 2020. Secure federated averaging algorithm with differential privacy. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP),* 1–6. DOI: https://doi.org/10.1109/MLSP49062.2020.9231531

[45] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. 2022. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems* 33, 12 (2022), 6999–7019. DOI: https://doi.org/10.1109/TNNLS.2021.3084827

[46] Zengpeng Li, Vishal Sharma, and Saraju P. Mohanty. 2020. Preserving data privacy via federated learning: Challenges and solutions. *IEEE Consumer Electronics Magazine* 9, 3 (2020), 8–16.

[47] Hongyu Liu and Bo Lang. 2019. Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences* 9, 20 (2019). DOI: https://doi.org/10.3390/app9204396

[48] Wai Weng Lo, Siamak Layeghy, Mohanad Sarhan, Marcus Gallagher, and Marius Portmann. 2022. E-graphsage: A graph neural network based intrusion detection system for IoT. In *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 1–9.

[49] Scott Lorenz, Stanley Stinehour, Anitha Chennamaneni, Abdul B. Subhani, and Damiano Torre. 2023. IoT forensic analysis: A family of experiments with Amazon Echo devices. *Forensic Science International: Digital Investigation* 45 (2023), 301541.

[50] Mirco Marchetti, Fabio Pierazzi, Michele Colajanni, and Alessandro Guido. 2016. Analysis of high volumes of network traffic for Advanced Persistent Threat detection. *Computer Networks* 109 (2016), 127–141. DOI: https://doi.org/10.1016/j.comnet.2016.05.018

[51] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*. PMLR, 1273–1282.
[52] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2016. Communication-efficient learning of deep networks from decentralized data. (2016). DOI: https://doi.org/10.48550/ARXIV.1602.05629
[53] Frantzy Mesadieu, Damiano Torre, and Anitha Chennameneni. 2024. Leveraging deep reinforcement learning technique for intrusion detection in SCADA infrastructure. *IEEE Access* 12 (2024), 63381–63399.
[54] Donald Michie, David J. Spiegelhalter, and C. C. Taylor. 1994. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood.
[55] David J. Miller, Zhen Xiang, and George Kesidis. 2020. Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks. *Proceedings of the IEEE* 108, 3 (2020), 402–433.
[56] Michele Mosca. 2018. Cybersecurity in an era with quantum computers: Will we be ready? *IEEE Security & Privacy* 16, 5 (2018), 38–41.
[57] Viraaji Mothukuri, Reza M. Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. 2021. A survey on security and privacy of federated learning. *Future Generation Computer Systems* 115 (2021), 619–640. DOI: https://doi.org/10.1016/j.future.2020.10.007
[58] Nour Moustafa. 2019. The Bot-IoT Dataset. Retrieved from https://doi.org/10.21227/r7v2-x988
[59] Vaikkunth Mugunthan, Anton Peraire-Bueno, and Lalana Kagal. 2020. PrivacyFL: A simulator for privacy-preserving and secure federated learning. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management*. Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (Eds.), ACM, 3085–3092. DOI: https://doi.org/10.1145/3340531.3412771
[60] Ali Nadian-Ghomsheh, Bahar J. Farahani, and Mohammad Kavian. 2021. A hierarchical privacy-preserving IoT architecture for vision-based hand rehabilitation assessment. *Multimedia Tools and Applications* 80, 20 (2021), 31357–31380. DOI: https://doi.org/10.1007/s11042-021-10563-2
[61] Euclides Carlos Pinto Neto, Sajjad Dadkhah, Raphael Ferreira, Alireza Zohourian, Rongxing Lu, and Ali A. Ghorbani. 2023. CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. (2023).
[62] Sang Ni, Quan Qian, and Rui Zhang. 2018. Malware identification using visualization images and deep learning. *Computers & Security* 77 (2018), 871–885. DOI: https://doi.org/10.1016/j.cose.2018.04.005
[63] Kartik Palani, Emily Holt, and Sean W. Smith. 2016. Invisible and forgotten: Zero-day blooms in the IoT. In *2016 IEEE International Conference on Pervasive Computing and Communication Workshops*. IEEE Computer Society, 1–6. DOI: https://doi.org/10.1109/PERCOMW.2016.7457163
[64] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *the 2017 ACM on Asia Conference on Computer and Communications Security*, 506–519.
[65] NhatHai Phan, Xintao Wu, Han Hu, and Dejing Dou. 2017. Adaptive Laplace mechanism: Differential privacy preservation in deep learning. In *2017 IEEE International Conference on Data Mining (ICDM)*, 385–394. DOI: https://doi.org/10.1109/ICDM.2017.48
[66] Le Trieu Phong, Yoshinori Aono, Takuya Hayashi, Lihua Wang, and Shiho Moriai. 2018. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security* 13, 5 (2018), 1333–1345. DOI: https://doi.org/10.1109/TIFS.2017.2787987
[67] Gopika Premsankar, Mario Di Francesco, and Tarik Taleb. 2018. Edge computing for the Internet of Things: A case study. *IEEE Internet of Things Journal* 5, 2 (2018), 1275–1284.
[68] Wang Ren, Xin Tong, Jing Du, Na Wang, Shan Cang Li, Geyong Min, Zhiwei Zhao, and Ali Kashif Bashir. 2021. Privacy-preserving using homomorphic encryption in Mobile IoT systems. *Computer Communications* 165 (2021), 105–111.
[69] Jihyeon Ryu, Keunok Kim, and Dongho Won. 2023. A study on partially homomorphic encryption. In *2023 17th International Conference on Ubiquitous Information Management and Communication (IMCOM)*. IEEE, 1–4.
[70] R. Sabitha, S. Gopikrishnan, B. J. Bejoy, V. Anusuya, and V. Saravanan. 2023. Network based detection of IoT attack using AIS-IDS model. *Wireless Personal Communications* 128, 3 (2023), 1543–1566.
[71] S. K. Sangeetha, Prasanna Mani, V. Maheshwari, Prabhu Jayagopal, M. Sandeep Kumar, and Shaikh Muhammad Allayear. 2022. Design and analysis of multilayered neural network-based intrusion detection system in the Internet of Things network. *Computational Intelligence and Neuroscience* 2022, 1 (2022), 9423395.
[72] S. V. N. Santhosh Kumar, M. Selvi, and A. Kannan. 2023. A comprehensive survey on machine learning-based intrusion detection systems for secure communication in Internet of Things. *Computational Intelligence and Neuroscience* 2023, 1 (2023), 8981988.
[73] B. S. Sharmila and Rohini Nagapadma. 2023. Quantized autoencoder (QAE) intrusion detection system for anomaly detection in resource-constrained IoT devices using RT-IoT2022 dataset. *Cybersecurity* 6, 1 (2023), 41.

