Studying the Robustness of Anti-Adversarial Federated Learning Models Detecting Cyberattacks in IoT Spectrum Sensors

Pedro Miguel Sánchez Sánchez, Alberto Huertas Celdrán, Timo Schenk, Adrian Lars Benjamin Iten, Gérôme Bovet, Gregorio Martínez Pérez, and Burkhard Stiller

Abstract—Device fingerprinting combined with Machine and Deep Learning (ML/DL) report promising performance when detecting spectrum sensing data falsification (SSDF) attacks. However, the amount of data needed to train models and the scenario privacy concerns limit the applicability of centralized ML/DL. Federated learning (FL) addresses these drawbacks but is vulnerable to adversarial participants and attacks. The literature has proposed countermeasures, but more effort is required to evaluate the performance of FL detecting SSDF attacks and their robustness against adversaries. Thus, the first contribution of this work is to create an FL-oriented dataset modeling the behavior of resource-constrained spectrum sensors affected by SSDF attacks. The second contribution is a pool of experiments analyzing the robustness of FL models according to i) three families of sensors, ii) eight SSDF attacks, iii) four FL scenarios dealing with anomaly detection and binary classification, iv) up to 33% of participants implementing data and model poisoning attacks, and v) four aggregation functions acting as anti-adversarial mechanisms. In conclusion, FL achieves promising performance when detecting SSDF attacks. Without anti-adversarial mechanisms, FL models are particularly vulnerable with > 16% of adversaries. Coordinate-wise-median is the best mitigation for anomaly detection, but binary classifiers are still affected with > 33% of adversaries.

Index Terms—Resource-constrained devices, cyberattacks, fingerprinting, federated learning, adversarial attacks, robustness

1 INTRODUCTION

In crowdsensing, large groups of individuals collaborate in a crowdsourcing fashion [24], typically leveraging devices as sensors [13]. Employing resource-constrained spectrum sensors (Raspberry Pis equipped with software-defined radio kits), the ElectroSense initiative marks an exemplary network for crowdsensing in particular [18]. However, the rapid growth of spectrum sensors also has accelerated the emergence of new and specialized cyberattacks, called spectrum sensing data falsification (SSDF) attacks [25]. In such a context, the privacy and integrity of sensors measurements are at risk.

In order to detect SSDF attacks affecting resource-constrained sensors, signature-based approaches present the limitation of not being effective against new attacks that have not been observed during the signature creation stage (zero-day attacks). To overcome this limitation, dynamic anomaly detectors considering fingerprinting are gaining relevance. This approach monitors device activities such as the usage of CPU, memory, network interfaces, or file system when there is no infection, and in a second stage, detects deviations produced by SSDF attacks [21]. The detection phase can be implemented using different techniques. One of the most lightweight in terms of resource consumption is based on rules, but creating precise rules requires expert knowledge and a relevant amount of time in complex crowdsensing scenarios [6]. Knowledge-based solutions have also been proposed in the literature, but they do not scale well with many sensors, requiring a lot of time to model and detect attacks [11]. Finally, machine and deep learning (ML/DL) techniques are gaining enormous relevance due to their detection performance, time, and relative simplicity [1].

Despite the benefits of anomaly detectors combining device fingerprinting and ML/DL, they present some characteristics limiting their applicability in crowdsensing scenarios where data belongs to different sensors and cannot be shared due to privacy concerns. Thus, federated learning (FL) becomes increasingly relevant [26]. Compared to centralized approaches, FL aims to train a federated model collaboratively but in a decentralized and privacy-preserving fashion. Each federation participant trains a model with its
own data and shares the model parameters to create the federated model. However, it presents limitations in terms of security since untrusted or malicious participants can launch adversarial attacks to destroy the model performance, add biases, or infer sensitive data of other participants. In this context, the literature has proposed several data and model falsification attacks consisting of poisoning data, labels, or weights during training to exchange fake model parameters with the entity (or entities) creating the federated model [20]. Different countermeasures have been proposed to overcome this problem, such as the usage of secure aggregation functions [16]. However, due to the novelty of the field, the combination of FL and behavioral fingerprinting for detecting SSDF attacks on spectrum sensor devices poses several open challenges.

In this context, the next open challenges serve as the motivation for the present work. First, there is an evident lack of FL-oriented datasets modeling fingerprints of resource-constrained spectrum sensors belonging to real platforms [19]. In this sense, it is critical to have datasets containing the behavior of heterogeneous sensors and SSDF attacks to propose novel and effective FL solutions. Second, there is no work measuring the performance of FL models using device fingerprinting to detect SSDF attacks affecting spectrum sensors. Furthermore, the detection performance of FL models and traditional ML/DL-based solutions have not been analyzed and compared in the field of SSDF attack detection. Most existing works consider spectrum data and consensus mechanisms to detect SSDF attacks. Still, they present limitations such as the need for redundant and trustworthy sensors and the difficulty of detecting attacks that add small perturbations to spectrum data [4]. Third, there is no work studying the robustness of FL-based solutions to detect SSDF attacks. Last but not least, the suitability of well-known anti-adversarial mechanisms reducing the impact of heterogeneous data and model poisoning attacks has not been analyzed in the field of spectrum sensors affected by SSDF attacks. This analysis is critical in the crowdsensing spectrum field, where malicious users can easily manipulate sensors.

To improve the previous challenges, this paper presents the following contributions:

1) The creation of a novel device behavioral fingerprinting dataset suitable for FL scenarios (publicly available in [22]). The dataset contains normal and under-attack behavior of four ElectroSense spectrum sensors, which are implemented in three families of Raspberry Pis connected to software-defined radio kits. About 75 internal events belonging to the usage of CPU, memory, network interface, file systems, and other relevant dimensions are monitored in each sensor for two different versions of normal behavior as well as eight SSDF attacks.

2) The usage of the dataset to evaluate and compare the SSDF attacks detection performance of (i) DL models under a horizontal FL scheme, and (ii) traditional DL approaches where a centralized aggregation neglects privacy. This evaluation comprises the definition of four federated scenarios dealing with anomaly detection (using Autoencoder), binary classification (with multilayer perceptron), and different participants.

3) The study of the federated model robustness in two of the previous federated scenarios under different conditions. These conditions vary in terms of (i) anti-adversarial aggregation mechanisms (two variants of trimmed mean, and coordinate-wise median), (ii) an increasing number of malicious participants (from 8 to 33%), and (iii) heterogeneous data and model poisoning attacks affecting both supervised and unsupervised FL models.

