Dilawer Ahmed*, Anupam Das, and Fareed Zaffar

Analyzing the Feasibility and Generalizability of Fingerprinting Internet of Things Devices

Abstract: In recent years, we have seen rapid growth in the use and adoption of Internet of Things (IoT) devices. However, some IoT devices are sensitive in nature, and simply knowing what devices a user owns can have security and privacy implications. Researchers have, therefore, looked at fingerprinting IoT devices and their activities from encrypted network traffic. In this paper, we analyze the feasibility of fingerprinting IoT devices and evaluate the robustness of such fingerprinting approach across multiple independent datasets — collected under different settings. We show that not only is it possible to effectively fingerprint 188 IoT devices (with over 97% accuracy), but also to do so even with multiple instances of the same make-and-model device. We also analyze the extent to which temporal, spatial and data-collection-methodology differences impact fingerprinting accuracy. Our analysis sheds light on features that are more robust against varying conditions. Lastly, we comprehensively analyze the performance of our approach under an open-world setting and propose ways in which an adversary can enhance their odds of inferring additional information about unseen devices (e.g., similar devices manufactured by the same company).

Keywords: Device fingerprint, generalizibility, Internet of Things

DOI Editor to enter DOI
Received ..; revised ..; accepted ...

1 Introduction

We live in an increasingly connected world, where we spend a large part of our time interacting with a wide range of IoT devices. Examples of such IoT devices include smart voice assistants, smart surveillance cameras, smart TVs, smart thermostats, smart locks and smart scales. While all of these IoT devices provide convenience through the automation of appliances, such convenience often comes with unforeseen security and privacy risks. For example, simply knowing what devices a consumer owns by itself can have serious security and privacy implications. Knowing that there is a heart rate monitor inside the house tells a lot about the inhabitant's health condition. Similarly, knowing the exact make and model of a security lock can potentially help an adversary launch targeted attacks, e.g., exploit unpatched known vulnerabilities.

Researchers have previously exploited network traffic to infer application and hardware-level metadata that are often sensitive. For example, researchers have shown how an adversary can use network traffic to infer the websites a user visits [10–12, 18, 22], content they watch [43], applications they run [13, 15, 46, 53, 55], devices they own [16, 36, 48], and even activities they perform on their devices [14, 44, 54]. In recent years, researchers have focused on identifying IoT devices [9, 48] and their device-level activities [36, 44, 54] from network traffic. However, most of these works focus on building and evaluating models that work well on a relatively small dataset (typically less than 50 devices) and lack any analysis of how such models generalize to other datasets, often collected under different settings. Moreover, they lack any comprehensive open-world analysis — something that an adversary is bound to face in any real-world setting.

In this paper, we aim to analyze the feasibility of fingerprinting IoT devices and test the robustness of such fingerprinting approach across independent datasets collected under different settings. In particular, we seek to answer the following questions: **RQ1:** ***How well can we fingerprint IoT devices at different granularities?*** Existing works conduct a study on a handful of devices; we perform analysis on 188 IoT devices — the largest analysis to the best of our knowledge. We also discuss effectiveness across different granularities, such as the manufacturer or device functionality regardless of vendor, while covering multiple devices of the same make-and-model. **RQ2:** ***How well do the fingerprints generalize across temporally and***

*: **Dilawer Ahmed:** North Carolina State University, E-mail: dahmed2@ncsu.edu
Anupam Das: North Carolina State University, E-mail: anupam.das@ncsu.edu
Fareed Zaffar: Lahore University of Management Sciences (LUMS), E-mail: fareed.zaffar@lums.edu.pk

spatially distant datasets? Does the data collection methodology itself impact fingerprinting accuracy? We evaluate to what extent classifiers are generalizable across various datasets collected from different locations at different timestamps. In other words, how well a classifier performs when trained and tested on different datasets. **RQ3:** ***How well do the classifiers perform in an open-world setting?*** We also analyze if our approach can effectively operate under an open-world setting when many of the devices remain unseen to the classifiers.

To answer these questions, we first collect seven datasets containing network traffic from a total of 188 IoT devices (of which 120 are unique make-and-model devices). Two of the datasets are public, while four datasets were collected by contacting the authors of existing literature. We also collect our own data. Next, we build our own device fingerprinting technique using well-known features, where we achieve similar, if not better, accuracy even for 3–7 times more devices than existing works. We then evaluate the robustness of our classifier under different settings, like analyzing the extent to which temporal, spatial and data-collection-methodology differences impact fingerprinting accuracy. We also perform open-world evaluations of our classifier. In summary, we make the following contributions:

- To the best of our knowledge, we perform fingerprinting analysis on the largest number of IoT devices (188 devices) to date. Our dataset covers not only different device types but also multiple instances of the same make-and-model device. Moreover, our analysis showcases fingerprinting accuracy at different granularities (§4).
- We evaluate various factors that can potentially impact accuracy, such as training and testing on different datasets to gauge the generalizability of our approach (i.e., determine features that are more generalizable across independent datasets). We analyze the impact of time, location and data-collection-methodology on fingerprinting accuracy (§5).
- We also perform a comprehensive analysis of fingerprinting IoT devices under open-world settings. We identify setups that can potentially boost an adversary's capability in inferring additional information about unseen devices. We conduct end-to-end evaluation under this setting (§6).

The remainder of this paper proceeds as follows. Section 2 describes related work. Section 3 characterizes the different datasets we use for our evaluations. Section 4 investigates scalability of our fingerprinting approach (**RQ1**). Section 5 investigates generalizability of our approach across different datasets (**RQ2**). Section 6 looks at open-world evaluations (**RQ3**). Section 7 summarizes our findings and lists the limitations of our approach. We conclude in Section 8. Our source code and datasets are open-sourced [1].

2 Background and Related Work

Device fingerprinting is the process of collecting application and/or hardware-level information from a remote computing device for the purpose of identification. Device fingerprinting is typically used for anomaly detection and digital rights management. However, device fingerprinting has also been used to track users online.

Inference through Network Traffic. There is a large body of work that exploits network traffic to fingerprint different websites that a given user visits [4, 10–12, 18, 21, 22, 28, 28, 29, 37, 38, 52, 56, 60]. Others have tried to infer contents from encrypted VOIP [57, 58], and video-streaming traffic [43, 47]. Researchers have also shown that it is possible to reduce the search space for guessing passwords in SSHv1 by exploiting the timing delays between subsequent IP packets [51].

Network Monitoring. For Internet-connected hosts, researchers have exploited TCP timestamps to estimate the clock skew of a device and consequently used the unique clock skews to detect devices [26, 33] remotely. Researchers have also analyzed wireless traffic to uniquely fingerprint wireless devices [17, 19, 27, 34, 39]. Furthermore, there is a rich literature on utilizing network traffic to detect anomalous activities [23, 25, 35, 50] and malware [6, 41].

Identifying IoT Devices. With the rapid adoption of IoT devices and their sensitive nature, researchers have recently explored the possibility of fingerprinting not only IoT devices but also device-level activities. Most IoT devices connect to the Internet either through a WiFi access point or through a hub/bridge (e.g., devices that use Zigbee/Zwave protocols), thus making it easy for an ISP to observe and potentially infer sensitive information from network traffic. Ren et al. [44] perform a comprehensive study on event-level traffic generated by various IoT devices from both the US and UK. They

1 https://github.com/dilawer11/iot-device-fingerprinting

identify the various Internet destinations for the traffic and determine how frequently communications are protected by encryption. Lastly, for traffic originating in the US and UK, they contrast regional differences between these results. Researchers have used DNS queries to infer IoT devices [8, 42]. They also model traffic characteristics to infer device-level activities. Furthermore, they propose traffic padding techniques to mitigate volume and timing-based inference attacks. Sivanathan et al. [48, 49] also leverage network traffic and build a multi-layer model to fingerprint IoT devices uniquely. OConnor et al. [36] and Trimananda et al. [54] both use packet direction and size to infer device-level activity. Others have utilized data from different layers of the network stack to identify IoT devices [31, 32].

Some have focused on specific types of devices and protocols. For example, Copos et al. [14] investigate network traffic to infer device-level activities from only Nest Thermostat and Nest Protect. Others focus on Zigbee/Z-Wave devices and leverage specialized Zigbee/Z-Wave sniffers [3, 62]. Alrawi et al. perform a systematic security analysis of IoT devices, where they assign scores to devices based on various security attributes, e.g., if proper encryption is used in different communications [5].

Countermeasures against Network Traffic Analysis. Wright et al. [59] suggest the use of traffic morphing to mitigate the risk of packet length-based inference techniques. Apthorpe et al. [7, 9] also evaluate VPN-based traffic shaping techniques to thwart traffic rate and volume-based attacks. Trimananda et al. [54] suggest a packet padding technique to prevent packet size-based inference attacks. However, Dyer et al. [18] show that traffic padding and morphing is still not effective in hiding critical information.

Distinction from Prior Work. Our work is inspired by the aforementioned work on fingerprinting IoT devices. However, our primary and differentiating goal is to evaluate the feasibility and robustness of fingerprinting a larger number of IoT devices (the largest evaluation to the best of our knowledge) across multiple datasets — collected across different settings to shed light on how well models generalize across different settings. Furthermore, existing literature has mainly performed closed-world evaluations and has suggested high success rates. We perform open-world evaluations to showcase the limitations under real-world settings. However, we also show that an attacker might be able to infer a coarser level of information for previously unseen devices. Table 20 in Appendix I shows an in-depth comparison with existing works in terms of features used, evaluation settings, detection granularity and datasets used.

3 Dataset and Methodology

In this section, we describe the setup of our analysis, along with the various datasets used and the data preprocessing steps involved in analyzing network traffic. We also discuss the features, machine-learning models and metrics used in our evaluations.

3.1 Threat Model

In this paper, we assume a passive network adversary who can observe and record all wide-area network traffic (i.e., encrypted traffic), including traffic coming in and going out from home gateway routers. The adversary, however, cannot view any local-area network traffic between devices behind the gateway router. The adversary also does not manipulate network traffic. Furthermore, the adversary does not rely on packet content but instead on traffic metadata extracted from TCP/IP headers and send/receive rates. Lastly, we assume the adversary can collect and analyze IoT traffic by deploying its own setup. This setup can temporally, geographically and otherwise be similar or different from the target. The adversary can also utilize existing publicly available datasets.

3.2 Datasets

We use seven different datasets in our experiments; two of them are publicly available, and four were made available upon request. The remaining one dataset is our own, where we collect data from IoT devices available in our lab. The collection setup is similar to existing approaches, where all of our IoT devices communicate through a WiFi router running OpenWRT [2] (we used a Linksys WRT1900ACS router with OpenWRT version 18.06.2). We captured and dumped traffic through the router in PCAP format on a connected host machine (using port mirroring). Following is a brief description of the different datasets used in our evaluations (Table 1 summarizes their traffic-level characteristics).

YourThings Dataset [5]. This dataset is one of the largest publicly available datasets. It contains continu-

Table 1. Characteristics of datasets used. In total we have 120 unique IoT device types and 188 IoT device instances (some device types are present multiple times in the same dataset or across different datasets). Some also contain non-IoT devices.

Dataset	Acronym	Country (State/city)	Capture Period	Capture Duration (∼days)	Unique IoT Devices	Total IoT Devices *	Total Non-IoT Devices	Unique Destinations	Total Download/ Upload	Fraction of IoT Traffic	Burst Size Download/ Upload	Burst Time Download/ Upload	Type◇
YourThings [5]	YT	US (GA)	Early 2018	11	45	45	3 †	5672	21.7/233.1 GB	0.98	0.7/1.6 KB	0.6/8.3 s	C
HomeSnitch [36]	HS	US (FL)	Early 2020	12	24	28	0	5600	3.4/47.2 GB	1.0	0.4/1.9 KB	1.4/7.2 s	C
PingPong [54]	PP	US (CA)	Late 2019	51	17	18	0	1434	0.24/1.65 GB	1.0	0.2/0.8 KB	2.2/24.4 s	E
Mon(IoT)r_US [44]	NE_US	US (MA)	Early 2019	14	41	41	0	1506	0.38/1.93 GB	1.0	0.6/1.9 KB	1.4/1.2 s	E
Mon(IoT)r_UK [44]	NE_UK	UK (London)	Early 2019	17	29	29	0	1514	0.33/2.18 GB	1.0	0.5/2.1 KB	1.8/1.2 s	E
UNSW [48]	SW	AU (Sydney)	Late 2016	21	19	19	7 ‡	5357	9.5/1.7 GB	0.18	0.5/0.4 KB	4.4/9.1 s	C
Our	Our	US (NC)	Early 2020	11	8	8	3 ♣	3743	7.4/2.6 GB	0.89	0.7/1.6 KB	3.1/28.9 s	C
Combined dataset					120	188	13						

*: The numbers may vary slightly from original source as we only consider devices with enough traffic to generate at least 10 samples. ◇ C: continuous and E: event-based
†: iPad, iPhone, Android tablet; ‡: Android tablet, Printer, MacBook, iPhone, Android phone, Laptop; ♣: Laptop, Desktop, Android Phone

ous traffic generated over a period of 11 days. While the exact device-level interactions are not explicitly listed, it covers traffic generated by device autonomously and through human interaction in a continuous manner.

