FL4IoT: IoT Device Fingerprinting and Identification Using Federated Learning

HAN WANG and DAVID EKLUND, RISE Research Institutes of Sweden, Sweden
ALINA OPREA, Northeastern University, U.S.A.
SHAHID RAZA, RISE Research Institutes of Sweden, Sweden

Unidentified devices in a network can result in devastating consequences. It is, therefore, necessary to fingerprint and identify IoT devices connected to private or critical networks. With the proliferation of massive but heterogeneous IoT devices, it is getting challenging to detect vulnerable devices connected to networks. Current machine learning-based techniques for fingerprinting and identifying devices necessitate a significant amount of data gathered from IoT networks that must be transmitted to a central cloud. Nevertheless, private IoT data cannot be shared with the central cloud in numerous sensitive scenarios. Federated learning (FL) has been regarded as a promising paradigm for decentralized learning and has been applied in many different use cases. It enables machine learning models to be trained in a privacy-preserving way. In this article, we propose a privacy-preserved IoT device fingerprinting and identification mechanisms using FL; we call it FL4IoT. FL4IoT is a two-phased system combining unsupervised-learning-based device fingerprinting and supervised-learning-based device identification. FL4IoT shows its practicality in different performance metrics in a federated and centralized setup. For instance, in the best cases, empirical results show that FL4IoT achieves ~99% *accuracy* and *F1-Score* in identifying IoT devices using a federated setup without exposing any private data to a centralized cloud entity. In addition, FL4IoT can detect spoofed devices with over 99% *accuracy*.

CCS Concepts: • **Security and privacy**; • **Computing methodologies** → *Distributed artificial intelligence*; *Neural networks*;

Additional Key Words and Phrases: Internet of things, federated learning, identification, fingerprinting, machine learning

ACM Reference format:
Han Wang, David Eklund, Alina Oprea, and Shahid Raza. 2023. FL4IoT: IoT Device Fingerprinting and Identification Using Federated Learning. *ACM Trans. Internet Things* 4, 3, Article 17 (July 2023), 24 pages.
https://doi.org/10.1145/3603257

1 INTRODUCTION

With the introduction of massive IoT, consisting of billions of connected devices with heterogeneous features from a variety of vendors, it is getting harder to detect unauthorized connected devices. This gets even more sensitive if such unauthorized devices are present in critical

This research has been partly supported by the EU H2020 projects ARCADIAN-IoT (GA 101020259) and VEDLIoT (GA 957197); Horizon Europe project HARPOCRATES (GA 10048312); and initiated by the RISE Cybersecurity KP.
Authors' addresses: H. Wang, D. Eklund, and S. Raza, RISE Research Institutes of Sweden AB, Isafjordsgatan 22, 164 40 Kista; emails: {han.wang, david.eklund, shahid.raza}@ri.se; A. Oprea, Northeastern University, 440 Huntington Avenue, 202 West Village H, Boston, MA 02115.

2577-6207/2023/07-ART17
https://doi.org/10.1145/3603257

infrastructure and can disrupt operations. For instance, in a recent NASA system hack [23], unidentified IoT devices resulted in an advanced persistent threat, where an undisclosed and unauthorized Raspberry Pi device connected to the Jet Propulsion Laboratory was the source of a hack of NASA servers where hackers were able to gain access to one of NASA's *major mission systems*. Unauthorized IoT devices with weak security can become a source of attacks on the global Internet [1]. It is, therefore, necessary to identify connected devices and remove unauthorized devices from the networks. Appropriate and well-designed device fingerprinting and identification can help network administrators to manage the network efficiently, ensuring the connections among devices are in order.

Motivation and Rationale: Due to IoT's heterogeneous ecosystem, device fingerprinting and identification are challenging tasks. Many approaches have been proposed along the growth of IoT networks, from the early traditional cryptographic authentication to the current machine-learning-based solutions. Existing ML-based works define the fingerprint of the devices as the set of features extracted either from the physical layer, mainly the radio frequency waveform, or the network traffic, usually the packet or flow patterns. Physical layer data are more sensitive to environmental factors and limited to individual wireless technologies. In comparison, network traffic data are more flexible but, at the same time, more complex. ML-based approaches usually involve various techniques such as data engineering for feature extraction and selection [2, 7, 21], signal processing [25], and natural language processing for the analysis of packet payload [29]. These kinds of device fingerprinting approaches usually require a deep study of the data, and are hence data-oriented and task-oriented, which makes these approaches not generic to the different use cases.

However, **deep learning (DL)** has recently emerged and become a trend in many domains, including device fingerprinting and identification. Rather than complex data engineering and data preprocessing, DL-based approaches are usually probabilistic models that learn the distributions or representations from the original data. Ortiz et al. [24] propose an LSTM-embedded autoencoder to learn the distribution from the packet payload content and compare the distributions of unknown devices with those known. However, they use a sniffer to analyze the packet information, which is infeasible when the packet is encrypted, and it raises privacy concerns. Other works [12, 36] use statistical methods to extract features as fingerprints and apply DL to train a neural network to identify the devices. DL-based device identification models are able to achieve better accuracy performance, but the way of producing fingerprints proposed in these works is the feature-extraction-based method, which is static, not dynamic. It might miss the potentially important factors possessed by the data.

Training a satisfactory DL-based model requires powerful computational resources; thus, it is common to push data to the cloud using its service. However, this creates vulnerabilities such as information leakage and abuse of data. New privacy laws such as the ePrivacy Directive and GDPR in the EU encourage (and, in some cases, mandate) sharing only minimal amounts of data, which enforces privacy-by-default and by-design. Bringing ML to the edge of the IoT network, where actual collection and/or actuation takes place, is becoming a new trend. Its advantages include efficient resource utilization, reducing latency, and so on. **Federated Learning (FL)** emerges as a promising paradigm for its decentralized and privacy-preserving ecosystem [9, 17]. It is a distributed approach that enables IoT edge devices to collaboratively train models in a decentralized way and keep the private data on the devices at the same time. In Section 2.2, we describe Federated Learning in more detail.

Goal: In this article, we introduce FL4IoT, a two-phased approach to device fingerprinting and identification, which can be applied in both centralized and federated settings. We build an unsupervised-learning-based model to generate the vectorized fingerprints by analyzing the device's traffic behavior. Since the generated fingerprints are vectorized, they are relatively lightweight to be stored on edge devices for further use. We then propose a supervised-learning-based model for device identification, where the generated fingerprints are involved in the training process. We finally evaluate FL4IoT on three real-world datasets to show the feasibility of applying our approach to FL, and we compare the overall performance between FL and centralized training. The main contributions of this article are listed as follows:

- We propose FL4IoT, a two-phased approach including generating lightweight 1D fingerprints offline and identifying new traffics with generated fingerprints online. FL4IoT is applicable in both the centralized and federated settings.
- We evaluate our method FL4IoT with three real-world datasets and compare with three baselines in a centralized setup and two baselines in a FL setup. The identification performance can reach ~99% of accuracy.
- We evaluate FL4IoT from different aspects with experiments including detection of spoofed devices and multiple clients in FL setup. FL4IoT reaches over 99% precision in detecting spoofed devices.
- To the best of our knowledge, FL4IoT is the first work focusing on applying FL to network-traffic-based device fingerprinting and identification tasks.

The remainder of the article is organized as follows: Section 2 summarizes related work. Section 3 defines the problem and the threat model. Section 4 introduces the proposed approach and the design of FL workflow used in this article. Section 5 summarizes the experiments from different setups and contains corresponding discussions. In Section 6, we analyze the security issues and challenges. Finally, Section 7 draws the conclusions and the future work.

2 RELATED WORK AND BACKGROUND

In this section, we survey the literature that closely relates to device fingerprinting and identification by using machine learning techniques, and we provide the background knowledge of federated learning.

2.1 Device Fingerprinting and Identification

Device identification and fingerprinting techniques have been used over the years in network security for device authentication, which is the process of determining whether a device is what it claims to be. Machine learning is increasingly employed to solve the problem. Existing ML approaches can be divided into two focuses: the physical-hardware-based method and the network-traffic-based method. Physical hardware-based fingerprinting is widely used in the domain of **Wireless Local Area Networks (WLANs)**. There are many studies on observing the physical information of the device, such as configuration [4, 13, 26], or investigating the **radio frequency (RF)** from the physical layer of the network [10, 14, 15, 27, 28, 38, 39]. In this article, we mainly focus on the network-traffic-based fingerprinting methods that study the network behavior of the IoT devices by extracting features from the packets or exploring the statistical information of the traffic flow [2, 7, 12, 19–22, 24, 25, 29, 33, 36].