The remainder of this article is organized as follows. Section 2 reviews solutions combining fingerprinting and ML/DL approaches able to detect cyberattacks affecting IoT. While Section 3 provides the details of the FL-oriented dataset created in this work, Section 4 evaluates and compares the performance of different FL and traditional DL models trained and evaluated in heterogeneous conditions and scenarios. Section 5 analyzes the robustness of FL models affected by different adversarial attacks. Finally, Section 6 draws conclusions and next steps.

2 RELATED WORK

This section reviews related work considering behavioral fingerprinting and ML/DL approaches, both centralized and federated, to detect cyberattacks affecting IoT.

In [21] a broad survey of device fingerprinting reviews a good number of works detecting cybersecurity issues in IoT devices. One of the main conclusions of this survey is that there is a current trend toward combining device fingerprinting and ML/DL/FL techniques to detect cybersecurity attacks. In this context, the work most related to the paper at hand in terms of attacks, devices, and behavioral events is proposed in [9]. The authors combine unsupervised ML/DL techniques and the usage of device resources (such as CPU, memory, file system, or the network interface, among others) to detect anomalies produced by seven SSDF attacks affecting different Raspberries Pi acting as ElectroSense sensors. A pool of experiments reports 80-100% TPR when detecting five of the seven SSDF attacks. In [2], the authors look at frequency distributions of protocol attributes and run clustering algorithms to capture particularities of botnet behaviors. They report 97-100% accuracy, and as in most works, networking features are leveraged as the behavioral source. The authors of [12] use ML techniques combined with network packets to detect heterogeneous malware in IoT devices. They achieve 95% accuracy on their test sets. The main difference between the previous three works and the paper at hand is that the proposed ML/DL models are created in a centralized manner, which means that the privacy of the data used to train the models has not been guaranteed, one of the main contributions of this work.

Dealing with solutions that use FL to detect malware affecting IoT devices in privacy-preserving scenarios. The authors of [23] propose a solution for industrial IoT that analyzes Android application samples and behavioral data in an industrial context. They report 97-100% accuracy when detecting different malware samples. [17] presents a different use case for FL in the field of intrusion detection with up to 97% accuracy. This work studies adversarial implications in FL and employs blockchain technology as an alternative to mitigate them. Therefore, this work focuses more on the accountability

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
of participants instead of reducing the impact of the attacks. Moreover, network data and FL have been applied for other network management tasks like packet scheduling [10]. The main difference between the previous approaches and the one proposed in this work is that they do not consider behavioral fingerprinting, do not consider the problem of malicious participants, and do not evaluate the robustness of their models against adversarial attacks.

Most related to this work, several works combine FL and device fingerprinting to detect cyberattacks. [8] presents an FL system to detect malware in Android. The authors train Support Vector Machine classifiers in a federated scenario using device features such as application programming interface (API) calls and permission configuration to obtain 94-96% F1-score. While API calls correspond to device behavioral source fingerprints, the work at hand analyzes behavioral data sources on a much lower level. In [15], a federated anomaly detection system is proposed for the IoT. It leverages device-type profiles of the communications to detect malware with 96% accuracy. In contrast to the work at hand, the previous solution is ineffective against malware affecting data availability, integrity, or confidentiality. The authors of [19] leverage the N-BaIoT dataset to train and evaluate FL models. They achieve good accuracy in federated scenarios dealing with network traffic events. In addition to that, adversarial impacts are measured for selected attacks, and different mechanisms of robust aggregation are evaluated. While the paper at hand considers adversarial attacks against federations, it applies and analyzes these concepts in a scenario with very different characteristics to the one used in [19]. In this sense, N-BaIoT contains network traffic from nine IoT devices such as webcams and smart doorbells. However, N-BaIoT does not consider spectrum sensors such as Raspberry Pis equipped with software-defined radio kits (as this work does), and does not model device behavioral fingerprinting events. Finally, the literature has proposed some works dealing with countermeasures to solve privacy and security issues of FL. In this sense, a robust zero-watermarking scheme is presented in [7], and different aggregation functions such as trimmed mean or coordinate-wise median are proposed in [27].

As can be seen in Table 1, none of the related work studies the detection performance and robustness of federated models detecting SSDF attacks. Only [9] covers the same attacks and devices considered in this work, but from a traditional ML/DL perspective and without considering privacy-preserving scenarios. Moreover, Table 1 shows that device behavioral fingerprints have not been used for the use case of federated malware detection. In conclusion, this literature review demonstrates the lack of works and datasets combining behavioral fingerprints and FL to detect cyberattacks in IoT devices similar to those used in crowdsensing platforms. Furthermore, to the best of our knowledge, there is no work studying the impact of adversaries on the robustness of the previous federated models.

### 3 Dataset Creation

This section describes the novel device fingerprinting dataset created for federated scenarios. In particular, it presents (i) the crowdsensing platform and the spectrum sensors used to create the dataset, (ii) the SSDF attacks affecting the deployed ElectroSense sensors, (iii) the events selected to create the fingerprints, and (iv) the exploration of the dataset content.

#### 3.1 ElectroSense Sensors & SSDF Attacks

ElectroSense is a real and collaborative crowdsensing platform that pursues the goal of monitoring the electromagnetic space [5]. ElectroSense is composed of a multitude of spectrum sensors built from cheap commodity hardware like Raspberry Pis equipped with software-defined radio scanners and antennas. Each sensor monitors the different bands and segments of the radio frequency spectrum within its location. Sensed spectrum data is periodically sent to a backend platform in charge of storing, pre-processing, and analyzing the data to provide services. These services range from spectrum

| Source | Device Types | Attack Type | Data/Fingerprints | ML Approach | Prediction | Privacy | Robustness |
|--------|--------------|-------------|-------------------|-------------|------------|---------|------------|
occupancy monitoring to transmission decoding. In this scenario, four physical spectrum sensors have been deployed in two locations. Table 2 summarizes the devices identifiers, hardware characteristics, and locations.

For each ElectroSense sensor, two versions of the official and publicly available software are used. The first version is the current sensing application, installed by default in the sensor. The second version of the ElectroSense sensor software is an old one, available on the official ElectroSense GitHub [3]. Additionally, eight different SSDF attacks are considered to infect the four sensors. These SSDF attacks are executed after modifying the ElectroSense sensor source code and compiling a new executable version. The main goal of these SSDF attacks is to manipulate the data of particular spectrum segments monitored by the sensors (in different ways) and send poisoned spectrum data to the ElectroSense backend platform. Despite the differences in terms of attack impacts, all affect the same number of spectrum segments (20 MHz). Table 3 summarizes the main aspects of the behaviors considered during the creation of the dataset. More details about the implementation and functionality of the SSDF attacks can be found in [9].

