IoT Device Fingerprinting: Machine Learning based Encrypted Traffic Analysis

Nizar Msadek, Ridha Soua, Thomas Engel
Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg
Esch-sur-Alzette, Luxembourg
Email: {nizar.msadek, ridha.soua, thomas.engel}@uni.lu

***Abstract*—Even in the face of strong encryption, the spectacular Internet of Things (IoT) penetration across sectors such as e-health, energy, transportation, and entertainment is expanding the attack surface, which can seriously harm users' privacy. We demonstrate in this paper that an attacker is able to disclose sensitive information about the IoT device, such as its type, by identifying specific patterns in IoT traffic. To perform the fingerprint attack, we train machine-learning algorithms based on selected features extracted from the encrypted IoT traffic. Extensive simulations involving the baseline approach show that we achieve not only a significant mean accuracy improvement of 18.5% and but also a speedup of 18.39 times for finding the best estimators. Obtained results should spur the attention of policymakers and IoT vendors to secure the IoT devices they bring to market.**

***Index Terms*—IoT Devices; IoT network Security; Device Type Fingerprinting; Machine Learning; Traffic Features**

I. Introduction

With more than 30 billion connected devices by 2020, a number expected to reach 500 billion by 2030 [1], the Internet of Things (IoT) is growing at a tremendous pace and promises to radically transform our daily lives. Leveraging the unprecedented number of smart devices deployed everywhere, and the availability of wireless communications technologies for constrained devices (e.g., RFID, Near Field Communication (NFC), Low-Power Wi-Fi, ZigBee, etc.), IoT is increasingly used for vital monitoring and control applications [2]. For instance, things can connect and interact with their surrounding space to send out streams of data to be aggregated and analyzed. The purpose is to deliver insight, which helps public authorities, smart-home owners, factory managers, take more informed decisions and actions. According to Cisco [3], its IoT connectivity management platform is used by over 15,000 companies and connects more than 66 millions things. Realizing IoTs' full market potential, a huge number of companies is providing such connected devices to end users.

This explosive growth in terms both of the number of connected devices and supported services, is accompanied by a concomitant rise in security attacks. Indeed, the peculiarities of IoT, such as the many heterogeneous operating environments coupled with the large scale of current IoT deployments, magnify security threats [4], [5]. Anomaly detection systems will attempt to find patterns in data which do not conform to expected normal behaviour. However, the broad heterogeneity of IoT devices, ranging from smart light bulbs and fingerprint sensor to smart locks, makes identifying unusual patterns an unpractical task. Unquestionably, misuse-based intrusion detection systems will also fail, given the fragmentation of the IoT device market. In addition, to deal with unsecured devices or equipment, it is common that security patches are applied through software and firmware updates. However, not all IoT devices support automated software update. A device-specific attack was demonstrated by [6] where authors managed to recover the private key used to authenticate firmware updates for Philips smart lightbulbs. Consequently, revealing the IoT device type could have significant impacts on the reliable and safe operation of a variety of crucial services. Because of this, in this paper we tackle the problem of IoT device fingerprinting. We demonstrate that machine learning techniques can be used to discover the types of the IoT devices transmitting traffic in the network. This fingerprinting is carried out using only features extracted from the encrypted network traffic. The key contributions of our work are as follows: (i) We propose a new IoT fingerprinting attack that is effective even when devices use encryption or when training sets are small (i.e., relatively to the baseline). The attack does not require handcrafted parameter tuning to segment traffic. This is done autonomously using a proposed novel sliding window technique. (ii) We study the characteristics of many IoT devices based on real dataset that was recently collected by IBM Research and Cisco. We demonstrate that, despite the broad adoption of transport layer encryption, informative features can be derived to disclose the type of IoT devices. (iii) We conduct different exploration and comparative evaluation measurements on the dataset. We find that our attack is more accurate and efficient than the baseline. It achieves a speedup ratio of 18.39 times for finding the best estimators, and has an accuracy improvement of 18.5% when compared to baseline.

The rest of this paper is organized as follows. Section II outlines our assumptions, problem statement, goal and the limitation of the baseline. Then, we detail our IoT device fingerprinting approach using machine learning over encrypted traffic. Section IV describes the dataset and our technique for feature analysis and selection. Experimental results and findings are detailed in Section V. We review the current literature on device fingerprinting in Section VI. Section VII concludes this paper.

II. Problem Statement and Baseline Method for Device Fingerprinting

A. Assumptions, Problem Statement, and Goal

In this work, we consider a common scenario where an attacker is a passive observer. He does not modify transmissions and he is not able to decrypt packets. The attacker is able to passively gather traffic activities about all IoT devices. Hence, he either monitors the gateway itself or compromised IoT devices. In addition, we assume the attacker possesses sufficient computational resources to train classifiers and make predictions from large training datasets. Based on this set of assumptions, we address the problem of IoT device fingerprinting by performing Machine Learning (ML)-based encrypted traffic Analysis. Our problem can be formalized as follows: Given a distributed system of several IoT devices $\mathcal{D}$ and a set of encrypted traffic $\mathcal{P}$, we treat the task of IoT device fingerprinting as a multi-class classification problem. Our aim is to map each encrypted traffic stream to the device type that has likely produced it. In this work, we limit the scope of our study to the case where $\mathcal{D}$ is finite and meta-information for all devices is available. Let $\mathcal{P}^{d_i} = \langle .., p^i_t, .. \rangle$ be the traffic generated by $d_i \in \mathcal{D}$ where p^i_t corresponds to one of its packets, t its arrival time, and $[p^i_t]$ its duration. The main goal is to look for a ML classification model that, once trained and benchmarked using its best tuning parameters, is able to identify the corresponding device d_i for any unseen encrypted traffic $\mathcal{P}^{d_i}$, and thus for $\forall d_i \in \mathcal{D}$. The first step towards this goal is to enable the extraction of features at runtime. The second step is to build a classifier that takes those features as input and makes classifications in order to infer the type of devices.

