Date of acceptance Grade

Instructor

IoT device fingerprinting with sequence-based features

Nishadh Aluthge

Helsinki December 12, 2017

UNIVERSITY OF HELSINKI
Department of Computer Science

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Nishadh Aluthge

Työn nimi — Arbetets titel — Title

IoT device fingerprinting with sequence-based features

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
	December 12, 2017	64 pages + 3 appendix pages

Tiivistelmä — Referat — Abstract

Exponential growth of Internet of Things complicates the network management in terms of security and device troubleshooting due to the heterogeneity of IoT devices. In the absence of a proper device identification mechanism, network administrators are unable to limit unauthorized accesses, locate vulnerable/rogue devices or assess the security policies applicable to these devices. Hence identifying the devices connected to the network is essential as it provides important insights about the devices that enable proper application of security measures and improve the efficiency of device troubleshooting. Despite the fact that active device fingerprinting reveals in depth information about devices, passive device fingerprinting has gained focus as a consequence of the lack of cooperation of devices in active fingerprinting. We propose a passive, feature-based device identification technique that extracts features from a sequence of packets during the initial startup of a device and then uses machine learning for classification. Proposed system improves the average device prediction F_1-*score* up to 0.912 which is a 14% increase compared with the state-of-the-art technique. In addition, We have analyzed the impact of confidence threshold on device prediction accuracy when a previously unknown device is detected by the classifier. As future work we suggest a feature-based approach to detect anomalies in devices by comparing long-term device behaviors.

ACM Computing Classification System (CCS):
Networks → Network management
Security and privacy → Network security
Security and privacy → Access control

Avainsanat — Nyckelord — Keywords
Internet of Things, Network security, Device fingerprinting, Network management

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — övriga uppgifter — Additional information

Contents

1 Introduction

The *Internet of Things* (IoT) enables the interconnection between people and physical objects by providing a means of sharing information and coordination. Previously isolated physical objects, join communication networks with wired/wireless technologies such as Ethernet, Bluetooth, Wifi, Zigbee, *etc.* and enhance the functionality with the support of embedded sensors and actuators. IoT has made a significant impact on both domestic and industrial sectors with applications ranging from as simple as smart coffee machines to advanced smart manufacturing processes. It is designed for uplifting the quality of life, better resource management and intelligent decision making.

Popular recent applications in IoT include smart homes, smart buildings, intelligent transportation systems, industrial automation systems, smart healthcare, smart grids and smart cities. A smart home enables the automatic control of the heating, ventilation and air-conditioning (HVAC) systems, lighting control, occupancy-aware control systems (*e.g.* air quality controllers), appliance controls (*e.g.* doors, windows, electronics) and improved security systems that enhance the consumers quality of life. *Lufthansa*, a leading airline service provider, employs an IoT-based strategy to optimize on-time performance and operation by aggregating the real-time aircrafts, airports and weather sensor data which is a classic example of industrial IoT implementation.

Since the inception of the IoT in the supply chain management using RFIDs in 1999, IoT has achieved a significant growth in terms of the number of connected smart devices, underlying technologies and offered services. A recent IHS forecasting [1] suggests that the IoT installed base would reach approximately 30.7 billion by the end of 2020, which is approximately a 50% increase compared to the predicted installed base in 2017 as indicated in Figure 1. The advancement of technology and innovativeness in electronics manufacturing industry have caused the production costs of IoT devices to be reduced significantly. These two factors undoubtedly draw the attention of new investments due to the promising market potential, thus resulting in novice device manufacturers entering into the IoT device production.

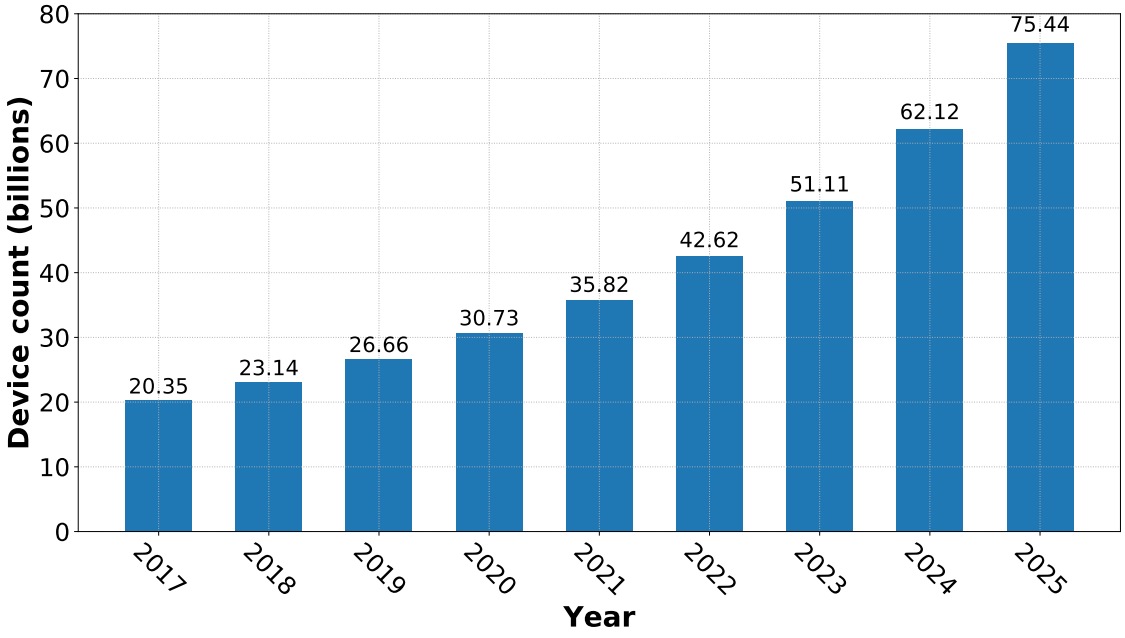


Figure 1: **Internet of Things installed base in global market in billions.** *Information for the figure was obtained from IHS White-paper on Internet of Things* [1].