UNSW Dataset [48]. This is another publicly available dataset from UNSW Sydney, Australia. The data contains continuous traffic traces in PCAP format using `tcpdump`. Data replicates a real living smart environment that covers data resulting from human interaction, e.g., smart bulb turning on by detecting motion and data that is not affected by human interaction, e.g., periodic status updates, DNS, or NTP.

Mon(IoT)r Dataset [44]. This dataset consists of data collected from two labs, one located in the US and another in the UK. We treat this dataset as two separate datasets and refer to the dataset from their US Lab as *NE_US* and the dataset from their UK Lab as *NE_UK*. The data is primarily traffic generated from specific device-level *events* such as asking Alexa to turn on the light. Each event or activity is separated into a separately labeled PCAP file.

PingPong Dataset [54]. This dataset consists of only *event-based* traffic similar to Mon(IoT)r dataset. Their dataset also includes internal traffic generated by the devices, but since we only focus on traffic from the point of view of an ISP, we ignore such traffic and only focus on Phone-to-Cloud and Device-to-Cloud traffic.

HomeSnitch Dataset [36]. This dataset continuously collects data from IoT devices where some traffic result from direct human interactions while some occur because of periodic updates.

Our Dataset. Our dataset was collected from our lab setting, which emulates a smart living room. Our dataset consists of traffic generated through both human interactions like turning on/off lights through voice assistants and idle traffic (i.e., no user interaction). For our dataset, we triggered normal operations (e.g., turning ON/OFF light/plug, TV or activating motion sensor and live camera feed) 3–4 times daily.

Table 1 summarizes the different traffic-level characteristics of all the datasets used in our evaluations. Across the datasets, we observed increased activity/traffic from 11 am to 3 pm (typical for lab settings). Specific activity depends on the mode of interaction, e.g., remote operation or in-person activation. More fine-grained details on exact activities performed are not provided by the data publishers. We can also see that some datasets contain more unique destinations (i.e., flows) and download/upload more data than others. For three datasets (i.e., PP, NE_US and NE_UK), data is collected in shorter intervals representing device-level operations like activating lights or voice assistants. We term these datasets as *event-based* datasets. The other four datasets collect data in a more continuous manner and are termed as *continuous*.

3.3 Data Pre-processing

We only extract header information from PCAP files. Such information includes IP addresses, ports, protocol number, timestamp and packet size. Next, we filter out any traffic that does not use any IP-based protocol, e.g., ARP. We also remove any internal traffic (traffic only visible to devices connected to the same local network) and consider only traffic visible to an external passive adversary (e.g., an ISP). In order to train machine learning models, we label devices using the device information provided by the different datasets. Some datasets also contain traditional non-IoT traffic generated by computers, printers, laptops, tablets and smartphones (see Table 1 for more details).

Table 2. List of different types of features used. Some features are single-valued summary statistics, whereas others are derived from the top most frequent elements in a given distribution (termed as multi-valued feature). The last three columns represent the accuracy, precision and recall of using each individual feature group derived from the combined dataset. 95% CI is provided in parentheses. All these features are extracted based on the 5-minute window.

Feature Group	Description	Single/Multi valued	Accuracy	Precision	Recall
Unique Packet Length	Distribution of unique packet sizes	Single-valued	95.63 (0.01)	95.05 (0.2)	94.15 (0.15)
Packet Delay	Distribution of inter packet delays	Single-valued	93.04 (0.02)	91.18 (0.17)	87.63 (0.15)
Protocols	Percentage of packets using different protocols	Single-valued	87.97 (0.01)	88.95 (0.24)	84.26 (0.16)
Burst Bytes	Total bytes transferred in a burst of packets	Single-valued	87.4 (0.01)	92.79 (0.24)	85.63 (0.11)
Packet Size List	List of most frequent packet sizes	Multi-valued	82.37 (0.02)	91.21 (0.12)	82.64 (0.08)
Total Packets	Total number of packets transferred	Single-valued	78.97 (0.02)	66.75 (0.12)	65.35 (0.1)
Burst Delay	Distribution of inter burst delay	Single-valued	76.44 (0.01)	86.04 (0.07)	73.03 (0.08)
Burst Time	Distribution of burst duration	Single-valued	72.98 (0.05)	87.39 (0.16)	71.58 (0.13)
IP List	List of unique IPs (first three octets)	Multi-valued	66.21 (0.03)	71.25 (0.32)	57.54 (0.1)
Burst Packets	Distribution of number of packets in a burst	Single-valued	65.86 (0.01)	75.84 (0.15)	62.3 (0.1)
Request-Reply Pair List	List of Request-Reply packet sizes	Multi-valued	64.12 (0.14)	72.35 (0.22)	58.08 (0.12)
TCP Flags	Percentages of TCP Flags in TCP Packets	Single-valued	62.22 (0.02)	72.11 (0.16)	56.28 (0.05)
Protocol List	List of protocols used in a window	Multi-valued	59.28 (0.01)	61.46 (0.25)	46.39 (0.07)
Traffic In and Out Ratio	Ratio of incoming vs. outgoing traffic	Single-valued	54.75 (0.03)	50.62 (0.14)	43.5 (0.14)
External Port List	List of external ports contacted	Multi-valued	49.8 (0.01)	51.19 (0.29)	42.81 (0.08)
HTTP(S) Traffic	Traffic to and from port 80 and 443	Single-valued	46.46 (0.02)	54.67 (0.25)	39.22 (0.09)
Hostname List	List of host names (TLD+1) contacted	Multi-valued	40.6 (0.01)	49.21 (0.36)	30.99 (0.04)
Unique IPs	Number of unique IPs contacted	Single-valued	39.97 (0.02)	35.93 (0.16)	33.42 (0.1)
Unique Ports	Number of unique ports contacted	Single-valued	32.27 (0.02)	30.38 (0.14)	26.12 (0.1)
Total Bytes	Total bytes transferred in a window	Single-valued	29.85 (0.05)	52.28 (0.41)	29.18 (0.12)

3.4 Feature Extraction

Using the pre-processed data, we extract features using a 5-minute window for each device-specific traffic. We determine device-specific flows using the egress IP address/port number, destination IP address/port number and the protocol in use. We tested three traffic window sizes: 2, 5, and 10-minute windows. The corresponding classification accuracies on the combined dataset were 97.79%, 97.81% and 98.19%, respectively. Given that most of the device-level activities were captured in short bursts (≤ 5 minutes), we chose a 5-minute window for our evaluations. Table 2 shows the various types of features extracted from a 5-minute window (Table 21 in Appendix J lists all the features used). All evaluation metrics are thus computed based on features extracted from a 5-minute window.

The types of features we extract can broadly be categorized into single-valued and multi-valued features. Single-valued features contain various statistical summaries (e.g., total, min, max, median, 25-th percentile, 75-th percentile, 90-th percentile, and standard deviation) computed over a 5-minute window, e.g., the total number of outgoing packets or the average delay between TCP outgoing packets. Multi-valued features are key-value pair objects that count different observed values or events, e.g., the number of times each domain is contacted or the number of times different ports are used. This is similar to the bag-of-words representation used by Sivanathan et al. [48]. For training and testing machine learning models, we only use at most the top N frequently observed values (we empirically set $N = 500$ and $N = 1000$ for protocol-specific and protocol-independent multi-valued features, respectively). Multi-valued features are transformed to numeric one-hot encoding for training and testing.

To understand how effective the features are in distinguishing devices, we generate a scatter plot considering samples from some of the same make-and-model devices as well as different make-and-model devices (using the combined dataset). We use t-Distributed Stochastic Neighbor Embedding (t-SNE) technique [30] to reduce the dimensionality of our feature vector. Figure 6 in Appendix B highlights our findings. We can see that, in general, samples from a given device are clustered together, irrespective of whether they are of the same make and model or not. However, the samples overlap significantly for some devices manufactured by the same vendor, like Google Home and Google Home Mini (this may result from using a similar software stack).

3.5 ML-Model and Metrics Used

In terms of machine learning models, we found Random Forests [40] to be most effective (we also tested SVM

and decision trees). We set the number of trees in the forest to 100 (i.e., *n_estimators*=100). We did not use deep-learning techniques for better explainability, like understanding why certain features rank at the top. We also use *forward feature selection* (FFS) [20, 61] mechanism to identify more generalizable features. To evaluate the effectiveness of our model, we use well-known metrics, like accuracy, precision and recall. We perform 5-fold cross-validation to account for the randomness inherent in the training data and repeat the whole process 10 times to account for any randomness inherent to the Random Forest model. We report the mean and 95% confidence interval (CI) over 10 runs unless mentioned otherwise.

4 Fingerprinting Devices

In this section, we study the effectiveness of fingerprinting IoT devices at *different granularity* using multiple datasets. First, we evaluate the effectiveness of our approach on each dataset separately. Next, we analyze the scalability of our approach against not only a larger number of devices compared to existing works but also devices that include multiple instances of the same make and model.

4.1 Effectiveness Across Datasets

As previously mentioned, we conducted our experiments using seven different datasets. We, therefore, first evaluate how well our proposed approach can fingerprint IoT devices on individual datasets (i.e., per-dataset basis). Table 3 summarizes our findings. We can see that, on average, we have close to 99% accuracy across all datasets (average precision and recall is close to 98%). Compared to existing literature, we observed similar performance. For example, Sivanathan et al. [48] report 99% accuracy in fingerprinting devices in their dataset. We obtain similar results as shown in Table 3 (these numbers can be considered as a baseline comparison for the remaining analyses to follow). Table 3 results suggest that IoT devices can be fingerprinted with high accuracy. However, given the relatively small number of devices available in each dataset, it still remains unclear whether such an approach can scale well with higher numbers of devices. We explore this question in the next section.

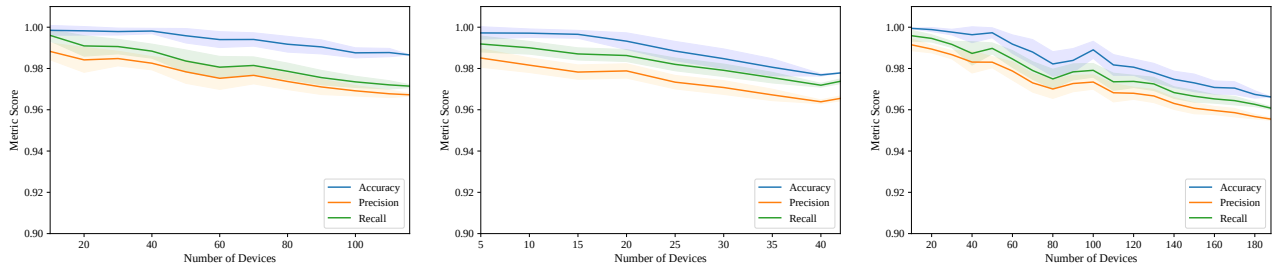
Table 3. Fingerprinting accuracy across individual datasets (95% CI is provided in parenthesis).

Dataset	Accuracy	Precision	Recall	Devices
UNSW	99.9 (0.0)	99.94 (0.0)	99.0 (0.16)	19
HomeSnitch	99.85 (0.0)	99.8 (0.01)	99.8 (0.01)	28
YourThings	97.91 (0.03)	97.89 (0.03)	97.82 (0.03)	45
NE_UK	97.39 (0.12)	96.94 (0.15)	96.52 (0.25)	29
NE_US	98.16 (0.14)	98.25 (0.21)	96.68 (0.35)	41
PingPong	99.58 (0.02)	93.83 (0.02)	93.92 (0.03)	18
Our	99.79 (0.01)	99.83 (0.0)	99.62 (0.01)	8
Average	98.94	98.07	97.62	

4.2 Scalability

In this section, we aim to measure how the classifier's performance varies as the number of devices increases. For this purpose, we merge all the datasets into one consolidated dataset (i.e., combined dataset). We consider three different settings. In the first case, we only consider one instance of a device, i.e., if two datasets have the same make-and-model device, we only choose one, picked randomly on each run of the experiment. This results in a total of 120 devices (also depicted in Table 1). We start with 10 devices chosen randomly and incrementally increase the number of devices in steps of 10 and perform 5-fold cross-validation over 10 runs per increment. Figure 1a shows how the accuracy, precision and recall varies as we increase the number of devices. We can see that there is a slight decay in the average metrics. However, even with 120 devices, we can achieve an average accuracy close to 99%.