B. Baseline method for device fingerprinting

1) Baseline Overview: As a baseline, we use a fingerprinting methodology approach [7] that has recently gained considerable attention, and is originated from a cooperative study between IBM, Cisco, and the University of New South Wales. It is based on a Dominant Protocol Analysis (DPA) feature Set, static segmentation, and Random Forest (RF) classification. The combination of DPA features and RF have been used for similar IoT traffic characterization approaches, such as IoT Sentinel [8]. The DPA features are derived from the meta-information in packet headers, which is segmented using the sliding window technique. Full details of the traffic features available in DPA can be found in [5]. The cross-validation method is applied, and the dataset is split into training and testing sets. This cross-validation is repeated 10 times and results are then averaged to produce a single performance metric. The drawbacks of this approach are twofold: (1) it relies on handcrafted parameter tuning and does not segment traffic autonomously as we do in this work. (2) it is only suited for large training datasets (i.e., $\geqslant$ 90% of total traffic) with noise-free traffic. For smaller training datasets (less than 90% of total traffic), this approach becomes very sensitive to noise and slow to find best estimators, i.e., the efficient set of parameters that allows the baseline to train optimally.

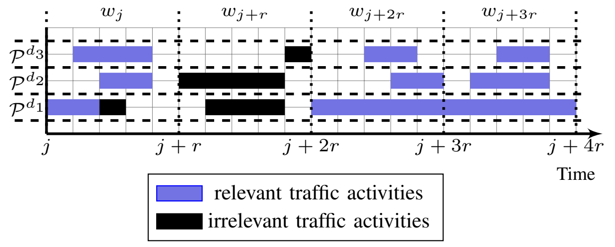


Fig. 1: Traffic segmentation using the original sliding window technique.

2) Main Limitations: For traffic segmentation, the concept of sliding windows is adopted in the baseline approach. Its main advantage is that it can easily separate traffic activities processed in the past from recent ones. It makes use of two important parameters: namely length l and shift r. Figure 1 gives an overview of this technique using an example of three devices $d_1, d_2, d_3 \in \mathcal{D}$, where $l = r$, j is the time at which the first window begins and $j + l$ the time at which it terminates and the next window begins. In the first time period, the traffic activities are extracted from window w_j. We denote by $\mathcal{P}^{d_i}_{[w_j]} = \langle p^i_j, ..., p^i_{j+l-1} \rangle$ the traffic activities of device d_i within window w_j. Then, the next window moves by shifting r from w_j to w_{j+r} and the traffic activities are extracted from $\mathcal{P}^{d_i}_{[w_{j+r}]} = \langle p^i_{j+r}, ..., p^i_{j+r+l-1} \rangle$. This process is attractively executed until all windows have been examined. Although this technique is simple and efficient, its also has some drawbacks: The first consists in the way l and r are paremeterized. In the baseline approach, These two parameters are fixed at design time and do not change during runtime. Second, not all traffic in each window is necessarily relevant – if we consider for instance the case of w_{j+r} where no relevant traffic activities exist. Furthermore, some devices might have long sequences of traffic (e.g., $\mathcal{P}^{d_1}$) whose tracing with a small window becomes completely intractable.

III. IoT device fingerprinting using ML over encrypted traffic

In this section, we describe our methodology for IoT device fingerprinting. It consists of the following steps: (i) a dynamic segmentation based on an extended sliding window technique; (ii) an analysis of encrypted traffic to extract relevant features, and finally (iii) a study of various classification algorithms.

A. Extended Sliding Window Technique

To overcome the limitations introduced in Section II-B2, we propose an improvement of the sliding window technique by making its length adaptive at runtime, according to the occurrence of relevant traffic. Its basic idea is roughly: "*whenever relevant traffic activities are becoming long, the window will grow autonomously; otherwise it will shrink to discard irrelevant traffic. Instead of shifting by r, it uses the next disjoint relevant activities to move over traffic*".

