FICDF: A Federated Incremental Learning Framework for IoT Device Fingerprinting

Shengli Ding*, Dong-Jun Han†, Christopher G. Brinton*, Keerthi Dasala*
*Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, USA
†Department of Computer Science and Engineering, Yonsei University, South Korea
*{ding393, cgb, kdasala}@purdue.edu, †djh@yonsei.ac.kr

***Abstract*—Although Internet of Things (IoT) devices have been widely used, their simple structure restricts the deployment of advanced security protocols, making them vulnerable to cyber-attacks. Consequently, network administrators must adopt a zero-trust approach to identify each IoT communication entity. Recently, radio frequency (RF) or traffic data based fingerprinting has proven to be an effective IoT identification technique. Nevertheless, existing fingerprinting methods face limitations due to privacy concerns, the extreme non-independent distribution of fingerprint data, and the dynamic updating of IoT devices, hindering real-world deployment. We propose a Federated IoT Continuous Device Fingerprinting (FICDF) mechanism to address these challenges. In the traffic data preprocessing stage, we design a binary encoding and temporal tensor channel stacking mechanism to enhance the device-specific features in each training sample. Within the framework of federated incremental learning, we introduce a k-means multi-centroid exemplar-based Gaussian noise feature-sharing mechanism to simultaneously address the extreme non-IID nature of the data and the issue of catastrophic forgetting. To the best of our knowledge, this is the first study to tackle federated incremental device fingerprinting under extreme non-IID conditions. The code for this paper can be found at https://github.com/Squiding/FICDF_WiOpt2024**

***Index Terms*—IoT device fingerprinting, Federated learning, Class incremental learning**

I. Introduction

As it is becoming an integral aspect of contemporary life, the Internet of Things (IoT) offers convenience and enhances efficiency through devices like smartwatches, smart speakers, and smart home appliances. These devices connect to the internet, providing users advanced services and automation capabilities. However, despite their popularity and benefits, IoT devices are notably vulnerable to malicious attacks. In the past three years, 69% of organizations have reported increased attacks targeting their IoT devices [1]. Additionally, over the past year, organizations have faced an average loss of $263,035 due to IoT attacks [1]. Due to the limitations imposed by simple firmware structures, IoT devices cannot deploy sufficient security protection mechanisms akin to those in computers. Some IoT devices also lack the capability for automatic system updates, preventing timely patching of security vulnerabilities. Moreover, the default administrator passwords for some IoT devices are often forgotten to be changed, allowing attackers to gain access to the devices effortlessly. These vulnerabilities in IoT devices lead to potential identity theft and unauthorized access, necessitating that network administrators monitor access requests to IoT devices with a zero-trust approach, even if they have verified identities.

Device fingerprinting is one promising solution to address these security concerns. Device fingerprinting involves collecting and analyzing the unique characteristics of a device's hardware and software configuration, e.g. radio frequency waveforms [2] and traffic or packet information [3]. This technique will create a unique identifier for each device based on these features and keep detecting anomalies by comparing the identifier against subsequent IoT running data. This enables seamless device behavior monitoring, making it difficult for attackers to impersonate legitimate devices. Unlike traditional authentication methods that rely on static credentials, which require frequent login verification, device fingerprinting can adapt to changes and provide a passive dynamic layer of protection. This approach improves security and helps maintain a seamless user experience by reducing the need for repeated authentication checks.

A. Motivation

Conventional device fingerprinting techniques often focus solely on the feature engineering of extracting and creating fingerprints, neglecting the emerging design requirements arising from the dynamic IoT device number and the complexity of wireless network structure [3]–[10]. For instance, an extensive enterprise network might include multiple gateways for the purpose of network segmentation, with data between these gateways unable to be shared due to privacy concerns. Additionally, some IoT devices, such as smartwatches or Raspberry Pi, may switch between different networks as users move. Therefore, there is a design requirement for fingerprinting mechanisms that protect user privacy within local networks while enabling fingerprint recognition of all IoT devices from a global perspective. However, most conventional device fingerprinting methods, centralized (e.g. [3], [8]) and distributed [11], require data to be transmitted from local gateways to a central controller, thus exposing it to the risk of privacy leakage. Furthermore, many current methods rely on sufficient datasets for effective feature engineering and must redo feature engineering when new devices appear. Nevertheless, as new IoT devices continuously join the system, the newly generated traffic data can burden the gateways' memory. When memory storage reaches its limit, old data may be deleted [12], leading

ISBN 978-3-903176-65-2 © 2024 IFIP

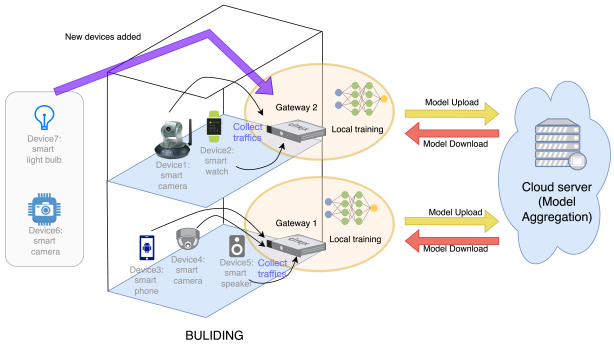

Fig. 1. An example with $J = 2$ gateways, each gateway is connected with a group of devices, and each device transmits data to the corresponding gateway. The gateway downloads the model from the server and trains it with local data. After training, the model is uploaded back to the server for aggregation. Additionally, when new IoT devices connect to the gateway, a task update is triggered.

to a decline in recognition accuracy for older devices, a phenomenon known as *catastrophic forgetting*.