The previous behaviors are sequentially executed in the devices of Table 2 for the time indicated in Table 3. To create the fingerprinting dataset, 75 internal events of each device were monitored in time windows of 50 s using the perf Linux command. These events belong to the following device data sources:

CPU, virtual memory, network, file system, scheduler, device drivers, and random numbers. Fig. 1 shows the events, classified per event type and family, contained in the datasets. In summary, the dataset includes four ElectroSense sensors, ten behaviors (two normal and eight SSDF attacks), 75 events belonging to eight event families, and a total of 73936 samples (approximately 60000 samples of normal and 13396 of malicious behavior). The dataset is publicly available in [22].

### Table 2: Details of the Devices Making up the Scenario

| Device ID | Type/Model | RAM | Location |
|-----------|------------|-----|----------|
| RPi3      | 3 Model B+ | 1GB | Zurich   |
| RPi4_1    | 4 Model B  | 2GB | Zug      |
| RPi4_2    | 4 Model B  | 2GB | Zug      |
| RPi4_3    | 4 Model B  | 4GB | Zurich   |

### Table 3: Behaviors Monitored during the Dataset Creation

| Behavior | Description                                      | Time |
|----------|--------------------------------------------------|------|
| Normalv1 | Current ElectroSense application sensing the spectrum | 5 days |
| Normalv2 | Old ElectroSense application sensing the spectrum | 5 days |
| Delay    | Sense different outdated spectrum data of affected segments | 4 hours |
| Confusion| Swap the spectrum data between affected segments   | 4 hours |
| Freeze   | Sense the same outdated spectrum data in affected segments | 4 hours |
| Hop      | Add random noise to random parts of affected segments | 4 hours |
| Mimic    | Copy the spectrum data of one segment into another segment | 4 hours |
| Noise    | Add random noise to the spectrum data of affected segments | 4 hours |
| Repeat   | Replicate the same spectrum data in all affected segments | 4 hours |
| Spoof    | Copy the spectrum data of one segment into another segment and add random noise | 4 hours |

3.2 Data Exploration

This section explores the created dataset to find data patterns and determine the suitability of ML/DL/FL techniques to detect SSDF attacks. This exploration also aims to determine if the data contained in the dataset is independent and identically distributed (IID) or non-IID. For that, three types of analysis are performed. The first one analyzes the evolution of data over time. The second focuses on the distributions of data belonging to different devices. Finally, the third explores data distributions according to various SSDF attacks.

The variation of behavioral data over time is essential to determine the stability of fingerprints, and the suitability of ML/DL/FL approaches to detect normal behavior and SSDF attacks. In this context and as an example, Fig. 2 shows the values of the kmem:mem_page_pcpu_drain event belonging to the Virtual Memory family across the time and for each device. As can be seen, the values are periodic, with some repetitive peaks. Exploring more in detail Fig. 2, it is also interesting to see the different distribution for RPi3 (in red) and RPi4s (in blue, orange, and green), indicating that behavioral data of similar devices is IID, and for different devices is non-IID. In particular, the range of values of

![Fig. 2. kmem:mem_page_pcpu_drain event for normal behavior in all devices.](image-url)
the kmen:mm_page_pcpu_drain event for the RPi3 is different from the range for the RPi4 devices. These characteristics are also visible in the majority of events, but they are not included due to room constraints.

For each device, the distribution of its events has been studied to analyze the differences between normal and under-attack behaviors. As a representative example, Fig. 3 shows for RPi4_1 and the urandon_read event how some attacks (hop, noise, and spoof) offer a higher number of random reads due to the generation of random noise. Another example can be seen in Fig. 4, where the writeback_mark_inode_dirty event of an RPi4_1 is differently affected by the copy and swap operations of some SSDF attacks (being disorder the attack with the lowest impact on the event values).

From the previous data exploration, it can be concluded that attacks do generally not impact the same features equally across different device types. Therefore, generalization across attacks and device types is challenging, and ML/DL/FL usage seems adequate for finding the events with the lowest impact on the event values.

4 FEDERATED SSDF ATTACKS DETECTION

This section evaluates the performance of different federated models detecting SSDF attacks and compares it with centralized ML/DL approaches where data privacy is not preserved. Two approaches have been considered to perform these tasks. The first approach detects anomalies using a supervised Autoencoder, while the second utilizes a supervised multilayer perceptron (MLP) to classify normal and under-attack behaviors. The pipeline followed to train and evaluate the federated models is also detailed in this section. Finally, four scenarios with different federation compositions (in terms of number and type of participants, behaviors, and detection tasks) are created to evaluate the performance of the previous FL models and compare them with centralized ML/DL approaches.

4.1 Federated ML Pipeline

The federated setting needs adaptations from the typical ML pipeline to handle distributed data and models. In particular, the scaling phase and the threshold selection have to be adapted to allow a global model to aggregate the knowledge of involved participants. Furthermore, a central coordinator needs to run the federated learning pipeline iteratively. The following subsections describe the necessary steps.

4.1.1 Dataset Splitting and Feature Preprocessing

Each federation participant creates the following datasets: one for training, one for validation and optimization of hyperparameters, and another for testing the model performance. Next, outlier filtering is performed on the training and validation sets. For that, Z-score is computed using the mean \( \mu \) and the standard deviation \( \sigma \) according to the formula \( \frac{x - \mu}{\sigma} \). Data points with an absolute z-score \( \geq 3 \) in any feature are excluded as they could impair the model performance. Besides, features with a correlation of 1 in the datasets are filtered.

4.1.2 Federated Feature Scaling

Feature scaling in a federated setup requires minimum communication efforts, as a global scaling for all participants must be implemented. Min-max scaling is employed using the formula \( \frac{x - \min}{\max - \min} \), and the minimum and maximum values are determined on the data of all the participants. Therefore, action from a central entity is required to coordinate the scaling process. A drawback of this approach is a certain loss of privacy since every participant must disclose each feature minimum and maximum value. This issue could be addressed using solutions such as homomorphic encryption, but it is out of the scope of this work.

4.1.3 Model Setup, Training and Evaluation

This work considers both supervised and unsupervised models. They require different data and methods to train the models and make predictions. However, both models are trained on a 68-dimensional input, corresponding to the number of relevant features after the preprocessing. Stochastic gradient descent (SGD) is used as the optimization algorithm with a learning rate of 1e-3 and a momentum term of 0.9.