Table 1 summarizes the existing works on network-traffic-based device fingerprinting and identification. We divide these works into two categories based on the type of ML method: Traditional ML or DL. From the table, we observe that there are two types of features: Packet features that look into the information contained in the packet header and payload, and network-flow features that consist of statistical information on traffic flow (such as mean and standard deviation of the packets transmission time). Moreover, we notice that most works define device fingerprint as the set of

Table 1. Existing Works on Network-traffic-based Device Fingerprinting and Identification

Category	Authors	Features type	ML Model	Key Contribution
Traditional machine learning	Miettinen et al. [21]	Packet features (Protocols in different layers, Packet Content, IP address...)	Random Forest	Propose a representation of device fingerprint that is a 23 by *n* matrix consisting of 23 features extracted by *n* packets.
	Hamad et al. [7]	Mixed features (Packet features and flow features)	Adaptive Boosting, LDA, K-Nearest Neighbors, Decision Tree, Nave Bayesian, SVM, Random Forest	The fingerprint comprises 67 features extracted from the packet header, payload, and statistical information of the traffic flow. Examine several different machine learning methods to choose the best model for the task.
	Bezawada et al. [2]	Packet features (Header features and payload features)	K-Nearest Neighbors, Decision Tree, Gradient Boosting	Generate a behavioral profile considering both packet header and payload. Utilize Miettinen et al.'s method to extract features from the header and propose three important payload features.
	Meidan et al. [20]	Flow features	Random Forest	Train a classifier on the set of whitelisted IoT devices and apply the inferred classifier to distinguish unauthorized devices.
	Meidan et al. [19]	Packet features (IP address, port numbers...)	Gradient Boosting, Random Forest, XGBoost	Propose a single-session binary classifier. Determine the optimal threshold for each classifier. Utilize majority voting on whether the session is generated by the device.
	Sivanathan et al. [29]	Mixed features (Flow features, domain names, port number, TLS handshake messages)	Ensemble learning	Statistically characterize network traffic in different terms by using bag-of-word, and text classification techniques. Propose a multi-stage ML framework to identify the IoT devices.
	Marchal et al. [25]	Flow features (Transform into periodic data)	K Nearest Neighbors	Analyze network flow and utilize Fourier transform to transform the time-series data into the signal format and extract fingerprint features by using signal processing methods.
	Msadek et al. [22]	Flow features (Availability time, inter-arrival time, packet size...)	K Nearest Neighbors, SVM, Random Forest, AdaBoost, Extra-Trees	Extract features of the fingerprint from the sequence of the packet header with the technique of sliding window.
Deep learning	Ortiz et al. [24]	Packet features (Sequence of Payload)	Ensemble learning	Propose LSTM-embedded autoencoder to learn the distribution over TCP flow, and use Bayesian model to compare distributions.
	Kumar et al. [12]	Flow features (Transmission time, inter-arrival time)	Neural network	Construct fingerprint by using statistical method. Train a neural network with generated fingerprints.

(Continued)

Table 1. Continued

Category	Authors	Features type	ML Model	Key Contribution
	Yang et al. [36]	Packet features (Protocols in different layers, packet payload...)	Neural network, LSTM	Propose the way of extracting features for fingerprinting from the application layer, transport layer, and network layer. Propose a three-level classifier for type, vendor, and product.
	Thom et al. [33]	Packet features (raw Pcap files)	Neural network	Convert Pcap files into Nilsimsa hash (Locality Sensitive Hasges) to generate fingerprint.

features extracted from the data and labeled with the corresponding device [2, 7, 20–22, 29, 36]. They train the ML models with these fingerprints to classify or identify previously unseen traffic. The key contribution of these works is mainly data engineering to parse and extract important features from the raw data. Some of the other works propose methods to transform the original data into another representation, such as periodic signal [25] or hash [33], or to learn the distribution [24] or representation [12] by using a neural network.

Traditional ML-based approaches to feature extraction for device fingerprinting seem promising, because they learn the raw data well. However, they require complex data pre-processing steps and are usually designed for specific data. For example, Miettinen et al. [21] propose a set of features that represent the device fingerprint, which covers 23 features, including protocols used in 4 different layers, such as ARP/LLC in the link layer, TCP/UDP in the transport layer, and other 8 specific protocols used for the application layer. Nevertheless, because of the fast growth of IoT, various protocols are released. Their approach could not cover all the protocols, for example, CoAP and MQTT for the application layer or DTLS for the transport layer. Sivanathan et al. [29] create a testbed to monitor and collect traffic among 28 IoT devices, and they utilize bag-of-word, a natural-language-processing method, to analyze the domain names and the content of the packet. This approach is designed delicately but is infeasible when the traffic is encrypted, which is a common drawback for the approaches that consider the content of the packet.

DL-based approaches, in contrast, do not require feature extraction based on one's domain knowledge. These works focus on designing generalized models to learn the representations or distributions that could represent the device as the fingerprint. For example, Ortiz et al. [24] propose an unsupervised-learning-based model, LSTM-embedded autoencoder, to learn the distribution from the content of the packet payload and compare the distributions of unknown devices with those that are known. However, to obtain the payload sequences to feed into the autoencoder, they use a sniffer to analyze the packet information, which is, again, infeasible if the packet is encrypted, and it also raises privacy concerns. However, Kumar et al. [12] first utilize the transmission time of a frame and inter-arrival time to construct a histogram for each frame type and assign a weight to each frame type. They treat the histogram and weight combination as the device's fingerprint and train a neural network with these generated fingerprints to be further used for device identification. One strength of their work is that they only exploit two main features to construct the fingerprint without looking into the content of the packet. Nevertheless, their way of fingerprint construction is static but not dynamic, which means it is possible to miss the potential factors possessed by the data. Nevertheless, these works serve as valuable references for us to preserve their merits and address their shortcomings. We build an unsupervised-learning-based model to learn vectorized fingerprints for each device based on their flow features instead of packet features. Our model is dynamic and capable of fingerprinting the device even if its traffic is encrypted. In addition, deep learning methods are adaptive and applicable in the federated learning setup.

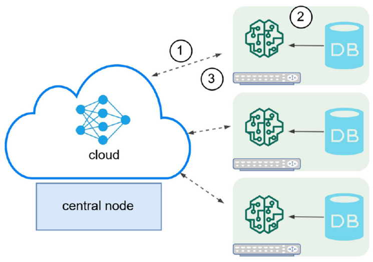


Fig. 1. Overview of federated learning setup: (1) The deployment of global model to clients, (2) Local training with private dataset on the clients, and (3) Model aggregation of the global model.

2.2 Federated Learning

Federated Learning has become a trend because of increasing data security and user privacy awareness, especially in IoT networks [37]. FL allows multiple entities to train a global model collaboratively without sharing their private data, assisted by a central node (or aggregator) that orchestrates the learning process without accessing the participants' datasets. The preliminary concept of FL emerged in 2015. Konecný et al. [8, 9] first introduced the prototype of federated learning, called federated optimization at that time. In 2017, Bonawitz et al. [3] designed a secure protocol, and McMahan et al. [17] improved the communication efficiency by introducing **Federated Averaging (FedAvg)**, which is now a state-of-the-art approach for the step of data aggregation in the FL process.

Figure 1 demonstrates the overview of FL setup. The training of an FL algorithm follows these steps: First, the participants receive some parameters from the central node and, with them, train the local model for a few epochs using their datasets. Then, the participants send the updated parameters after the training to the central node. Finally, the central node combines the parameters the participants sent using some aggregation rule, such as FedAvg, and sends back the aggregated model to the participants. This process is repeated until some level of convergence is attained or a maximum number of training rounds is reached [8].

FL has been applied in different domains, including IoT networks. However, applying FL to IoT device fingerprinting and identification is less discussed. Wu et al. [35] propose an FL-based RF fingerprinting and identification for IoT devices. They train a **convolutional neural network (CNN)** on the original signals collected from various IoT devices for fingerprint recognition. And they use a dynamic sample selection algorithm to accelerate the convergence of the learning progress. In their assumption, the RF signals are already the fingerprints of the devices, and their work is to learn to recognize them. Their work is only the existing work on applying FL on device fingerprinting and identification, so they do not compare their method to the other RF fingerprinting and identification techniques. Even so, they demonstrate the feasibility of FL applying to this specific task. To the best of our knowledge, our proposed work is the first to investigate network-traffic-based fingerprinting and identification with FL.

3 PROBLEM FORMULATION AND THREAT MODEL

3.1 Problem Formulation

This article considers the use case in IoT networks comprising different kinds of IoT devices and edge devices. However, the proposed algorithm can be applied in different contexts where device fingerprinting and identification are needed. Our system is general and is not limited to specific types of devices. It can be applied in different use cases, such as smart homes, smart buildings, and industrial IoT. For instance, various interconnected devices in industrial IoT systems create a highly complex environment where it is difficult to manage and monitor the behavior of sensors or instruments. Therefore, our system is able to help the central controller, usually employed on the edge device, to automatically identify the IoT devices with their detailed profiles. The edge devices, in the scenario we consider, are usually the gateways that monitor traffic flow and authenticate or authorize the devices. Therefore, the edge device can be seen as the client in FL setup. We assume there are n clients that own data, denoted as $Cl = \{cl_1, \ldots, cl_n\}$. Each client has its on-device traffic dataset $\mathcal{D}_k = \{\boldsymbol{x}_j, y_j\}_{j=1}^{|\mathcal{D}_k|}$, where $|\mathcal{D}_k|$ is the total number of data samples in the dataset on client k, $\boldsymbol{x}_j$ indicates the jth sample, and y_j is the corresponding label. In general, the objective in FL task can be formulated as minimizing the loss function of the global model $\mathcal{M}_{\mathcal{G}}(\mathcal{W}_{\mathcal{G}})$:

$$\mathcal{M}_{\mathcal{G}}\left(\mathcal{W}_{\mathcal{G}}\right) = \frac{1}{N}\sum_{k=1}^{N}\mathcal{M}_k\left(\mathcal{W}_k; \mathcal{D}_k\right), \tag{1}$$

where $\mathcal{W}$ denotes the parameters of the model, and N indicates the total number of the clients.

3.2 Threat Model

Our target is a general IoT network deployed in smart homes and smart buildings. It typically comprises various IoT devices and one edge device. The edge device is usually a wireless or wired gateway router with IP-enabled devices in the network. Therefore, we assume that all the communication links among IoT devices and the channel between the edge device and the cloud are secure. Moreover, we assume when all IoT devices are initially connected to the network, they may possess vulnerabilities but are considered harmless, indicating that they have not been compromised by adversaries.

We take privacy issues into consideration with the help of FL. The sensitive data or messages collected on one device will not be shared with anyone. One of the adversaries to our approach is the unauthorized devices that are already compromised and try to perform reconnaissance behavior. FL4IoT is able to identify them in the first place by means of device identification. Another adversary can be the spoofed device trying to hide its identity within the IoT network. FL4IoT also is able to distinguish it from its original traffic. Therefore, in this article, rather than introducing a new authentication method or an intrusion detection system, we propose an approach to generate a lightweight fingerprint with a privacy-preserving method to prevent adversaries from compromising the vulnerable device and further exploiting it to launch advanced attacks. Our approach is agnostic to the characteristics of data in the network traffic as well as the type of protocols.