[74] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 1310–1321.
[75] Srinivas Sridharan. 2006. *Implementing Scalable Locks and Barriers on Large-Scale Light-Weight Multithreaded Systems*. Ph.D. Dissertation. University of Notre Dame.
[76] M. R. Suma and P. Madhumathy. 2022. Brakerski-Gentry-Vaikuntanathan fully homomorphic encryption cryptography for privacy preserved data access in cloud assisted Internet of Things services using glow-worm swarm optimization. *Transactions on Emerging Telecommunications Technologies* 33, 12 (2022), e4641.
[77] Ankit Thakkar and Ritika Lohiya. 2021. A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges. *Archives of Computational Methods in Engineering* 28, 4 (2021), 3211–3243.
[78] Damiano Torre, Mauricio Alférez, Ghanem Soltana, Mehrdad Sabetzadeh, and Lionel C. Briand. 2021. Modeling data protection and privacy: Application and experience with GDPR. *Software and Systems Modeling* 20, 6 (2021), 2071–2087. DOI: https://doi.org/10.1007/s10270-021-00935-5
[79] Damiano Torre, Anitha Chennamaneni, and Alex Rodriguez. 2023. Privacy-preservation techniques for IoT devices: A systematic mapping study. *IEEE Access* 11 (2023), 16323–16345. DOI: https://doi.org/10.1109/ACCESS.2023.3245524
[80] Damiano Torre, Frantzy Mesadieu, and Anitha Chennamaneni. 2023. Deep learning techniques to detect cybersecurity attacks: A systematic mapping study. *Empirical Software Engineering* 28, 3 (2023), 76.
[81] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction APIs. In *USENIX Security Symposium*, Vol. 16, 601–618.
[82] Imtiaz Ullah and Qusay H. Mahmoud. 2021. Design and development of a deep learning-based model for anomaly detection in IoT networks. *IEEE Access* 9 (2021), 103906–103926.
[83] Imtiaz Ullah and Qusay H. Mahmoud. 2022. An anomaly detection model for IoT networks based on flow and flag features using a feed-forward neural network. In *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 363–368.
[84] Devrim Unal, Mohammad Hammoudeh, Muhammad Asif Khan, Abdelrahman Abuarqoub, Gregory Epiphaniou, and Ridha Hamila. 2021. Integration of federated machine learning and blockchain for the provision of secure big data analytics for Internet of Things. *Computers & Security* 109 (2021), 102393. DOI: https://doi.org/10.1016/j.cose.2021.102393
[85] Mi Wen, Rong Xie, Kejie Lu, Liangliang Wang, and Kai Zhang. 2021. FedDetect: A novel privacy-preserving federated learning framework for energy theft detection in smart grid. *IEEE Internet of Things Journal* 9, 8 (2021), 6069–6080.
[86] Zeyue Xue, Pan Zhou, Zichuan Xu, Xiumin Wang, Yulai Xie, Xiaofeng Ding, and Shiping Wen. 2021. A resource-constrained and privacy-preserving edge-computing-enabled clinical decision system: A federated reinforcement learning approach. *IEEE Internet of Things Journal* 8, 11 (2021), 9122–9138. DOI: https://doi.org/10.1109/JIOT.2021.3057653
[87] Abbas Yazdinejad, Reza M. Parizi, Ali Dehghantanha, and Hadis Karimipour. 2021. Federated learning for drone authentication. *Ad Hoc Networks* 120 (2021), 102574. DOI: https://doi.org/10.1016/j.adhoc.2021.102574
[88] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. 2022. Enhanced membership inference attacks against machine learning models. In *the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022*. Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi (Eds.), ACM, 3093–3106. DOI: https://doi.org/10.1145/3548606.3560675
[89] Pengfei Zhang, Xiang Cheng, Sen Su, and Ning Wang. 2022. Task allocation under geo-indistinguishability via group-based noise addition. *IEEE Transactions on Big Data* 9, 3 (2022), 860–877.
[90] Yilei Zhang, Peiyun Zhang, Yonglong Luo, and Liya Ji. 2020. Towards efficient, credible and privacy-preserving service QoS prediction in unreliable mobile edge environments. In *International Symposium on Reliable Distributed Systems, SRDS 2020*. IEEE, 309–318. DOI: https://doi.org/10.1109/SRDS51746.2020.00038

Received 8 February 2024; revised 19 July 2024; accepted 21 August 2024