In the anomaly detection scenarios, an Autoencoder with a single hidden layer of size 32 is used. After the first linear layer, batch normalization is applied and GELU is used as an activation function on the hidden state. A second linear layer transforms the hidden state back to its original size, followed by a GELU activation function that returns the reconstructed input. After the training phase, the anomaly threshold is determined based on the mean (\( \mu \)) and
standard deviation (σ) of the reconstructed mean square error (MSE). The formula used to select the threshold is shown in

\[ \text{threshold} = \mu + 3 \cdot \sigma. \]  

(1)

The prediction determines if the MSE of the recreated input is greater than the threshold. If so, it is considered an anomaly and, therefore, behavior under attack. Otherwise, it is considered normal behavior.

In the binary classification scenarios, an MLP is used. A linear layer produces a hidden state of size 256. Subsequently, batch normalization and the GELU activation function are applied to the hidden state. A second linear layer then returns a single output neuron. A Binary Cross Entropy Loss function with logits (\texttt{BCEWithLogitsLoss}) is used during training, which applies the sigmoid activation function and minimizes the logarithmic difference of the output to the encoded label (0 for normal behavior and 1 for attack behavior). Early stopping is applied when there is no loss decrease greater than \(1e^{-4}\) on the validation set.

For the federated training, \texttt{FederatedAveraging} (FedAvg) is used [14]. The federation is trained for 15 aggregation rounds with five local epochs per participant if not stated otherwise. It is important to note that the models are relatively small and thus can also be trained on resource-constrained hardware. Further, early stopping is implemented per participant.

4.1.4 Federated Threshold Selection

For anomaly detection, each participant sends its locally computed threshold to the central coordinator, which determines a global threshold. This work considers the mean \(\mu\) and standard deviation \(\sigma\) of the list of thresholds the participants send to the coordinator. Only thresholds that have an absolute \(z\)-score that is \(\leq 1.5\) are considered, choosing the maximum of those filtered values as the global threshold.

4.2 Federated Scenarios and Detection Performance

This section creates four federated scenarios where heterogeneous FL models are trained and evaluated following the previous pipeline. In addition, the detection performance of these models is compared to the one obtained by centralized approaches where data privacy is not preserved. For the sake of fairness, both the federated and central models use the same algorithms, training and testing data, and hyperparameters. Since the test sets for each behavior are separated, the experiments show the accuracy of the model for each behavior. The main goals of this section are to i) evaluate the suitability of the FL-oriented dataset presented in Section 3, ii) compare the performance of FL and centralized ML/DL models, iii) establish a baseline for the robustness analysis performed in Section 5.

The scenarios consider the dataset explained in Section 3 to create the federations. To decide the number of sensors participating in each scenario, each participant must have enough data to achieve meaningful convergence in its local training loop. Therefore, the scenarios explained in this section restrict the number of participants per device type to a maximum of 4. Below, each scenario details the exact number and type of sensors used in its federation and the behaviors considered for training and testing. Fig. 5 depicts the federated architecture pipeline together with the scheme of the training and testing division of devices in each scenario.

4.2.1 Scenario 1: Federated Anomaly Detection with Balanced Device Type

This scenario detects zero-day attacks when there is a balanced federation of different sensor types (RPi3, RPi4 2GB, and RPi4 4GB). In particular, four participants per sensor type are generated to set a total of 12. The 12 participants of the federation train a privacy-preserving Autoencoder following the pipeline previously explained. Each participant uses 1500 samples of its normal behavior for training and 150 normal samples for the threshold selection task. Once the federated Autoencoder is trained, each participant evaluates 75 samples per behavior (normal, normal_v2, and eight SSDF attacks).

Table 4 reports the accuracy achieved by the federated Autoencoder per device type and behavior. The parentheses denote the difference with the accuracy of a central model. A positive difference means that the federation outperforms the central approach, whereas a negative one is the opposite. Finally, it is important to note that RPi4_2 is excluded from training and only used for testing.

As seen in Table 4, both models (federated and centralized) perform almost identically, a good signal for the federated Autoencoder. More in detail, both models cannot detect freeze and repeat attacks due to their low impact on the device behavior, but the rest of the attacks are classified correctly (\geq 96%). An important aspect is that the accuracy on the second normal behavior (normal_v2) is also high (96.00-99.33%) despite not being used during training.

4.2.2 Scenario 2: Federated Anomaly Detection with New Device Type

This scenario evaluates whether a federated model can be useful for a new device type joining the federation and detecting zero-day SSDF attacks. Thus, the federated anomaly detection model is trained with eight participants belonging to two device types. Subsequently, the model is evaluated with behavioral samples (normal and under-
attack) of the new third device type. As indicated in Table 5, this is done for the three possible combinations of device types, generating three federated Autoencoder. Each Autoencoder is trained following the previous pipeline and the same number of samples as in the previous scenario.

Table 6 shows the accuracy of the three federated Autoencoders and the difference with the centralized one. As an example, the first column displays the accuracy of Autoencoder 3 (see Table 5).

As can be seen in Table 6, knowledge transfer to unseen device types is possible if there are similarities in the hardware configuration. Since the behaviors of RPi3 and RPi4s are quite different (non-IID data), the knowledge transfer to RPi3 is not possible, and all samples are classified as abnormal. In contrast, the detection performance on unsee RPi4s with different RAM is generally high. As in the previous scenario, freeze and repeat are not properly detected due to their high similarity to normal behavior. When comparing the federated model to the centralized approach, there are no major differences, performing the federated model slightly better when detecting normal behavior. It is a good achievement for FL, since data privacy is preserved.

### 4.2.3 Scenario 3: Federated Binary Classification with Balanced Device Type

It analyzes the capabilities of a federated binary classifier to transfer SSDF attacks knowledge between the federation. In particular, this scenario creates a federation of four participants per device type (12 in total) with the same behavioral data (normal and under-attack) per device type. More in detail, one participant per device type holds only normal data while the other three hold normal and delay, normal and freeze, and normal and noise, respectively. Each participant has 1000 samples of each selected behavior in its training set, 100 of each selected behavior in its validation set, and 250 of each existing behavior (two normal and eight attacks) in the test set. With this configuration and following the previous pipeline, a federated MLP is trained and evaluated. Table 7 shows the detection accuracy of the federated MLP model and the difference with the centralized approach. As usual, RPi4_2 is only used during testing.

As can be seen in Table 7, the federated MLP transfers the attack knowledge quite well. It even improves the accuracy of the central model for the disorder attack (+11-45%), solving a potential minor overfitting problem with normal behavior. As in the two previous scenarios, freeze and repeat are neither detected by federated nor centralized models due to their similarity with normal behavior. It happens even when these attacks are considered during training. For the rest of behaviors, no major difference (>5%) is observed between federated and centralized approaches.