In the second case, we want to distinguish between only the same make-and-model devices, i.e., the same devices that potentially have different firmware or operate under different conditions. For this purpose, we first determined the number of instances per make and model of an IoT device. Figure 5 (in Appendix A) shows the distribution of the number of devices with one or more instances. Most of the devices (80) have only one instance, and the remaining 40 devices have more than one instance (Table 19 in Appendix H list all unique device types). In this setting, we only consider the devices with more than one instance (where each instance gets its own unique label). We then perform a similar incremental step with increments of 5 devices and perform 5-fold cross-validation with 10 runs per increment. Figure 1b shows the trends observed. Again we see that the average accuracy decreases slightly, but even with devices with multiple instances, we can achieve around 98% accuracy. To better understand why the same make-and-model devices are easily distinguishable, we look at the



(a) Unique instances **(b)** Multiple instances **(c)** All instances

Fig. 1. Change in evaluation metrics as we increase the number of devices under different settings. In general, we see that the metrics decrease ever slightly as we increase the number of devices. However, in all cases the average accuracy is over 97%.

classifier's top features (i.e., based on Gini importance). For example, for the three instances of 'Geeni Cameras' inside the HS dataset, we found that in/out ratios, burst behavior and IP list (see Table 2 for detail) are the top features. This implies that devices have slightly different fingerprints based on activities, e.g., a camera in the garage might get activated more often than a camera in the kitchen and hence have different burst and in/out ratios. Figure 6 in Appendix B also shows how the three devices form individual clusters.

Finally, for the last setting, we consider all the device types and instances altogether, which sums up to a total of 188 devices (where each device, irrespective of make, model and dataset, gets its own unique label). We again perform an incremental step of 10 devices, and perform 5-fold cross-validation over 10 runs. Figure 1c shows the accuracy, precision and recall under this setting. Again, we see that the average accuracy reduces as we increase the number of devices, but even with 188 devices, we can achieve an average accuracy of around 97%. These results suggest that we can not only fingerprint many IoT devices but also do so even with multiple instances of the same make-and-model device.

4.3 Fingerprinting Granularity

In the next series of experiments, we explore the feasibility of fingerprinting IoT devices at different granularities. For this purpose, we consider *five* different levels of granularity: 1) individual device level (individual devices, irrespective of make and model, are assigned unique identifiers); 2) unique make and model (devices of the same make and model are grouped together); 3) company-specific category (devices of similar functionality, but produced by the same company are grouped together (e.g., Amazon's Echo Dot and Echo Plus grouped together); 4) same company (devices manufactured by the same vendor are grouped together irrespective of their functionality (e.g., Amazon Echo Dot and Amazon Fire TV are both manufactured by Amazon); and 5) device category (devices of similar functionality, irrespective of the manufacturer are grouped together, e.g., Google Home and Amazon Echo are both voice assistants). We then compute accuracy across all these different levels of granularity. The main goal of testing across different granularities is to see if we can get some insight into how device functionality and device manufacturer play a role in IoT device fingerprinting. As a motivating example, let us consider the case of Amazon Echo Dot, Amazon Echo Spot and Amazon Echo Plus. These devices are from the same manufacturer and provide similar functionality (i.e., smart voice assistants). Due to code reuse and similar activity patterns, these devices may generate very similar network traces and hence be mistaken for one another. We want to test if grouping such devices together can improve the performance of the classifier. Table 4 shows the performance metrics across the five different levels of *granularity*. The results show that as we move towards coarser granularity (i.e., grouping devices of similar functionality together), the accuracy does increase slightly but not with the same proportion with which the number of labels decreases.

Table 4. Effectiveness of fingerprinting IoT devices at different granularity. Higher accuracy is achieved for the general device category (95% CI is provided in parenthesis).

Granularity	Accuracy	Precision	Recall	# of labels
Individual Device	96.62 (0.01)	96.08 (0.07)	95.55 (0.06)	188
Unique make and model	97.85 (0.01)	97.02 (0.14)	96.05 (0.09)	121*
Company-specific category	98.91 (0.0)	98.05 (0.18)	96.8 (0.12)	99*
Same company	99.8 (0.01)	99.09 (0.15)	98.05 (0.12)	66*
General-device category	98.89 (0.01)	98.87 (0.01)	98.28 (0.02)	16*

* Including one label for all non-IoT devices.

To obtain a better understanding of the sources of inaccuracies, we plot the confusion matrix in Fig-

ure 12 in Appendix F for 'general-device category'. We can see that we obtain near-perfect scores for all the device categories except for the device category 'motion sensors'. Further analysis revealed that the classifier wrongly predicted 'Belkin motion sensors' traces as 'Belkin switches'. This could be an artifact of products manufactured by the same vendor contacting the same backend infrastructure [45]. To explore this further, we performed another experiment in which we grouped all devices from the same company. Table 4 shows that we achieve almost perfect accuracy in such a case suggesting that devices from the same vendor possibly have similar fingerprints in terms of traffic they generate.

Takeaway. The general takeaway from this section is that our classification approach achieves as good as, if not better, results compared to existing IoT device fingerprinting approaches. Our classifier also scales well when the number of devices increases to well above what an average smart home or even a small to medium-sized office environment might have. Furthermore, we showcase that our approach can distinguish devices at various granularities, including devices manufactured by the same vendor and even similar functioning devices.

5 Fingerprinting Generalizability

Existing work on fingerprinting IoT devices lacks analysis of generalizability across different real-world factors. To determine to what extent classifiers are generalizable across different factors, we first explore whether top-ranked features are consistent across different datasets. We use forward feature selection (FFS) [20, 61] to rank the features for each dataset using accuracy as the performance metric. Figure 7 in Appendix C highlights the top 20 features across each individual dataset. As we can see from the figure, very few features are common across all the datasets. Features such as the maximum incoming packet length (*max_in_len*) and the maximum outgoing packet length (*max_out_len*) are common across all datasets, indicating that packet length may be one of the more useful features. Table 2 also shows similar results, where we report the accuracy of classifying 120 unique IoT devices (from the combined dataset) using only one feature group at a time. The lack of overlap among the top 20 features across different datasets suggests that a classifier tuned for one dataset will likely not be equally effective on other datasets. In this paper, we analyze the impact of the following factors on generalizability: 1) temporality, 2) locality and 3) data collection methodology. Our analysis sheds light on potentially more robust features across different varying conditions.

5.1 Temporal Impact

We, first, investigate if the fingerprints change temporally, as a software update, version upgrade, or evolving backend infrastructure might change the fingerprints. To evaluate this, we collected another round of data from our lab in January 2021, and we also obtained another round of data from the HomeSnitch team [36] in May 2021. We next compare our new datasets to the old datasets. We trained our model on the 2020 version of the datasets and tested it on the 2021 version of the datasets. We found the average accuracy, precision and recall to be 94.72%, 85.01%, 89.83%, respectively, for Our dataset and 87.76%, 85.31%, 85.53%, respectively in the case of the HomeSnitch dataset. This suggests that the classifier generalizes relatively well even with possible firmware/software upgrades to the devices. Figure 8 (in Appendix D) shows the confusion matrix in this setting for Our dataset. As we can see, inaccuracies arise from devices from similar companies (e.g., both Ring and Echo Look are Amazon products) or devices with similar functionality (e.g., LG TV has a voice assistant built-in). Figure 9 in Appendix D shows the confusion matrix for the HomeSnitch dataset. Upon further investigation in the inaccuracies, we found that these were due to drastic changes in traffic patterns compared to the previous year and/or similar traffic to another device in the latest version of the dataset. For example, Ultraloq Lock Bridge and Lockly Lock Hub had different patterns in 2020 traffic capture, whereas in 2021, Ultraloq Lock Bridge had almost identical traffic patterns for Lockly Lock Hub; e.g., in/out ratios, packet lengths, burst times were similar. Similarly, Nightowl Doorbell had a 40:60 in/out ratio in 2021, while in 2020, it was 14:86. It also contacted 768 unique ports in 2020, while in 2021, it only contacted 71, which was closer to Geeni Camera. Samsung SmartThings Camera also had a different pattern compared to last year, and in 2021 in/out ratio and bytes sent per burst were similar to Arlo Base Station. In 2020 it also had 590 unique packet lengths per window, while in 2021, it only had 101, which was closer to Arlo Base Station than its older version.

Next, we perform forward feature selection to determine features that generalize well temporally, and the results are summarized in Table 6. As we can see, the optimal set of features can still achieve an accuracy up to 99% for Our dataset and 94% for the HomeSnitch

Table 5. Effectiveness of classifier when training and testing on temporally distant datasets with similar setup.

Train	Test	Accuracy	Precision	Recall	Devices
Our (2020)	Our (2021)	94.72 (0.82)	85.01 (1.13)	89.83 (1.14)	8
HS (2020)	HS (2021)	82.61 (3.45)	83.96 (1.27)	82.34 (2.51)	15

Table 6. Top feature groups that achieve the best performance under temporally generalized settings. Features bolded comprise the optimal feature set.

Rank	Feature	Cum. Acc.	Rank	Feature	Cum. Acc.
1	**Hostname List**	**95.53**	11	Unique Ports	98.17
2	**Unique IPs**	**96.43**	12	Burst Time	97.82
3	**External Ports List**	**98.36**	13	Packet In and Out Ratio	97.62
4	**HTTP(S) Traffic**	**96.57**	14	Protocols	96.97
5	**Packet Sizes List**	**98.55**	15	Unique Packet Lengths	97.05
6	**Request-Reply Pkt Sizes**	**98.72**	16	Total Packets	95.76
7	**IP List**	**98.95**	17	Packet Delay	96.2
8	TCP Flags	98.87	18	Burst Delay	96.1
9	Protocol List	98.68	19	Burst Packets	95.3
10	Total Bytes	98.35	20	Burst Bytes	93.28

Table 7. Effectiveness of classifier when training and testing on temporally distant datasets with similar setup using only the optimal features.

Train	Test	Accuracy	Precision	Recall	Devices
Our (2020)	Our (2021)	98.94 (0.16)	93.95 (2.44)	95.7 (0.55)	8
HS (2020)	HS (2021)	94.24 (0.28)	89.36 (0.33)	92.88 (0.35)	15

dataset. Delay and burst-based features seem less reliable, whereas unique hosts/IPs and packet sizes remain stable.

5.2 Spatial Impact

Next, we analyze the impact of training and testing on datasets collected from different geographic regions. This analysis is important as it will inform the adversary to what extent the training data needs to emulate a target's location and highlight more generalizable features across different geolocations. This becomes relevant if the adversary is towards the weaker end of the spectrum and is unable to collect data in the same geographical location as the target. For this analysis, we need to select datasets collected around the same time using a similar setup across two different geographic regions. The NE_US and NE_UK datasets are ideal for this evaluation. Ren et al. [44] state that their devices were procured from the corresponding countries (US, UK) where they performed data collection. 46 devices were purchased from US stores (US devices) and deployed in the US testbed; 35 were purchased from UK stores (UK devices) and deployed in the UK testbed. There were 26 common devices (note that we removed 5 devices because of a low number of data samples) across the two labs. We use the common devices to analyze the impact of fingerprinting devices across different geolocations. Some devices may have variants based on region, but the authors do not explicitly confirm or deny this.

Table 8 shows the accuracy when trained and tested on geographically distant datasets. We can see that the average accuracy drops to around 80%. Interestingly, we see better results when trained on NE_UK and tested on NE_US. Ren et al. [44] have shown in their study that a majority of their IoT device traffic terminated in the US for both the US and UK labs due to reliance on infrastructure with limited geodiversity. Thus, while devices from the UK contacted the major US-based cloud infrastructures, they also contacted some EU infrastructures. As a result, NE_UK is in some sense a superset of NE_US. However, the diversity in the infrastructure can still cause errors. Figure 10 and 11 (in Appendix E) show the confusion matrix when the classifier is trained on one location and tested on the other. The majority of the errors resulted from imbalanced data across the two datasets, e.g., Sousvide device was extensively being mislabelled when trained on NE_US and tested on NE_UK but not so much when trained on NE_UK and tested on NE_US. Sousvide device in the UK contacted AWS 27660 times throughout the data collection period, whereas it contacted the same service only 92 times in the US.

To understand which features are more stable across geographically distant datasets, we also rank the features using FFS. For each round of FFS we rank the feature based on average accuracy across different train/test combinations. For example, suppose training on NE_US and testing on NE_UK gives 90% accuracy for a feature and training on NE_UK and testing on NE_US gives 92% accuracy. In that case, we consider the average accuracy to be around 91%. This gives preference to features that generalize well for different settings instead of a single train/test set. We select the set of features that return the best accuracy as our optimal feature set. Table 9 shows a ranking of the features. We can see that most features except for traffic bursts are more suitable for this setting. To further validate our feature selection process, we reevaluate the performance metrics only using the top features (i.e., ones that are bolded in Table 9). We found the accuracy, precision and recall to increase to 94.76%, 93.38% and 92.68%, respectively, when trained on NE_UK and tested on NE_US. Similarly, the accuracy, precision and recall improved to 77.71%, 81.85% and 77.65%, respectively, when trained on NE_US and tested on NE_UK. Thus, in both cases

Table 8. Effectiveness of classifier when training and testing on geographically distant datasets with similar setup.

Train	Test	Accuracy	Precision	Recall	Devices
NE_UK	NE_US	89.4 (1.14)	89.31 (1.09)	85.25 (1.83)	21
NE_US	NE_UK	69.6 (1.03)	73.46 (2.08)	69.58 (1.08)	21

Table 9. Top feature groups that achieve the best performance under geographically generalized settings. Features bolded comprise the optimal feature set.