Recently, researchers have begun addressing these issues and proposed some solutions, such as using federated learning (FL) [13] to protect user privacy and reduce communication overhead for data transmission to a central server [14]. Additionally, to mitigate the problem of declining accuracy due to the influx of new device data, incremental learning (IL) has been applied to fingerprint recognition [2]. However, these methods typically consider only one aspect of the challenges encountered in practical applications. Therefore, combining FL and IL into Federated Class-Incremental Learning (FCIL) to comprehensively address user privacy and dynamic device types becomes an intuitive yet feasible idea. However, the non-IID nature of user data in FL settings exacerbates the catastrophic forgetting problem in IL. The mediocre performance of existing FCIL structures based on image classification [15] also proves the challenging nature of the problem. Thus, it is meaningful and challenging to design a dedicated FCIL device fingerprinting framework addressing the extreme non-iid of IoT device data while meeting accuracy requirements.

B. Problem Definition

To better understand the problem, we define some key concepts. Considering the complexity of the federated incremental learning framework, we start with fingerprint recognition under a single gateway, then expand to federated learning under multiple gateways, and finally introduce incremental learning components to complete the final explanation. Figure 1 illustrates a use case of device fingerprinting with federated and incremental requirements.

1) Single Gateway: Suppose there is a network with a gateway G that requires the deployment of a global device fingerprinting system. Under this gateway, there may be multiple routers or wireless access points (APs). Each AP or router may have IoT devices D_d that need fingerprint recognition, where d represents the ID of the IoT device. All D_d under this gateway form a set $\mathcal{D} = \{D_d\}_{d=1}^{K}$, where K is the total number of IoT devices. In this paper, we do not need to consider the complex hierarchical structure between IoT devices and the gateway, nor do we need to know the number of APs or routers. Because any traffic information $\boldsymbol{x}_d$ generated by $D_d \in \mathcal{D}$ will eventually pass through the gateway and be collected, the gateway will preprocess (III-A) the raw data $\boldsymbol{x}_d$ into $\dot{\boldsymbol{x}}_d$ and generate labels $\boldsymbol{y}_d$ through one-hot encoding. Ultimately, the gateway obtains the training dataset for device fingerprinting, which is the training task $\mathcal{T} = \bigcup_{d=1}^{K} \left\{ \{\dot{\boldsymbol{x}}_{d,i}, \boldsymbol{y}_{d,i}\}_{i=1}^{N_d} \right\}$, where N_d is the number of training samples for each D_d. Here, we assume that $\mathcal{T}$ is collected and created after D_d passed the identity validation, which is safe and has not been modified maliciously. Lastly, the neural network model Θ used for classification will train on $\mathcal{T}$ at the gateway and eventually be used for device fingerprint recognition.

2) Federated Learning under Multiple Gateways: As mentioned in the motivation section, there are multiple gateways in the actual network $\mathcal{G} = \{G_g\}_{g=1}^{J}$, where J is the total number of gateways ($J \geq 2$) and G_g represents each individual gateway. Similarly, each G_g will have its own training task $\mathcal{T}_g = \bigcup_{d_g=1}^{K_g} \left\{ \{\dot{\boldsymbol{x}}_{d_g,i}, \boldsymbol{y}_{d_g,i}\}_{i=1}^{N_{d_g}} \right\}$. Under the framework of federated learning, each G_g will be assigned the same local model Θ_g and will perform local training using $\mathcal{T}_g$. Above $\mathcal{G}$, a central server $\mathcal{S}$ will aggregate the trained Θ_g models to obtain the updated global model $\Theta_{\mathcal{S}}$. The central server $\mathcal{S}$ can obtain the total number of D_{d_g} as $\sum_{g=1}^{J} K_g$ to initialize $\Theta_{\mathcal{S}}$, but it has no access to $\mathcal{T}_g$. Notably, we label each training sample with the ID of its IoT device (d) and the gateway (g). Due to network segmentation and security requirements between different gateways, even though some movable D_{d_g} may switch between different G_g, the secure data used for training will only be obtained and stored from the G_g where D_{d_g} undergoes its initial network verification. Therefore, the $\mathcal{T}_g$ of different G_g will be highly non-independent and identically distributed (non-iid), as well as each D_{d_g} will only appear in its corresponding $\mathcal{D}_g$, that is,

$$\mathcal{D}_g \cap \mathcal{D}_{g'} = \emptyset \quad \forall\, g, g' \in \{1, 2, \ldots, J\} \quad \text{with} \quad g \neq g' \tag{1}$$

3) Federated Incremental Learning for Increasing Number of IoT Devices: In actual fingerprint recognition, $\mathcal{D}_g$ is not fixed. More D_{d_g} will be added to $\mathcal{D}_g$. We use $t = 1, 2, 3, \ldots, T$ to represent the t-th task, where T represents the current task number. When the number of new D_{d_g} reaches a certain amount, $\mathcal{S}$ will update the model $\Theta_{\mathcal{S}}^{t}$ and start a new round of training. Intuitively, $\mathcal{T}_g^t$ in G_g at time t should be equal to $\bigcup_{t'=1}^{T} \mathcal{T}_g^{t'}$. However, due to the limited memory space of G_g, many old data $\bigcup_{t'=1}^{T-1} \mathcal{T}_g^{t'}$ will be discarded, causing the data imbalance problem that leads to catastrophic forgetting, i.e.,

$$\left| \bigcup_{t'=1}^{T-1} \mathcal{T}_g^{t'} \right| \ll \left| \mathcal{T}_g^{T} \right| \tag{2}$$

C. Major Contributions

Based on the challenges mentioned in the motivation and problem definition, especially (1) and (2), we propose the **F**ederated **I**oT **C**ontinuous **D**evice **F**ingerprinting (FICDF) mechanism. Our main contributions are as follows:

1) We design a k-means multi-centroid exemplar sample selection method for IoT device traffic packets. Compared to the existing single-mean-based exemplar selection, our method can better cover the traffic packets generated by the same IoT device with diverse purposes and features. Therefore, under the constraint of the limited memory of the training clients, our method can more comprehensively retain the traffic characteristics of old IoT devices.
2) We propose a binary encoding and channel stacking-based preprocessing method for IoT traffic data. This method binarizes the packet information and stacks the sequentially received packets in different channels of a tensor according to their arrival time, forming a training sample similar to an RGB color image. The data using the stacked channel sample ensures that the original temporal information is retained even after the task transition when exemplar selection causes fragmentation of originally continuous samples.
3) We address the extreme non-iid problem between training clients and the catastrophic forgetting issue in incremental learning by incorporating a Gaussian noise-based exemplar data-sharing mechanism. Our designed FICDF mechanism meets the requirements of device privacy protection and dynamic updates of device numbers in real deployments for device fingerprint recognition. This is also the first paper to address extreme non-iid device fingerprint recognition under a federated incremental learning framework. Through detailed simulation experiments, we demonstrated the effectiveness of our method.

This paper is organized as follows. Section II discusses related work. Section III introduces the proposed method, Section IV discusses the experimental setup and results, and Section V summarizes the paper and future research directions.

II. Related Work

This section will mainly discuss conventional device fingerprinting techniques, including existing methods that apply the FL and IL frameworks.

A. Wireless Device Fingerprinting

In 2005, Kohno et al. from the University of California, San Diego, first invented the technique of remote physical device fingerprinting using micro-clock offsets in device hardware [6]. Since then, numerous new technologies for device fingerprinting have been proposed. Initially, most of the algorithms were based on statistics [16]. In statistical methods, researchers use statistical functions such as the expected value [7] to extract device fingerprint features and perform classification. Recently, with the rapid development of Machine Learning/Deep Learning (ML/DL) algorithms, many researchers have begun to explore the potential of using these new technologies for device fingerprinting. In [17], Duan et al. developed a kNN fingerprint recognition algorithm named ByteIoT. In the work [10] by Kostas et al., six ML algorithms were applied to fingerprint recognition. However, ML algorithms, especially the kNN algorithm, are difficult to operate in a distributed environment where data cannot be shared due to privacy protection. Recently, Feng et al. proposed a fingerprint recognition model based on RNN [3]. They converted IoT device packets into grayscale images and used the sequential arrangement of packets to enhance the distinctiveness of features. However, in practical applications, due to the limited memory of the training client, which is usually a router or gateway, old data is selectively cleared. When the fingerprint recognition system needs to be updated, the accuracy decreases due to the fragmentation of the time series of old data samples.

B. Federated Learning and Incremental Learning for Device Fingerprinting

Federated learning has advantages in privacy protection and communication cost savings by prohibiting local user data from being shared with a central server while enabling distributed collaboration between users and the central server [18], [19]. In a recent paper, Wang et al. [14] applied a federated learning approach for device fingerprinting. The FL4IoT model adopts a two-step structure for device fingerprinting and recognition, utilizing FedAvg [13] for federated learning aggregation. Additionally, Wu et al. [20] proposed an RF fingerprinting method based on FL. The input to this model is statistically processed I/Q samples, and the global model for FL is a convolutional neural network (CNN). Although these two studies demonstrate the effectiveness of FL in fingerprinting applications, neither of them considered the extreme non-iid data across different training clients, as we highlighted in (1).

In (2), we mentioned that the introduction of new IoT devices would pose the challenge of catastrophic forgetting for the fingerprinting system. This is the most common challenge in the field of IL. Recently, Liu et al. proposed a channel separation incremental learning (CSIL) method for device fingerprinting [2]. They used Automatic Dependent Surveillance-Broadcast (ADS-B) signals from commercial aircraft as the input for fingerprinting. CSIL mitigates catastrophic forgetting by adding Elastic Weight Consolidation (EWC) [21] to the loss function, which calculates and stores the Fisher Information Matrix of old tasks to protect important parameters from being overly modified. However, in federated learning, especially under the conditions mentioned in (1), the EWC cannot correctly calculate the importance of parameters globally due to the unavailability of complete global data.

C. Fusion of Federated Learning and Incremental Learning

To comprehensively address (1) and (2), we need to combine FL and IL. However, since (1) and (2) influence each other, designing an FCIL suitable for fingerprinting is significantly

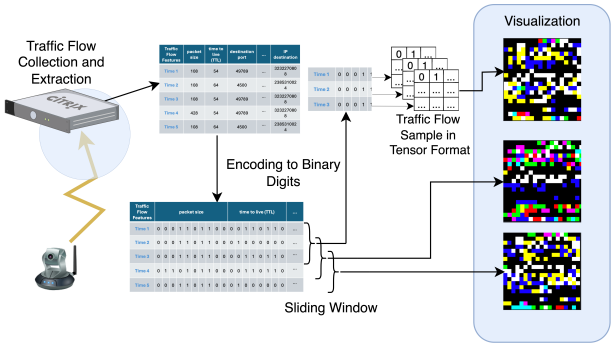

Fig. 2. Binary Encoding. The illustration shows traffic flow collection and extraction from an IoT device, encoding to binary digits, stacking time channel tensor using sliding window, and visualization.

challenging. In [15], Dong et al. proposed a Global-Local Forgetting Compensation (GLFC) based on image recognition. To mitigate the forgetting of old classes on local clients, it utilizes two loss functions: class-aware gradient compensation loss and class semantic relation distillation loss [15]. It employs a proxy server to select the best old global model to assist local relation distillation, thereby alleviating global forgetting caused by the class imbalance between clients. However, this model is not specifically designed for device fingerprinting. In general federated learning applications, extreme non-iid data (1) is not commonly seen, but it is a norm in our IoT device fingerprinting problem. Therefore, building a federated incremental learning system tailored explicitly for IoT device fingerprint recognition is necessary.