1.1 Challenge of device identification

The rapidly increasing number of heterogeneous IoT devices joining the networks, imposes an additional burden on network administrators to manage the networks in terms of security, network resources and device troubleshooting unless they manage to identify the devices present in the network. That is the reason for device identity management to be considered as a key challenge in the Internet of Things [2, 3, 4, 5]. Hence identifying devices connected to a network (*device fingerprinting*) becomes an essential component in modern network management tools as it provides important information such as product vendor, product type, operating system and software version of a particular device.

The successful fingerprinting of a device enables,

- Network administrators (defensive purposes);
  - an inventory of the devices connected to the network
  - locate vulnerable, rogue devices or unauthorized devices and provide a proper security assessment of the network

- Adversaries (offensive purposes);
  - gain information about the device's characteristics (driver/firmware) that the attacker intends to attack

Defensive device fingerprinting is the base for vulnerability assessment for device models using popular vulnerability repositories, security enforcement of the IoT devices as in [6] and detection of device anomalies through continuous device behavior monitoring which is described in our future work. In offensive device fingerprinting, attacker gains more insights about the node which he intends to attack, thus the vulnerabilities are exposed precisely.

Device fingerprinting is the process of gathering device information in order to characterize it. In device fingerprinting the aim is to extract information about device's software, operating system or hardware components in order to create a signature which also called as a fingerprint. Fingerprinting can be divided into three main categories; Operation system fingerprinting, Host fingerprinting and Device fingerprinting. In our work we will be focusing on device fingerprinting in order to identify the IoT devices present in a network. The concept of fingerprinting was first introduced with Operating system (OS) fingerprinting tools, such as Network Mapper (Nmap) [7], Xprobe [8], Passive OS fingerprinting [9] and SinFP [10]. They analyze Internet Protocol (IP) packets to infer information related to the operating system versions used in the hosts, the type of hosts present in the network and the types of applications/ services used in hosts.

Device fingerprinting can be categorized into two main types; Active and Passive. In active device fingerprinting the *fingerprinter* which can be a measurer, an attacker or an adversary must be able to initiate a connection with the *fingerprintee* (the device which needs to be identified). The aim of this approach is to trigger a response from the *fingerprintee* which is unique to that particular device. For instance, Bratus et al. [11] measure the responses to a crafted set of IEEE 802.11 frames and Sieka [12] fingerprints the wireless access points by measuring the timings related to the authentication procedure for MAC layer communications. In passive device fingerprinting, the *fingerprinter* passively listens to the network traffic of the *fingerprintee* and extracts information to produce a unique representation for the device. For example, [6, 13, 14, 15, 16, 17, 18, 19] and [20] propose different passive device identification techniques. In certain occasions a combination of active and passive approaches are used where *fingerprinter* initially establishes a connection with the *fingerprintee* and continue to use the same connection over time to gather required

Active fingerprinting	**Passive fingerprinting**
Reveals in depth information about fingerprintee.	Limited information about a device compared to active counterpart.
Introduces monitoring traffic to the network.	Doesn't introduce any additional traffic to the network.
Needs cooperation from the fingerprintee.	Monitors and analyses the traffic.
Can be applied even when the devices are behind NAT or firewalls.	Doesn't need to establish a connection with fingerprintee.
Can be applied over a long period of time.	Undetectable to the fingerprinted device.

Table 1: **Comparison of Active and Passive device fingerprinting approaches.**

information about the device [17]. The selection of the device fingerprinting type depends on the depth of information required about a device as well as the accessibility of the device in the network. Table 1 compares the two main fingerprinting techniques in detail.

1.2 Thesis contribution

In our study we are focusing on identifying IoT devices connecting to a network in order to manage them properly. Despite the fact that active fingerprinting provides additional insights about the devices, need for cooperation from the *fingerprintee* limits its usage as devices might be reluctant to cooperate due to security concerns. Further, if an adversary device joins the network, identification technique should have the capability to identify the device and apply appropriate security policies rather than waiting for the unknown device to cooperate in identification. Since passive fingerprinting approach can be applied for any device connecting to the network, irrespective of the cooperation from device, it suits our application better than active fingerprinting approach. Further it keeps the network bandwidth utilization intact as no additional monitoring traffic is introduced to the network which is advantageous in large scale deployments. Hence we have selected to deploy a passive fingerprinting technique to identify IoT devices. Passive fingerprinting techniques use unique deviations of devices' clock skews, unique characteristics of USB hard-

ware, features extracted from protocol headers to generate a unique fingerprint for a device. A detailed analysis of techniques that use above measurements are discussed in Section 2.

We have performed a critical evaluation of existing device fingerprinting techniques and identified limitations of each technique. Further, we have analyzed important features which are missing in the existing studies to improve the performance of device fingerprinting. The proposed device fingerprinting approach extracts features from a sequence of packets (*sequenced-based features*) to generate a fingerprint that uniquely represent a device. The feature set includes summary statistics (minimum, maximum, mean, variance *etc.*) of feature types such as packet inter-arrival time, Ethernet packet size, IP header size *etc.* (detailed feature description is provided in Section 3, Table 4). Then the generated fingerprints are used to train a multi-class machine learning classifier which can distinguish different IoT devices. Our technique is a passive, payload independent, fast and scalable approach for device fingerprinting based on the features extracted from the Ethernet, IP and TCP/IP headers. As a result of using the information available in the protocol headers, our approach can extract features from encrypted traffic as well. But unlike the state-of-art feature-based device identification approach in [6], we have focused on extracting features from a sequence of packets during the initial setup of a device rather than extracting features from each individual packet. Furthermore, they have not included features which are known to be vital for traffic classification, such as the summary statistics and Fourier transform of packet inter-arrival times, the summary statistics of packet sizes, and the direction of the packet sequences. Our purpose was to utilize the above mentioned important sequence-based characteristics that are missing in the IoT Sentinel approach, to improve the device fingerprinting performance. Our system improves the average device prediction F1-score up to 0.912 which is a 14% increase compared with the state-of-the-art technique. Our technique doesn't require high computation power as in [16] and doesn't need any special protocols, hardware stacks or IP header options (SIP, USB, TCP timestmap option *etc.*) as in [14], [15] and [17], thus suited for a vast range of devices.