Rank	Feature	Cum. Acc.	Rank	Feature	Cum. Acc.
1	**Request-Reply Pkt Sizes**	**63.88**	11	Unique IPs	80.72
2	**Protocol List**	**70.59**	12	Total Packets	81.01
3	**Hostname List**	**73.43**	13	Packet In and Out Ratio	80.36
4	**Unique Packet Lengths**	**77.66**	14	Burst Packets	79.03
5	**HTTP(S) Traffic**	**78.89**	15	Protocols	78.0
6	**IP List**	**79.88**	16	Packet Delay	77.49
7	**External Ports List**	**80.35**	17	Burst Time	76.43
8	**TCP Flags**	**81.02**	18	Burst Delay	75.25
9	Total Bytes	80.75	19	Burst Bytes	75.2
10	Unique Ports	80.9	20	Packet Sizes List	74.81

where we only use optimal features, we see performance improvement compared to using all features.

5.3 Impact of Data Collection Approach

We now analyze if the data collection methodology itself impacts performance. For this, we train and test on datasets that have different collection characteristics (e.g., event-based compared to continuous and vice versa). We selected NE_US and YourThings as our candidates for the event-based and continuous dataset, respectively, as they provide the largest number of device overlap (also both located on the east coast of the US). While the datasets were collected at a different point in time, we have already showcased that temporal changes have minimal impact on our fingerprinting technique in Section 5.1. We next train and test on datasets that have different collection characteristics. Table 10 highlights the fingerprinting accuracy when trained and tested on different datasets. We see that, on average data collection method impacts accuracy significantly, leading to poor classifier performance. Next, we perform forward feature selection to determine features that generalize well, and the results are summarized in Table 11. As we can see, the top set of features can still achieve an accuracy close to 80%. Packet size, hostnames and ports seem more useful in this setting. To verify that these features really boost performance, we reevaluated the results of Table 10, but this time only using the features identified in Table 11 (i.e., bolded features). Table 12 highlights our findings. We can see that when training on YourThings and testing on NE_US we get slightly better results compared to the opposite setting. Similar results were observed when training on other continuous datasets and testing on the event-based datasets. This indicates that, in general, training on continuous datasets is advantageous for an adversary as it contains not only event-specific traffic information but also periodic or idle traffic characteristics.

Table 10. Effectiveness of classifier when training and testing on datasets with different collection characteristics.

Train	Test	Accuracy	Precision	Recall	Devices
NE_US	YT	28.32 (0.87)	53.79 (2.56)	31.92 (0.79)	13
YT	NE_US	28.89 (2.0)	58.45 (2.99)	42.51 (2.04)	13

Table 11. Top feature groups that achieve the best performance for datasets with different collection characteristics. Features bolded comprise the optimal feature set.

Rank	Feature	Cum. Acc.	Rank	Feature	Cum. Acc.
1	**Packet Sizes List**	**71.05**	11	Total Bytes	76.71
2	**Unique Ports**	**77.09**	12	Burst Packets	74.12
3	**HTTP(S) Traffic**	**78.38**	13	TCP Flags	70.53
4	**Packet Delay**	**79.06**	14	Burst Time	71.17
5	**Packet In and Out Ratio**	**78.71**	15	Unique Packet Lengths	65.64
6	**Hostname List**	**78.49**	16	Burst Bytes	62.4
7	**External Ports List**	**79.49**	17	Burst Delay	59.65
8	Total Packets	79.37	18	Unique IPs	59.3
9	IP List	77.44	19	Protocol List	51.73
10	Request-Reply Pkt Sizes	76.96	20	Protocols	41.92

Table 12. Effectiveness of classifier when training and testing on datasets with different collection characteristics, but using only the robust features from Table 11.

Train	Test	Accuracy	Precision	Recall	Devices
NE_US	YT	54.87 (2.15)	53.68 (1.36)	55.06 (1.84)	13
YT	NE_US	50.29 (2.61)	60.25 (0.89)	57.38 (1.7)	13

Takeaway. To the best of our knowledge, we are the first to study how well IoT device fingerprinting generalizes across different datasets. Our analysis shows that temporality, locality and data collection method have varying effects on the generalizability of fingerprinting. We show that while temporality does not significantly impact performance, data collection method and locality do to some extent. However, when data collection methodology, locality and temporality match between datasets, we can see almost perfect accuracy. For example, both our dataset and HS dataset are continuous in nature and were collected in the US (east coast) during early 2020. The accuracy, precision and recall across the two datasets (i.e., even reversing the training and test sets) is around 98-99% (Table 15 in Appendix E shows more details). On the other hand, when two or more of these properties change, the model does not general-

ize well, but certain features are more robust to varying conditions (see Table 9 and 11).

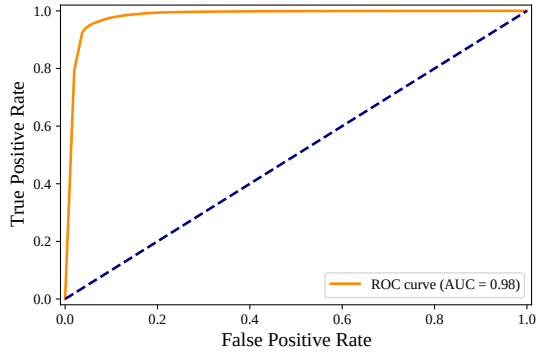
6 Open-World Evaluations

Previous works on IoT device fingerprinting have mostly evaluated a classifier where all test samples correspond to devices included in the training set (i.e., closed-world setting). In reality, this will not always be the case, as unseen and new devices will appear, resulting in unseen test samples. For example the SmartHomeDB currently lists 1251 different IoT devices [1]. To tackle this problem, we perform a series of experiments to first distinguish between IoT and non-IoT devices, followed by IoT devices that are known versus unknown to the adversary. The end goal is to infer additional information about IoT devices that an adversary may not have seen before. For this analysis, we use the combined dataset as this provides us with not only the most diverse number of devices but also non-IoT traffic to emulate real-world settings (note that not all datasets had non-IoT traffic as shown in Table 1).

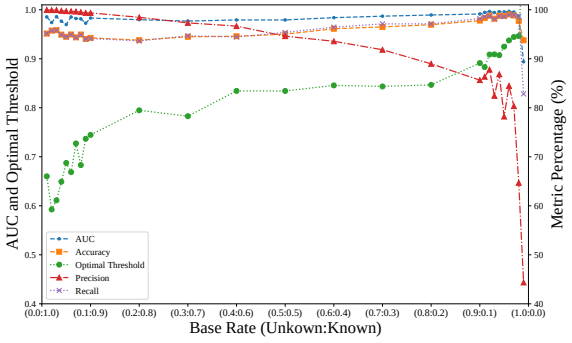
6.1 Distinguishing Known from Unknown

An open-world classifier must distinguish between samples that are not a part of the training set. For this purpose, we randomly split the devices into known and unknown sets. We use the samples from the known set of devices to train the classifier and then evaluate the classifier on a test set consisting of both known and unknown samples. We modify our classifier to use the probability of the predicted class as a threshold to decide whether a device is known or unknown (i.e., it is essentially a binary classifier). Next, we compute true positive (the number of known-device samples classified as known) and false positive (the number of unknown-device samples classified as known) rates for different thresholds of classification probability. We then determine the optimal threshold using a ROC curve, i.e., the threshold that minimizes false positives while maximizing true positives. For open-world evaluations, we only consider devices of unique make and model (i.e., 120 devices as shown in Table 1).

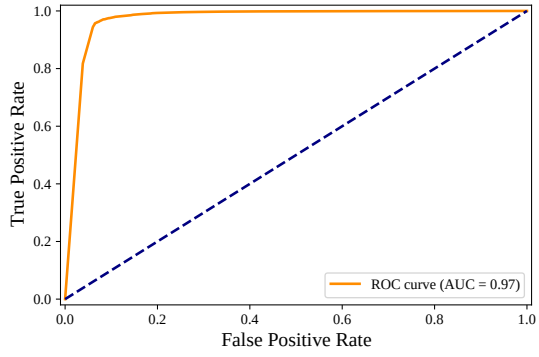
Basic Setting. To emulate the open-world setting, we randomly assign 50% of the devices to the known set and the remaining 50% to the unknown set. Next, we use 70% of the samples from the known set to train the classifier, whereas samples from the unknown set are never seen by the classifier. We then split the remaining 30% data into a test and validation set of equal size (15% each). Figure 2a highlights the ROC curve over 10 runs using the validation dataset. The *area under curve* (AUC) is 98%, which suggests the classifier can effectively distinguish between known and unknown devices. The optimal threshold for the binary classifier, in this case, is around 0.80.



(a) ROC curve for random 50% split



(b) ROC curve with varying base rate



(c) ROC curve under strategic setting

Fig. 2. ROC curve under open-world setting: a) with 50% random device split; b) varying base rate; c) strategic setting

Next, we look at how changing the base rate (i.e., the fraction of known devices) impacts AUC and the optimal threshold. For this purpose, we vary the fraction of known and unknown devices (e.g., the fraction of known devices from 1% to 99%) and compute AUC and the optimal threshold under such settings. We also analyze how accuracy, precision and recall vary with the base rate. Figure 2b highlights our findings. We can see that AUC remains more or less stable for a varying number of unknown IoT devices. However, the optimal threshold (in terms of classification probability) increases as we increase the number of unknown devices, which intuitively makes sense as the classifier only focuses on a very small number of devices (i.e., less diversity). In terms of performance metrics, we see that recall slowly increases, while precision starts to descend as we increase the fraction of unknown devices. They both crossover at the 50% base rate, indicating the importance of balanced samples for the open-world setting. Both precision and recall suffer when the fraction of known devices is 1%. However, an attacker can potentially focus on popular devices, thereby increasing their chance of boosting the known device portion.

Strategic Setting. We consider the case where instead of randomly splitting the devices, the adversary adopts a more strategic split with the goal to *maximize* their chance of inferring more meaningful information about unknown samples. In this setting, the adversary attempts to cover at least one company-specific device type, e.g., including one of Amazon's voice assistants, among many (like Echo Dot, Echo Plus, Echo Look), in the known set. In this way, the adversary can maximize their odds of covering a *similar functional* device in the training set. Again devices of similar type are *randomly* split between the known and unknown set. For an odd number of devices belonging to a given category, we assign the extra device to the known and unknown set alternatively, thus favoring the adversary to cover more devices in the case of an uneven number of categories containing an odd number of devices.

Now, such a setting is likely going to adversely impact the classifier's performance as for each device type in the known set, there will be a *similar* device present in the unknown set (e.g., Google Home is in the known set and Google Home Mini is in the unknown set). As a result, unknown samples can be classified as known. This would also result in a higher optimal threshold for the binary classifier. We perform a similar open-world evaluation to test how the strategic setting impacts the various performance metrics. Around 50% of the devices are known and unknown, even under this setting.[2] Figure 2c highlights the ROC curve under this setting. We can see that the average AUC is 97%. The optimal threshold for the binary classifier changes from 0.80 to 0.88 as we alter from the basic to the strategic setting. AUC changes from 0.98 to 0.97. In the following section, we describe how the strategic setting can help an adversary gain more insights about the unknown samples.

6.2 End-to-end Open-world classifier

In this section, we present an end-to-end system that first determines whether a device sample is an IoT device or not. Once the adversary knows the device is an IoT device, it can then predict if it is a known or unknown device. If known, they try to predict the actual device model name. If unknown, they try to *infer* meaningful information about the unknown test sample from the predicted labels. We consider two possible outcomes in this setting: 1) predict the company-specific device label for the unknown sample (e.g., Amazon voice assistant), and 2) predict the general-device category for the unknown sample (e.g., voice assistant). Thus, this process combines two binary classifiers (i.e., IoT vs. Non-IoT and known vs. unknown) followed by two multi-class classifiers, where one predicts the exact device label if the device is predicted to be known and the other predicts the company-specific or general-device category if the device is predicted to be unknown. The latter is trained using the company-specific/general-device category labeling from the training set. Figure 3 highlights the overall end-to-end process for inferring details on unknown samples under the open-world setting.

For these experiments, we only consider devices listed in Table 17 [3] and Table 18 [4] of Appendix H, i.e., devices that can be grouped with at least one other similar functioning device. Next, we strategically split devices into known and unknown sets, where the known set contains at least one device per company-specific category or general-device category (i.e., we consider these two possible settings). For non-IoT devices, we adopt a similar approach, where we randomly assign one device type per category to the known and unknown set (e.g., iPhone to known and Android phone to unknown set).