III. Methodology

In this chapter, we will discuss in detail how we addressed the challenges of device fingerprinting encountered in (1) and (2). We will first introduce the data preprocessing of IoT device traffic. Then, we will explain the prototype sample selection, data sharing, and data down sampling. Finally, we will present the complete FICDF's workflow.

A. Data Preprocessing

Feature extraction and fingerprint creation are usually the most crucial steps in traditional device fingerprinting design. However, we believe that neural networks can uncover latent features in IoT device traffic that even humans cannot identify. Therefore, in this paper, we will leave all feature extraction tasks to be handled by a convolutional neural network.

As described in Problem Definition I-B and Figure 2, for all G_g in $\mathcal{G}$, traffic information generated by D_{d_g} communication over a period will be read. As shown in Table III, we selected 19 packet features visible to G_g and restricted their value ranges. If packet raw data exists, its value is marked as 1; otherwise, it is marked as 0. Then, each value is converted to a fixed-length binary number according to the defined range to ensure adequate expression of potential features. If the binary number is shorter than the required length, it is padded with zeros. By performing the above operation for each traffic packet generated by D_{d_g}, we will ultimately obtain a table as shown in Figure 2.

TABLE I
Selected Values from IoT Packets

No.	Field	Range	Bits Required
1	Packet Size	(0, 65535)	16
2	IP Source Address	$(0, 2^{32}-1)$	32
3	IP Destination Address	$(0, 2^{32}-1)$	32
4	Source Port	(0, 65535)	16
5	Destination Port	(0, 65535)	16
6	Time to Live	(0, 255)	8
7	Protocol	(0, 255)	8
8	Ethernet Frame Type	(0, 65535)	16
9	Packet Raw Data	(0, 1)	1
10	Epoch Timestamp	$(0, 2^{32}-1)$	32
11	TCP Sequence Number	$(0, 2^{32}-1)$	32
12	TCP Acknowledgment Number	$(0, 2^{32}-1)$	32
13	TCP Flags	(0, 255)	8
14	TCP Window Size	(0, 65535)	16
15	UDP Length	(0, 65535)	16
16	UDP Checksum	(0, 65535)	16
17	IP Identification	(0, 65535)	16
18	IP Fragment Offset	(0, 8191)	13
19	IP Header Checksum	(0, 65535)	16

It is important to note that each matrix only contains a single traffic packet sent by an IoT device. However, it is evident that the semantic relationships and temporal order between packets also contain rich potential features. Therefore, we use a sliding window with a size of three and a step of one on the binary table in Figure 2. Each binary packet sequence is then padded and reshaped into a $19 * 19$ matrix, and the three consecutive packets selected by the sliding window are combined into a three-channel tensor. By doing so, we obtain $\mathcal{T}_g^t$ for each G_g, which meets the input layer requirements of Θ_g.

B. Components in FCIL

1) k-means multi-centroid: As discussed in Section I-B, $\mathcal{T}^t$ is continuously updated over time. The decline in classification accuracy of old categories due to the loss or active deletion of old data is a common issue in IL. The most common solution currently is iCaRL proposed by Rebuffi et al. [22]. iCaRL is based on data replay, which protects the performance of old categories in new tasks by storing a certain amount of old samples. However, its problem is that its exemplar data selection is only based on the mean of data features. Since IoT device packets can be classified into different types, choosing a single mean as the standard cannot comprehensively summarize the communication characteristics of IoT devices. Therefore, we use k-means to classify data samples to obtain multiple feature centroids. We then select a set of exemplar samples based on the distance to the centroids.

As shown in algorithm 1 We begin by extracting features for all images in the dataset $\mathcal{T}_g$ using the feature extractor ϕ from the client model Θ_g. For each class D_{d_g} in the set of current classes $\mathcal{D}_g$, we perform k-means clustering on the feature vectors $\mathbf{f}_d$ to obtain centroids $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_H\}$. The distance between a feature vector $\mathbf{f}$ and a centroid $\mathbf{c}$ is calculated as:

$$d(\mathbf{f}, \mathbf{c}) = \|\mathbf{f} - \mathbf{c}\|, \tag{3}$$

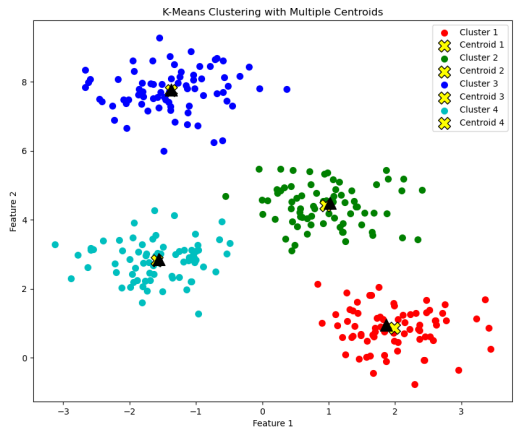

Fig. 3. An illustrative diagram of multi-centroid selection in k-means. The cross represents the centroid. The triangle represents the closest samples to the centroid, i.e., the exemplar.