The successful identification of IoT devices present in a network enables the efficient management of the network resources and security policies. If a device's normal behavior can be modeled using a set of features, it can be used as a baseline to detect abnormal behavior of a device of the same type. The abnormality detection by comparing traffic patterns can reveal effects due to errors/faults of a device other than attacks, that are undetectable to the security tools in general. In that

case, accurately detecting the device type is critical as the selection of security policies depends upon it. Further, an incorrect device prediction not only hinders the performance of a device but also jeopardizes the security state of the network. Therefore, the main purpose of our research is precisely identify devices present in a network to support detection of abnormal behaviors of IoT devices. We suggest a feature-based approach to detect abnormal device behaviors through long-term behavior comparison.

In a summary, our contribution can be stated as follows.

- A critical comparison of existing device fingerprinting techniques and identify the limitations, improvements.
- A novel device fingerprinting approach using packet header related features that improves the average device classification over F_1-*score* the state-of-the-art solution [6] by 14%.
- Analysis of the impact of confidence threshold on fingerprinting unknown devices.
- Proposal for a feature-based anomaly detection technique by behavior comparison.

1.3 Roadmap

The rest of the document is organized as follows. Section 2 introduces the existing researches on device fingerprinting and provides a critical comparison of existing techniques. The proposed novel device fingerprinting approach technique is explained in Section 3 followed by the implementation procedure in Section 4. Section 5 evaluates the proposed system covering classification model selection, performance comparison with the state-of-the-art technique and analysis of the impact of confidence threshold on identifying unknown devices. Existing work on device anomaly detection and a proposal for anomaly detection using long-term traffic behavior comparison are explained in Section 6, followed by the our conclusion in Section 7.

2 State-of-the-art device fingerprinting techniques

Existing work on device fingerprinting has used both active and passive approaches. In this section we will be covering few notable fingerprinting techniques under each approach.

2.1 Active device fingerprinting

2.1.1 Analyzing responses to crafted MAC frames

An active fingerprinting approach which is based on identifying different responses of different Media Access Control (MAC) layer implementations to non-standard events. In [11] a series of crafted non-standard or malformed 802.11 MAC frames (stimulus) are sent to devices and the differences of responses are analyzed to classify a wireless device. By inspecting the values of MAC header 8 bit Frame Control (FC) bits; To DS, From DS, More Fragments, Retry, Power Management, More Data, Protected Frame and Order of response frames, a set of non-standard and unusual combinations of FC bits are derived, which are capable of distinguishing different device-types. Then a set of frames are crafted ensuring that they elicit aforementioned non-standard and unusual combinations of FC bits in the responses. Following are few scenarios (possible tests) identified in the study.

- Probe FC test - Test response to Probe requests with varying FC flags
- Authentication FC test - Test response to Authentication requests with varying FC flags
- Beacon test
- Association Request test
- Re-association Request test
- Action test

A series of tests covering all of the 8-bit FC flag combinations (256 in total) as responses is applied to a device using the crafted set of frames. Each of theses tests were run 5 times, thus the output for each test would be a value between 0 - 5 where

0 stands for no responses received with a particular FC bit combination and 5 for always receiving a response with that FC bit combination. All the responses are selected as a 256 dimensional feature vector. For n device types the feature vectors are generated k times and the feature vectors are concatenated in to a 256 * (n * k) dimensional matrix. Then using singular vector decomposition (SVD), this matrix is decomposed in a product of three matrices. After that using the first 2 components of the decomposition, each feature vector is projected in to a 3-dimensional space.

When an unknown device is being scanned, its responses are captured and the 256 dimensional feature vector is generated. Then the feature vector is projected in to the 3-dimension space using the pre-computed decomposition. After that similarity with the known signatures are computed calculating the cosine distance metric. These distance metrics are then compiled in to a decision tree structure to derive on the final prediction. The implementation consists of two main components; the scanning platform which prepares and send the stimuli and the monitoring platform which sniffs the response and reformat the responses to suit the tests in decision tree.

Decision tree structures are deployed due to its simplicity and better human readability. Nonetheless, in the proposed technique decision tree generation is a manual process, which would be a bottleneck of the system in the long run. Hence suitable measures needs to be taken to automatically generate the tree structure. Cosine distance metric however, claimed to be overly biased by the features with higher values irrespective of the number of features two vector share as explained by Li *et al.* [21]. Hence instead of using *'hard'* cosine distance metric, a similar vector space distance model could be used such as Weighted-cosine metric [21] which is unbiased or soft-cosine metric [22] which takes in to account the similarity of features. This technique has been evaluated with wireless access points, however when extending it for client stations the consent from the owners to participate in fingerprinting is required. Since this technique actively communicates with the fingerprintee, there is a possibility that the communication might be detected and blocked by intrusion detection systems. Furthermore, vulnerable devices might face functionality limitations (cease of services) due to the active testing.