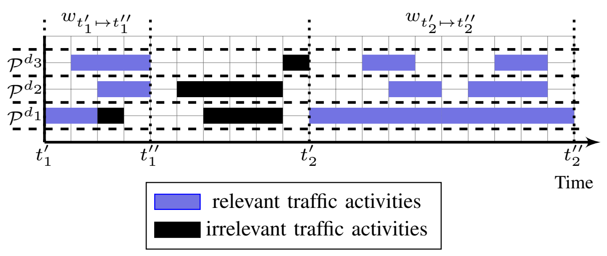


Fig. 2: Simplified illustration of our approach using the same example of Fig 1

Figure 2 illustrates our approach using the same example mentioned above. Initially, we iteratively analyze the temporal overlaps between relevant activities. Let $\mathcal{S}^{d_k}$ and $\mathcal{S}^{d_l}$ be subsequences of traffic, where $\mathcal{S}^{d_k} \subseteq \mathcal{P}^{d_k}$, $\mathcal{S}^{d_l} \subseteq \mathcal{P}^{d_l}$, $\forall d_k, d_l \in \mathcal{D}$. We say that $\mathcal{S}^{d_k}$ and $\mathcal{S}^{d_l}$ are disjoint if they do not overlap in time. We write this as $\mathcal{S}^{d_k} \bowtie \mathcal{S}^{d_l}$ and define it as Formula 1.

$$\mathcal{S}^{d_k} \bowtie \mathcal{S}^{d_l} \equiv \{t_u+[p^k_{t_u}] < t_v \vee t_v+[p^l_{t_v}] < t_u | \forall p^k_{t_u} \in \mathcal{S}^{d_k} \text{and} \, \forall p^l_{t_v} \in \mathcal{S}^{d_l}\} \tag{1}$$

Then the algorithm is split into two phases, the first of which is similar to the original approach, but we now use our overlap analysis to segment the traffic into fragments of disjoint activities. For each fragment j, we determine the start and termination times, and store these values pairwise (t'_j, t''_j) in list $\mathcal{T}$. The algorithm then enters its second phase, in which it iterates through $\mathcal{T}$ to autonomously adjust l and r. It creates a temporal window $w_{t'_j \mapsto t''_j}$ for every element $(t'_j, t''_j) \in \mathcal{T}$. The length and shift are computed autonomously using formulas 2 and 3, respectively.

$$l_j = t''_j - t'_j \tag{2}$$

$$r_j = \begin{cases} 0 & \text{if } j = 1 \\ t''_j - t'_{j-1} & \text{otherwise} \end{cases} \tag{3}$$

B. Feature Extraction

In this phase, we look for features that are time-invariant, or at least do not undergo any change for a reasonably long time. One simple technique would be to consider the payload of packets. However, as discussed earlier, with encrypted traffic we cannot inspect the payload content. We therefore extract our features only from packet headers, using the two following techniques:

- **Basic Features based on dominant protocol analysis**: We analyze the different OSI protocol layers that the devices use to interact. We identify the set of dominant protocols that can be used to extract features, which implicitly reveal unique characteristics of the considered devices. We study their impact in terms of predictive performance and tune their selection iteratively. The set of features considered in this study includes the types of dominant protocols, port intervals, packet size, packet quantity, availability time, inter-arrival time, cumulative count, etc.
- **Derived features based on statistical traffic rates**: Features are obtained by converting the flow of basic features into statistical distributions. This set includes minimum, maximum, mean, median, standard deviation, variance, etc.

C. Machine Learning Algorithms

We now introduce the list of considered algorithms along with their corresponding tuning parameters. Our selection was based on the studies described in [9], [10], and [11].

1) **K-nearest Neighbours (k-NN)** [12] are simple non-parametric learning algorithms which do not require a model to be fitted. Given an input sample x, the k training examples closest in distance to x are determined. Sample x is classified using a majority vote among these k neighbours. The parameter k characterizing this number of neighbours is also known as *n_neighbors*, when using the terminology of scikit-learn [13]. The other parameter used to determine the manner of prediction is *weights*. If it is set to *uniform*, then all points in each neighbourhood are weighted equally. Otherwise, it is set to *distance*, in which the weight points are calculated based on the inverse of their distance. This means that nearer neighbours of a query point will have a greater influence than those that are further away.
2) **Support Vector Machine (SVM)** [14] finds an optimal boundary between two different classes by maximizing the margin between patterns belonging to the classes. The SVM algorithm not only performs linear classification, but also non-linear classification using a *kernel function*, i.e. cases where the decision function is not a linear function of the data. When training a SVM, two parameters must be considered: *gamma* and C. The *gamma* parameter defines how much influence a single training example has. The parameter C, in contrast, trades off misclassification of training examples against simplicity of the decision function, i.e, a low C makes the decision function smooth, while a high value of C aims to classify all training examples correctly.
3) **Random Forest (RF)** [15] is an ensemble learning technique that combines numerous decision trees at training time using several bootstrapped samples of the training data. The class prediction is the result of combining these individual tree predictions. This algorithm makes use of a randomly-chosen subset of features to find the best split for each node and is robust against overfitting. Two important parameters must be considered: *max_depth* and *max_features*. The former represents the maximum number of trees, while the latter stands for the maximum number of features considered for node splitting.
4) **AdaBoost (AB)** [16] invokes weak learners in a series of iterations to maintain a set of weights over the training set. Initially, all weights are set equal. After each iteration, the weights of incorrectly classified samples are increased so that each learner is forced to focus on the misclassified instances. A confidence value is given to each learner

based on its predictive performance, and a new pattern is classified through a weighted voting mechanism that merges the prediction of all the weak learners. The most decisive parameter is *learning_rate* whose value is within the interval $[0, 1]$ and defines the step size while learning.