4 APPROACH

4.1 System Design

Figure 2 briefly depicts the architecture of the proposed system, named FL4IoT, with its position in the FL setup. There are two modules of FL4IoT: **Device fingerprinting (DF)** and **device identification (DI)**. DF is the module that fingerprints the traffic flow data and produces a set of vectorized fingerprints representing every authorized device; DI is the online module that identifies the devices from the newly observed records with the help of stored fingerprints. The module of DF is first trained offline with the traffic flows of authorized devices to generate the fingerprints, and each vectorized fingerprint can represent one device. DI is trained with another set of traffic flow of authorized devices and their generated fingerprints. The module of DI can be used during the inference phase to predict the new-observed traffic. DI is not needed to be retrained even if the fingerprint is updated. Moreover, each module has its own model aggregator implemented on the central node or cloud.

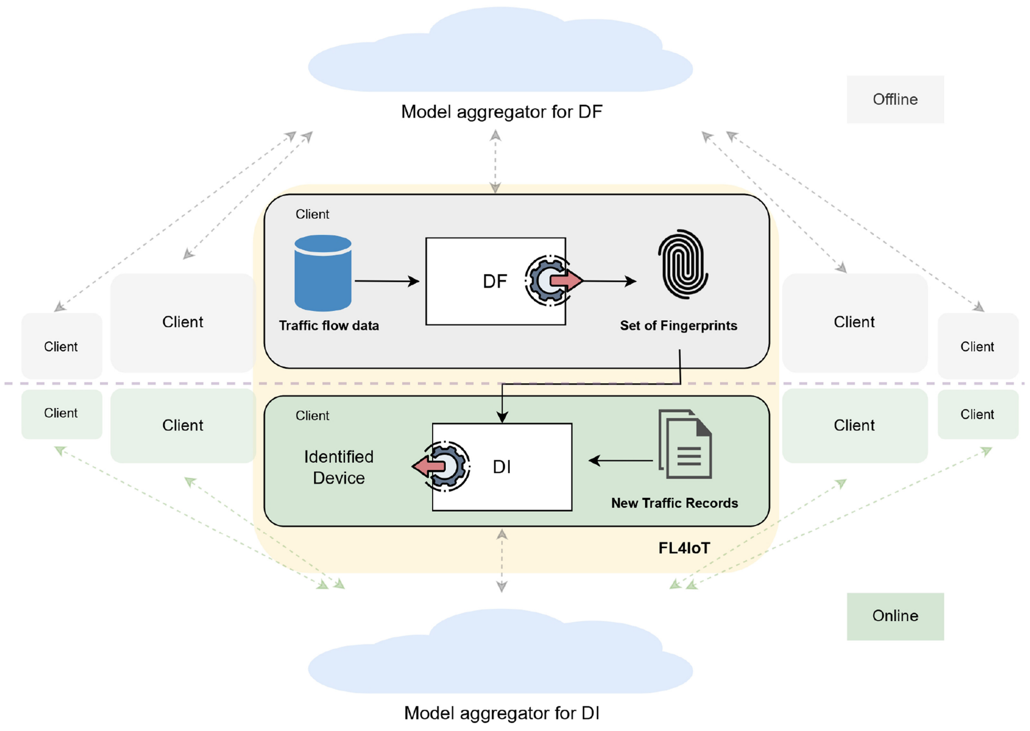


Fig. 2. An overview of FL4IoT architecture.

4.2 Device Fingerprinting

The goal of device fingerprinting is to build a reconstructed representation of a device. This representation is seen as the fingerprint that should be concise and unique. A fingerprint represents a specific device with only limited information. IoT network generates a huge amount of traffic traces, and also, they are usually high-dimensional data. Therefore, we proposed a generic unsupervised approach. Our method is flexible regarding the size of the dataset and the dimension of feature space and is adaptable to the IoT edge device.

We design an unsupervised learning algorithm to generate the vectorized device fingerprint. As depicted in Figure 3, we observe the traffic flow comprising of sets of packets as the input dataset denoted as $P = \{p_1, p_2, \ldots, p_n\}$. We first apply **Principal component analysis (PCA)**, the green part in the figure, to denoise the dataset and reduce the feature dimensions of the raw input. PCA is a linear transformation method for dimension reduction that can find the directions (components) and variables that explain the most variance in the data. We regard PCA as data pre-processing to facilitate the training steps. The output from PCA is fed into the encoder part of autoencoder, which is the yellow part in the figure. It is to reconstruct the features into a lower dimensional space as in the code layer. Finally, the reconstructed set of codes P' is clustered by the K-means algorithm to find the centroid of each cluster, denoted as f, as the representation of the devices.

4.2.1 Autoencoder. An autoencoder is employed to protect the generated fingerprint from being traced back to raw data. At the same time, the feature dimension of fingerprints is reduced to the appropriate size for communication and storage.

An autoencoder is a neural-network-based data compression algorithm that finds a compressed representation of the data together with a reconstruction function. The objective is known as reconstruction, and an autoencoder generally consists of an encoder and a decoder.

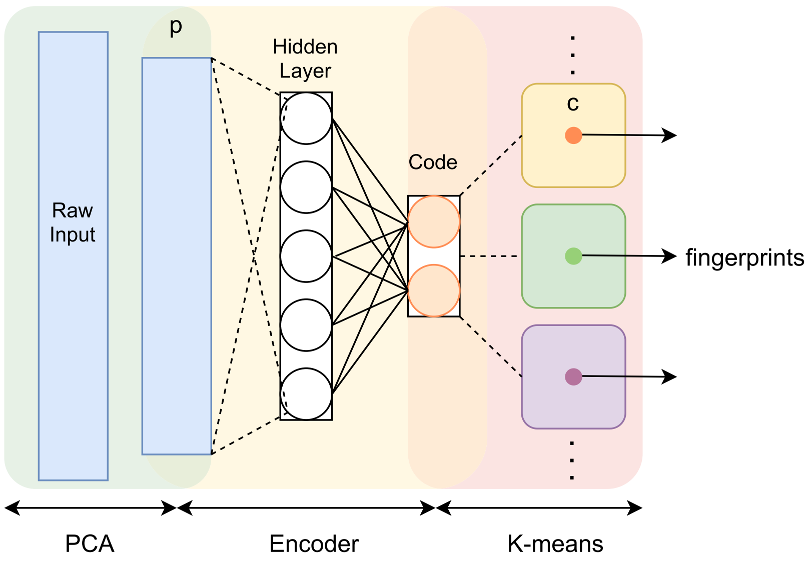


Fig. 3. The architecture of device fingerprinting includes three phases: PCA is used for denoising and feature dimension reduction, Encoder is used to generate the representation code of the input data, and K-means is applied to cluster the code and find the centroid as the fingerprint of the device.

(a) An encoder learns the data representation in lower-dimensional space, i.e., extracting the most salient features of the data. The encoder can mathematically be described as:

$$p' = h_e(p), \tag{2}$$

where p' is the learned code by the encoder h_e from input data point p.

(b) A decoder learns to reconstruct the original data based on the learned code by the encoder. Mathematically, it can be represented as:

$$\hat{p} = h_d(p'), \tag{3}$$

where $\hat{p}$ is the reconstructed data by the decoder h_d based on the learned representation code p'.

The loss function $\mathcal{L}$ to the autoencoder is defined as follows:

$$\mathcal{L}(p, \hat{p}) = \frac{1}{N} \sum_{i=1}^{N} \|p_i - \hat{p}_i\|^2, \tag{4}$$

where N is the number of training samples, $p = (p_1, \ldots, p_N)$ are the input data points, and $\hat{p} = (\hat{p}_1, \ldots, \hat{p}_N)$ are the reconstructed data points by the decoder.

In this article, we extend and apply the idea from Song et al. [31] to generate fingerprints combining an autoencoder and the K-means clustering algorithm. The benefit is that it helps every newly constructed code to get close to its cluster center, which means the fingerprints are distilled more precisely. The output codes from the encoder are taken as the input to the K-means model. K-means works iteratively and aims to assign each data sample to one cluster minimizing the distance between the cluster centroid and every point within the cluster. The centroid $c_i^* \in \mathbb{R}^{\mathbb{N}}$ can be formulated as follows:

$$c_i^* = \arg\min_{c_j^{t-1}} \left\| p_i' - c_j^{t-1} \right\|^2, \tag{5}$$

ALGORITHM 1: Algorithm of Device Fingerprinting

```
Input  : Dataset: $P = \{p_1, \dots, p_n\}$, Hyper-parameters: Number of components: $com$, Number of clusters:
         $k$, Learning rate: $\eta$, Number of epochs: $E$.
Output : Set of centroids: $C$
1  $C = \{c_1, \dots, c_k\}$ /* Initialize each cluster centroid */
2  $P_{pca} = PCA(P, com)$
3  $e = 0$
4  repeat
5      Update the mapping network by minimizing Equation (5) with stochastic gradient descent and get
       $p_i'$ as in Equation (1).
6      for $j \in \{1, \dots, k\}$ do
7          Let $c_j$ be the mean of the $j$th cluster.
8      end
9      Partition $P' = \{p_1', \dots, p_n'\}$ into $k$ clusters and update the sample assignment $c_j$ via Equation (4).
10     $e = e + 1$
11 until $e \geq E$;
```

where p_i' indicates the code generated by autoencoder and c_j^{t-1} represents the centroid of the jth cluster in the $t-1^{th}$ iteration. In our case, the clusters represent different devices, and the centroid from each cluster is regarded as the fingerprint for each device.

According to Reference [31], the low dimensionality of the code produced from the encoder resolves the drawback of the clustering algorithm. Most of these algorithms work effectively in a large-scale dataset but would lose their effectiveness on high-dimensional data [5]. Therefore, the loss function of the autoencoder is combined with the K-means model, which results in the following objective:

$$\min_{W,b} \gamma \cdot \sum_{i=1}^{N} \|p_i' - c_i^*\|^2 + \frac{1}{N} \sum_{i=1}^{N} \|p_i - \hat{p}_i\|^2, \tag{6}$$

where γ indicates the tradeoff importance between the encoder and K-means. This objective function ensures that the data representations from the encoder are close to their corresponding centroids in the cluster, and meanwhile, the reconstruction error is small. Note that Reference [31] contains a typo in the form of a sign error in the objective function (6).