2 This was possible as we had an even number of device categories.

3 36 devices for which we found at least another similar functional device from the same company

4 118 devices categorized into 12 different functionalities

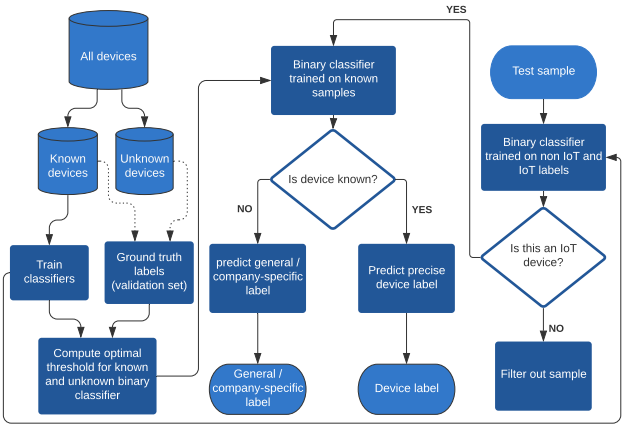


Fig. 3. Overall procedure followed in the end-to-end evaluation under the open-world setting.

Table 1 highlights the different non-IoT device types. We then repeat experiments 10 times and report the average metrics with a 95% confidence interval.

Table 13 summarizes the performance of the different classifiers under this strategic open-world setting. Table 13 showcases performance at both the company-specific and general-device category level. First, the effectiveness of the binary classifier in distinguishing IoT devices from non-IoT devices is shown under different settings. We see that the average accuracy is 94% and 96% when devices are split into known and unknown sets based on company-specific and general-device categories, respectively. The precision and recall are around 95% and 97% under company-specific setting, while they are around 97% and 99% under the general-device setting. Features such as unique packet length and packet delay are dominant in distinguishing IoT traffic from non-IoT traffic (see Table 16 in Appendix G for the ranking of feature groups). Next, for the binary classifier that distinguishes between known and unknown IoT devices, we see that the average accuracy, precision and recall are around 90% under the company-specific setting, while this number is 95% when devices are distributed based on general-device category. These numbers indicate that an adversary can effectively distinguish known and unknown devices.

Finally, the table reports the accuracy, precision and recall of the *multi-class* classifier for the *predicted* known and unknown samples (following the procedure shown in Figure 3). We see around 88% accuracy in properly classifying the predicted known samples into individual device labels (note that the adversary wishes to determine the exact device label for predicted known samples, not the company-specific or general-device category labels). The errors, in this case, are caused by some

Table 13. Open-world evaluations under strategic setting. Over 45% of the unknown samples are correctly classified at the company-specific level.

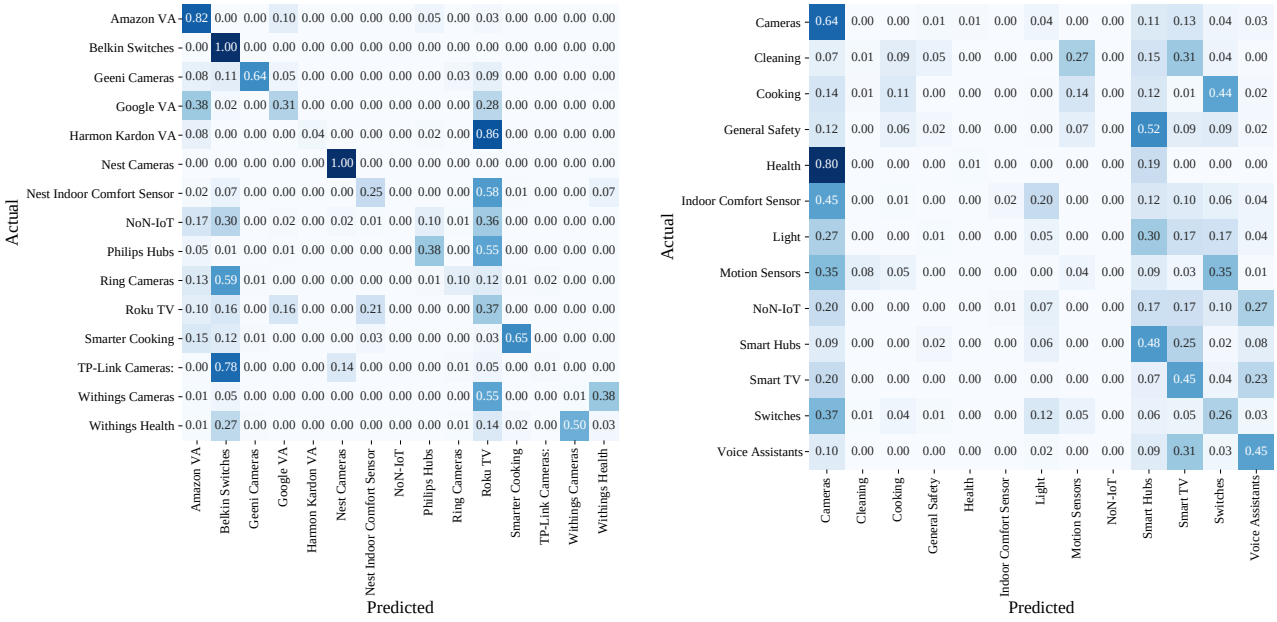
Setting	Prediction	Accuracy	Precision	Recall
Company-specific	IoT vs. Non-IoT	93.56 (1.76)	95.35 (1.9)	97.16 (1.49)
	Known vs. Unknown	90.17 (1.06)	90.18 (1.07)	90.16 (1.17)
	Known samples	87.81 (1.05)	69.34 (3.27)	74.08 (3.23)
	Unknown samples	45.45 (5.78)	55.51 (7.46)	48.56 (4.76)
General category	IoT vs. Non-IoT	95.72 (0.79)	96.68 (0.85)	98.82 (0.65)
	Known vs. Unknown	95.45 (0.73)	95.39 (0.74)	95.5 (0.74)
	Known samples	94.18 (1.1)	81.37 (1.14)	84.03 (0.92)
	Unknown samples	37.56 (3.05)	28.75 (3.4)	26.14 (3.54)

of the unknown samples that are classified as known by the binary classifier. Upon investigation, we found that the *true unknown* samples that are classified as known, actually belong to similar functional device groups in the known set (e.g., Google Home mini being labeled as Google Home — both devices using the same protocols and ports; only packet size was significantly different). This is validated by 100% accuracy when we consider only company-specific device labels for the predicted known samples. For the *predicted unknown samples*, we see that around 45% of them are correctly classified, suggesting the adversary can reasonably identify unknown samples from similar functional devices (7 times more accurate than a random guess).[5] We see similar results when devices are split into known and unknown sets based on general-device category (close to 5 times more accurate than a random guess).[6] These results suggest that the adversary can benefit by covering one or more devices from each company-specific category or general-device category into the known set.

To obtain deeper insight into the inaccuracies, we generate the confusion matrix for the *unknown* samples. Figure 4 highlights the confusion matrices for the company-specific categorization and general-device categorization. We see that in the company-specific category, traffic from 'Harmon Kardon Voice Assistant' gets mapped to 'Roku-TV'. This could be because 'Roku TV' has a voice-based remote, which mimics similar functionality as a voice assistant. We also see that 'Withings Health' and 'Withings Cameras' are being labeled as one another. This observation could be attributed to a manufacturer-specific software stack (e.g., sending updates to the same backend server at the same peri-

5 We had 15 company-specific device category, so a random guess would be successful 6.67% of the time.

6 We had 13 general-device categories, so a random guess would be successful 7.69% of the time.



(a) Predicted Unknown: Company-specific category

(b) Predicted Unknown: General-device category

Fig. 4. Confusion matrix for predicted unknown samples under open-world setting. Adversary is able to perform better when training data contains similar functional devices from the same vendor.

odic rate). We observed a similar phenomenon in Section 4 (in Figure 12), where 'Belkin Motion Sensors' were being predicted as 'Belkin Switches'. We also observe that 'Ring Camera' and 'TP-link Camera' traffic are predicted as 'Belkin Switches'. This could result from cameras being controlled through smart switches. Another source of error is Google voice assistants being predicted as Amazon voice assistants. This could be due to the similar functionality of these devices, for example, contacting the same backend services for streaming news or music. Lastly, we observe multiple devices being predicted as 'Roku TV', this could imply that traffic from 'Roku TV' is noisy and therefore harder to predict.

For the general-device category prediction (Figure 4b), we again see that devices in the health category are being labeled as cameras. This is likely due to traces from 'Withings Health' being mislabeled as 'Withings Cameras' (see Table 18). We see similar cases arise between – 'cooking' and 'motion sensor' category (because of 'Belkin WeMo Crockpot' and 'Belkin Motion Sensors'); 'cooking' and 'switches' category (because of 'Belkin WeMo Switch' and 'Belkin WeMo Crockpot'); 'motion sensors' and 'switches' category (because of 'Belkin WeMo Motion Sensor' and 'Belkin WeMo Switch'); 'indoor comfort sensor' and 'camera' category (because of 'Nest Thermostat' and 'Nest Cam'); 'motion sensor' and 'camera' category (because of 'D-Link Motion Sensor' and 'D-Link Camera'); 'switches' and 'camera' category (because of 'TP-Link WiFi plug' and 'TP-Link Day Night Camera'). Traffic from the 'cleaning' category also gets matched to the 'motion sensor' category, possibly because cleaning devices use motion sensors to sense their surroundings and periodically send status updates to the backend server. 'General Safety' devices like motion detectors or door/window open detectors are typically controlled through hubs, and thus we see overlap between 'General Safety' and 'Smart hubs'. We also see voice assistant traffic being classified as smart TVs and vice versa; this could be because most smart TVs have voice assistant functionalities nowadays. In general, we see that the adversary obtains better performance when the training set contains at least one device per company-specific device category.

6.3 Targeted Attacks

In the real world, an adversary can have many different goals and likely will not be interested in all devices but some specific device or group of devices. This group can be the same make and model devices or devices from a different vendor of the same functionality. For example, an adversary might want to know if this household has a Google Voice Assistant or a Smart Bulb from any manufacturer. This could be used to exploit a particu-

Table 14. Effectiveness of classifier under targeted device settings

Group	Accuracy	Precision	Recall
Geeni Cameras	98.03 (0.44)	89.39 (4.61)	69.26 (10.09)
Google Home Devices	99.74 (0.12)	99.66 (0.64)	88.98 (5.5)
Roku TV	98.6 (0.38)	91.37 (4.98)	61.42 (12.62)
Ring Doorbells	98.46 (0.47)	99.94 (0.05)	65.9 (10.45)
Amazon Echo Devices	97.78 (0.53)	88.47 (3.9)	69.72 (9.59)
Belkin WeMo Devices	99.73 (0.21)	99.68 (0.37)	96.9 (2.48)
Smart Switches	95.29 (0.59)	77.24 (4.86)	60.67 (6.34)

lar device vulnerability. Hence, we analyze different possible groups to demonstrate the performance improvement from previous sections where the adversary is not focusing on specific devices. For this, we first select a target group of devices and label all the samples from these devices as the target and, consequently, all other samples from all other devices (IoT or non-IoT) as non-target. Then we do a standard 80:20 train test split to test the effectiveness of the classifier. For training, we randomly select half of the devices from the target and non-target sets. For example, if we have Google Home devices as the selected target group, then we select either Google Home or Google Home Mini (since this group only has two devices) to train the classifier, and the other device is unseen during the training phase. Similarly, from other non-target devices, we do an even half split, and the classifier does not see half of the devices during the training phase. Finally, we test using the test dataset and compute the metrics. We have evaluated this attack on different groups, including similar category (i.e., functionality), same company (vendor), same company-category, and same make-and-model to illustrate different settings possible under this attack. Table 14 shows the results.

The accuracy is high across the board. Precision is on the higher side compared to recall. For certain devices, we see that the recall is comparatively low. These cases occur for target devices that are spuriously classified as other devices. For example, from Figure 11 we can see that Roku TV is mislabelled as other devices (e.g., as Nest Thermostat). Similarly, Amazon Echo devices are labeled as Samsung SmartThings Hub. Overall, however, the results show that an adversary can get better results when compared to previous approaches when targeting devices, even when devices are unknown.

Takeaway. Our open-world analysis showcases that an adversary is able to improve inference by 5–7 times than a *random guess*. Alternatively, an adversary can focus on detecting the presence of a specific device (e.g., with some known vulnerability) and build device-specific binary classifiers (i.e., a test sample is either a target device or not). In general, we see that an adversary benefits when the training set contains at least one device type per company, enabling them to predict the company-specific device category.

7 Discussion

Summary. We analyze the feasibility of fingerprinting IoT devices under many real-world settings. This work advances the state-of-the-art by providing the following insights: (1) it is possible to fingerprint IoT devices with high accuracy at different granularities, even with multiple instances of the same make-and-model devices; (2) temporality, locality and data collection method affect the generalizability of fingerprinting; however, some features are more resistant to varying datasets; (3) an adversary can increase their odds of inferring meaningful information about unseen devices by training a model on at least one IoT device from each company-specific category or general-device category. We have open-sourced our datasets and code [7].