2.1.2 Fingerprinting by timing analysis

The physical differences in devices affect the way they communicate with other entities due to the differences in network interface cards (NICs) board layout (paths,

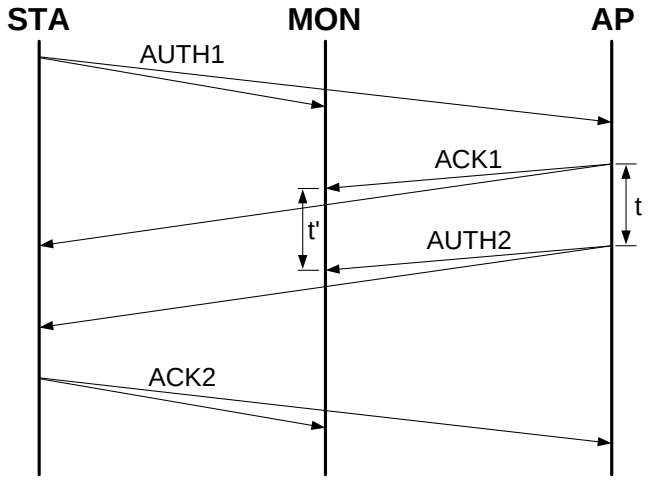


Figure 2: **Radio frames exchanges during 802.11 authentication procedure.** *STA, MON and AP refers to client station, Monitoring node and Access point respectively.*

dimensions, soldering, *etc.*), turn-on/off transient time variability of elements and differences in oscillators. Identifying and exploring these physical differences could enable the characterization of each unique device. Sieka [12] propose a technique that statistically analyzes the precise timing measurements related to radio communication and classify wireless devices using machine learning. Proposed technique explores the variations of timing measurements required for a device to perform various communication functions, specifically timings related to the authentication procedure for 802.11 MAC layer communications. The basic frame sequence of 802.11 MAC layer authentication procedure is shown in Figure 2.

This approach focuses on exploiting the timing difference between the transmissions of first acknowledgment (ACK1) and first Authentication response (AUTH2) from a device. This time lapse is indicated in the Figure 2 as t. Since t cannot be measured unless accessing the node being fingerprinted, t' is measured from the Monitoring node. This approximation is possible assuming that the timing difference of signal propagation between monitoring node and access point is negligible. STA initiates the authentication procedure by transmitting AUTH1 message and MON node keeps recording all the communication between the STA and AP including timestamps. The process is iterated multiple times to generate enough data for the statistical analysis. The mean and the difference of each sample with the mean (deviations) are calculated for set of samples. A fingerprint is generated by calculating a relative histogram of the deviations to uniquely represent a device and a classifier is built

using Support Vector Machines (SVM).

This approach was tested with a set of wireless access points and has achieved 86% prediction accuracy. Nevertheless, the test device set was extremely limited (5 to be exact), thus with the addition of more devices prediction accuracy might be reduced further. One major consideration under this approach is the validity of the assumption $t = t'$. We can assume this to be true, if the path between MON and AP remains same during the transmission of ACK1 and AUTH1. But in an environment where MON and AP are mobile the assumption might be invalid. Furthermore, the timing variability of MON due to the other processes running in MON, can falsify the assumption.

2.1.3 Use of association redirection

Four types of frames are transmitted between an Access point and a client station during the initial connection; authentication request, authentication response, association request and association response. Association redirection happens when the access point uses another address in the association response than the one mentioned in the authentication response frame. The approach explained in [23] exploits the association redirection to fingerprint clients connecting to a wireless access point. It has been noted that during an association redirection of a wireless access point, connecting clients behave differently to each other. In this technique, when the AP is responding to the association request it changes source address, BSSID address and both of them together in the association response to generate unique responses from the clients. An extension of the approach was implemented by applying the redirection on authentication responses from the access points which increased the detection of unique devices. The technique was able to fingerprint 9 different devices during the experiment with both authentication and association redirection mechanisms. A dysfunction of association could happen as a result of this modification in the association process, which could be a major drawback in this technique.

2.2 Passive device fingerprinting

2.2.1 Clock-skew deviation based fingerprinting

Basic concept behind this approach is to exploit the minute deviations in a device's clock skew to uniquely represent that device. Vern Paxon [24] defines clock skew as

"A clock's skew at a particular moment is the frequency difference (first derivative of its offset with respect to true time) between the clock and national standards". It is a result of minute deviations between the clock oscillators in different devices. Moon *et al.* [25] explains that the clock skews are relatively stable and constant for a particular device. This phenomena of variability in clock skews has been used with slight differences for fingerprinting devices in [17, 18, 19] and [20].

Transmission control protocol (TCP) includes a notion of the time of the device in its TCP flows as *Timestamp option* based on RFC 1323. In [17], TCP Timestamp option in the TCP header is used to derive the clock skew of a device and generate a fingerprint of the device. It has been noted that in all modern operating systems TCP Timestamp option is implemented which is a 32-bit timestamp generated by the origin. The article explains that the clock skews can be measured by comparing the timetstamps in the TCP header for two consecutive packets with timestamp values at the fingerprinter for the same two packets. Passive, active and semi-passive variants of this technique has shown promising results in fingerprinting devices when they are behind NAT/ firewall, independent of the distance or access technology. Nonetheless, scalable deployment of this approach is hindered by the necessity of having the TCP timestamp option enabled. Older devices which do not have the TCP timestamp option and the devices that have disabled it will not be able to fingerprinted with this approach. Furthermore, time consumption of fingerprinting process due to the deep packet inspection is significant.