Our method for device fingerprinting is explained in Algorithm 1. The input to the algorithm consists of the dataset P and several hyper-parameters including the number of components for PCA, the number of clusters k for K-means, the learning rate η, the tradeoff parameter λ, and the number of epochs E. The output will be the fingerprints, which are the set of centroids C of the devices. The centroids are initialized randomly, and PCA is applied to the input P picking out the first com principal components as shown in lines 1 and 2. Subsequently, the training starts with the objective function of minimizing the loss (line 5). In each round, the set of codes, P', is clustered to find the centroid c_j in line 7. Then, in line 9, we reassign the $p' \in P'$ into a new cluster. The training process stops after all the epochs are completed.

4.3 Device Identification

Regarding to device identification, we apply a feed-forward neural network in supervised learning to identify the devices in IoT network. We illustrate our method in Figure 4. A new set of labeled traffic X and Y are fed into the network. To evaluate the effectiveness of the generated fingerprints, we use them in the training process. The objective of the network is to maximize the similarity

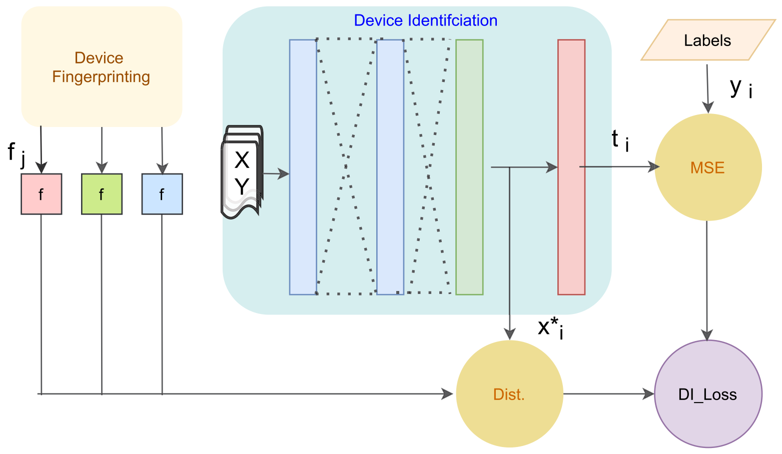


Fig. 4. The model design for device identification uses generated fingerprints. We take input (X, Y) to obtain x_i^*. We calculate the distance between x_i^* and fingerprints. We get t_i from the output layer to calculate the Mean Squared Error (MSE) with labels y_i.

between the data samples and the fingerprint f_j of the corresponding device generated from the proposed method. In other words, the model learns to minimize the distance between data and the class centroid. The loss function is defined as:

$$DI\_loss(X, Y) = \min_{W,b} \lambda \cdot \sum_{i=1}^{N} \left\| x_i^* - f_j \right\|^2 + \frac{1-\lambda}{N} \sum_{i=1}^{N} \left\| t_i - y_i \right\|^2, \tag{7}$$

where x_i^* is the ith output from the second last layer in the neural network, f_j denotes the fingerprint of the jth device, t_i denotes the output target, and y_i is the ground truth. Also, λ is the control factor for the impact caused by the generated fingerprint.

4.4 FL4IoT in Federated Learning

By taking advantage of FL, we propose to implement the model on the security gateway where the data from the devices are aggregated and the whole IoT network is monitored. Therefore, we utilized the FL approach to implement the distributed learning of models from several clients. As the workflows of FL show in Figure 5(a) and (b), we have different strategies for device fingerprinting and identification.

4.4.1 Device Fingerprinting in FL. Regarding device fingerprinting, we assume that the input data to the proposed model are unlabeled, which is close to the real-world setup. Thus, our proposed model is trained in an unsupervised manner. However, the research topic of clustering in FL has yet to be explored deeply because of the heterogeneity in behavior and hardware of the IoT device. The decentralized data can be expected to be skewed and imbalanced due to the various clients. It makes the averaging of the centroid for clustering more difficult. Soliman et al. [30] first proposed an adaptive K-means clustering method for the FL setup.

Let n be the number of clients that own data, and let $Cl = \{cl_1, \ldots, cl_n\}$ be the set of clients. The global model is denoted by M_g. As illustrated in Figure 5(a), the process of device fingerprinting in FL can be divided into five steps: **(1)** An initial model M_g from the central node is first deployed to each client in Cl that has joined FL. **(2)** A model M_{cl} is trained locally with an on-device dataset D_{cl}, and a set of fingerprints $F_{cl} = \{f_1, \ldots, f_n\}$ is generated. The set of fingerprints is the representation

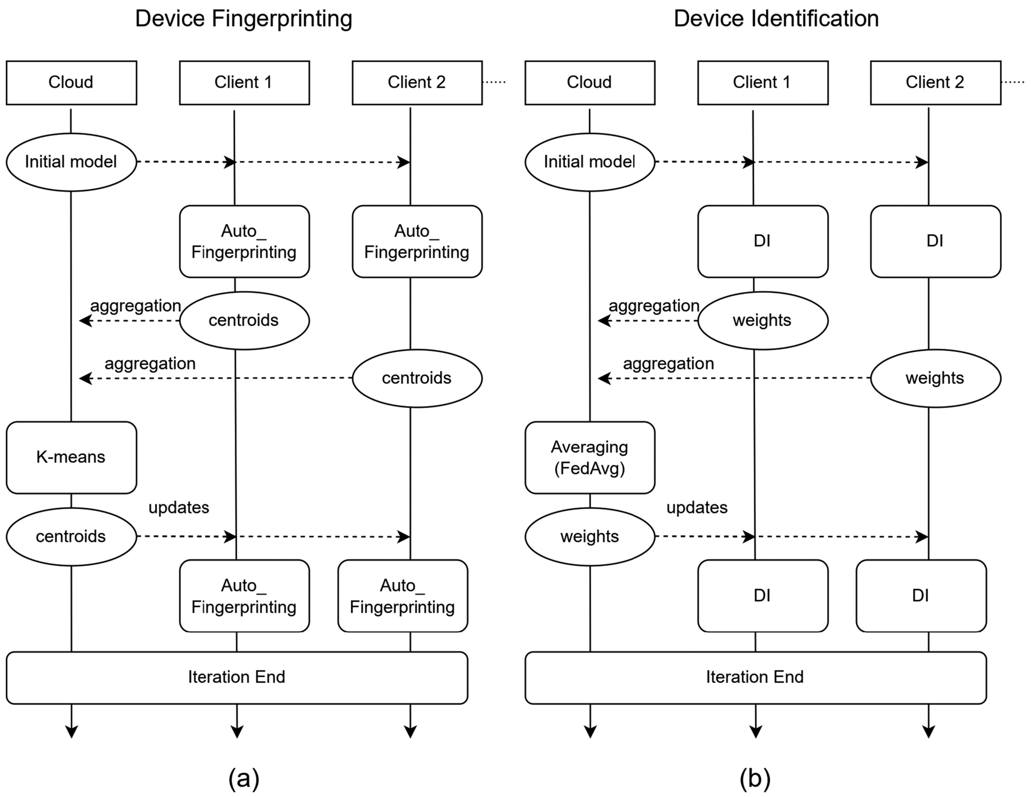


Fig. 5. The workflow of (a) Device fingerprinting and (b) Device identification in FL setup.

ALGORITHM 2: Algorithm of Federated K-means

```
Input : The union $C_u$ of all fingerprints from all the clients. Number of clusters: $K = \{k_1, \ldots, k_N\}$
        from clients 1 through $N$.
Output: Updated Centroids: $C_{n,u}$ for $1 \leq n \leq N$
1 for $k \in K$ do
2     $C_{n\_u} = Kmeans(C_u, k)$
3 end
```

of a set of centroids for n devices. **(3)** When the central node calls for the aggregation, the clients send back the set of centroids F_{cl} to the central node. **(4)** After collecting the centroids from the clients, Federated-Kmeans as illustrated in Algorithm 2 is performed on the central node to find the updated centroids, denoted as C_{cl_u} for each client cl. **(5)** Finally, the algorithm returns k centroids back to the corresponding client, and a new round of training starts.

4.4.2 Device Identification in FL. Compared to device fingerprinting, device identification is much simpler in the FL setup, since we adapt supervised learning to build a feed-forward neural network. As depicted in Figure 5(b), the main steps are similar between the two approaches. However, in this phase, the clients return the weights w instead of centroids to the cloud. Then, we apply the FedAvg algorithm for the aggregation. FedAvg simply computes the average of the weights W_g integrated from clients Cl. It can be formulated as follows:

$$W_g = \frac{1}{N}\sum_{n=1}^{N} w_n. \tag{8}$$

Finally, the models M_{cl} on the clients are updated by the global weights W_g.

4.4.3 Computational Complexity. The complexity of computing device fingerprinting and device identification should be discussed independently. Regarding device fingerprinting, in every local training round, FL4IoT executes an autoencoder and K-means to generate and group the codes in P'. Therefore, the computational complexity of the device fingerprinting is the summation of the two phases, that is, $O(e(w + kd))$, where w is the number of parameters of the model, k is the number of the clusters, d is the time for computing distance between the codes, and e is the number of the training epoch. Regarding to FL setup, the overall complexity takes the additional model aggregation phase into account, so it becomes $O(g(w_{up} + cd + e(w + kd)))$, where g represents the global communication epoch, w_{up} indicates the model updates, and c is the number of centroids. In contrast, the computational complexity of device identification is simpler, that is, $O(e(w + k))$. The complexity correlates with the size of the neural network, the number of fingerprints involved, and the number of training epochs. When putting device identification in FL setup, the complexity takes the number of global epochs additionally into account. So the complexity becomes to $O(g(e(w + k)))$.