Potential Countermeasures. We also emulated how packet padding and traffic morphing (i.e., controlling traffic delay and volume through dummy packets) would impact our device fingerprinting approach. We emulate the impact of such countermeasures by disabling/dropping features that can be impacted by the corresponding countermeasure(s) — assuming countermeasures fully hide a subset of the features. We consider a closed-world setting to obtain an upper bound success rate in the presence of countermeasures. We found that even when dropping any features related to packet size, burst or delay, we can still uniquely identify IoT devices with 96% accuracy when considering the close-world setting on the combined dataset. When looking at the remaining features that enabled such high accuracy, we found that simply using the list of IPs, protocols and external ports were sufficient to detect devices uniquely. Thus, a *VPN/proxy* based approach may be useful in hiding the remaining useful features. A VPN service tunnels traffic through an encrypted connection and routes all traffic from the gateway router to the VPN service. As a result, an adversary intercepting traffic between the gateway router and the VPN service will only see the same set of IPs, protocols, ports and hostnames for all traffic generated by different IoT devices. How-

7 https://github.com/dilawer11/iot-device-fingerprinting.

ever, full-fledged analysis of the existing countermeasure may still reveal sensitive information, as demonstrated by Dyer et al. [18]. We leave such analysis as future work.

Potential Sources of Errors. In both the open-world and generalization settings, we observe more false positives and false negatives when compared to the simple closed-world approach. Upon further investigation, we found that these errors arise from a couple of factors, including but not limited to similar devices (e.g., Google Home and Google Home Mini, Amazon Echo family), devices from the same company (e.g., Belkin WeMo devices) and devices from a different company but with similar functionalities (e.g., Google Home and Amazon Echo). These errors will always be present, especially in open-world settings.

Some special cases may create further confusion. For example, the Alexa app running on a smartphone can be confused with an Alexa device. However, we found that their network fingerprints are very different (features such as packet length, burst vary highly between the two) even when Alexa is performing the same task, and hence we could easily distinguish between them with over 99% accuracy.

Comparison with Website Fingerprinting. Juarez et al. [24] state that website fingerprinting is prone to decaying accuracy over time since the content of the webpage changes very rapidly. According to their paper, the accuracy falls under 50% in just ten days. However, in our experiments on IoT devices, we do not observe a decay or drift for IoT devices because most changes come from firmware updates that primarily fix small bugs and do not introduce significant protocol/packet-level changes, as evident from the traces. We saw that most devices still had similar delays and packet sizes; e.g., Smart WiFi Plug had an average packet size of 110.42 previously (in 2020), and in the new dataset (in 2021), it was 110.08. Similarly, the Amazon Echo Plus previously had an average packet size of 274, and in the newer dataset, it was 281. In terms of unique destinations contacted, we also saw similar trends where most devices contacted similar numbers of endpoints, e.g., Geeni Camera previously had contacted 5.55 unique destinations on average in each window, whereas in the new dataset, it has 5.04. Our analysis on temporal impact showcased that the fingerprints do not change drastically over time, thus allowing an adversary to periodically retrain the model (maybe every once a year) and maintain high accuracy. However, specific categories of devices may receive more updates over any given span of time, potentially impacting fingerprint stability. However, a motivated and resourceful adversary can frequently retrain the model using updated data (whenever feasible).

Limitations. There are a few limitations to our work. First, we manually categorized devices into different groups based on their functionality, but some devices have dual functionality, and we had to choose one using our best judgment. Second, since six of the datasets were collected from external sources, there is the possibility that some of the devices are wrongly labeled. However, looking at the high fingerprinting accuracy on the combined dataset (Table 4) this is likely negligible. Thirdly, multiple devices of the same make but different models/versions may be labeled as the same device. This can result in poor generalized accuracy and poor open-world results in some cases. However, with better labeling, classification results will likely improve. We also have a limited number of non-IoT devices in our dataset, whereas non-IoT traffic is expected to be much higher in the wild.

8 Conclusion

In this paper, we analyze the feasibility of fingerprinting IoT devices across several diverse datasets. We show that it is possible to uniquely fingerprint many IoT devices (188 devices in total) and do so when there are multiple devices of the same make and model. We also explore the possibility of predicting devices at different levels of granularity. We, next, analyze the extent to which temporality, locality and data-collection-methodology impact classification. Our analysis sheds light on potentially more robust features across varying conditions. Lastly, we showcase how an adversary can design an open-world classifier to obtain additional insights about unseen devices. We find that an adversary can guess the proper company category seven times more accurately than a random guess by exploiting the metadata of encrypted network traffic.

Acknowledgement

We thank our anonymous reviewers for their valuable feedback. This material is based upon work supported in parts by the National Science Foundation under grant number CNS-1849997 and the Center for Accelerated

Real Time Analytics (CARTA) - NCSU Research Site. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Smarthomedb. https://www.smarthomedb.com/. Accessed: 2021-12-02.

[2] OpenWrt Project, 2020. https://openwrt.org/.

[3] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A. Sadeghi, and A. S. Uluagac. Peek-a-boo: I see your smart home activities, even encrypted! *CoRR*, abs/1808.02741, 2018.

[4] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel. Fpdetective: dusting the web for fingerprinters. In *Proceedings of the 20th ACM SIGSAC conference on Computer and Communications Security (CCS)*, pages 1129–1140, 2013.

[5] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose. Sok: Security evaluation of home-based IoT deployments. In *Proceedings of the 40th IEEE Symposium on Security and Privacy (SP)*, pages 1362–1380, 2019.

[6] B. Anderson and D. McGrew. Identifying encrypted malware traffic with contextual flow data. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security (AISec)*, pages 35–46, 2016.

[7] N. Apthorpe, D. Y. Huang, D. Reisman, A. Narayanan, and N. Feamster. Keeping the smart home private with smart(er) IoT traffic shaping. *Proceedings on Privacy Enhancing Technologies*, 2019(3):128–148, 2019.

[8] N. Apthorpe, D. Reisman, and N. Feamster. A smart home is no castle: Privacy vulnerabilities of encrypted IoT traffic. *CoRR*, abs/1705.06805, 2017.

[9] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster. Spying on the smart home: Privacy attacks and defenses on encrypted IoT traffic. *CoRR*, abs/1708.05044, 2017.

[10] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine. Privacy vulnerabilities in encrypted http streams. In *Proceedings of the 5th International Conference on Privacy Enhancing Technologies (PETS)*, pages 1–11, 2005.

[11] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS)*, pages 605–616, 2012.

[12] S. Chen, R. Wang, X. Wang, and K. Zhang. Side-channel leaks in web applications: A reality today, a challenge tomorrow. In *Proceedings of the 31st IEEE Symposium on Security and Privacy (SP)*, pages 191–206, 2010.

[13] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde. Can't you hear me knocking: Identification of user actions on android apps via traffic analysis. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy (CODASPY)*, pages 297–304, 2015.

[14] B. Copos, K. Levitt, M. Bishop, and J. Rowe. Is anybody home? inferring activity from smart home network traffic. In *IEEE Security and Privacy Workshops (SPW)*, pages 245–251. IEEE, 2016.

[15] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song. Networkprofiler: Towards automatic fingerprinting of android apps. In *Proceedings of the 32nd IEEE INFOCOM*, pages 809–817, 2013.

[16] A. Das, N. Borisov, and E. Chou. Every move you make: Exploring practical issues in smartphone motion sensor fingerprinting and countermeasures. *Proceedings on Privacy Enhancing Technologies*, 2018(1):88–108, 2018.

[17] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee. Identifying unique devices through wireless fingerprinting. In *Proceedings of the 1st ACM conference on Wireless Network Security*, pages 46–55, 2008.

[18] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Proceedings of the 33rd IEEE Symposium on Security and Privacy (SP)*, pages 332–346. IEEE, 2012.

[19] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker. Passive data link layer 802.11 wireless device driver fingerprinting. In *Proceedings of the 15th Conference on USENIX Security Symposium (USENIX Security)*, volume 3, pages 16–89, 2006.

[20] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, Mar. 2003.

[21] J. Hayes and G. Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security)*, pages 1187–1203, 2016.

[22] D. Herrmann, R. Wendolsky, and H. Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *Proceedings of the 2009 ACM workshop on Cloud Computing Security*, pages 31–42, 2009.

[23] Y. Jin, E. Sharafuddin, and Z.-L. Zhang. Unveiling core network-wide communication patterns through application traffic activity graph decomposition. In *Proceedings of the 11th International Joint Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 49–60, 2009.

[24] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS)*, page 263–274, 2014.

[25] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: Multilevel traffic classification in the dark. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, page 229–240, 2005.

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

[27] Z. Li, W. Xu, R. Miller, and W. Trappe. Securing wireless systems via lower layer enforcements. In *Proceedings of the 5th ACM workshop on Wireless Security (WiSec)*, pages 33–42, 2006.

[28] M. Liberatore and B. N. Levine. Inferring the source of encrypted http connections. In *Proceedings of the 13th ACM conference on Computer and Communications Security (CCS)*, pages 255–263, 2006.

[29] L. Lu, E.-C. Chang, and M. C. Chan. Website fingerprinting and identification using ordered feature sequences. In *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS)*, pages 199–214, 2010.

[30] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[31] S. Marchal, M. Miettinen, T. D. Nguyen, A. Sadeghi, and N. Asokan. Audi: Toward autonomous IoT device-type identification using periodic communication. *IEEE Journal on Selected Areas in Communications*, 37(6):1402–1412, 2019.

[32] 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 *Proceedings of the 37th IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 2177–2184, 2017.

[33] S. B. Moon, P. Skelly, and D. Towsley. Estimation and removal of clock skew from network delay measurements. In *Proceedings of the 18th IEEE Conference on Computer Communications (INFOCOM)*, volume 1, pages 227–234, 1999.

[34] N. T. Nguyen, G. Zheng, Z. Han, and R. Zheng. Device fingerprinting to enhance wireless security using nonparametric bayesian method. In *Proceedings of the 30th IEEE INFOCOM*, pages 1404–1412. IEEE, 2011.

[35] T. T. T. Nguyen and G. Armitage. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys Tutorials*, 10(4):56–76, 2008.

[36] T. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.-R. Sadeghi. Homesnitch: behavior transparency and control for smart home IoT devices. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, pages 128–138, 2019.

[37] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle. Website fingerprinting at internet scale. In *Proceedings of the 23rd Annual Network and Distributed System Security Symposium (NDSS)*, 2016.

[38] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 103–114, 2011.

[39] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 99–110, 2007.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[41] R. Perdisci, W. Lee, and N. Feamster. Behavioral clustering of http-based malware and signature generation using malicious network traces. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI)*, page 26, 2010.

[42] R. Perdisci, T. Papastergiou, O. Alrawi, and M. Antonakakis. Iotfinder: Efficient large-scale identification of iot devices via passive dns traffic analysis. In *Proceedings of the 5th IEEE European Symposium on Security and Privacy (EuroS&P)*, 2020.

[43] A. Reed and M. Kranch. Identifying https-protected netflix videos in real-time. In *Proceedings of the 7th ACM on Conference on Data and Application Security and Privacy*, pages 361–368, 2017.

[44] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi. Information exposure from consumer IoT devices: A multidimensional, network-informed measurement approach. In *Proceedings of the 19th Internet Measurement Conference (IMC)*, pages 267–279, 2019.

[45] S. J. Saidi, A. M. Mandalari, R. Kolcun, D. J. D. Hamed Haddadi, D. Choffnes, G. Smaragdakis, and A. Feldmann. A haystack full of needles: Scalable detection of IoT devices in the wild. In *Proceedings of the 20th Internet Measurement Conference (IMC)*, 2020.

[46] B. Saltaformaggio, H. Choi, K. Johnson, Y. Kwon, Q. Zhang, X. Zhang, D. Xu, and J. Qian. Eavesdropping on fine-grained user activities within smartphone apps over encrypted network traffic. In *Proceedings of the 10th USENIX Conference on Offensive Technologies (WOOT)*, pages 69–78, 2016.

[47] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, T. Kohno, et al. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *Proceedings of the 16th USENIX Security Symposium (USENIX Security)*, pages 55–70, 2007.

[48] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2018.

[49] 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 *IEEE Conference on Computer Communications Workshops*, pages 559–564, 2017.

[50] R. Sommer and V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *Proceedings of the 30th IEEE Symposium on Security and Privacy (SP)*, pages 305–316, 2010.

[51] D. Song. Timing analysis of keystrokes and ssh timing attacks. In *Proceedings of the 10th USENIX Security Symposium (USENIX Security)*, 2001.

[52] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings of the 23rd IEEE Symposium on Security and Privacy*, pages 19–30, 2002.

[53] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic. Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *Proceedings of the 1st IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 439–454, 2016.

[54] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky. Packet-level signatures for smart home devices. In *Proceedings of the 27th Annual Network and Distributed System Security Symposium (NDSS)*, 2020.

[55] T. van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. Choffness, M. van Steen, and A. Peter. FlowPrint: Semi-Supervised Mobile-App Fingerprinting on Encrypted Network Traffic. In *Proceedings of the 27th Annual Network and Distributed System Security Symposium (NDSS)*, 2020.