Algorithm 1 Select Prototypes Using k-means Multi-centroid

Require: $\mathcal{G}$: Set of gateways, Θ_g: Client model, $\mathcal{T}_g$: Dataset for each gateway, $\mathcal{D}_g$: Set of current classes for each gateway, H: Number of prototypes per class

Ensure: $\mathcal{P}$: Prototypes

for each gateway $G_g \in \mathcal{G}$ **do**
  Extract features for all images in $\mathcal{T}_g$ using feature extractor ϕ from Θ_g
  for each class $D_{d_g} \in \mathcal{D}_g$ **do**
    Perform k-means clustering on $\mathbf{f}_d$ to get centroids $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_H\}$
    for each centroid $\mathbf{c}_h$ **do**
      Compute distances $d(\mathbf{f}_d, \mathbf{c}_h)$
      Select index j with minimum distance, append to $\mathcal{P}$, and remove selected $\mathbf{f}_d$ from dataset
    end for
  end for
end for
return $\mathcal{P}$

where $\|\cdot\|$ denotes the Euclidean norm. For each centroid $\mathbf{c}_h$, we compute the distances $d(\mathbf{f}_d, \mathbf{c}_h)$ and select the index j with the minimum distance to append to the set of exemplar or prototypes $\mathcal{P}$, then remove the selected $\mathbf{f}_d$ from the dataset to avoid redundancy. This process ensures that the selected exemplars are the closest samples to the centroids while at the same time ensuring that there is a certain amount of variability between each of the centroid-related samples, providing a representative set of exemplars for each class. Figure 3 shows an illustrative diagram of multi-centroid Selection in k-means.

2) Data Sharing: As mentioned in (1), the disjoint relationship exists between $\mathcal{D}_g$ of different G_g. A common solution is data sharing. In GLFC [15], Dong et al. used a prototype gradient-based communication to alleviate the data distribution imbalance between local clients (i.e., $\mathcal{D}_g$). The reason for not using raw samples directly is for privacy protection. However, in our device fingerprinting method, we first 1) encode the data in binary and 2) overlay the channels. After preprocessing steps 1) and 2), the original privacy information, such as IP addresses, has already been perturbed. Considering that these methods can still be reverse-engineered, we are inspired by the addition of noise in neural network training to enhance generalization ability. We further add irreversible Gaussian noise to the tensor samples to achieve data encryption. Thus, by sharing a small amount of prototypes selected by k-means multi-centroid and encrypted with Gaussian noise between $\mathcal{D}_g$, we can alleviate the catastrophic forgetting problem caused by (1).

To add Gaussian noise to the dataset $\mathcal{T}_g = \bigcup_{d_g=1}^{K_g} \{\{\dot{\boldsymbol{x}}_{d_g,i}, \boldsymbol{y}_{d_g,i}\}_{i=1}^{N_{d_g}}\}$, we apply the following transformation:

$$\dot{\boldsymbol{x}}'_{d_g,i} = \dot{\boldsymbol{x}}_{d_g,i} + \mathcal{N}(0, \sigma^2), \tag{4}$$

where $\mathcal{N}(0, \sigma^2)$ represents Gaussian noise with mean 0 and variance σ^2. This ensures that the perturbed data $\dot{\boldsymbol{x}}'_{d_g,i}$ maintains the essential features while adding a layer of encryption for privacy. The updated dataset will be denoted as $(\mathcal{T}_g^t)'$.

3) Data Down Sampling: Researchers often use balanced datasets with the same number of samples for different classes when testing the performance of new neural network architectures. However, in practical IoT device fingerprinting, different IoT devices generate different numbers of packets due to varying levels of activity. Additionally, we use a data-sharing scheme to address (1), which will share only a small number of prototypes to reduce communication overhead. Thus, there is a significant numerical difference between the shared class dataset and the dataset collected by G_g in the current task, i.e.,

$$\frac{|\bigcup_{t'=1}^{T-1} \mathcal{T}_g^{t'} \cup \bigcup_{t'=1}^{T-1} (\mathcal{T}^{t'})' \cup (\mathcal{T}^T)'|}{|\mathcal{D}^*|} \ll \frac{|\mathcal{T}_g^T|}{|\mathcal{D}^*|} \tag{5}$$

where $|\mathcal{D}^*|$ is the number of device in any $\mathcal{T}$. The four sets on the numerator represent the four datasets that a G_g contains in the current task. The left side of the inequality has 1) $\bigcup_{t'=1}^{T-1} \mathcal{T}_g^{t'}$ the old classes' prototypes retained by G_g from previous tasks, 2) $\bigcup_{t'=1}^{T-1} (\mathcal{T}^{t'})'$ the classes' encrypted prototypes shared by other $G_{g'}$ from previous tasks, and 3) $(\mathcal{T}^T)'$ the classes' encrypted prototypes shared by other $G_{g'}$ in the current task. The right side represents 4) $|\mathcal{T}_g^T|$ the samples of the current classes in the new task for G_g. Through the denominator $|\mathcal{D}^*|$, this inequality compares the number of samples in each specific IoT device rather than the set of multiple devices.

Our experiments show that unbalanced data can lead to classification accuracy fluctuations between different $\mathcal{T}_g^t$. Moreover, a large sample size will increase training duration and impose higher demands on hardware performance. In order to strike a balance between latency and fingerprint recognition performance, we introduce a data down sampling method combined with the proposed k-means multi-centroid prototypes approach. Similar to Algorithm 1, we filter the data.

However, instead of obtaining only one sample closest to the centroid, we obtain a batch of samples ranked by distance to the centroids. We then select a set of most representative samples for formal training. In our code, we let each centroid have two closest samples, ensuring that:

$$\frac{|\mathcal{T}_g^T|}{|\mathcal{D}^*|} = a\frac{|\bigcup_{t'=1}^{T-1}\mathcal{T}_g^{t'}|}{|\mathcal{D}^*|} = a\frac{|\bigcup_{t'=1}^{T-1}(\mathcal{T}^{t'})'|}{|\mathcal{D}^*|} = a\frac{|(\mathcal{T}^T)'|}{|\mathcal{D}^*|} \tag{6}$$

where we choose $a = 1$ so that every component has the same size. The components in the second row share the same data size because they are all filtered using Algorithm 1. This ensures that in any G_g, the size of the dataset under each label is balanced, and at the same time, less time and computing power will be utilized for learning the IoT device fingerprints.