5 IMPLEMENTATION AND EVALUATION

We implement FL4IoT in Python. We utilize PyTorch, which is an ML library, especially for deep learning. With the help of PyTorch, we implement the autoencoder embedded with K-means and the neural network from scratch. We use Scikit-learn tools to build PCA and split the dataset. Regarding applying FL, we implement our aggregation algorithm by utilizing PySyft library that is based on the PyTorch framework. PySyft provides a toolkit for FL with several additional features.

5.1 Experiment Setup

5.1.1 Setup. We conduct experiments on a Kubernetes cluster equipped with Nvidia-Gtx-2080ti GPU and 10 GB CUDA memory on the cloud service provided by RISE ICE cloud service. The evaluation consists of four sets of experiments: the quality of device fingerprinting in the centralized and FL settings and the performance of device identification in centralized training and FL setups.

5.1.2 Dataset. In this article, we evaluate FL4IoT with three open datasets: *N-BaIoT* dataset [18], *IoTSentinel* [21], and *UNSW BoT-IoT* [29]. The denotations of each device are listed in Table 2. *N-BaIoT* dataset consists of real-world network traffic flow from nine commercial IoT devices. Please check Reference [18] for more details on these nine devices. It includes security cameras (Provision PT 838 Security Camera, Provision PT737E Security Camera, SimpleHome XCS7 1002 WHT Security Camera and SimpleHome XCS7 1003 WHT Security Camera), two doorbells (Danmini and Ennio), an Ecobee thermostat, a Philips B120N/10 baby monitor, and a Samsung SNH 1011 N webcam. There are 23 incremental statistical features, such as the mean and the variance of the packet size, and the amount of time between packet arrivals. The features are extracted from five different time windows (100 ms, 500 ms, 1.5 sec, 10 sec, and 1 min). This dataset is originally collected for IoT Botnet detection. It captures both benign and malicious traffic carried by two botnets (Mirai, BASHLITE). We mainly consider the benign traffic to represent device behavior in the network, and different devices will be selected in different experiment sets.

Table 2. Notation for the Chosen Devices in Experiments

N-BaIoT				
PT1: Provision PT-838 Security Camera	PT2: Provision PT-737E Security Camera	DB: Danmini Doorbell	XC1: SimpleHome XCS7-1002-WHT Security Camera	XC2: SimpleHome XCS7-1003-WHT Security Camera
UNSW BoT-IoT				
BM: Withings Smart Baby Monitor	AE: Amazon Echo	NW: Netatmo Welcome Camera	MS: Belkin WeMo motion sensor	
IoTSentinel				
DT: D-Link Switch	WL: WeMo Link	HS: Hue Switch	WS: WeMo Insight Switch	

The *IoTSentinel* dataset consists of network traffic traces captured from 31 IoT devices with four types of connection technology, such as WiFi, ZigBee, and Z-Ware. The dataset covers common device classes related to smart lighting, home automation, security cameras, and so on. The traffic is collected on a security gateway. Miettinen et al. [21] propose a method to extract 23 features as a representation for each packet. Out of the 23 features, 16 are binary, which are indicators of the chosen protocols. These 16 protocols are typically used over WiFi. Other features include packet size, IP address, and so on. The features form an n by 23 matrices recording n packets received at a device. In Reference [21] this matrix is used as the device fingerprint. In contrast, we regard these features as the input to our device fingerprinting model.

The *UNSW BoT-IoT* is a real-world dataset published by University of New South Wales. The dataset consists of the network traffic captured by Sivanathan et al. [29] from setting up a smart environment infrastructure with campus facilities, including 28 IoT devices such as cameras from different vendors, switches, motion sensors, and so on. These devices communicate with Internet servers via a gateway. They start logging traffic from 1-Oct-2016 to 13-Apr-2017 for 26 weeks. The traffic is stored as a PCAP file containing packet header and payload information. We use Miettinen et al.'s [21] method to extract features from the raw data to make sure that the features have the same structure as the second dataset.

5.1.3 Models. First, we configure the device fingerprinting of FL4IoT to be a three-hidden-layer autoencoder (symmetric setting in both encoder and decoder) combined with a K-means algorithm. Some fixed hyper-parameters for the model are set. For example, the tradeoff γ between reconstruction error and cluster error defined in Equation (6) is set to 0.4, and the number of training epochs E is set to 100. Regarding the number of clusters k and the learning rate η, we have different setups for each experiment set. Second, we configure the device identification of FL4IoT to be a feed-forward neural network with 3 hidden layers. We choose *ReLU* to be the activation function for the first 3 layers and Sigmoid activation for the last one. We consistently use *Adam* optimizer for both models. The control factor λ defined in Equation (7) is set to 0.4. Besides, the setting for FL setup is 5 local training epochs and 50 global communication epochs.

5.2 Results and Discussion

To have an overall evaluation to FL4IoT, we conduct three main experiment sets, including (i) Quality of Device Fingerprinting, (ii) FL4IoT in centralized learning, and (iii) FL4IoT in federated learning. When we evaluate the performance of FL4IoT in centralized learning, we measure the performance in three scenarios: (1) different device types, (2) different vendors but the same device

Table 3. Performance Comparison between FL4IoT and Baselines in Centralized Learning

Experiments	DF	DI		Base_RF [20, 21]		Base_KNN [22, 25]		Base_NN [12]	
	Purity	Test Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score	Accuracy	F1 Score
N-BaIoT									
PT/DB	65.63%	99.68%	0.9953	99.97%	1.0	99.78%	1.0	99.38%	0.9907
PT/XC	83.32%	99.86%	0.9978	99.99%	1.0	99.95%	1.0	99.85%	0.9963
PT1/PT2	50.80%	95.57%	0.9429	98.28%	0.9838	93.45%	0.9322	93.02%	0.9126
UNSW BoT-IoT									
BM/AE	64.98%	94.57%	0.9555	97.06%	0.9611	94.95%	0.9525	94.01%	0.9515
BM/NW	80.02%	99.93%	0.9992	99.95%	1.0	99.83%	1.0	99.79%	0.9976
BM/MS	67.71%	95.28%	0.9668	97.28%	0.9632	95.97%	0.9613	94.47%	0.9576
IoTSentinel									
DT/WL	69.93%	94.93%	0.9452	98.51%	0.9818	97.41%	0.9731	90.06%	0.9029
DT/WS	67.33%	93.31%	0.9481	97.30%	0.9671	96.14%	0.9603	91.29%	0.8802
DT/HS	75.18%	99.23%	0.9948	99.58%	0.9923	99.43%	0.9911	99.06%	0.9937

types, and (3) different modules from the same vendor. These three scenarios are binary classification problems. We also evaluate a hybrid comparison in a centralized setting, which is a multi-class classification problem discussed in Section 5.2.3. Furthermore, we demonstrate FL4IoT is capable to detect spoofed devices, which is shown in Section 5.2.4. When we evaluate the performance of FL4IoT in federated learning, except for the general model performance, we also observe the impact caused by the number of clients, which is discussed in Section 5.2.7.

5.2.1 Quality of Device Fingerprinting. We first evaluate the quality of our approach to device fingerprinting in both centralized and federated settings. Note that the goal of this experiment is mainly to evaluate the quality of the clustering. Therefore, in this experiment, we are not comparing with any other baselines; we rather demonstrate the strength of the fingerprints generated by the proposed method by showing its utility in device identification, which will be discussed in the following sections. The overall performance is shown in Table 3. We measure the output instances from clusters with the corresponding labels and compute a purity value. Purity is the ratio between the dominant class in the cluster and the size of cluster. High purity means that the clusters are so pure that each cluster only contains data from one class. It can be formulated as:

$$\text{purity} = \frac{1}{N} \sum_{k} \max_{j} |\omega_k \cap c_j|, \tag{9}$$

where N denotes the number of data samples, k is the number of clusters, ω_k indicates the data in cluster k, and c_j is the ground truth of class j.

As shown in Table 3, we conduct three experiments per dataset. In every experiment, we pick two devices according to their device type, vendor, and model. The notation for each device is listed in Table 2. We take *N-BaIoT* dataset as an example. *PT/DB* indicates the experiment conducted with two devices that have different device type, where *PT* is a security camera and *DB* is a doorbell; *PT/XC* represents the experiment conducted with two security cameras produced by two different vendors; *PT1/PT2* indicates the experiment conducted with two security cameras from the same vendor but different product modules. Note that there are no devices that are produced by the same vendor but different product modules in *UNSW BoT-IoT* and *IoTSentinel*, so we do not include experiments for this scenario in these two datasets. We set the hyper-parameters such as the various learning rates η in three different datasets ($1e-4$ for *N-BaIoT* and *IoTSentinel*, $1e-3$ for *UNSW BoT-IoT*), and the cluster number k for K-means is set to 2.

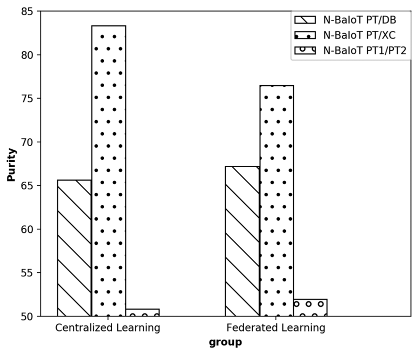


Fig. 6. The purity performance in device fingerprinting.

According to Table 3, Table 4, and Figure 6, we find the performance of purity of clusters are over 60% both in centralized and federated manner except for the case of *N-BaIoT PT1/PT2.* This phenomenon is expected, because two devices from the same vendor but different modules behave much similarly compared to the other devices. It is much more difficult to be clustered. Even so, Table 3 shows that although the model for device fingerprinting gets only 50% in purity in *N-BaIoT PT1/PT2* group, the generated fingerprint can help FL4IoT achieve over 95% accuracy in the same case.