### 4.2.4 Scenario 4: Federated Binary Classification with New Device Type

It is a combination of Scenario 2 and 3, and evaluates the capabilities of a federated binary classifier transferring attack knowledge from the federation to a new device type as Scenario 2 (see Table 5). Following the previous pipeline, a federated MLP model per federation (3 in total) is trained. Table 8 shows the accuracy of the three federated MLP and their differences compared to the centralized versions.

The results of Table 8 are very similar to those of Scenario 2. The knowledge transfer capability between RPi4s works well for most behaviors, while it does not work at all for RPi4 and RPi3 due to non-IID data. Further, the model does not

### TABLE 5: Federated Models Used in Scenario 2 and 4

| Model ID          | Training Devices | Testing Devices |
|-------------------|------------------|-----------------|
| Autoencoder/MLP 1 | RPi3 & RPi4_1    | RPi4_3          |
| Autoencoder/MLP 2 | RPi3 & RPi4_3    | RPi4_1 & RPi4_2 |
| Autoencoder/MLP 3 | RPi4_1 & RPi4_3  | RPi3            |

### TABLE 6: Accuracy of Scenario 2 Autoencoder Models and Difference with a Centralized Approach (In Parentheses)

| Behavior  | RPi3 (diff.) | RPi4_1 (diff.) | RPi4_2 (diff.) | RPi4_3 (diff.) |
|-----------|--------------|----------------|----------------|----------------|
| normal    | 96.0% (2.7%) | 100% (1.3%)    | 100% (2.7%)    | 99.3% (0.7%)   |
| normal_v2 | 96.0% (3.3%) | 96.7% (2.7%)   | 99.3% (0.7%)   | 98.7% (6.7%)   |
| delay     | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| disorder  | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| freeze    | 0.7% (-8.7%) | 4.00% (-0.7%)  | 2.0% (0%)      | 0% (-1.3%)     |
| hop       | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| mimic     | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| noise     | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| repeat    | 6.0% (-4.0%) | 3.3% (-0.7%)   | 2.7% (-2.0%)   | 2.7% (-2.0%)   |
| spoof     | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |

### TABLE 7: Accuracy of Scenario 3 MLP Model and Difference with a Centralized Approach (In Parentheses)

| Behavior | RPi3 (diff.) | RPi4_1 (diff.) | RPi4_2 (diff.) | RPi4_3 (diff.) |
|----------|--------------|----------------|----------------|----------------|
| normal   | 0% (0%)      | 98.0% (4.0%)   | 97.33% (4.0%)  | 99.3% (9.3%)   |
| normal_v2| 0% (0%)      | 98.00% (2.7%)  | 99.3% (4.0%)   | 96.0% (2.7%)   |
| delay    | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| disorder | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| freeze   | 100% (0%)    | 4.7% (-2.7%)   | 0.7% (-6.0%)   | 0.7% (-4.7%)   |
| hop      | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| mimic    | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| noise    | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| repeat   | 100% (0%)    | 2.00% (-5.3%)  | 4.00% (-6.0%)  | 1.3% (-2.7%)   |
| spoof    | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |

### TABLE 8: Accuracy of Scenario 3 MLP Model and Difference with a Centralized Approach (In Parentheses)

| Behavior | RPi3 (diff.) | RPi4_1 (diff.) | RPi4_2 (diff.) | RPi4_3 (diff.) |
|----------|--------------|----------------|----------------|----------------|
| normal   | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| normal_v2| 100% (4.0%)  | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| delay    | 100% (0%)    | 100% (2.2%)    | 100% (3.1%)    | 100% (0%)      |
| disorder | 90.6% (45.4%)| 95.4% (15.4%)  | 97.8% (11.3%)  | 97.3% (20.2%)  |
| freeze   | 0% (-6.5%)   | 6.8% (-5.7%)   | 4.3% (-3.7%)   | 5.5% (-4.7%)   |
| hop      | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| mimic    | 100% (0%)    | 100% (1.2%)    | 100% (5.1%)    | 100% (0%)      |
| noise    | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
| repeat   | 2.5% (-2.5%) | 4.1% (-2.0%)   | 4.8% (-3.3%)   | 13.5% (-0.2%)  |
| spoof    | 100% (0%)    | 100% (0%)      | 100% (0%)      | 100% (0%)      |
detect the behaviors freeze and repeat as attack in all RPi4, as expected from the previous results. For these two behaviors, the centralized model slightly outperforms the federated model (< 5%). However, it cannot be considered as a significant improvement due to the low scores (~ 0%), which are aligned with those obtained in the previous three scenarios. Finally, the federated model shows better performance (+ 10%) for the disorder behavior on an RPi4 with 4GB of RAM.

5 ROBUSTNESS AGAINST ADVERSARIAL ATTACKS

This section evaluates the robustness of the FL models created in Scenario 1 and 3 of Section 4. In federated learning, both the server and the participants could be malicious. However, this work focuses on adversarial participants degrading the effectiveness of federated models. Therefore, a completely honest server is assumed, and the following adversarial attacks are considered. The impact of each attack is evaluated under an increasing number of adversarial devices and widely used aggregation functions acting as anti-adversarial mechanisms.

- **Data poisoning.** Adversaries use malicious data to train their local models and send fake model weights to the aggregation server.
  - **Behavior Injection.** It is used for anomaly detection, and adversaries train their local models using SSDF attack data as if it were normal. In this work, the first adversarial device uses spoof behavior to train (instead of normal), the second mimic, the third delay, and the fourth disorder. These SSDF attacks are selected according to their median MSE in the corresponding federation without adversaries, choosing the ones more dissimilar from the normal behavior. Freeze and repeat behaviors are not injected due to their similarity to normal behavior.
  - **Label Flipping.** It is used for classification, and adversaries train their models by flipping the labels of normal and SSDF attacks. In the following experiments, the first adversary flips the labels of normal behavior, the second normal and delay, the third normal and freeze, and the fourth normal and noise.

- **Model Canceling.** It is a model poisoning attack (affecting anomaly detectors and classifiers) where adversaries send fake weights to bring the global model parameters to zero. In the following experiments, adversaries upload the parameters of the last known global model multiplied by a factor \( a \) that is determined according to the formula based on the number of participants \( K \) and the number of adversaries \( f \): \( K - f + a \cdot f = 0 \).

- **Threshold Upload.** It is used for anomaly detection, where the threshold is responsible for deciding whether a given sample is abnormal or not. Adversaries send manipulated values of local thresholds to influence the global one. In this work, adversaries overstate their thresholds by creating random values from the uniform distribution in the range \([10^0, 10^9]\). In addition, this attack is combined with model canceling.