C. Workflow

Algorithm 2 details the entire workflow of FICDF. First, each G_g will generate the initial training dataset using the data preprocessing method described in Section III-A and download Θ. G_g will first use the unfiltered data to train and update the feature extractor ϕ. Then, G_g will use Algorithm 1 to filter prototypes and update the training dataset through data sharing and down sampling. All new task follows this procedure.

Algorithm 2 Workflow in FICDF

Require: $\mathcal{G}$: Set of gateways, $\mathcal{S}$: Central server, $\Theta_{\mathcal{S}}$: Global model

Ensure: Updated global model $\Theta_{\mathcal{S}}$

1: **for** each gateway $G_g \in \mathcal{G}$ **do**
2: **Data Preprocessing:** Generate initial training dataset $\mathcal{T}_g$ using data preprocessing method described in Section III-A.
3: **Model Initialization:** Download $\Theta_{\mathcal{S}}$ from $\mathcal{S}$ and initialize ϕ using unfiltered data.
4: **Initial Training:** Train ϕ on $\mathcal{T}_g$ to get initial feature extractor.
5: **Prototype Selection:** Use Algorithm 1 to filter prototypes and create $\mathcal{P}$.
6: **Data Sharing and down sampling:** Share encrypted prototypes and down-sample the data to update $\mathcal{T}_g$.
7: **Training on Updated Dataset:** Train ϕ on updated $\mathcal{T}_g$.
8: **end for**
9: **Model Aggregation:** Aggregate updated ϕ from each G_g to update global model $\Theta_{\mathcal{S}}$.
10: **return** $\Theta_{\mathcal{S}}$

IV. Experimental Results

In this section, we will discuss our experimental setup and results.

A. Dataset Selection and Setup

We used the publicly available IoT Traffic Traces dataset, IoT Sentinel, from Aalto University [8], [23]. This very popular IoT dataset includes different types of IoT devices, such as smart cameras, plugs, and weight scales. It also includes different products from the same manufacturer (e.g., D-Link Smart plug and D-Link WiFi Motion sensor) as well as the same type of products from different manufacturers (e.g., Edimax Smart Plug Switch, WeMo Switch, and Homematic pluggable switch) [8], [23]. This similarity can increase the difficulty of device fingerprinting, helping verify the robustness of the algorithm. We created and tested the algorithm. Table II shows the platform and hardware on which our code ran on.

TABLE II
CONFIGURATION OF HARDWARE AND SOFTWARE

Software/Hardware	**Configuration**
Operation System	Linux 5.4.0
CPU	Intel(R) Xeon(R) Gold 6242 CPU @ 2.80GHz
Memory	32GB
GPU	NVIDIA Tesla V100S PCIe 32GB
CUDA	10.1

TABLE III
COMPARISON OF DIFFERENT SETTINGS

Setting 1						
	$\mathcal{T}^1$	$\mathcal{T}^2$	$\mathcal{T}^3$	$\mathcal{T}^4$	$\mathcal{T}^5$	$\mathcal{T}^6$
G_1	0, 1	5	10	16	23, 24	27, 29
G_2	2, 4	6, 9	11	15, 18, 19	21	25, 26
G_3	3	7, 8	12, 13, 14	17	20, 22	28

Setting 2						
	$\mathcal{T}^1$	$\mathcal{T}^2$	$\mathcal{T}^3$	$\mathcal{T}^4$	$\mathcal{T}^5$	$\mathcal{T}^6$
G_1	4	7, 6	14, 13	19, 17, 15	21	26, 28
G_2	1	9, 5	11	16	20, 23	27
G_3	0, 3, 2	8	10, 12	18	22, 24	25, 29

Setting 3				
	$\mathcal{T}^1$	$\mathcal{T}^2$	$\mathcal{T}^3$	$\mathcal{T}^4$
G_1	2, 3	8	16, 19	22, 27
G_2	0, 1, 5	7	17, 18	21, 26
G_3	4, 6	9, 10, 11, 12, 13	14, 15, 20	23, 24, 25

Setting 4						
	$\mathcal{T}^1$	$\mathcal{T}^2$	$\mathcal{T}^3$	$\mathcal{T}^4$	$\mathcal{T}^5$	$\mathcal{T}^6$
G_1	3	8	12	19	20, 22	28
G_2	0	6	10, 13	16	21	26
G_3	4	7, 9	11	18, 15	24	27
G_4	1, 2	5	14	17	23	25, 29

To better simulate the distribution and dynamic increase of IoT devices in real networks, we designed four different network settings. As shown in Table III, the rows of the table represent different G_g, and the columns represent different T^t. The numbers represent the IDs of each D_{d_g}. These D_{d_g} assignments were randomly generated. Setting 1, 3, and 4 used a random seed of 2024, and setting 2 used a random seed of 3024.

B. Results

Since we are the first article on fingerprint recognition under federated incremental learning, there is no direct baseline. We choose the image recognition FCIL method GLFC [15] and the IL device fingerprinting method CSIL for comparison [2]. The same IoT device distribution settings are allocated for all approaches. For fairness, we applied the same data pre-