5.2.2 Centralized Device Identification. In every experiment, we have a new set of data different from the one used in device fingerprinting and a set of fingerprints that were generated from the previous experiments. We split the dataset into the training set and the testing set. The training set is used to train a model with the loss function described in Equation (7). In the following experiments, we adopt accuracy and F_1-score to evaluate our model. Accuracy is defined as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{10}$$

where TP denotes the number of true positives, TN denotes the number of true negatives, FP denotes the number of false positives, and FN denotes the number of false negatives. The F_1-score is defined as:

$$F_1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}, \tag{11}$$

where

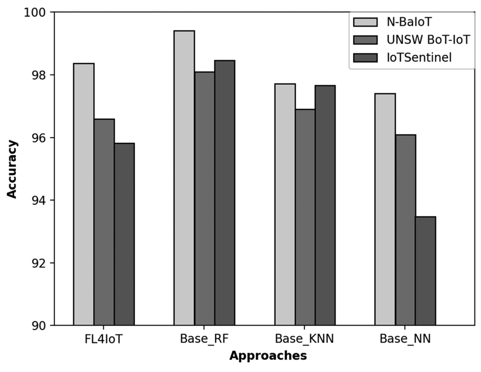
$$\text{precision} = \frac{TP}{TP + FP}, \tag{12}$$

$$\text{recall} = \frac{TP}{TP + FN}. \tag{13}$$

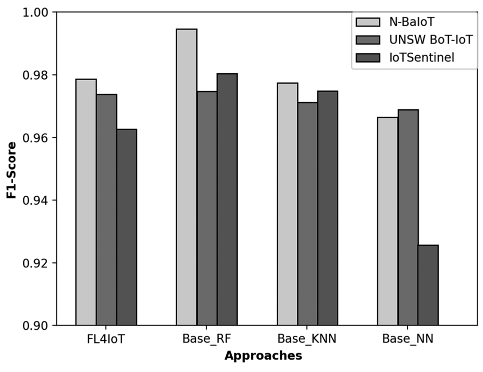
F_1-score conveys the balance between the precision and the recall. It shows how exact and complete the model performs.

We compare FL4IoT with three baselines of device identification. One is Random Forest used in Reference [21], denoted as *Base_RF,* one is K-nearest Neighbors, which is used in References [22, 25], denoted as *Base_KNN,* and a feed-forward neural network consisting of three layers (1 input layer, 1 hidden layer, and 1 output layer) that has been utilized by Reference [12], denoted as *Base_NN.*

An overall centralized performance of FL4IoT is shown in Table 3. Figures 7(a) and 7(b) show the comparison of accuracy and F1-score among FL4IoT, *Base_RF, Base_KNN,* and *Base_NN* on three datasets. From Table 3, we can observe both accuracy in training and testing are over 93% in every



(a)



(b)

Fig. 7. Performance comparison of FL4IoT in centralized learning on three datasets: (a) Accuracy performed in device identification (b) The F_1-Score performed in device identification.

case. Especially, FL4IoT can achieve over 99% accuracy in some cases, such as *N-BaIoT PT/DB*, *N-BaIoT PT/XC*, *UNSW BoT-IoT BM/NW*, and *IoTSentinel DT/WS*. From this experiment, the results of purity of the clusters and the performance of identification accuracy are positive correlation. It is expected, since the more the data distributions of the two devices are deviated, the better result of accuracy it performs. It also reflects that it is easier to identify devices that are different types or produced by the different vendors than the same device type and produced by the same vendors but they are different modules.

Observing from Figures 7(a) and 7(b), we find *Base_RF* and *Base_KNN* are two competitive baselines that work slightly better than FL4IoT. It is expected, because Random Forest and K-nearest neighbors, which are traditional ML methods, typically work well on the small datasets. On the contrary, DL-based method perform better on the big dataset. Overall, FL4IoT works comprehensively better than *Base_NN*. However, the traditional ML-based methods such as Random Forest and K-nearest are very difficult to be applied in federated manner. Adapting traditional ML models to FL is a rather interesting and challenging research area that can be explored further in the future study. Therefore, we highlight that one of contributions of this article is we propose a device fingerprinting and identification scheme that can be applied in both settings.

5.2.3 *Hybrid Comparison.*

In this experiment, we measure how FL4IoT performs on mixed-typed devices. We use all devices from *N-BaIoT*, listed in Table 2, to conduct an experiment. There are totally five devices: two sets of security cameras produced by same vendors but different models (*PT1/PT2* and *XC1/XC2*) and one doorbell (*DB*). We first generate fingerprints for every device and use them in device identification. The result is demonstrated as confusion matrix in Figure 8(a). The overall accuracy is 96.28% among all 22,107 data samples. In the confusion matrix, each row of the matrix represents the data samples in ground truth, while each column represents the data samples in a predicted class. Therefore, we find that the trained model has the best performance on identifying doorbell, because it gets the best precision (99.51%). Also, the classification between *PT1* and *XC1* achieves 100% accuracy. This experiment shows FL4IoT performs well in both mixed types and grouped subset.

5.2.4 *Spoofed Device Detection.*

In this experiment, we investigate whether FL4IoT can detect compromised or spoofed devices. Therefore, we choose the data that are malicious traffic carried by Mirai Botnet from *N-BaIoT*. More specifically, the data are traffic of automatically scanning vulnerable devices from *XC1*, denoted as *XC_A*. And, we use the benign traffic data from three

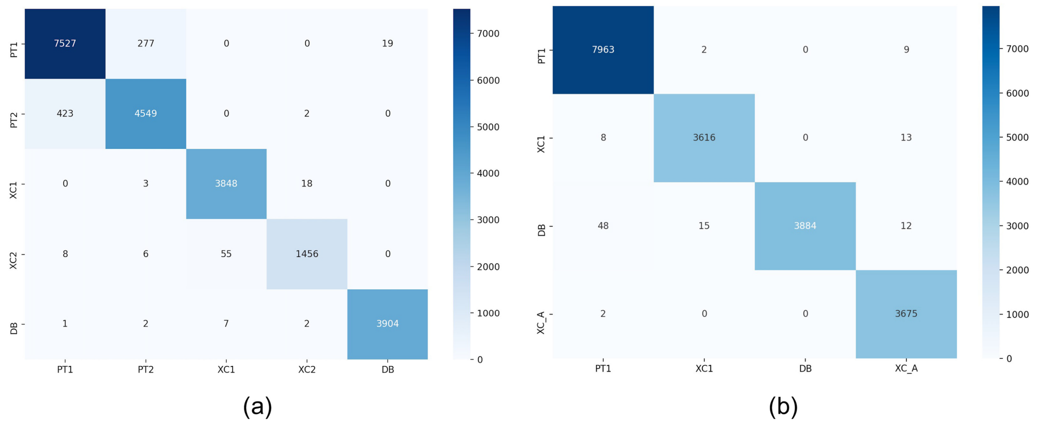


Fig. 8. The confusion matrix shows the FL4IoT's performance on two experiments: (a) hybrid comparison and (b) spoofed device detection among different types of devices from *N-BaIoT*. The denotation of devices are listed in Table 2. The bar of color gradient represents the number of data samples.

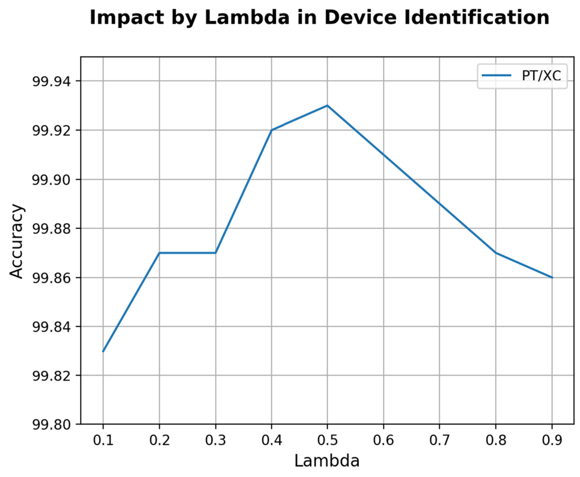


Fig. 9. Test Accuracy in FL4IoT along different number of λ.

devices (*PT1*, *XC1,* and *DB*) in classification. The confusion matrix is shown in Figure 8(b). The overall accuracy achieves 99.43% among 19,247 traffic data samples. Especially, from the confusion matrix, we find that only 13 malicious traffic data are identified as benign traffic from *XC1*, and the precision of detecting *XC_A* can reach 99.08%. Therefore, the results show FL4IoT is capable to detect spoofed or compromised devices from the suspicious traffic generated from them.

5.2.5 Impact by Lambda. To measure how much generated fingerprint should be involved in FL4IoT, we investigate the accuracy changes along with various values of λ in Equation (7), since λ is a control tradeoff between the fingerprint and the labels. We take experiment in the *N-BaIoT PT/XC* group as an example. As shown in Figure 9, we observe that when the lambda is set to 0.5, our approach can perform best, and it performs worse for both extreme values in λ, such as 0.1 and 0.9. This phenomenon can be explained that both the fingerprints and labels are critical for FL4IoT.

5.2.6 Efficiency of Federated Learning. To observe the performance and efficiency of FL, we conduct a set of experiments to evaluate FL4IoT in a federated setup. First, to set up federated

Table 4. Performance Comparison between FL4IoT and Baselines in Federated Learning

Experiments	FL4IoT			Base_NN		Base_FLKmeans [11]	
	DF	DI					
	Purity	Accuracy	F-Score	Accuracy	F-Score	Accuracy	F-Score
N-BaIoT							
PT/DB	67.18%	99.66%	0.9929	99.11%	0.9925	99.51%	0.9914
PT/XC	76.47%	99.92%	0.9989	99.89%	0.9981	99.77%	0.9962
PT1/PT2	51.96%	93.89%	0.9211	92.27%	0.9022	92.77%	0.9175
UNSW BoT-IoT							
BM/AE	80.22%	95.19%	0.9438	93.74%	0.9482	90.63%	0.9088
BM/NW	72.37%	99.51%	0.9935	99.6%	0.9928	94.85%	0.9431
BM/MS	80.35%	96.03%	0.9649	91.39%	0.9277	96.64%	0.9575
IoTSentinel							
DT/WL	70.15%	94.67%	0.9438	92.74%	0.9166	93.15%	0.9219
DT/WS	82.48%	92.12%	0.9183	91.42%	0.8855	92.02%	0.9022
DT/HS	74.89%	99.03%	0.9944	98.36%	0.9889	98.77%	0.9892

learning, we design our model structure on the cloud and deploy the initial model to the clients. For every experiment, we distribute the dataset to two clients and set the parameters the same way as in the previous experiments in centralized learning. And, we compare FL4IoT with two baselines. One is a general neural network, denoted as *Base_NN*, which is able to be applied in FL setup, and FLKmeans proposed by Kumar et al. [11], denoted as *Base_FLKmeans*, which is a federated K-means algorithm. It is to investigate the performance of the proposed federated K-means in FL4IoT. In this experiment set, we consider both general model performance and the impact from the number of clients.