Another clock-skew based device fingerprinting technique called 'GTID' has been introduced in [20] which passively captures traffic of wireless devices on a wired segment between the end destination and the access point. The process involves three main steps; feature extraction, signature generation and similarity measure. Packet inter-arrival time (IAT) is selected as the feature under this approach which represents the delay between two successive packets in the flow. Signature generation is focusing on identifying the distribution of the IAT vector due to the repetitive nature of network traffic. It is achieved by a binning process which uses N (empirically found to be 300) equally sized bins and measuring the number of IAT values falling in each of the bins (frequency). Once the signature generation is completed, a closeness measure is used to compare the similarity between two signatures using Artificial Neural Networks (ANNs). Generated frequency vector representing a device is input to the ANN and it compares the frequency vector with the master database, returning a closeness value between 0 - 1 where 1 is the perfect match. Proposed technique is capable of identifying device-types as well as unique devices.

Unlike most of the device fingerprinting approaches, GTID has the ability to identify previously unseen devices. Further it supports the IP level encrypted traffic and compatible with any IP enabled device. Similar to most of the packet timing based work, GTID also get affected by the buffering in switches and routers which limits its deployment when devices are connected through Internet. In addition, capturing traffic might be difficult sometimes due to restricted network access, as the traffic capturing needs to take place on the wired segment in between the AP and end destination. In terms of the machine learning mechanism used in this technique, ANNs require a greater computational resources [26], thus is not easily deployable at the edge networks. Apart from that, in order to achieve the optimum state learning with minimal error it could take a considerable duration which slows down the classification model generation process.

2.2.2 Packet inter-arrival time-based fingerprinting

In this technique, shifts in the inter-arrival times of a packet train caused by device's internal architectural differences are analyzed to create a unique fingerprint for a device. [13] and [27] analyze the differences of responses by different devices when they are subjected to a packet train, to fingerprint devices. These differences in responses are due to the heterogeneity of device components in chip-set, firmware and drivers. The technique introduced in [13], fingerprints wireless Access Points (APs) in four steps as shown in Figure 3. In the feature extraction phase, a packet train is sent through the access point and the egress traffic is captured where each individual packet is shifted in time. Then for the captured packet train, packet inter-arrival times (time delay between two consecutive packets) are calculated and an inter-arrival time (IAT) sequence is generated. Then in the next step, IAT sequence is subjected to wavelet transform that decomposes the IAT vector in to wavelet coefficients in order to generate a signature. The resultant set of coefficients is used as a master signature to represent the device-type and by repeating the same steps, a collection of master signatures are generated representing each AP type. When a signature of an unknown AP needs to be classified, it is compared with a set of available master signatures in the similarity measure step, by measuring the circular-cross correlation. The device-type with the highest circular-cross correlation is predicted as the type of the unknown device.

It has suggested that the prediction accuracy could be improved by splitting the packet transmission so that packet train is sent through a wire and received wire-

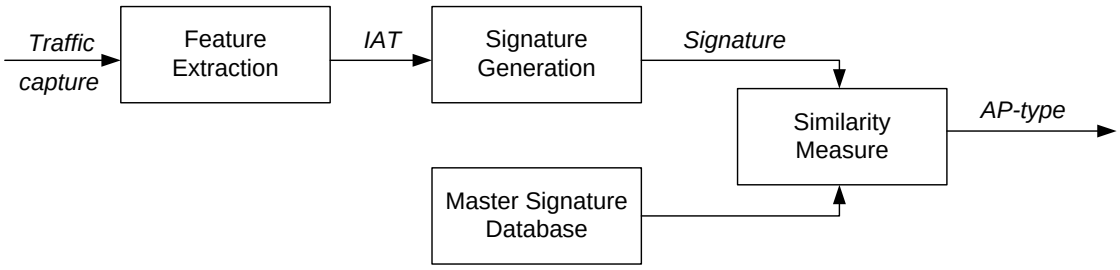


Figure 3: **Overview of wireless AP fingerprinting by analyzing packet inter-arrival times.**

lessly, reducing the delay introduced by accessing the wireless link twice. This passive technique has been evaluated using emulated packets for 6 different wireless APs with an accuracy reaching 100% with the increase of the number of packets. Even though this technique has shown impressive results, the size of the test device set is rather small. With more devices using live packets instead of emulated traffic the prediction correctness might get affected.

2.2.3 Host identification via USB fingerprinting

A novel approach of device fingerprinting is explained in [14] which exploits the unique characteristics of the USB hardware and software stacks due to its complexity, variability and ubiquity. These unique characteristics allow distinguishing between model identifiers, operating systems and state (real or virtual) of the devices. It has been noticed that the sequence of USB transactions and inter-arrival timing of commands get affected due to aforementioned characteristics. A set of computers are used as a test set and the USB traces for each device are captured for 15 seconds by connecting the computer to an USB protocol analyzer to capture high precision transactions. This approach can perform both operation systems fingerprinting and machine-type fingerprinting.

- Operating system fingerprinting
  Enumeration and *bulk IN-OUT transfer*, USB transactions have shown to be critically important in identifying the operating systems of devices. Enumerations happens at the start of a communication between two devices where the sequence of transactions depends on the operating system. After a device is configured, during the Bulk In-OUT transfer requests, the type and size of

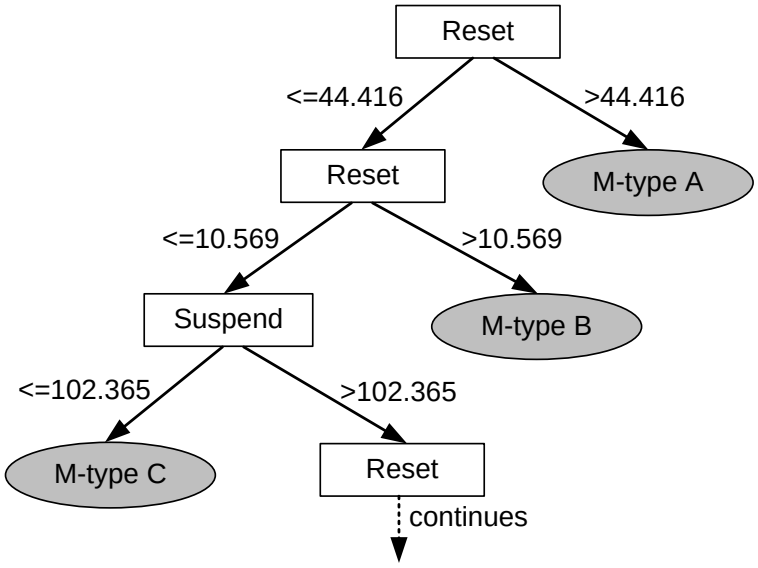


Figure 4: **Decision tree classifier generated from thumb drive characteristics.** *All the timings for Reset and Suspend operations are in milliseconds. Fingerprint characteristics of a machine is compared to intervals in the decision to derive on the machine-types as indicated by Elliptical nodes.*