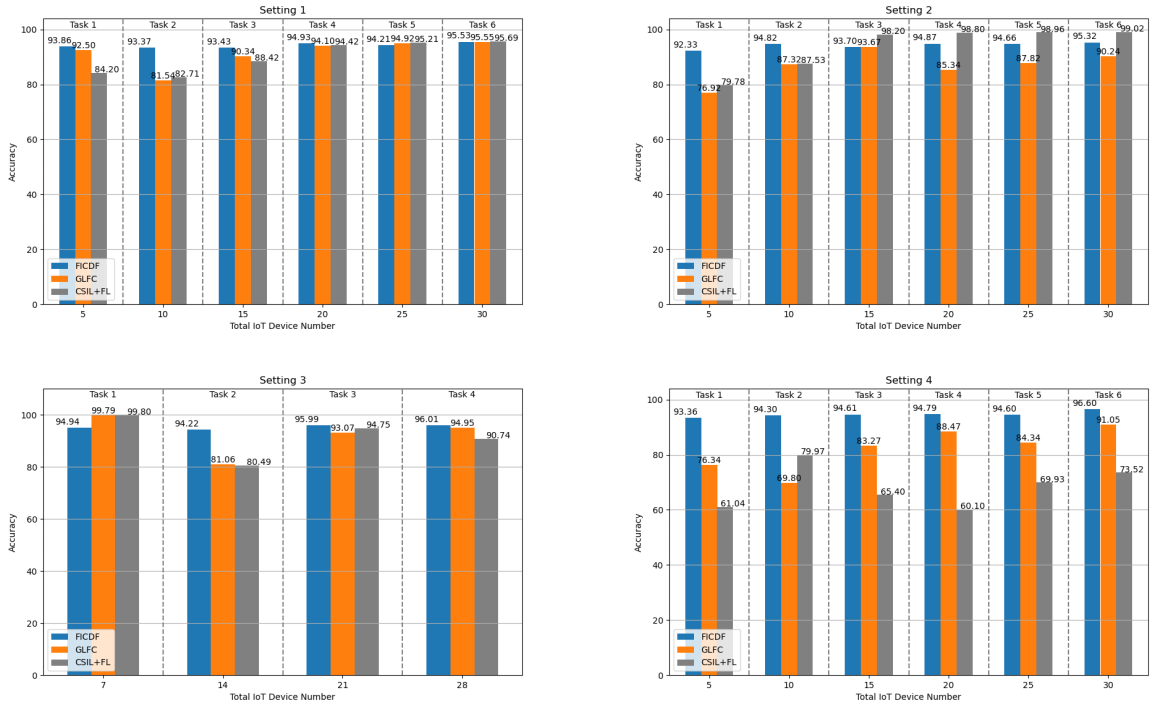

Fig. 4. Comparison of FICDF and GLFC Accuracy Under Different Settings.

processing method for GLFC as FICDF and added the FL structure into the CSIL. The experimental results are shown in the Figure 4. As a result, in the majority of tasks, the stability and accuracy of FICDF exceeds that of GLFC and CSIL, with an average accuracy of about 94%. Although CSIL has higher accuracy in some tasks, it comes at the cost of the memory overhead associated with using all the samples during training. The GLFC method uses fewer samples for training by filtering the data, thus requiring lesser computing power and memory. In Setting 4, where the number of gateways increases to 4, both CSIL and GLFC show a dramatic performance drop. However, our algorithm benefits from the encrypted data-sharing mechanism, and this negative effect due to the extreme non-iid of data between gateways is significantly weakened.

We also noticed some very interesting phenomena. When the total number of IoT devices increases, the accuracy does not decrease as we expected. On the contrary, all three methods show some increase. We think the possible reason is that different IoT devices share some common features. When the total number of IoT devices is small, the feature extractor in the convolutional network learns fewer overall sample features. However, as the total number of IoT devices increases, the feature extractor will have a more comprehensive understanding of each IoT device's unique characteristics and the latent relationships between each feature. At the same time, both our method and the GLFC method use the replay method of storing old class prototypes, which allows the feature extractor to continue learning and extracting features of old samples in the new task. Of course, because we used a k-means multi-centroid exemplar selection method and noise data sharing, the feature extractor can better capture IoT packet features in the current task, achieving more stable accuracy.

TABLE IV
ABLATION ANALYSIS FOR FICDF USING SETTING 1

Tasks	$\mathcal{T}^1$	$\mathcal{T}^2$	$\mathcal{T}^3$	$\mathcal{T}^4$	$\mathcal{T}^5$	$\mathcal{T}^6$
kC + DS	90.61	91.27	92.75	93.83	93.14	93.93
~~kC~~ + DS	69.89	72.68	69.53	77.58	74.94	77.67
kC + ~~DS~~	77.20	70.97	74.33	82.38	82.42	85.81

TABLE V
IMPACT OF COEFFICIENT a IN (6)

Tasks	$\mathcal{T}^1$	$\mathcal{T}^2$	$\mathcal{T}^3$	$\mathcal{T}^4$	$\mathcal{T}^5$	$\mathcal{T}^6$	**Time (s)**
a_1	93.86	93.37	93.43	94.93	94.21	95.53	66849.84
a_4	90.94	93.69	93.47	95.37	95.00	95.99	97493.19
a_8	87.70	93.97	94.98	96.52	94.96	96.39	111665.12
All Data	99.04	97.58	96.94	82.97	95.96	96.82	222035.23

C. Ablation Analysis for FICDF

We used Setting 1 for ablation experiments and studied the impact of our algorithm's components on device fingerprinting performance by observing the accuracy (in percentage) of each $\mathcal{T}^t$. In Table IV, kC represents the k-means multi-centroid, and DS represents Data Sharing. In the second row of Table IV, we used a single mean value to select samples; in the third row, we removed the data sharing mechanism. It can be observed that both approaches negatively affect accuracy, with the prohibition of k-means multi-centroid having a more significant impact. Tabl V shows the impact of the coefficient a on accuracy and training time, where a equals the value of its subscript. It has been found that there are always fluctuations in accuracy when the data is imbalanced. Moreover, balanced datasets can achieve similar accuracy to a_4, and a_8 in a

shorter time. Although the use of the complete dataset shows extremely high accuracy in $\mathcal{T}^1$ and $\mathcal{T}^2$, its training time is three times that of the balanced data.

In conclusion, the experiments confirmed that our method shares the capability of achieving reliable device fingerprinting accuracy in different network settings and maintaining its performance when new IoT devices are continuously added.