In this set of experiments, we evaluate overall FL4IoT performance in FL. The detailed results are shown in Table 4. To be simpler, we demonstrate the result with the setup of two clients joining, even though FL4IoT is applicable to multiple clients. Therefore, there are 10 additional data points sampled from the Gaussian distribution on the central node to be fed into K-means for clustering, together with aggregated fingerprints. Moreover, we show more details of multiple clients in the sub-experiment.

Figures 10(a) and 10(b) show the comparison between FL4IoT and two baselines: *Base_NN* and *Base_Kmeans*. We can observe that the performance of FL4IoT in accuracy and F_1-Score are both the best among the three methods, and it performs approximately matching with how it does in centralized learning. From Table 4, we can find that test accuracy can reach over 92% in every case, especially 99% in some cases (*N-BaIoT PT/DB*, *N-BaIoT PT/XC*, *UNSW BoT-IoT BM/NW*, and *IoTSentinel DT/WS*). Compared to *Base_NN*, the better performance of FL4IoT and *Base_FLKmeans* also show the effectiveness and utility of the generated fingerprints in device identification. Moreover, our proposed federated K-means algorithm outperforms *Base_FLKmeans*, which means our proposed scheme of federated K-means is a better choice in our scenario.

5.2.7 *Impact from the Number of Clients.*

In this experiment, we aim to investigate the performance impact brought by the change in the number of clients. Therefore, we conduct a set of experiments with different numbers of clients and take the experiment in the *UNSW BoT-IoT BM/NW* group as an example. We observe the accuracy when the number of clients ranges from 2 to 15. The result is shown in Figure 11. When there are three clients, the model achieves 99.9%

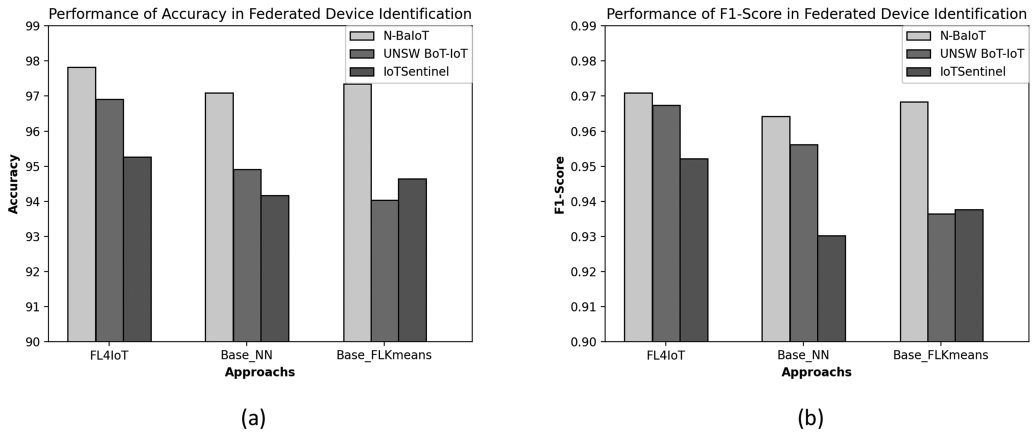


Fig. 10. Performance comparison of FL4IoT in federated learning on three datasets: (a) Accuracy performed in device identification (b) The F_1-Score performed in device identification.

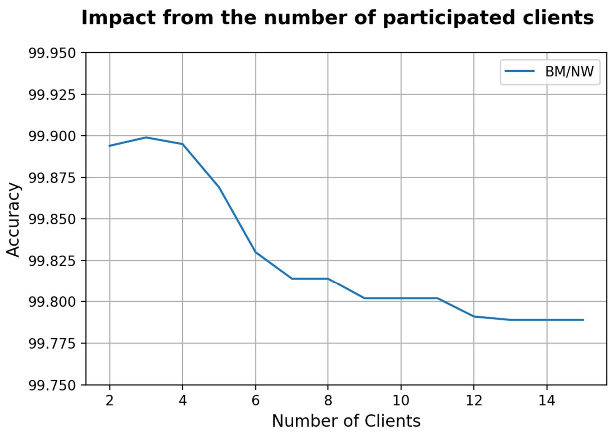


Fig. 11. Comparison of accuracy of FL4IoT along different number of clients in FL.

accuracy. The result shows as we expect, that the performance gets slightly lower as the number of clients increases.

6 SECURITY AND PRIVACY ANALYSIS

FL4IoT brings a privacy-preserving way to address the problem of device fingerprinting and identification. The purpose of FL4IoT is to help identify IoT devices newly joining the network and avoid attacks such as the NASA system hack [23]. The empirical result shows that FL4IoT can detect spoofed or compromised devices. We discuss the advantages and disadvantages of FL4IoT from different aspects. In the aspect of the algorithm, FL4IoT assumes that the number of legitimate devices remains the same. It means that if there is a new legitimate device, then FL4IoT needs to be retrained. However, the merit of FL4IoT is that the generated fingerprints are not just the features extracted from the raw traffic traces, but rather another representation reconstructed by FL4IoT. It increases the difficulties for attackers to retrieve the raw data for a target device, but at the same time, lowers the risks during the process of aggregation is performed in FL4IoT.

The fingerprint generated by FL4IoT is a vectorized representation that is more lightweight to be stored in any edge device. This ensures the privacy as the data never leave the perimeter of the organization. However, compared with the centralized approach, there are still some risks. According to Reference [17], there is no full guarantee for data privacy, even if the data is anonymized. Due to the aggregation for updates of parameters (such as FedAvg), frequent communication between clients and the cloud is needed. Therefore, the algorithm is potentially vulnerable to adversarial attacks where a compromised client can poison the model [16].

From the clients' perspective, the cloud may not be a trusted entity. There are some privacy techniques such as perturbation of local models where noise is added to the parameters; from the cloud's perspective, adversaries may manipulate the result of averaging and aggregation such as poisoning attacks [32]. For FL4IoT, it is possible that the aggregated centroids or the parameters are poisoned. Therefore, even though it is out of the scope of this article, we agree a well-designed method for averaging and the protection of the cloud is crucial.

7 CONCLUSION AND FUTURE WORK

Devastating consequences can arise from the presence of unidentified devices in the network, highlighting the need to identify and fingerprint IoT devices that are connected to private or critical networks. Machine learning-based device fingerprinting and identification methods usually rely on collecting substantial data from IoT networks and sending it to a central cloud. Nonetheless, private IoT data cannot be shared with the central cloud in many sensitive situations. Therefore, we introduced a two-phased system called FL4IoT, which involves generating fingerprints and identifying devices by analyzing their traffic behavior. The fingerprint generated by FL4IoT is comparatively lightweight and can be used to identify devices by device types, vendors, and product modules. We evaluated FL4IoT with real-world datasets, and the empirical results show that it achieves 99% accuracy and F_1-Score in some cases and over 93% in every case. Moreover, FL4IoT has the capability to identify spoofed devices with a precision rate exceeding 99%. However, FL4IoT is applicable in federated learning. From the experimental results, we observed that FL4IoT performs similarly in centralized and federated settings and outperforms the other baselines.

The use of ML/DL techniques to solve a general security problem is becoming prevalent with the increasing scale and complexity of the networks and systems [6, 34]. Fog/edge computing also becomes a big trend in the next generation due to the enforcement of GDPR. It brings a large number of advantages, including privacy preservation, fast response, short latency, cheap cost, and so on. Gill et al. [6] provides a high-level summarization of the potential challenges related to fog/edge or serverless computing. Except for these, implementing ML/DL on these comparatively resource-constrained devices is challenging already. For example, heterogeneous IoT devices lead the produced data to be in **non-independent and identical distribution (non-IID)**, hindering the ML/DL applications. Many open research directions are left for future works, such as performing on-device learning without losing significant utility and learning a decent model on non-IID data. We believe the growth of fog/edge/serverless computing will also impact the problem of device fingerprinting and identification, which is an exciting research topic worth exploring.

However, the previous discussions on fog/edge computing also facilitate the emergence of federated learning discussed in this article. Although FL has many benefits, it also brings new specific challenges and vulnerabilities. Compared to centralized learning, FL models are directly prone to adversarial attacks. Sharing models also leads to further issues, such as how to protect the confidentiality of the model. Detecting compromised clients or adversarial inputs in an efficient manner remains an unresolved matter. A solution needs to be computationally cheap without significantly degrading the model's performance. We plan to address these issues in future work and improve the security of our aggregation algorithms by developing mitigation techniques against adversarial attacks.

REFERENCES

[1] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis et al. 2017. Understanding the Mirai Botnet. In *Proceedings of the 26th USENIX Security Symposium (USENIX Security'17)*. 1093–1110.

[2] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. 2018. Behavioral fingerprinting of IoT devices. In *Proceedings of the Workshop on Attacks and Solutions in Hardware Security (ASHES'18)*. Association for Computing Machinery, New York, NY, 41–50. DOI: https://doi.org/10.1145/3266444.3266452

[3] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*, Association for Computing Machinery, Dallas, Texas, 1175–1191. DOI: https://doi.org/10.1145/3133956.3133982

[4] Yushi Cheng, Xiaoyu Ji, Juchuan Zhang, Wenyuan Xu, and Yi-Chao Chen. 2019. DeMiCPU: Device fingerprinting with magnetic signals radiated by CPU. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19)*, Association for Computing Machinery, 1149–1170. DOI: https://doi.org/10.1145/3319535.3339810

[5] C. Chio and D. Freeman. 2018. *Machine Learning and Security- Protecting System With Data and Algorithms*. O'Relliy Media Inc. https://mlsec.net/.