[56] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. Effective attacks and provable defenses for website fingerprinting. In *Proceedings of the 23rd USENIX Security Symposium (USENIX Security)*, pages 143–157, 2014.

[57] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson. Uncovering spoken phrases in encrypted voice over ip conversations. *ACM Transactions on Information and System Security (TISSEC)*, 13(4):1–30, 2010.

[58] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson. Language identification of encrypted VOIP traffic: Alejandra y roberto or alice and bob? In *Proceedings of the 16th USENIX Security Symposium (USENIX Security)*, volume 3, pages 43–54, 2007.

[59] C. V. Wright, S. E. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *Proceedings of the 16th Annual Network and Distributed System Security Symposium (NDSS)*, 2009.

[60] J. Yan and J. Kaur. Feature selection for website fingerprinting. *Proceedings on Privacy Enhancing Technologies (PETS)*, 2018(4):200–219, 2018.

[61] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pages 412–420, 1997.

[62] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu. Homonit: Monitoring smart home apps from encrypted traffic. In *Proceedings of the 25th ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 1074–1088, 2018.

Appendix

A Distribution of Device Instance

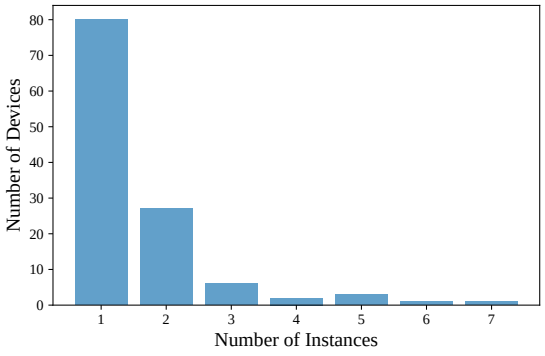


Fig. 5. Number of instances per make-and-model device.

B Example of Device Clusters

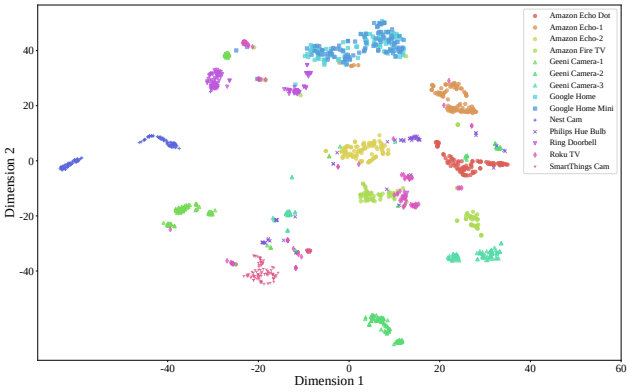


Fig. 6. Scatter plot of samples from different devices. Samples from individual devices are closely clustered together.

C Top Ranked Features across Datasets

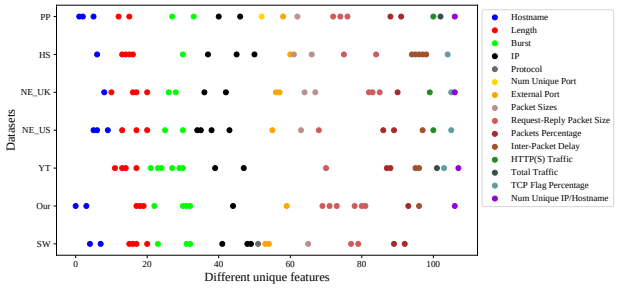


Fig. 7. Top 20 features in each dataset. Different sets of features appear as the top 20 features for each dataset.

D Temporal Generalizability

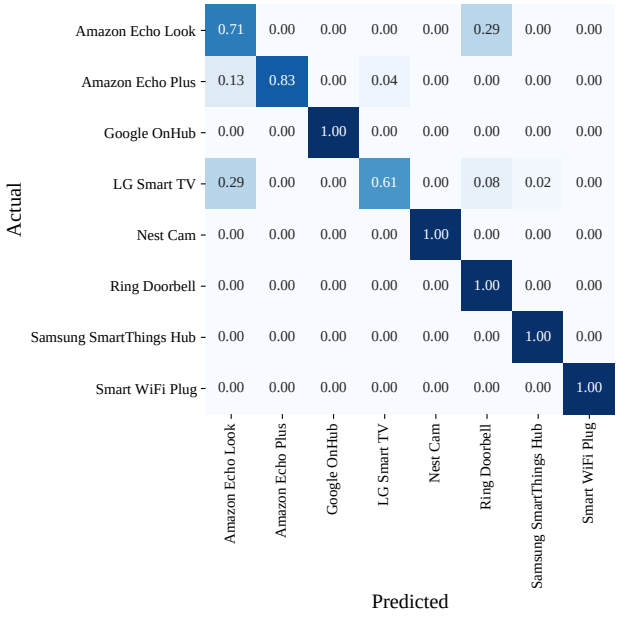


Fig. 8. Confusion matrix when training on 'Our' dataset from early 2020 and testing on 'Our' dataset from early 2021.

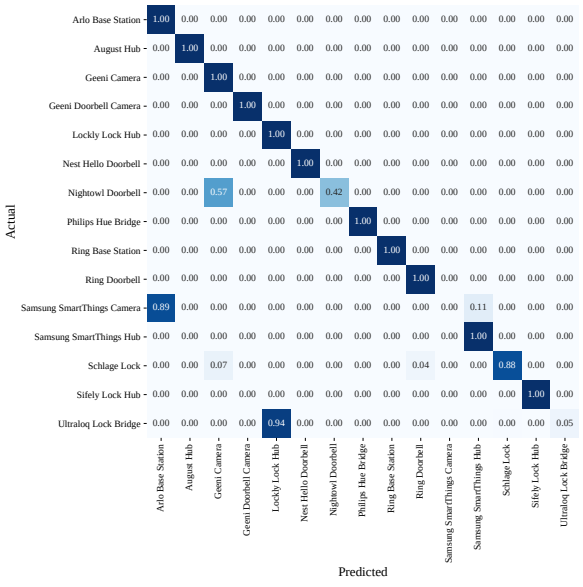


Fig. 9. Confusion matrix when training on HomeSnitch dataset from early 2020 and testing on HomeSnitch dataset from early 2021.

E Spatial Generalizability

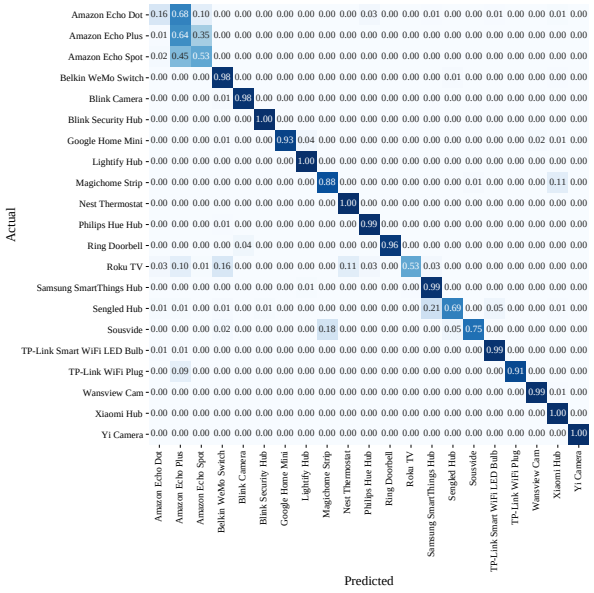


Fig. 10. Confusion matrix when training on NE_UK dataset and testing on NE_US dataset.

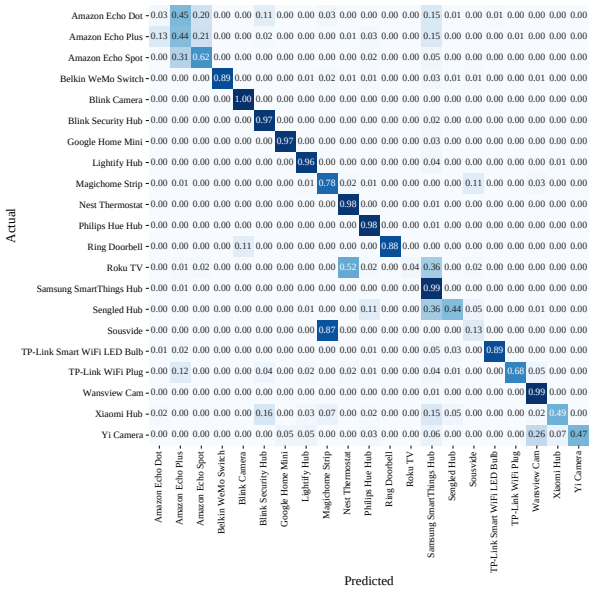


Fig. 11. Confusion matrix when training on NE_US dataset and testing on NE_UK dataset.

Table 15. Effectiveness of classifier when training and testing on different continuous datasets.

Train	Test	Accuracy	Precision	Recall	Devices
HS	Our	98.43 (0.0)	97.99 (0.0)	98.74 (0.0)	2
Our	HS	99.98 (0.01)	99.98 (0.01)	99.98 (0.01)	2
YT	Our	79.66 (5.24)	83.65 (3.58)	76.94 (6.0)	4
Our	YT	91.67 (3.14)	93.73 (1.97)	92.04 (3.0)	4
YT	HS	33.06 (1.05)	38.03 (4.49)	22.59 (2.0)	5
HS	YT	75.41 (4.67)	82.87 (4.23)	74.99 (4.95)	5
YT, Our	HS	71.82 (1.05)	67.33 (2.81)	55.58 (0.72)	5
HS, Our	YT	70.85 (3.44)	79.55 (3.29)	71.28 (3.56)	7
HS, YT	Our	59.31 (2.92)	77.95 (1.05)	64.3 (2.56)	4

F Confusion Matrix under General-device Category

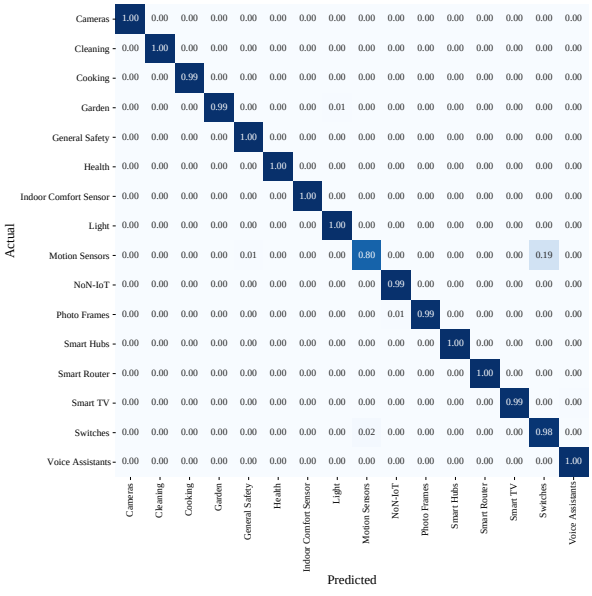


Fig. 12. Confusion matrix for classifying devices under general category. All device categories achieve high accuracy, except for the 'motion sensor' category.

G IoT vs. Non-IoT Classifier

Table 16. Top features for IoT vs. Non-IoT classifier.

Features	Importance		
Unique Packet Lengths	0.16	Burst Delay	0.04
Packet Delay	0.10	Protocol List	0.04
Packet Sizes List	0.08	External Ports List	0.04
Burst Bytes	0.08	Request-Reply Pkt Sizes	0.04
Burst Packets	0.07	Total Packets	0.02
Hostname List	0.07	Unique IPs	0.02
Protocols	0.06	Packet In and Out Ratio	0.02
IP List	0.06	Unique Ports	0.01
Burst Time	0.05	Total Bytes	0.01
TCP Flags	0.04	HTTP(S) Traffic	0.01

H Distribution of IoT Devices

Table 17. Company-specific distribution of IoT devices where each categories has at least two devices. We use these devices for predicting company-specific category labels under open-world setting.

Nest Cameras	**Ring Cameras**	**Belkin Switches**
Nest Cam IQ	Ring Doorbell	Belkin WeMo Link
Nest Hello Doorbell	Ring Doorbell Chime	Belkin WeMo Switch
Nest Cam	Ring Doorbell Pro	Belkin WeMo Insight
Nest Indoor Comfort Sensor	**TP-Link Cameras**	**Withings Cameras**
Nest Thermostat	TP-Link Kasa Camera	Withings Home
Nest Protect smoke alarm	TP-Link Day Night Camera	Withings Smart Baby Monitor
Geeni Cameras	**Smarter Cooking**	**Roku Smart TV**
Geeni Doorbell Camera	iKettle	Roku TV
Geeni Camera	Smarter Coffee Machine	Roku 4
Philips Hubs	**Google VA**	**Amazon VA**
Philips Hue Bridge	Google Home	Echo, Echo Dot, Dot Kids
Philips Hue Hub	Google Home Mini	Echo Show, Echo Plus
		Echo Spot, Echo Look
Harmon Kardon VA	**Withings Health**	
Allure Speaker	Withings Smart scale	
Harmon Kardon Invoke	Withings Aura smart sleep sensor	

Table 18. Distribution of IoT devices into general device categories where each categories has at least two devices. We use these devices for predicting general-device category labels under open-world setting.