- **Random Weight Upload.** It is another model poisoning attack (affects anomaly detectors and classifiers), where adversaries send random weights to the aggregation server. In this work, adversaries generate their weights from a normal distribution with a mean of zero and a standard deviation of three.

In addition to FedAvg, this work evaluates the following secure aggregation functions to increase the robustness of federated models:

- **Trimmed mean.** It extends the classic FedAvg in such a way that the highest and the lowest weight updates are excluded from calculating the mean of the local models weights.

- **Trimmed mean_2.** It is a particular configuration of the previous one that filters out the two highest and lowest entries for the averaging.

- **Coordinate-wise median.** It uses the median of every weight instead of the average. Therefore, outliers belonging to adversarial inputs can be excluded very effectively if there is a sufficient number of honest updates.

Finally, to measure the impact of the attacks and the robustness provided by the aggregation functions when evaluating normal and under-attack behaviors, behavioral data of each participant is concatenated, and the F1-score metric is calculated as

\[
F1 - score = \frac{TP}{TP + \frac{1}{2} (FP + FN)}
\]

TP: True Positives

TABLE 8

| Behavior       | RPi3 (diff.) | RPi4_1 (diff.) | RPi4_2 (diff.) | RPi4_3 (diff.) |
|----------------|-------------|---------------|---------------|---------------|
| normal         | 100% (100%) | 100% (1.1%)   | 100% (0%)     | 99.3% (3.00%) |
| normal_v2      | 100% (100%) | 100% (0%)     | 100% (0%)     | 100% (0%)     |
| delay          | 0% (-100%)  | 100% (0%)     | 100% (0%)     | 100% (0%)     |
| disorder       | 0% (-100%)  | 97.3% (0.6%)  | 100% (1.4%)   | 90.0% (-10.0%)|
| freeze         | 0% (-2.1%)  | 8.5% (-2.4%)  | 5.7% (-2.8%)  | 3.1% (-4.6%)  |
| hop            | 2.4% (-97.6%) | 100% (0%) | 100% (0%) | 98.9% (1.1%) |
| mimic          | 0% (-100%)  | 100% (0%)     | 100% (0%)     | 100% (0%)     |
| noise          | 0% (-100%)  | 100% (0%)     | 100% (0%)     | 100% (0%)     |
| repeat         | 3.1% (-3.1%) | 3.5% (-2.0%)  | 10.4% (-3.1%) | 4.3% (-2.9%)  |
| spoof          | 0% (-100%)  | 100% (0%)     | 98.5% (-1.5%) | 100% (0%)    |
Positive, TN: True Negative, FP: False Positive, FN: False Negative)

5.1 Robustness of Scenario 1

5.1.1 Attack Behavior Injection

From zero to four participants per device type (0% to 33% of the federation) are turned into adversaries. This adversary setup is repeated three times (one per device type) and adversaries use attack samples to train the federated model. The first row of Fig. 6 shows for each device type the F1-score of the federated Autoencoder according to the implemented aggregation function, and the number of adversaries belonging to the RPi3. The second row shows the same, but when the adversaries belong to the RPi4 2GB family. Due to space constraints and similarities with the RPi4 2GB family, it is not shown when attackers belong to the RPi4 4GB type.

As can be seen in the two first rows of Fig. 6, for FedAvg (blue line), one adversary (8% of the federation) decreases the F1-score of each device below 70%, and four adversaries (33%) destroy the model performance (~50%). Furthermore, the injecting device type matters, and in general, attacks performed by RPi3 have more impact than when they are executed by RPi4. This is because RPi3 are fewer in the federation, and their relevance in the federated models is higher. In particular, with four malicious RPi3, the models are destroyed (first raw). In contrast, due to the non-IID data and the higher number of RPi4, when four RPi4 are malicious, their impact on RPi3 is lower. Comparing the aggregation functions, coordinate-wise median (in red) performs the best in general, achieving an F1-score above 60% for all test sets up to two adversaries. Especially in the case of adversarial RPi3, the aggregation function achieves excellent robustness with F1-scores above 80% for up to 4 adversaries. It reflects that coordinate-wise median is able to filter most of those vectors having extreme values due to malicious device manipulations. In contrast, trimmed mean (in green) shows unstable performance. It performs well when the injecting device is the RPi3 (as coordinate-wise median), but bad when the injecting device is the RPi4 (like FedAvg).

5.1.2 Model Canceling and Threshold Attack

In contrast to the previous attack, the impact of model canceling does not depend on the device type executing it. For this attack, up to six adversaries affect the model robustness. It means that the number of participants varies from 12 (no adversaries) to 18 (with six malicious actors, 33%). It is important to remember that the model canceling attack is combined with an overstatement of the threshold.

The third row of Fig. 6 shows how FedAvg aggregation is only capable of defending against one adversary. Most importantly, the threshold overstatement can destroy the model performance once one manipulated threshold is not filtered. In this scenario, the coordinate-wise median provides a very robust defense since the federation maintains very good performance even with six adversaries. However, trimmed mean techniques (in orange and green) only increase the robustness while the number of filtered values is lower than the adversaries. In conclusion, while the mean is shifted heavily towards the attackers for the FedAvg and trimmed mean aggregations, the median can be more stable against largely different adversarial model weights. However, in the case of ≥ 50% adversarial percentage, the median would also lose effectiveness.

5.1.3 Random Weight Uploads

The fourth row of Fig. 6 shows the impact of the adversaries per aggregation function. Here, similar results to the previous attack are observed. Coordinate-wise median performs the best, followed by trimmed mean and the basic trimmed mean. Nonetheless, in this attack where adversaries produce random weights, it is not as obvious as for model canceling.
how the aggregation function can filter the exact weight values. Random weights can be in a completely honest range for some layers or hidden units, but they can also be extreme values for others. It depends on which distribution the random values are sampled and whether they are extreme values compared to the honest weights. However, random weights have no significant impact on the median.

This scenario has shown that data poisoning attacks performed by RPi3 have more impact than when they are executed by RPi4 due to their fewer presence in the federation. Then it has also demonstrated that robust aggregation methods improve the model resilience against adversaries. In all attacks, coordinate-wise median filters more malicious weights, offering the best robustness. It maintains the model performance almost unaltered in 3 of 4 adversarial attacks, only decreasing (still better than the other aggregation methods) when RPi4_1 performs a data poisoning attack. Besides, trimmed mean_2 also shows good robustness when the number of adversaries is low. However, when this number increases, this aggregation method becomes unstable, and the attack succeeds.

5.2 Robustness of Scenario 3

5.2.1 Label Flipping

From zero to four adversaries of each device type take part in this experiment. This adversary setup is repeated three times (once per device type), and adversaries flip their data labels for training.