requests are determined by the operating systems. By comparing the above two features operating system fingerprinting is achieved.

- Machine-type fingerprinting
  In uniquely identifying machine-types, this study has focused on analyzing time intervals between USB requests (inter-arrival time of commands). It has been discovered that for different machine-types the intervals fall in different ranges. In this study Suspend, Reset and Retry USB requests are analyzed to compute time intervals to fingerprint a machine-type. Since the comparison is not a simple one-to-one mapping as in operation system fingerprinting, a decision tree structure is used as shown in Figure 4.

Experiments have shown that this technique achieves 100% accuracy in OS fingerprinting and over 95% in machine-type fingerprinting. Despite the higher prediction accuracy, this technique has several limitations. In order to acquire high precision timing data, a costly and bulky USB protocol analyzer is required along with a physical connection to the fingerprintee, which might not be feasible at all times. Furthermore, the stability of the features selected have shown inconsistency over time, for an example Retry time intervals have shown a deviation after two weeks

during the evaluation.

2.2.4 Timing analysis of probe request frames

Probe request frames are used by wireless devices in 802.11 active scanning process, to search for wireless access points (APs) present in the wireless vicinity as soon as the initialization of wireless Network Interface Card (NIC) of the device. Even after connecting to an AP, probe request frames are continuously transferred between the AP and the device, but in a lesser frequency. Non-standard implementations of the active scanning process, has led different drivers to perform active scanning in slightly different techniques which creates a mean to distinguish different devices. As a result, even though all probe request frames are periodic, the periodic intervals vary between different NIC drivers as seen in Figure 2 of Franklin *et al.* [28], for a D-Link driver for the D-Link DWL-G520 and a CISCO driver for Aironet. This concept has been exploited for fingerprinting wireless drivers and devices under [16, 28, 29] and [30].

In [28], wireless driver fingerprinting happens in two steps; Trace capture and Fingerprint generation. In trace capture phase, time intervals between all successive probe request frames between the driver and an AP are captured. Then the sequence of values are binned using a binning approach to assign them to discrete bins. Two attributes related to probe rate are selected to fingerprint a wireless driver namely, bin frequency and the average time interval of the bin. First attribute represents the size of the bin and second the mean value of the bin. These two attributes creates a master signature for a wireless device. When an unknown wireless driver needs to be identified, it's signature is compared with the signatures in the master signature database, to determine the wireless driver type. Hardware abstraction layer implementations hinder the fingerprinting ability of this technique as it reduces the diversity of the wireless drivers. As an example, the MadWifi driver for Linux works well with most of the Atheros chipsets creating a more homogeneous environment.

Desmond *et al.* [16] explain that the probe request intervals are not only dependent on NIC drivers but also device-type and operating systems used. It uses the distribution of inter-burst latencies to fingerprint wireless devices. Even though this technique has a device prediction accuracy of 0.81, due to the slower rate of active scanning process, higher amount of time and lengthy traces are required. For an example, as per the evaluation results the average time required is approximately 1 hour to capture sufficient data to fingerprint a device and this could go up to 2

Type	**Feature**
Link layer protocol (2)	ARP, LLC
Network Layer Protocol (4)	IP, ICMP, ICMPv6, EAPoL
Transport layer protocol (2)	TCP, UDP
Application layer protocol (8)	HTTP, HTTPS, DHCP, BOOTP, SSDP, DNS, MDNS, NTP
IP options (2)	Padding, Router Alert
Packet content (2)	Size (int), Raw data
IP address (1)	Destination IP counter (int)
Port class (2)	Source (int) / Destination (int)

Table 2: **List of extracted features to uniquely represent a device.** *All features except the ones marked with "int" are binary features.*

hours for devices such as Intel NICs. In addition lossy nature of the wireless communication due to shadowing, interference *etc.* acts as a major barrier, causing delays and packet losses. Furthermore, timing measurements of both the above mentioned approaches are vulnerable to effects of temperature changes.

2.2.5 Fingerprinting with packet-based features

IoT Sentinel [6] uses differences of feature values present in the headers (Ethernet, IP, TCP/UDP) of packets originated from the device during its initial startup, to distinguish device-types. When a newly observed MAC address is detected, the system records the packets originated from the device, during the device setup phase. By analyzing the Ethernet, IP and Transport packet headers, 23 features are extracted per packet as indicated in Table 2.

The resultant set of feature vectors were stored in a 23 x `N` matrix `F`, where `N` is the number of packets processed. Consecutively identical packets were discarded when generating matrix `F`. Since `F` depends on the number of packets in the trace, a fixed-size fingerprint $\mathtt{F}_1$ was built. $\mathtt{F}_1$ was generated using the first 12 unique feature vectors from `F` resulting in a 276-dimensional fingerprint (12 vectors x 23 features). When a particular device did not have sufficient packets to generate a 276 dimensional fingerprint, it was padded with 0 values to make it consistent. This fixed-size fingerprint $\mathtt{F}_1$ was used to train the machine learning classification model.