Smart TV	**Switches**	**Light**
LG Smart TV, Amazon Fire TV, Apple TV, Roku TV, Roku 4, nVidia Shield, Samsung SmartTV	TP-Link WiFi Plug, Belkin WeMo Link, D-Link Plug, Belkin WeMo Switch, Belking WeMo Insight, Amazon Plug, Smart WiFi Plug	Bulb 1, Koogeek Lightbulb, TP-Link Smart WiFi LED Bulb, LIFX Virtual Bulb, Sengled Bulb, Magichome Strip,Xiaomi Strip Philips Hue Bulb
Cleaning	**Health**	**Motion Sensors**
Xiaomi Cleaner, Roomba,	Withings Smart scale, Withings Aura smart sleep sensor,	Belkin WeMo Motion Sensor, Chamberlain myQ Garage Opener, D-Link Motion Sensor,
General Safety	**Indoor Comfort Sensor**	**Cooking**
Ring Security System, Nest Guard, Schlage Lock, D-Link Alarm,	Ecobee Thermostat, Nest Thermostat, Netatmo weather station, Nest Protect smoke alarm, Honeywell Thermostat,	Smarter Coffee Machine, Belkin WeMo Crockpot, iKettle, Brewer, Fridge, Sousvide, Microwave
Voice Assistants (VA)	**Smart Hubs**	**Cameras**
Allure Speaker, Apple HomePod, Sonos, Google Home, Google Home Mini, Canary, Amazon Echo, Amazon Echo Dot, Amazon Dot Kids, Bose SoundTouch 10, Amazon Echo Plus, Amazon Echo Show, Amazon Echo Spot, Amazon Echo Look, iHome, Harmon Kardon Invoke, Triby Speaker,	Sengled Hub, Blink Security Hub, Insteon Hub, Wink 2 Hub, Logitech Harmony Hub, Philips Hue Hub, Caseta Wireless Hub, Samsung SmartThings Hub, Google OnHub, August Lock Hub, MiCasaVerde VeraLite, Lightify Hub, Xiaomi Hub, Arlo Base Station, Ultraloq Lock Bridge, Lockly Lock Hub, Sifely Lock Hub, Ring Base Station, Philips Hue Bridge, Blink Security Hub,	Belkin Netcam, Cloudcam, Chinese Webcam, Yi Camera, Luohe Spycam, Zmodo Doorbell, Blink Camera, Lefun Cam, TP-Link Day Night Camera, Netatmo Welcome, Nest Cam IQ, Dropcam, Nest Cam, Piper NV, Withings Smart Baby Monitor, Amcrest Cam, Withings Home, D-Link Camera, Insteon Camera, Wansview Cam, Charger Camera, Samsung SmartThings Camera, Night Owl Doorbell Camera, Bosiwo Camera, Xiaomi Camera 2, Ring Doorbell Chime, Netgear Arlo Cam, TP-Link Kasa Cam, Nest Hello Doorbell, August Doorbell Cam, Logitech Logi Circle, Geeni Camera, Ring Doorbell, Ring Doorbell Pro, Geeni Doorbell Cam, Microseven Cam,

Table 19. All unique make and model devices grouped into general-device categories.

Cameras	**Smart Hubs**	**Voice Assistants**
Belkin Netcam, Cloudcam	Sengled Hub, Insteon Hub	Allure Speaker, Sonos
Chinese Webcam, Blink Camera	Blink Security Hub	Apple HomePod
Yi Camera, Lefun Cam	Philips Hue Bridge	Triby Speaker, Google Home
Luohe Spycam, Zmodo Doorbell	Wink 2 Hub, Lightify Hub	Bose SoundTouch 10
TP-Link Kasa Camera	Logitech Harmony Hub	Google Home Mini, Canary
Ring Doorbell Chime	Philips Hue Hub	iHome, Harmon Kardon Invoke
TP-Link Day Night Cloud camera	Caseta Wireless Hub	Amazon Echo
Netatmo Welcome, Ring Doorbell	Samsung SmartThings Hub	Amazon Echo Dot
Nest Cam IQ, Dropcam	Google OnHub, Xiaomi Hub	Amazon Dot Kids
Nest Cam, Piper NV	MiCasaVerde VeraLite	Amazon Echo Plus
D-Link Camera, Insteon Camera	Ring Base Station	Amazon Echo Show
Netgear Arlo Camera	Sifely Lock Hub	Amazon Echo Spot
Bosiwo Camera, Charger Camera	Arlo Base Station	Amazon Echo Look
Logitech Logi Circle	Ultraloq Lock Bridge	**Health**
Samsung SmartThings Camera	August Lock Hub	Withings Smart scale
Ring Doorbell Pro	Lockly Lock Hub	Withings Aura smart sleep sensor
Amcrest Cam, Withings Home	**Indoor Comfort Sensor**	**Cleaning**
Withings Smart Baby Monitor	Nest Thermostat	Xiaomi Cleaning, Roomba
August Doorbell Cam	Netatmo weather station	**Light**
Wansview Cam, Geeni Camera	NEST Protect smoke alarm	Magichome Strip, Sengled Bulb
Microseven Camera	Ecobee Thermostat	Xiaomi Strip, Philips Hue Bulb
Geeni Doorbell Camera	Honeywell Thermostat	Bulb 1, Koogeek Lightbulb
Xiaomi Camera 2	**Smart TV**	TP-Link Smart WiFi LED Bulb
Nest Hello Doorbell	Apple TV, Roku TV	LIFX Virtual Bulb
Night Owl Doorbell Camera	Samsung SmartTV	**Motion Sensors**
Smart Router	nVidia Shield	Belkin WeMo Motion Sensor
Securifi Almond	LG Smart TV, Roku 4	Chamberlain myQ Garage Opener
Switches	Amazon Fire TV	D-Link Motion Sensor
Smart WiFi Plug	**Photo Frames**	**Garden**
Amazon Plug	Pix-Star Photo Frame	Rachio Sprinkler
D-Link Plug	**Cooking**	**General Safety**
TP-Link WiFi Plug	Belkin WeMo Crockpot	Nest Guard
Belkin WeMo Link	iKettle, Sousvide, Brewer	Schlage Lock
Belkin WeMo Switch	Microwave, Fridge	D-Link Alarm
Belkin WeMo Insight	Smarter Coffee Machine	Ring Security System

I Comparison with Related Work

Table 20. Comparison with existing work on IoT device fingerprinting. Symbols convey the following meanings – ○: not analyzed, ◐: partially analyzed, ●: analyzed.

		Apthorpe et al. [9]	Ren et al. [44]	Sivanathan et al. [48]	Copos et al. [14]	HomeSnitch [36]	PINGPONG [54]	Our work
Features used	Packet timing	○	●	◐	○	●	○	●
	Packet sizes	○	●	○	○	●	●	●
	Traffic direction	○	○	○	●	●	●	●
	Burst timing	○	○	○	○	●	○	●
	Burst sizes	○	○	○	○	●	○	●
	Flow timing	●	○	●	○	●	○	○
	Flow sizes	●	○	●	●	●	○	○
	External ports	○	○	●	○	○	○	●
	External IPs	○	○	○	●	○	○	●
	External domains	○	○	●	○	○	○	●
	Protocols	○	○	○	○	○	○	●
	DNS queries	●	○	●	○	○	○	●
	NTP queries	○	○	●	●	○	○	●
	MAC Addresses	●	○	○	○	○	○	○
Evaluation settings	Closed-world	●	●	●	●	●	●	●
	Open-world	○	○	○	○	○	◐	●
Granularity of detection	Device-activity *	●	●	○	●	●	●	○
	Device make-and-model *	●	○	●	○	●	●	●
	Company-specific ◇	○	○	○	○	○	○	●
	General-device †	○	○	○	○	○	○	●
Datasets used (geographic origin)		1 (US)	2 (US, UK)	1 (AU)	1 (US)	2 (US)	3 (US, AU)	7 (US, UK, AU)
Number of unique device types		7	55	28	2	20	19	120

* Inferring individual device-level activity; * Individual devices (different make and model); ◇ Devices from the same vendor with similar functionality, e.g., Google Home and Google Home Mini are grouped together; † Company agnostic device category, e.g., Amazon Echo and Google Home are smart voice assistants;

J Full Feature List

Table 21. List of all features grouped based on different attributes.

Burst Time	Burst Bytes	Burst Packets	Burst Delay	Packet Delay	Unique IPs	Unique Ports
out_mean_bursttime	out_mean_burstbytes	out_mean_burstnumpkts	out_mean_interburstdelay	mean_interpktdelay	num_unique_ip	in_numuniquesrcport
in_mean_bursttime	in_mean_burstbytes	in_mean_burstnumpkts	in_mean_interburstdelay	median_interpktdelay	num_unique_ip_3octet	out_numuniquedstport
out_median_bursttime	out_median_burstbytes	out_median_burstnumpkts	out_median_interburstdelay	25per_interpktdelay	num_unique_hostname	
in_median_bursttime	in_median_burstbytes	in_median_burstnumpkts	in_median_interburstdelay	75per_interpktdelay		
out_25per_bursttime	out_25per_burstbytes	out_25per_burstnumpkts	out_25per_interburstdelay	90per_interpktdelay		
in_25per_bursttime	in_25per_burstbytes	in_25per_burstnumpkts	in_25per_interburstdelay	std_interpktdelay	**Packet Size List**	**Request-Reply Pair List**
out_75per_bursttime	out_75per_burstbytes	out_75per_burstnumpkts	out_75per_interburstdelay	max_interpktdelay	Packet Sizes TLSTCP	Req-Reply Pair TLSTCP
in_75per_bursttime	in_75per_burstbytes	in_75per_burstnumpkts	in_75per_interburstdelay	min_interpktdelay	Packet Sizes DNS	Req-Reply Pair DNS
out_90per_bursttime	out_90per_burstbytes	out_90per_burstnumpkts	out_90per_interburstdelay		Packet Sizes UDP	Req-Reply Pair UDP
in_90per_bursttime	in_90per_burstbytes	in_90per_burstnumpkts	in_90per_interburstdelay	**External Port List**	Packet Sizes NTP	Req-Reply Pair NTP
out_std_bursttime	out_std_burstbytes	out_std_burstnumpkts	out_std_interburstdelay	External Port List		
in_std_bursttime	in_std_burstbytes	in_std_burstnumpkts	in_std_interburstdelay	External Port TLSTCP		
out_max_bursttime	out_max_burstbytes	out_max_burstnumpkts	out_max_interburstdelay	External Port DNS		
in_max_bursttime	in_max_burstbytes	in_max_burstnumpkts	in_max_interburstdelay	External Port UDP		
out_min_bursttime	out_min_burstbytes	out_min_burstnumpkts	out_min_interburstdelay	External Port NTP		
in_min_bursttime	in_min_burstbytes	in_min_burstnumpkts	in_min_interburstdelay			
Protocols	**Total Packets**	**Total Bytes**	**Unique Packet Length**	**Packet In and Out Ratio**	**TCP Flags**	**HTTP(S) Traffic**
out_tls1pkts_percentage	out_totalpkts	out_totalbytes	mean_out_uniquelen	out_percentage	out_tcpack_percentage	pkts_80_443_percentage
in_tls1pkts_percentage	in_totalpkts	in_totalbytes	mean_in_uniquelen	in_percentage	out_tcpsyn_percentage	byte_per_pkt_80_443
out_tls12pkts_percentage			median_out_uniquelen		out_tcpfin_percentage	
in_tls12pkts_percentage	**Hostname List**	**IP List**	median_in_uniquelen		out_tcprst_percentage	
out_tcppkts_percentage	Hostname List	IP List	25per_out_uniquelen		out_tcppsh_percentage	
in_tcppkts_percentage	Hostname TLSTCP	IP TLSTCP	25per_in_uniquelen		out_tcpurg_percentage	
out_udppkts_percentage	Hostname DNS	IP DNS	75per_out_uniquelen		in_tcpack_percentage	
in_udppkts_percentage	Hostname UDP	IP UDP	75per_in_uniquelen		in_tcpsyn_percentage	
out_dnspkts_percentage	Hostname NTP	IP NTP	90per_out_uniquelen		in_tcpfin_percentage	
in_dnspkts_percentage			90per_in_uniquelen		in_tcprst_percentage	
out_ssdppkts_percentage			len_out_uniquelen		in_tcppsh_percentage	
in_ssdppkts_percentage			len_in_uniquelen		in_tcpurg_percentage	
out_sslpkts_percentage			max_out_len			
in_sslpkts_percentage			max_in_len			
out_icmppkts_percentage			min_out_len			
in_icmppkts_percentage			min_in_len			
out_ntppkts_percentage						
in_ntppkts_percentage			**Protocol List**			
out_numuniqueprotocol			Protocol List			
in_numuniqueprotocol						