V. Conclusion

This paper introduces a federated class-incremental learning system named FICDF, which robustly handles the challenges associated with IoT device fingerprinting. Our system preserves privacy while supporting dynamic model updates and addressing the inherent diversity of IoT devices. By introducing methods centered around k-means multi-centroid exemplar selection for sample memory storage, encrypted data sharing between gateways, and down sampling, we successfully tackled issues arising from IoT devices' extreme non-iid distribution and data imbalance. Additionally, we incorporated a binary encoding and temporal channel overlay method tailored for convolutional neural networks to increase the information entropy of samples, which also provides a degree of data encryption. Our experimental evaluation demonstrates that the system consistently maintains reliable accuracy, even as the number of devices increases, thereby proving its scalability and adaptability. Overall, our results validate the proposed approach's effectiveness in enhancing IoT device fingerprinting, especially considering the practical network structure.

Acknowledgement

This work was supported in part by the National Science Foundation (NSF) under grant ITE-2326898, and by the Office of Naval Research (ONR) under grant N000142112472.

The authors would like to acknowledge the use of ChatGPT-4o, an artificial intelligence (AI) language model developed by OpenAI, for assistance in text polishing and grammar correction.

References

[1] 2023 State of IoT Security Report. Technical report.

[2] Yongxin Liu, Jian Wang, Jianqiang Li, Shuteng Niu, and Houbing Song. Class-Incremental Learning for Wireless Device Identification in IoT. *IEEE Internet of Things Journal*, 8(23):17227–17235, December 2021. Conference Name: IEEE Internet of Things Journal.

[3] Yuming Feng, Yu Zhang, Hui He, Weizhe Zhang, and Desheng Wang. An IoT Device Identification Method Using Extracted Fingerprint From Sequence of Traffic Grayscale Images. *IEEE Transactions on Dependable and Secure Computing*, pages 1–18, 2024.

[4] Kunal Sankhe, Mauro Belgiovine, Fan Zhou, Shamnaz Riyaz, Stratis Ioannidis, and Kaushik Chowdhury. ORACLE: Optimized Radio clAssification through Convolutional neuraL nEtworks, December 2018. arXiv:1812.01124 [cs, eess].

[5] Guillem Reus-Muns, Dheryta Jaisinghani, Kunal Sankhe, and Kaushik R. Chowdhury. Trust in 5G Open RANs through Machine Learning: RF Fingerprinting on the POWDER PAWR Platform. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–6, December 2020. ISSN: 2576-6813.

[6] T. Kohno, A. Broido, and K.C. Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, April 2005. Conference Name: IEEE Transactions on Dependable and Secure Computing.

[7] Suman Jana and Sneha K. Kasera. On Fast and Accurate Detection of Unauthorized Wireless Access Points Using Clock Skews. *IEEE Transactions on Mobile Computing*, 9(3):449–462, March 2010. Conference Name: IEEE Transactions on Mobile Computing.

[8] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N. Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 2177–2184, June 2017. ISSN: 1063-6927.

[9] Samuel Marchal, Markus Miettinen, Thien Duc Nguyen, Ahmad-Reza Sadeghi, and N. Asokan. AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication. *IEEE Journal on Selected Areas in Communications*, 37(6):1402–1412, June 2019.

[10] Kahraman Kostas, Mike Just, and Michael A. Lones. IoTDevID: A Behavior-Based Device Identification Method for the IoT. *IEEE Internet of Things Journal*, 9(23):23741–23749, December 2022.

[11] Vijayanand Thangavelu, Dinil Mon Divakaran, Rishi Sairam, Suman Sankar Bhunia, and Mohan Gurusamy. DEFT: A Distributed IoT Fingerprinting Technique. *IEEE Internet of Things Journal*, 6(1):940–952, February 2019. Conference Name: IEEE Internet of Things Journal.

[12] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics Training: Multi-Class Incremental Learning Without Forgetting. pages 12245–12254, 2020.

[13] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, April 2017. ISSN: 2640-3498.

[14] Han Wang, David Eklund, Alina Oprea, and Shahid Raza. FL4IoT: IoT Device Fingerprinting and Identification Using Federated Learning. *ACM Transactions on Internet of Things*, 4(3):17:1–17:24, July 2023.

[15] Jiahua Dong, Lixu Wang, Zhen Fang, Gan Sun, Shichao Xu, Xiao Wang, and Qi Zhu. Federated Class-Incremental Learning. pages 10164–10173, 2022.

[16] Pedro Miguel Sánchez Sánchez, José María Jorquera Valero, Alberto Huertas Celdrán, Gérôme Bovet, Manuel Gil Pérez, and Gregorio Martínez Pérez. A Survey on Device Behavior Fingerprinting: Data Sources, Techniques, Application Scenarios, and Datasets. *IEEE Communications Surveys & Tutorials*, 23(2):1048–1077, 2021. Conference Name: IEEE Communications Surveys & Tutorials.

[17] Chenxin Duan, Hao Gao, Guanglei Song, Jiahai Yang, and Zhiliang Wang. ByteIoT: A Practical IoT Device Identification System Based on Packet Length Distribution. *IEEE Transactions on Network and Service Management*, 19(2):1717–1728, June 2022.

[18] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated Learning: Strategies for Improving Communication Efficiency, October 2017. arXiv:1610.05492 [cs].

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

[20] Weiwei Wu, Su Hu, Yuan Gao, and Jiang Cao. Federated Learning Based Distributed Algorithms for RF Fingerprinting Extraction and Identification of IoT Devices. In Xianbin Wang, Kai-Kit Wong, Shanji Chen, and Mingqian Liu, editors, *Artificial Intelligence for Communications and Networks*, pages 3–18, Cham, 2021. Springer International Publishing.

[21] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. arXiv:1612.00796 [cs, stat].

[22] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning, April 2017. arXiv:1611.07725 [cs, stat].

[23] Samuel Marchal. captures_iot_sentinel(.zip), April 2017.