5) **Extra-Trees (ET)** [17] is an ensemble classification method made up of several tree classifiers trained independently. It performs like an ordinary RF, but the top-down splitting in the tree learner is done randomly. Instead of determining the best split combination (based on, e.g., entropy or the Gini impurity) a random value is selected for the split. As in RF, the *max_depth* parameter represents the maximum depth of the tree, whereas *max_features* stands for the maximum feature number to be considered when looking for the split.

In the current study, we favoured algorithms with ensemble methods, given that their classifiers generally achieve better accuracy than other single classifiers [10], [18]. The performance measurements were conducted using the scikit-learn package library [13]. To take the advantage of parallel computations, we set the parameter *n_jobs* to -1, meaning that the computations are run simultaneously on all cores of the machine. All other parameters are set to the default values provided by scikit version 0.19.1.

IV. Data Collection and Analysis

A. Dataset

The dataset [19] used in this study is obtained from the University of New South Wales [20], [21], and the same dataset was recently used by IBM Research and Cisco Systems in [7]. Figure 3 illustrates the testbed adopted by these studies:

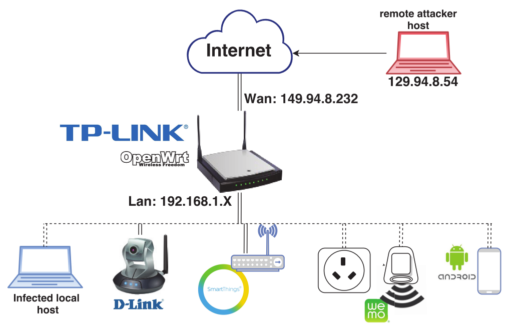


Fig. 3: Testbed showing the IoT devices and gateway [21].

campus network was instrumented with a diversity of IoT devices. Table I provides more detailed information about the devices. These devices included cameras, lights, activity sensors, health and well-being monitors. The TP-Link router – shown in Figure 3 – serves as a gateway to the public Internet. It was flashed with OpenWrt [22] as well as some other packages to enable traffic capture, including for instance the *tcpdump* tool. The WAN interface of the gateway is connected to the global Internet while the IoT devices are attached to the LAN/WLAN interfaces. The aim of this experiment was to passively collect the traffic activities of all devices. These activities were collected over a period of three weeks and stored as pcap files on an external hard drive attached to the gateway. We make use of these pcap files to analyze relevant activities and to extract informative features with unique patterns, enabling us to accurately distinguish between the devices.

Category	**Device**	**Wireless/Wired**
Hubs	Smart Things	Wired
	Amazon Echo	Wireless
Cameras	Netatmo Welcome	Wireless
	TP-Link Day Night Cloud camera	Wireless
	Samsung SmartCam	Wireless
	Dropcam	Wireless
	Insteon Camera	Wired/Wireless
	Withings Smart Baby Monitor	Wired
Switches & Triggers	Belkin Wemo switch	Wireless
	TP-Link Smart plug	Wireless
	iHome	Wireless
	Belkin wemo motion sensor	Wireless
Air quality sensors	Nest Protect smoke alarm	Wireless
	Netatmo weather station	Wireless
Healthcare devices	Withings Smart scale	Wireless
	Blipcare Blood Pressure meter	Wireless
	Withings Aura smart sleep sensor	Wireless
Light Bulbs	LiFX Smart Bulb	Wireless
Electronics	Triby Speaker	Wireless
	Pix-Star Photo-frame	Wireless
	HP Printer	Wireless

TABLE I: List of IoT devices in the smart campus environment.

B. Feature Analysis and Selection

To identify and select relevant features, we first study the different traffic characteristics of IoT devices by considering the following aspects:

1) Dominant Protocol Analysis (DPA): Our focus here is on the application layer protocol, as seen in Figure 4, where we examine the most used destination ports for TCP/UDP packets. We found that the port number distribution is not uniform across all packets. Some devices tend to send more of their packets using well-known ports than other ports. We therefore decided to group the ports into ranges: 1) *system ports* from 0 to 1023, 2) *registered ports* from 1024 to 49151, and finally 3) *dynamic ports* from 49152 to 65535.

Based on this arrangement, we define feature $f_{range}(port)$ as a function of $port$ as follows:

$$f_{range}(port) = \begin{cases} 1 & \text{if } port \in [0, 1023] \\ 2 & \text{if } port \in [1024, 49151] \\ 3 & \text{if } port \in [49152, 65535] \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

In the remainder of this paper, we focus on the range [0,1023] due to the fact that *system ports* cover the official list of well-known protocols. A deep inspection of this interval has allowed us to find out that TCP port 443 – the well-known port for HTTPS – is the TCP port most used by all devices. Around 72% of packets are sent through this port. This means that the majority of packets contents are encrypted and therefore we cannot rely on inferring sensitive data solely