One classifier per device-type is generated using the fixed-size fingerprints ($\mathtt{F}_1\mathtt{s}$)

based on Random Forest classification algorithm. When training a classifier for a device type, all collected fixed-size fingerprints were split into two classes where all the fingerprints for that particular device is considered as one class and a subset of fingerprints from all other device types as the other class. The ratio of the number of fingerprints of device class to other class was 1:10 and a discrimination threshold of 0.2 was used to determine the outcome of a classifier where if the probability of prediction is over 0.2 it was declared accepted and rejected otherwise. Random Forest classifier was implemented with 50 decision trees at a depth of 3, using `Scikit-learn` [31]. This classifier creation was repeated for all device types and a set of classifiers were generated. An unknown fingerprint was subjected to all the classifiers and the results were collected. If the unknown fingerprint is positively detected by only one classifier, the unknown fingerprint is considered to be one of that class. But if multiple classifiers show positive results another separation step was followed.

When more than one classifier accepts some unknown fingerprint, edit distance based metric was used to discriminate among them. In this case *Damerau-Levenshtein* edit distance was used to compute the edit distance. Edit distance was calculated between the unknown fingerprint `F` and five fingerprints for each device type it got a match for. The summed up distances per device-type were used to derive on a global dissimilarity score, where the lowest dissimilarity score gives the final prediction.

This passive approach is aimed at Internet of Things and suitable for devices communicating over WiFi or any channel supporting TCP/IP. Fingerprinting can be performed in a security gateway on the edge networks as it doesn't need high processing power or memory. In average it requires 21 packets to generate a fingerprint, thus is faster and scalable. Since it extracts features from header fields, prior knowledge of syntax or data fields are not required and encrypted traffic also can be used. The technique was evaluated using a set of 27 off-the-shelf IoT devices and achieved 0.815 global prediction accuracy. Nonetheless, the machine learning algorithm used involves time consuming edit distance calculations which could be simplified further by using a multi-class classifier rather than using 27 binary classifiers. In addition it doesn't consider important classification features related to distribution of packet sizes, inter-arrival times and Fast Fourier Transform of inter-arrival times.

	Gao *et al.* [13]	Bratus *et al.* [11]	Sieka *et al.* [12]	Letaw *et al.* [14]	Desmond *et al.* [16]	Franklin *et al.* [28]	Kohno *et al.* [17]	Uluagac *et al.* [20]	Miettinen *et al.* [6]	Francois *et al.* [15]
Active(A)/Passive(P)	P	A	A	P	P	P	P	P	P	P
Scalable	✓	×	-	-	-	-	-	✓	✓	✓
Wireless fingerprinting	✓	✓	✓	×	✓	✓	✓	×	✓	✓
Stability of features	✓	✓	-	×	×	×	✓	×	✓	✓
Feature type: Time(T)/ Header(H)	T	H	T	T	T	T	T	T	H	H
Identify device-type(T)/unique device(U)	T	T	U	T	U	U	U	T/U	T	T
Dependencies	×	✓	×	✓	×	×	✓	×	×	✓
Detection of unseen devices	×	×	-	-	-	-	-	✓	✓	✓
Need for special packets	×	✓	×	×	×	×	×	×	×	✓
Need for special equipment	×	×	×	✓	×	×	×	×	×	✓
Applicable to all devices	✓	✓	×	×	×	×	-	✓	✓	×
Evaluated with IoT devices	×	×	×	×	×	×	×	-	✓	×
Prediction accuracy	0.9	-	0.86	0.95	0.81	-	-	-	0.81	0.86
Size of test data set (number of devices)	6	-	5	5	3	-	-	-	27	26

Table 3: **Comparison of state-of-the-art device fingerprinting techniques.** *First column lists down the set of parameters to compare the performance.*

2.3 Summary

A comparison of aforementioned device fingerprinting technique can be seen in Table 3. Active fingerprinting technique proposed by Bratus *et al.* [11], is less scalable as with the increase of devices it will significantly utilize the useful bandwidth of the network. In contrast, passive approaches [6, 13, 20] and [15] are capable of scaling up with the addition of devices. As indicated in the Table 3, Letaw *et al.* [14] and Uluagac *et al.* [20] cannot be used to fingerprint wirelessly as they need a physical connectivity with the device being fingerprinted. Packet timing based fingerprinting techniques experience stability issues in features due to the errors caused by buffering in switches and routers and effects of temperature.

The fingerprinting techniques proposed in [12, 13, 14] and [16] exhibit accuracies higher than 0.8, however the number of devices used for the evaluation is fairly less. Even though Francois *et al.* [15] have used a set of 26 devices to validate their claim,

all the devices used were hardphones or SIP servers.

Among device fingerprinting techniques discussed in Section 2, despite active fingerprinting techniques have a high accuracy, usage of it is limited, because of the lack of cooperation of devices. Further, the ability to fingerprint devices without cooperation from devices should be a vital component of a fingerprinting technique as one of their main purposes is to identify rogue/unauthorized devices in the network. Hence we have considered on implementing a passive technique excluding the active techniques discuss prior and analyzed the suited technique among clock-skew based, packet inter-arrival time/ timing based or packet header information based techniques. Even though USB based fingerprinting [14] seems to be promising its implementation is hindered because of the necessity of having a USB stack and the need for costly, bulky equipment to capture high precision data.