The first two rows of Fig. 7 show that for FedAvg, the attack does not have a significant impact on the model performance, especially when RPi3 acts as adversary. This is because certain attack characteristics are already available in the federation. For example, if there are two adversarial RPi3, the normal and freeze behavior labels are flipped for this specific device type only. However, the knowledge about these behaviors is fully present for the other two device types. It explains the much higher F1-score than in other attack scenarios. Apart from that, the trimmed mean function is the one providing more robustness for all devices. In contrast, coordinate-wise median shows a different pattern where the model performs poorly, especially for RPi3. It can be appreciated how performance even improves with the presence of some adversaries. However, with too many adversaries flipping labels, the selection of participants weights becomes a random task, and the coordinate-wise median is not a valid countermeasure. This occurs because the random values fall on different sides of the median, shifting the “honest” values away from the center of the distribution.

5.2.2 Model Canceling

In contrast to the anomaly detection experiment, the threshold cannot be attacked in this case. Therefore, the third row of Fig. 7 reports the attack results regardless of the device type acting maliciously. As can be seen, while the performance for trimmed mean excluding one extreme value is very similar to FedAvg aggregation, the exclusion of two extreme values (trimmed mean_2) helps to protect from one more adversary. Still, the performance drops below 20% for three or more adversaries. Coordinate-wise median performs better than the other aggregation functions with four or more adversaries but does not present a viable solution either (the F1-score ranges between 20-60%). It might be explained by the fact that the median filters out the good weights of RPi3 because this device type represents a minority in the federation.

5.2.3 Random Weight Uploads

It considers from zero to six adversaries executing random weight model uploads. The fourth row of Fig. 7 reports the
FI-score of the federated MLP for different aggregation functions and adversaries. As can be seen, the trimmed mean function provides the most robust results in general (and especially for RPi4). In this attack, adversaries generate random weights that are not always malicious or extreme values for the federated model. It causes some instability, which is also impacted by the fact that random adversarial weights cancel out each other. Indeed, the more adversaries are introduced, the more random the global model becomes. Therefore, FedAvg is highly unstable. The second trimmed mean variant provides a better defense as more adversaries can be filtered, being especially in favor of RPi4. Finally, coordinate-wise median does not perform well for low numbers of adversaries but provides an effective countermeasure for four or more malicious participants.

This scenario has shown that label flipping attacks add randomness and instability to the models performance regardless of the device type executing the attacks. In general, the attack does not have a significant impact because flipped behaviors are already present in others devices, and when the number of adversaries is high, they cancel their impact each other. In addition, robust aggregation methods are not as effective as in Scenario 1. Here, there is not a clear aggregation method better than the rest. Trimmed mean is the one offering the best results under label flipping attacks, while trimmed mean_2 and coordinate-wise median have the best results for model canceling and random uploads attacks, but still with a significant performance loss compared to no-attack situations.

6 SUMMARY, CONCLUSIONS, AND FUTURE WORK

This work first creates an FL-oriented dataset composed of samples from eight different SSDF attacks and two versions of normal behavior for a total of four physical spectrum sensors of ElectroSense. After that, four federated scenarios based on anomaly detection and binary classification are created to evaluate and compare the detection performance of FL models and DL models where data privacy is neglected. Finally, this work analyzes the impact of different amounts of malicious participants executing data and model poisoning attacks against FL models equipped with different aggregation mechanisms (federated averaging, trimmed mean, and coordinate-wise median).

As the main conclusion of this study, FL achieves a detection performance that can compete with centralized DL approaches without significant limitations. Anomaly detection is the best approach when detecting SSDF attacks in the selected scenario due to its training simplicity, the detection of zero-day attacks, and its robustness. In terms of robustness and the impact of anti-adversarial mechanisms, the FedAvg is particularly vulnerable to extreme values as they distort the selected global weight average entirely. The trimmed mean aggregation may be able to filter extreme values up to some extent, but if there are sufficiently many adversaries, not all weight uploads can be excluded. Further, there is the concern of excluding honest participants’ weights instead of adversaries. Thus, the best number of updates to exclude from the averaging is very difficult to determine in trimmed mean. Coordinate-wise-median is the best mitigation for anomaly detection, but binary classifiers are still affected with > 33% of adversaries. In summary, coordinate-wise median is the method offering the best robustness for unsupervised scenarios (anomaly detection), while trimmed mean-based aggregations are the ones offering the best results in supervised scenarios (classification).

As future work, there is still room for further research about robust aggregation mechanisms. In the case of anomaly detection, domain-specific aggregation functions could be used to filter adversaries more effectively by leveraging further knowledge about common distributions and fingerprint patterns of normal behavior. For instance, it could be utilized that the threshold of an honest federation participant should be in a certain range for a given device type. Lastly, a larger dataset could greatly enhance the exploration of the FL use case. It could allow researchers to test more extensively how heterogeneity and non-IID data influence federated model performance.

REFERENCES

[1] A. Aldweesh, A. Derhab, and A. Z. Emam, “Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues,” Knowl.-Based Syst., vol. 189, 2020, Art. no. 105124.

[2] A. Blaise, M. Bouet, V. Conan, and S. Secchi, “BotFP: Fingerprints clustering for bot detection,” in Proc. IEEE/IFIP Netw. Operations Manage. Symp., 2020, pp. 1–7.

[3] R. Calvo, “es-sensor,” 2020, Accessed: Jan. 31 2022. [Online]. Available: https://github.com/electrosense/es-sensor

[4] P. S. Chatterjee, “Systematic survey on SSDF attack and detection mechanism in cognitive wireless sensor network,” in Proc. IEEE Int. Conf. Intel. Technol., 2021, pp. 1–5.

[5] ElectroSense, “Collaborative spectrum monitoring,” 2021, Accessed: Jan. 31 2022. [Online]. Available: https://electrosense.org

[6] A. Hanza, H. H. Gharakheili, and V. Sivaraman, “Combining MUD policies with SDN for IoT intrusion detection,” in Proc. Workshop IoT Secur. Privacy, 2018, pp. 1–7.

[7] B. Han, R. Jhaveri, H. Wang, D. Qiao, and J. Du, “Application of robust zero-watermarking scheme based on federated learning for securing the healthcare data,” IEEE J. Biomed. Health Inform., to be published, doi: 10.1109/JBIH.2021.3129936.

[8] R. H. Hsu et al., “A privacy-preserving federated learning system for android malware detection based on edge computing,” in Proc. IEEE 15th Asia Joint Conf. Infotech. Invertur., 2020, pp. 128–136.