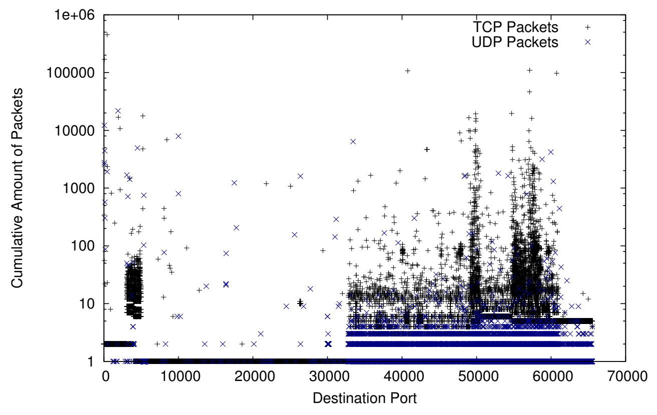


Fig. 4: The correlation between used ports and amount of packets collected from one representative day (September 28th chosen for illustrative purpose).

from payloads. The second dominant port is TCP port 80 – representing HTTP – which constitutes 27% of TCP traffic. Our analysis shows also that TCP Ports 853/53, 445, 548 and 22 – indicative of DNS, SMB, AFP and SSH respectively – are present among the other used protocols. Moving away from TCP, we can observe UDP packets in the range of *system ports* as well. They are associated essentially with UDP port 1900, indicative of the SSDP protocol. This protocol is used for advertising the presence of devices in the network, and for discovering new services. It accounts for 20% of UDP packets. DNS and NTP packets are also visible among the UDP packets using ports 53/853 and 123 respectively. We choose 18 protocols to define features. The complete list of considered protocols is given in Table II. The output of each feature is boolean and can thus take either the value of 0 or 1. It is set to 1 if the protocol is used, otherwise it has the value of 0.

OSI Layer	Protocol	Feature type
Application layer (10)	HTTP / HTTPS / DHCP	bool. (0/1)
	SSDP / DNS / MDNS	bool. (0/1)
	NTP / SMB / AFP / SSH	bool. (0/1)
Transport layer (2)	TCP / UDP	bool. (0/1)
Network layer (4)	IP / ICMP	bool. (0/1)
	ICMPV6/IGMP	bool. (0/1)
Data link layer (2)	ARP / LLC	bool. (0/1)

TABLE II: The protocols chosen for defining features.

2) Statistical Analysis From Traffic Rates: There are additional features that can be used to identify the devices. These include – but are not limited to – the mean and standard deviation of traffic volume, as well as the inter-arrival time of packets. Regarding the latter attribute, we found that there is a unique pattern for some devices. For instance, inter-arrival times of 90, 60 and 20 seconds occur respectively for the HP Printer, the iHome switch, and the Netatmo Welcome. Note that such an occurrence of inter-arrival times appears with a probability greater than 70%. The other attribute we consider is the traffic volume. Figure 5 shows our analysis. The values on the x-axis represent the mean traffic volume of devices and their standard deviation is shown on the y-axis.

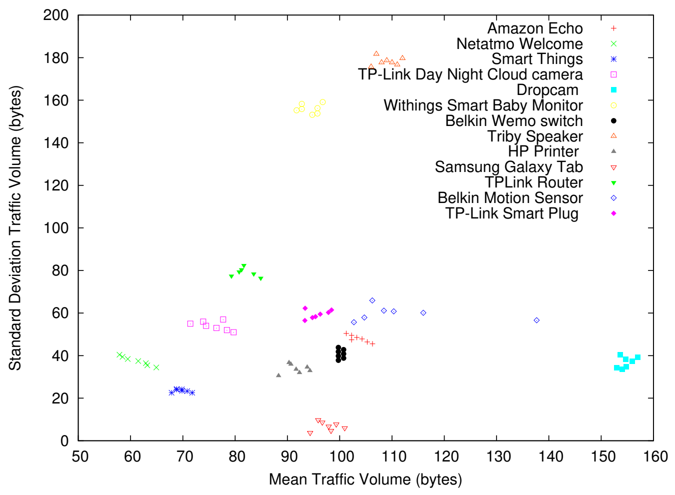


Fig. 5: The mean and standard deviation of packet volume belonging to devices that are responsible for dominant protocols

It is clear that the usage of both features combined achieves an accurate classification output, and the devices are well clustered. For instance, the mean and standard deviation of Belkin Wemo and Smart Things differ by almost an order of magnitude over the dataset period and thus helps to decisively distinguish between the devices.

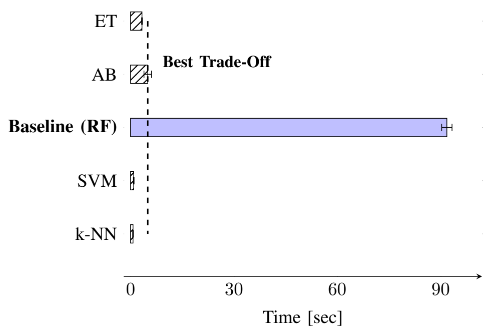
V. Fingerprinting Attack:Experimental Results

To evaluate our proposed fingerprinting attack, we developed a framework for device fingerprinting that allows the consideration of different classifiers and parameters at runtime, during the benchmarking process. This was a necessary step to avoid unfairness in comparing our new techniques, which may impact the overall evaluation result with the baseline approach. Hence, a series of measurements were conducted. The aim was to determine the optimal tuning parameters for each learning classifier that could achieve a performance gain over the baseline. The methodology for investigating these measurements consists of the following two steps:

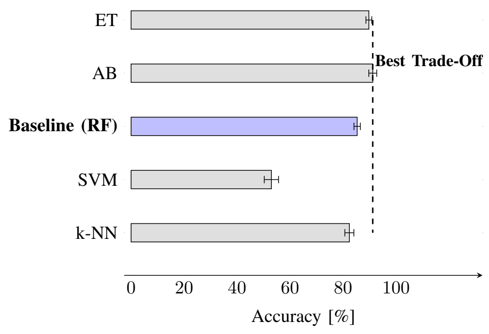
- Exploration Evaluation: is used to find best estimators which refer to the set of efficient parameter values allowing classifiers to train optimally. The training set is used to conduct these measurements.
- Comparative Evaluation: the classifiers are trained based on their best estimators found in the exploration phase. They are then benchmarked with the baseline according to a set of predictive metrics. The testing set is used to perform these measurements.

A. Exploration Evaluation

The first barrier to comparing classifiers is obviously their different tuning parameters, also known in the literature as



(a) Average time needed for training models



(b) Maximum predictive accuracy achieved

Fig. 6: Evaluation results showing the best trade-off between speedup and accuracy during the training process. The accuracy is calculated using the metric function *best_score_*, provided by *RandomizedSearchCV*, to determine the max score of each best estimator.

hyperparameters. These can greatly influence the time needed to train models and make good predictions. The baseline was reproduced according to the best parameter results achieved in [7]. This corresponds to a model that employs RF based on a daily segmentation without overlapping. The parameters used were $l = 24$ hours and $r = 24$ hours. For other classifiers, there is no need to set l and r, since their traffic segmentation is done autonomously, as described in Section III-A. We trained all classifiers and tuned parameters via 10-fold cross-validation with the *RandomizedSearchCV* function from scikit-learn. The *RandomizedSearchCV* is used to find best estimators from a set of parameter settings. In contrast to *GridSearchCV*, not all parameter values are explored, but rather a fixed number of parameter settings is sampled from the specified distributions. We set 60% of the dataset as a training set and 40% as a testing set. The split was performed with a fixed random strarting state. This has the benefit that the experiments are reproducible and independent of any special properties of e.g. interference delay between cores. The parameter *n_iter* is set to 60. We also set AUC (Area Under the Curve) as the scoring metric in *RandomizedSearchCV*. In the following the best estimators are reported and the performance results are presented in Figure 6.

- Best estimator found for k-NN:

```
KNeighborsClassifier(algorithm='auto',
     leaf_size=30, metric='minkowski',
     metric_params=None, n_jobs=1, n_neighbors=4,
     p=2, weights='distance')
```

- Best estimator found for SVM:

```
SVC(C=5.40948345269, cache_size=200, class_weight=
     None, coef0=0.0, decision_function_shape='ovr'
     , degree=3, gamma=0.129798709295, kernel='rbf'
     , max_iter=-1, probability=False, random_state
     =None, shrinking=True, tol=0.001, verbose=
     False)
```

- Best estimator found for RF (**Baseline**):

```
RandomForestClassifier(bootstrap=True, class_weight
     =None, criterion='gini', max_depth=80,
     max_features='auto', max_leaf_nodes=None,
     min_impurity_decrease=0.0, min_impurity_split=
     None, min_samples_leaf=1, min_samples_split=2,
      min_weight_fraction_leaf=0.0, n_estimators
     =400, n_jobs=1, oob_score=False, random_state=
     None, verbose=0, warm_start=False)
```

- Best estimator found for AB:

```
AdaBoostClassifier(algorithm='SAMME.R',
     base_estimator=None, learning_rate=0.05,
     n_estimators=100, random_state=None)
```

- Best estimator found for ET:

```
ExtraTreesClassifier(bootstrap=False, class_weight=
     None, criterion='gini', max_depth=6,
     max_features=0.7, max_leaf_nodes=None,
     min_impurity_decrease=0.0, min_impurity_split=
     None, min_samples_leaf=1, min_samples_split=2,
      min_weight_fraction_leaf=0.0, n_estimators
     =50, n_jobs=1, oob_score=False, random_state=
     None, verbose=0, warm_start=False)
```

Figure 6 shows the trade-off between training time and maximum accuracy achieved by each classifier during the learning process. It can be seen in Figure 6-a that all classifiers yield an excellent time performance compared to the baseline. The best improvement is achieved by k-NN; it has a speedup ratio of 129.28 times, whereas the worst improvement is associated with AB, with a speedup ratio of 18.39 times – although this is still highly significant compared to the baseline. ET achieves a speedup ratio of 25.55% times. However, examining Figure 6-b, it becomes clear that AB and ET achieve the best trade-off between speedup and accuracy. AB and ET make respectively an accuracy improvement of 5.88% and 2.32% compared to the baseline, making them very applicable to situations with high traffic volume.