[6] Sukhpal Singh Gill, Minxian Xu, Carlo Ottaviani, Panos Patros, Rami Bahsoon, Arash Shaghaghi, Muhammed Golec, Vlado Stankovski, Huaming Wu, Ajith Abraham, Manmeet Singh, Harshit Mehta, Soumya K. Ghosh, Thar Baker, Ajith Kumar Parlikad, Hanan Lutfiyya, Salil S. Kanhere, Rizos Sakellariou, Schahram Dustdar, Omer Rana, Ivona Brandic, and Steve Uhlig. 2022. AI for next generation computing: Emerging trends and future directions. *Internet Things* 19 (2022), 100514. DOI: https://doi.org/10.1016/j.iot.2022.100514

[7] Salma Abdalla Hamad, W. Zhang, Quan Z. Sheng, and Surya Nepal. 2019. IoT device identification via network-flow based fingerprinting and learning. *2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE)* (2019), 103–111.

[8] Jakub Konecný, Brendan McMahan, and Daniel Ramage. 2015. Federated optimization: Distributed optimization beyond the datacenter. *CoRR* abs/1511.03575 (2015).

[9] Jakub Konecný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. In *Proceedings of the NIPS Workshop on Private Multi-party Machine Learning*. Retrieved from https://arxiv.org/abs/1610.05492.

[10] Jaidip Kotak and Yuval Elovici. 2021. IoT device identification using deep learning. In *Proceedings of the 13th International Conference on Computational Intelligence in Security for Information Systems (CISIS'20)*. Springer International Publishing, Cham, 76–86.

[11] H. H. Kumar, K. V. R, and M. K. Nair. 2020. Federated K-means clustering: A novel edge AI based approach for privacy preservation. In *Proceedings of the IEEE International Conference on Cloud Computing in Emerging Markets (CCEM'20)*. IEEE Computer Society, 52–56. DOI: https://doi.org/10.1109/CCEM50674.2020.00021

[12] Kaushal Kumar, Asish Kumar Dalai, Saroj Kumar Panigrahy, and Sanjay Kumar Jena. 2017. An ANN based approach for wireless device fingerprinting. In *Proceedings of the 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT'17)*. 1302–1307. DOI: https://doi.org/10.1109/RTEICT.2017.8256809

[13] Andreas Kurtz, Hugo Gascon, Tobias Becker, Konrad Rieck, and Felix Freiling. 2016. Fingerprinting mobile devices using personalized configurations. *Proc. Privac. Enhanc. Technol.* 1 (2016), 4–19.

[14] Pengfei Liu, Panlong Yang, Wen-Zhan Song, Yubo Yan, and Xiang-Yang Li. 2019. Real-time Identification of Rogue WiFi Connections Using Environment-Independent Physical Features. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 190–198. DOI: https://doi.org/10.1109/INFOCOM.2019.8737455

[15] Y. Luo, H. Hu, Y. Wen, and D. Tao. 2020. Transforming device fingerprinting for wireless security via online multitask metric learning. *IEEE Internet Things J.* 7, 1 (2020), 208–219.

[16] Chuan Ma, Jun Li, Ming Ding, Howard H. Yang, Feng Shu, Tony Q. S. Quek, and H. Vincent Poor. 2020. On safeguarding privacy and security in the framework of federated learning. *IEEE Network* 34, 4 (2020), 242–248. DOI: https://doi.org/10.1109/MNET.001.1900506

[17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 54)*. PMLR, 1273–1282. Retrieved from https://proceedings.mlr.press/v54/mcmahan17a.html.

[18] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. 2018. N-BaIoT–Network-based detection of IoT botnet attacks using deep autoencoders. *IEEE Pervas. Comput.* 17, 3 (2018), 12–22. DOI: https://doi.org/10.1109/MPRV.2018.03367731

[19] Yair Meidan, Michael Bohadana, Asaf Shabtai, Juan David Guarnizo, Martín Ochoa, Nils Ole Tippenhauer, and Yuval Elovici. 2017. ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis. In *Proceedings of the Symposium on Applied Computing (SAC'17)*. Association for Computing Machinery, New York, NY, 506–509. DOI: https://doi.org/10.1145/3019612.3019878

[20] Y. Meidan, M. Bohadana, A. Shabtai, M. Ochoa, N. O. Tippenhauer, J. D. Guarnizo, and Y. Elovici. 2017. Detection of unauthorized IoT devices using machine learning techniques. *CoRR abs/1709.04647* (2017).

[21] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. IoT SENTINEL: Automated device-type identification for security enforcement in IoT. In *IEEE 37th International Conference on Distributed Computing Systems (ICDCS'17)*. 2177–2184. DOI: https://doi.org/10.1109/ICDCS.2017.283

[22] Nizar Msadek, Ridha Soua, and Thomas Engel. 2019. IoT device fingerprinting: Machine learning based encrypted traffic analysis. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC'19)*. 1–8. DOI: https://doi.org/10.1109/WCNC.2019.8885429

[23] NASA. 2019. *Cybersecurity Management and Oversight at the Jet Propulsion Laboratory* (Report No. IG-19-022). Retrieved from https://oig.nasa.gov/docs/IG-19-022.pdf.

[24] Jorge Ortiz, Catherine Crawford, and Franck Le. 2019. DeviceMien: Network Device Behavior Modeling for Identifying Unknown IoT Devices. In *Proceedings of the International Conference on Internet of Things Design and Implementation (IoTDI'19)*, Association for Computing Machinery, Montreal, Quebec, 106–117. DOI: https://doi.org/10.1145/3302505.3310073

[25] T. D. Nguyen, A. Sadeghi, S. Marchal, M. Miettinen, and N. Asokan. 2019. AuDI: Toward autonomous IoT Device-type Identification using periodic communication. *IEEE J. Select. Areas Commun.* 37 (2019).

[26] Iskander Sanchez-Rola, Igor Santos, and Davide Balzarotti. 2018. Clock Around the Clock: Time-Based Device Fingerprinting. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS'18)*, Association for Computing Machinery, Toronto, 1502–1514. DOI: https://doi.org/10.1145/3243734.3243796

[27] Kunal Sankhe, Mauro Belgiovine, Fan Zhou, Luca Angioloni, Francesco Restuccia, Salvatore D'Oro, Tommaso Melodia, Stratis Ioannidis, and Kaushik Chowdhury. 2020. No radio left behind: Radio fingerprinting through deep learning of physical-layer hardware impairments. *IEEE Trans. Cogn. Commun. Netw.* 6 (2020).

[28] Kunal Sankhe, Mauro Belgiovine, Fan Zhou, Shamnaz Riyaz, Stratis Ioannidis, and Kaushik Chowdhury. 2019. ORACLE: Optimized radio classification through convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Communications*. DOI: https://doi.org/10.1109/infocom.2019.8737463

[29] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. 2019. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Trans. Mob. Comput.* 18, 8 (2019), 1745–1759.

[30] Amira Soliman, Sarunas Girdzijauskas, M.-R. Bouguelia, Sepideh Pashami, and Slawomir Nowaczyk. 2020. Decentralized and adaptive K-means clustering for non-iid data using hyperloglog counters. In *Proceedings of the 24th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. 343–355. DOI: https://doi.org/10.1007/978-3-030-47426-3_27

[31] Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. 2013. Auto-encoder based data clustering. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, Springer Berlin Heidelberg, Berlin, Heidelberg, 117–124.

[32] Gan Sun, Yang Cong, Jiahua Dong, Qiang Wang, Lingjuan Lyu, and Ji Liu. 2022. Data poisoning attacks on federated machine learning. *IEEE Internet Things J.* 9, 13 (2022), 11365–11375. DOI: https://doi.org/10.1109/JIOT.2021.3128646

[33] Jay Thom, Nathan Thom, Shamik Sengupta, and Emily Hand. 2022. Smart Recon: Network traffic fingerprinting for IoT device identification. In *Proceedings of the IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC'22)*. 0072–0079. DOI: https://doi.org/10.1109/CCWC54503.2022.9720739

[34] Han Wang, Luis Barriga, Arash Vahidi, and Shahid Raza. 2019. Machine learning for security at the IoT edge—A feasibility study. In *Proceedings of the IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems Workshops (MASSW'19)*. 7–12. DOI: https://doi.org/10.1109/MASSW.2019.00009

[35] Weiwei Wu, Su Hu, Yuan Gao, and Jiang Cao. 2021. Federated learning based distributed algorithms for RF fingerprinting extraction and identification of IoT devices. In *Artificial Intelligence for Communications and Networks*. Springer International Publishing, Cham, 3–18.

[36] Kai Yang, Qiang Li, and Limin Sun. 2019. Towards automatic fingerprinting of IoT devices in the cyberspace. *Comput. Netw.* 148 (2019), 318–327. DOI: https://doi.org/10.1016/j.comnet.2018.11.013

[37] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.* 10, 2 (Jan. 2019). DOI: https://doi.org/10.1145/3298981

[38] Xinyu Zhou, Aiqun Hu, Guyue Li, Linning Peng, Yuexiu Xing, and Jiabao Yu. 2019. Design of a robust RF fingerprint generation and classification scheme for practical device identification. In *IEEE Conference on Communications and Network Security (CNS'19)*. 196–204. DOI: https://doi.org/10.1109/CNS.2019.8802783

[39] Chaoshun Zuo, Haohuang Wen, Zhiqiang Lin, and Yinqian Zhang. 2019. Automatic fingerprinting of vulnerable BLE IoT devices with static UUIDs from mobile apps. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS'19)*, Association for Computing Machinery, 1469–1483. DOI: https://doi.org/10.1145/3319535.3354240

Received 17 December 2021; revised 13 March 2023; accepted 17 May 2023