[9] A. Huertas Celdrán, P. M. Sánchez Sánchez, G. Bovet, G. Martínez Pérez, and B. Stiller, “CyberSpec: Intelligent behavioral fingerprinting to detect attacks on crowdsensing spectrum sensors,” 2022, arXiv:2201.05410.

[10] M. Iemmili, M. Depken, W. Bouilla, R. H. Jhaveri, G. Srivastava, and T. R. Gadekallu, “A novel model based on window-pass preferences for data-emergency-aware scheduling in computer networks,” IEEE Trans. Ind. Informat., to be published, doi: 10.1109/TII.2022.3149896.

[11] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: Techniques, datasets and challenges,” Cyber-phys. Syst., vol. 2, no. 1, pp. 22–29, 2019.

[12] A. Kumar and T. J. Lim, “EDIMA: Early detection of IoT malware network activity using machine learning techniques,” in Proc. IEEE 5th World Forum Internet Things, 2019, pp. 289–294.

[13] J. Liu, H. Shen, H. S. Narman, W. Chung, and Z. Lin, “A survey of mobile crowdsensing techniques: A critical component for the Internet of Things,” ACM Trans. Cyber-Phys. Syst., vol. 2, no. 3, pp. 1–26, 2018.

[14] B. McManeh, E. Moore, D. Ramage, S. Hampson, and B. A. Y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Proc. Int. Conf. Artif. Intell. Statist., 2017, pp. 1273–1282.

[15] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, “DiO: A federated self-learning anomaly detection system for IoT,” in Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst., 2019, pp. 756–767.
K. Pillutla, S. M. Kakade, and Z. Harchaoui, “Robust aggregation for federated learning,” IEEE Trans. Signal Process., vol. 70, pp. 1142–1154, 2022.

D. Preuveneers, V. Rimmer, I. Tsingopoulos, J. Spooren, W. Joosen, and E. Ilie-Zudor, “Chained anomaly detection models for federated learning: An intrusion detection case study,” Appl. Sci., vol. 8, no. 12, 2018, Art. no. 2663.

S. Rajendran et al., “Electrosense: Open and big spectrum data,” IEEE Commun. Mag., vol. 56, no. 1, pp. 210–217, Jan. 2018.

V. Rey, P. M. Sánchez Sánchez, A. Huertas Celdrán, and G. Bovet, “Federated learning for malware detection in IoT devices,” Comput. Netw., vol. 204, 2022, Art. no. 108693.

N. Rodríguez-Barroso, D. J. López, M. V. Luzón, F. Herrera, and E. Martínez-Cámara, “Survey on federated learning threats: Concepts, taxonomy on attacks and defenses, experimental study and challenges,” 2022, arXiv:2201.08135.

P. M. Sánchez et al., “A survey on device behavior fingerprinting: Data sources, techniques, application scenarios, and datasets,” IEEE Commun. Surveys Tut., vol. 23, no. 2, pp. 1048–1077, Apr.–Jun. 2021.

T. Schenk and A. Iten, “Robust and privacy-preserving ai in crowdsensing platforms,” 2022, Accessed: Jan. 31 2022. https://github.com/Leitou/map-experiment/

R. Taheri, M. Shojaifar, M. Alazab, and R. Tafazolli, “Fed-IoT: A robust federated malware detection architecture in industrial IoT,” IEEE Trans. Ind. Inform., vol. 17, no. 12, pp. 8442–8452, Dec. 2021.

L. Xu, X. Zhou, X. Li, R. H. Jhaiveri, T. R. Gadekallu, and Y. Ding, “Mobile collaborative secrecy performance prediction for artificial IoT networks,” IEEE Trans. Ind. Inform., vol. 18, no. 8, pp. 5403–5411, Aug. 2022.

K. Yadav, S. D. Roy, and S. Kundu, “Defense against spectrum sensing data falsification attacker in cognitive radio networks,” Wirel. Pers. Commun., vol. 112, pp. 849–862, 2020.

Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, “Federated learning,” Synth. Lectures Artif. Intell. Mach. Learn., vol. 13, no. 3, pp. 1–207, 2019.

D. Yin, Y. Chen, R. Kannan, and P. Bartlett, “Byzantine-robust distributed learning: Towards optimal statistical rates,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 5650–5659.

Pedro Miguel Sánchez Sánchez received the MSc degree in computer science from the University of Murcia, Spain. He is currently working toward the PhD degree in computer science with the University of Murcia. His research interests include focused on continuous authentication, networks, 5G, cybersecurity and the application of machine learning, and deep learning to the previous fields.

Alberto Huertas Celdrán received the MSc and PhD degrees in computer science from the University of Murcia, Spain. He is currently a post-doctoral fellow with the Communication Systems Group CSG, Department of Informatics IfI with the University of Zurich UZH. His scientific interests include IoT, BCI, cybersecurity, data privacy, continuous authentication, semantic technology, and computer networks.

Adrian Lars Benjamin Iten received the BSc degree in computer science from the University of Zurich, Switzerland evaluating compact Machine Learning (ML) models for Natural Language Processing (NLP). He is currently working toward the MSc degree in software systems with the University of Zurich and has previously worked as a software engineer in the industry for five years. His scientific interests include the applications of ML for cybersecurity, and NLP.

Gérome Bovet received the PhD degree in networks and computer systems from Telecom ParisTech, France, in 2015, and the executive MBA degree from the University of Fribourg, Switzerland, in 2021. He is the head of data science for the Swiss Department of Defense, where he leads a research team and a portfolio of about 30 projects in the context of Cyber-Defence and intelligence. His work focuses on Machine and Deep Learning approaches, with an emphasis on anomaly detection, adversarial and collaborative learning applied to data gathered by IoT sensors.

Gregorio Martínez Pérez is full professor with the Department of Information and Communications Engineering, University of Murcia, Spain. His scientific activity is mainly devoted to cybersecurity and networking. He is working on different national (14 in the last decade) and European IST research projects (11 in the last decade) related to these topics, being Principal Investigator in most of them. He has published more than 160 papers in national and international conference proceedings, magazines and journals.

Burkhard Stiller received the informatik-diplom (MSc) degree in computer science and the Dr. rer. nat. (PhD) degree from the University of Karlsruhe, Germany, in 1990 and 1994, respectively. Since 2004 he chairs the Communication Systems Group CSG, Department of Informatics IfI, University of Zurich UZH, Switzerland as a full professor. His main research interests include published in well more than 300 research papers and include systems with a fully decentralized control (blockchains, clouds, peer-to-peer), network and service management, Internet-of-Things, and telecommunication economics.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.