B. Comparative Evaluation

Initially, the classifiers are trained with optimal parameter values determined from the exploration phase (see Section V-A). Their predictive results are then benchmarked on the testing set, with the baseline according to the following Key Performance Indicators (KPIs):

- Precision (Pr) $Pr = \frac{TP}{TP+FP}$
- Recall (Re) $Re = \frac{TP}{TP+FN}$

Learning Models		Extra-Trees (ET)	AdaBoost (AB)	**Baseline** – Random Forest (RF)
KPIs in [%]	$MAcc$	79.5 (77.5–81.8)	95.5 (94.1–97.6)	78 (76.7–80.8)
	$Misc$	20.5 (18.2–22.3)	4.5 (3.2–5.9)	22 (20.8–24.7)
	Pr	80 (77–82.5)	96 (95–97)	78 (75–80.2)
	Re	79 (76–81)	95 (94–96)	78 (76–80.4)
	F_1	79.5 (74.2–82.8)	95.5 (93.2–97.9)	78 (76.7–80.9)
Improvement in $MAcc$ over Baseline		**1.5%**	**18.5%**	—%

Performance Indicators	[%]	Support Vector Machine (SVM)	k-Nearest Neighbours (k-NN)
KPIs in [%]	$MAcc$	39 (37–41)	78.5 (76.1–81.4)
	$Misc$	61 (59–63.9)	21.5 (19.1–23.7)
	Pr	40 (37–42)	78 (75.6–79.9)
	Re	38 (36–41)	79 (77.2–80.1)
	F_1	38.9 (36.2–42.3)	78.4 (75.2–80.2)
Improvement in $MAcc$ over Baseline		-39%	**0.5%**

TABLE III: Baseline approach vs. our proposed classifiers

- F_1 score (F_1) $F_1 = 2 \cdot \frac{Pr \cdot Re}{Pr+Re}$
- Mean Accuracy ($MAcc$) $MAcc = \frac{(Pr+Re)}{2}$
- Misclassification ($Misc$) $Misc = 1 - MAcc$

where TP denotes the number of devices whose types are correctly classified as positive (True Positive), FN is the number of devices whose types are incorrectly classified as negative (False Negative), TN stands for the number of devices whose types are correctly classified as negative (True Negative), and FP denotes the number of devices whose types are incorrectly classified as positive (False Positive). Finally, the mean accuracy ($MAcc$) is defined as the overall score of the predictions, whereas the Misclassification ($Misc$) is calculated as its inverse probability.

The experimental results of the baseline and proposed methods are shown in Table III. All classifiers, except the one based on SVM, achieve improvements in $MAcc$ over the baseline on test set. Indeed, our autonomous segmentation and feature extraction are informative for IoT device fingerprinting, since the result demonstrates a good predictive performance for most of the classifiers. The strongest performance, 95.5% $MAcc$, is obtained using the learning model AB. This $MAcc$ offers a significant relative improvement of 18.5% compared to the baseline. When comparing, however, between training and testing accuracies, it is clear that AB remains among the top two best classifiers. This demonstrates that the performance of AB is not susceptible to noise during training, which is consistent with the results obtained by other studies [23].

VI. Related Work

Device fingerprinting has gained a considerable attention from academic and industrial research communities. General device fingerprinting has been reported in [24], [25], [26], which examined various features ranging from packet header to physical properties including clock skews and radio frequency signatures. Wireless device fingerprinting methods have been discussed in [27], [28], [29]. These studies investigated the device type identification by considering the implementation differences of a common protocol such as SIP, across similar devices. However, research studies focusing on IoT device fingerprinting are still in their infancy stages due to the evolving nature of the IoT industry. Our IoT fingerprinting approach is similar in spirit to the work presented by Sivanathan et al. [7], considered as baseline in this paper. It involves a dominant protocol analysis feature set, static segmentation, and RF classification. The drawbacks of this approach are twofold: (1) it relies on handcrafted parameter tuning and does not segment traffic autonomously as we do in this work. (2) it is only suited for large training datasets (i.e., $\geqslant$ 90%) with noise-free traffic. For smaller training datasets (less than 90%), this approach becomes very sensitive to noise and slow to find best estimators. Miettinen et al. in [8] proposed IoT Sentinel, a framework capable of identifying devices from an IoT network and enforcing mitigation measures for those that have potential security vulnerabilities. Their approach does not analyze the behavior of a device as described in the problem statement. Our work is complementary to theirs, it can be used along with their approach to fingerprint devices and provide stronger security. Maiti et al. [30] introduced a device type fingerprinting technique relying on analysis of encrypted traffic. This technique was evaluated on 10 IoT devices and required at least 30,000 frames to be effective. Under a standby state of operation, an IoT device can take days to generate such a volume of traffic. Siby et al. described IoTScanner in [31], an architecture that sniffs network traffic at data link layer, and analyzes it using packet header information extracted from sliding windows. Their work is more concerned with discerning the distinct devices and their presence in network. A drawback of this approach is that two identical device types could be identified as two different device types due to the variations in traffic generated from different interfaces. Our technique, however, is not restricted to a specific interface and adapts the fingerprinting autonomously when a new traffic interface is discovered. Consequently and in contrast to previous work, it identifies the type of an IoT device under any used IoT communication.

VII. Conclusion

Many IoT device vendors cease their technical support in terms of patches or firmware updates shortly after product release. This gives attackers an enormous opportunity to compromise millions of unsecured IoT devices widely used nowadays across many industries. Revealing the IoT device type will leave user devices and data at high risk. To engage the attention of security IoT policymakers and vendors, we demonstrate in this paper, the high accuracy of machine learning techniques in fingerprinting IoT devices over encrypted traffic. We have shown that, without requiring decryption of the underlying traffic, (basic/derived) DPA features and automated sliding windows can be efficiently used in revealing the IoT device type with an accuracy improvement of 18.5% compared to the baseline.

References

[1] Cisco, "Internet of Things: Connected Means Informed," Tech. Rep., Feb. 2016. [Online]. Available: https://www.cisco.com/c/dam/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015.

[3] *DC MarketScape Report Names Cisco a Leader in Worldwide IoT Platforms*, Cisco, January 2018, https://blogs.cisco.com/digital/idc-marketscape-report-names-cisco-a-leader-in-worldwide-iot-platforms.

[4] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in internet of things: The road ahead." *Computer Networks*, vol. 76, pp. 146–164, 2015.

[5] A. R. Sfar, E. Natalizio, Y. Challal, and Z. Chtourou, "A roadmap for security challenges in the internet of things," *Digital Communications and Networks*, vol. 4, no. 2, pp. 118 – 137, 2018.

[6] E. Ronen, A. Shamir, A. Weingarten, and C. OFlynn, "Iot goes nuclear: Creating a zigbee chain reaction," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 195–212.

[7] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying iot traffic in smart cities and campuses," in *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May 2017, pp. 559–564.

[8] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi, and S. Tarkoma, "Iot sentinel: Automated device-type identification for security enforcement in iot," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 2177–2184.

[9] S. B. Kotsiantis, "Supervised machine learning: A review of classification techniques," in *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2007, pp. 3–24.

[10] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15.

[11] M. Woniak, M. Graña, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Inf. Fusion*, vol. 16, pp. 3–17, Mar. 2014.

[12] M.-L. Zhang and Z.-H. Zhou, "A k-nearest neighbor based algorithm for multi-label classification," in *2005 IEEE International Conference on Granular Computing*, vol. 2, July 2005, pp. 718–721 Vol. 2.

[13] A. Gron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 1st ed. O'Reilly Media, Inc., 2017.

[14] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their Applications*, vol. 13, no. 4, pp. 18–28, July 1998.

[15] M. S. Alam and S. T. Vuong, "Random forest classification for detecting android malware," in *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, Aug 2013, pp. 663–669.

[16] T. Jan, "Ada-boosted locally enhanced probabilistic neural network for iot intrusion detection," in *Complex, Intelligent, and Software Intensive Systems*, L. Barolli, N. Javaid, M. Ikeda, and M. Takizawa, Eds. Cham: Springer International Publishing, 2019, pp. 583–589.

[17] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, Apr 2006.

[18] R. Maclin and D. W. Opitz, "Popular ensemble methods: An empirical study," *CoRR*, vol. abs/1106.0257, 2011. [Online]. Available: http://arxiv.org/abs/1106.0257

[19] *IoT traffic analysis*, UNSW Sydney, January 2019, https://iotanalytics.unsw.edu.au/.

[20] F. Loi, A. Sivanathan, H. H. Gharakheili, A. Radford, and V. Sivaraman, "Systematically evaluating security and privacy for consumer iot devices," in *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*, ser. IoTS&#38;P '17. New York, NY, USA: ACM, 2017, pp. 1–6.

[21] A. Sivanathan, D. Sherratt, H. H. Gharakheili, V. Sivaraman, and A. Vishwanath, "Low-cost flow-based security solutions for smart-home iot devices," in *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Nov 2016, pp. 1–6.

[22] *OpenWrt Project*, January 2019, https://openwrt.org/.

[23] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag, "Collaborative hyperparameter tuning," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ser. ICML'13. JMLR.org, 2013, pp. II–199–II–207. [Online]. Available: http://dl.acm.org/citation.cfm?id=3042817.3042916

[24] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. A. Beyah, "Who's in control of your control system? device fingerprinting for cyber-physical systems," in *NDSS*, 2016.

[25] R. Lippmann, D. Fried, K. Piwowarski, and W. W. Streilein, "Passive operating system identification from tcp / ip packet headers *," 2003.

[26] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, April 2005.

[27] C. Arackaparambil, S. Bratus, A. Shubina, and D. Kotz, "On the reliability of wireless fingerprinting using clock skews," in *Proceedings of the Third ACM Conference on Wireless Network Security*, ser. WiSec '10. New York, NY, USA: ACM, 2010, pp. 169–174. [Online]. Available: http://doi.acm.org/10.1145/1741866.1741894

[28] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. Van Randwyk, and D. Sicker, "Passive data link layer 802.11 wireless device driver fingerprinting," in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, ser. USENIX-SS'06. Berkeley, CA, USA: USENIX Association, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267336.1267348

[29] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 94–104, Firstquarter 2016.

[30] R. R. Maiti, S. Siby, R. Sridharan, and N. O. Tippenhauer, "Link-layer device type classification on encrypted wireless traffic with cots radios," in *Computer Security – ESORICS 2017*, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds. Cham: Springer International Publishing, 2017, pp. 247–264.

[31] S. Siby, R. R. Maiti, and N. O. Tippenhauer, "Iotscanner: Detecting and classifying privacy threats in iot neighborhoods," *CoRR*, vol. abs/1701.05007, 2017. [Online]. Available: http://arxiv.org/abs/1701.05007