Most of the passive techniques discussed in Section 2 are based on measuring the timing of packets. Higher time consumption, necessity of deep packet inspection, errors caused by buffering in switches and routers, limitations introduced by protocol alterations and effect of the temperature, however have raised complexities for the deployment of timing based fingerprinting techniques. Furthermore, these techniques have been evaluated by fingerprinting APs or computers, thus compatibility with off-the-shelf IoT remains to be unanswered. IoT sentinel [6], however has the ability to fingerprint a device passively and faster by analyzing the headers of a small set of packets. It can detect previously unseen devices and it can be applicable to all devices. More importantly it has been evaluated with off-the-shelf IoT devices. Even though the average prediction accuracy is comparatively lower then other work, it has considered a diversified set of devices whereas Sieka *et al.* [12], Gao *et al.* [13] and Letaw *et al.* [14] have achieved higher accuracies with a narrow test set for evaluation. Despite the fact that the average device prediction accuracy is 0.81, IoT Sentinel has a prediction accuracy of 100% for 10 device-types which is still greater than the test sets of aforementioned studies. Francois *et al.* [15] has achieved a prediction accuracy of 0.86 with a set of 26 devices which is at first seems to overtake IoT Sentinel. But the devices were limited to a specific category that uses SIP as a protocol, thus limiting the scalability. Hence compared to the state-of-the-art techniques, IoT Sentinel edges ahead of other work and we have identified its importance.

The IoT Sentinel is a lightweight, scalable passive fingerprinting approach. However, the prediction accuracy for a set of devices (10 device-types) under this technique

remains around 50% and there is a possibility of devices from same vendor to be confused with each other. In order to overcome these limitations of the existing techniques, We propose a device fingerprinting technique, which extracts features from headers of a sequence of packets, to generate a fingerprint for a device and use machine learning to classify the fingerprints.

3 Sequence-based device fingerprinting

3.1 Overview

We propose a device-type fingerprinting technique, based on extracting the important distinguishable features from the headers (Ethernet, IP, TCP/UDP) of packets during the initial communication of a device. Since different vendors request different information/ parameters from external entities during its initial communication phase, capturing the initial communication patterns become a perfect criteria for fingerprinting devices. Furthermore, in our approach we use a sequence, a set of packets ordered by timestamps to extract the features. A sequence of packets can be considered in two variants; bidirectional sequences (set of packets between a device and other external entities where device can be present as either source or destination) and source-originated sequences (set of packets originated from a device where device is always the source). In our approach we have considered features from both bidirectional and source-originated sequences. In the *training phase*, using the extracted features, a *fingerprint* (collection of features which represents a device) is generated for each device and a multi-class classifier is trained using machine learning algorithms as shown in Figure 5. When an unknown device needs to be identified in the *prediction phase*, a fingerprint is generated using the packet sequence during its initial setup with the same set of features as indicated in Figure 6. Then the generated fingerprinting is subjected to the classifier derived earlier and a list of probabilities are be predicted. The device-type related to the highest probability is assigned as the type of the test device.

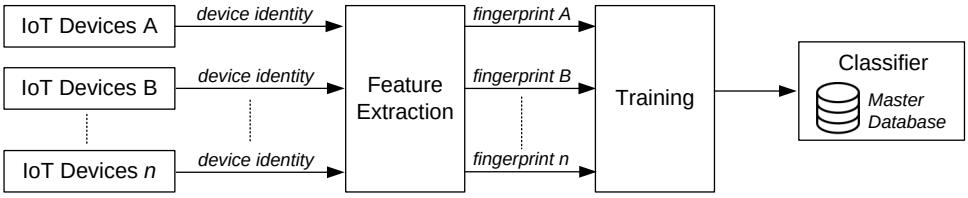


Figure 5: **Training phase - classification model generation.** *From each IoT device, its identity and packet sequence are inputs for feature extraction and generating a fingerprint. Then fingerprints of all devices are used in the training phase to train a classifier.*

From this point onwards we will be referring to the IoT Sentinel [6] fingerprinting approach as packet-based device fingerprinting approach. Our sequence-based de-

Figure 6: **Prediction phase - Device identification.** *Feature extraction generates a fingerprint with the packet sequence of the test device. Then it is subjected to the classifier which use a master database to provide a list of predictions. The device-type related to the highest probability (P) is predicted as the type of the test device.*

vice fingerprinting approach differs from that of the packet-based mainly in feature selection criteria. While packet-based approach focuses on extracting features from each individual packet separately, we have focused on using summary statistics and Fast Fourier Transform components of measurements as features. We identified that the summary statistics and Fast Fourier Transform components are vital in classification as suggested by Nguyen *et al.* [32]. In addition, we introduced separate feature sets based on the directionality of the traffic considering packets originated from source (source-originated) and packets between source and other entities (bidirectional). In contrast packet-based technique only employ packets originated from source to generate features. Further we have simplified the two-fold classification mechanism of packet-based technique which includes 27 binary classifiers and 5-7 tie breaking edit distance calculations in to a single step. We used a multi-class Random Forest classification algorithm which outputs a list of probabilities among which the device label with the highest probability is assigned as the predicted device-type. Reduction to a single step reduces the time required for classification significantly due to the omission of the edit distance calculation. Furthermore, we have employed F_1*-score* as the metric for the evaluation of the performance of sequence-based technique which considers the effect of False Positives and False Negatives compared to the accuracy metric used in packet-based approach.

3.2 Feature selection

In search for directions to define a feature set, we referred to articles related to feature-based device fingerprinting techniques as well as feature-based Internet traffic classification techniques to identify promising sequence-based features. IoT Sentinel [6] uses 23 features extracted from packet headers at the initial startup of a device as shown in Figure 7, with their importance to the classification. It can be

seen that the accuracy of device fingerprinting depends mainly on four features; *Packet_size, Destination_IP, Source_port* and *Destination_port*. A survey paper on feature-based Internet traffic classification by Nguyen *et al.* [32] summarizes a list of prominent work on Internet traffic classification. It suggests that summary statistics (*maximum, minimum, variance, quartiles, average etc.*) of traffic measurements (*packet size, packet inter-arrival time etc.*) can be used as features to represent a class. As an example following list of features are frequently used in Internet traffic classification as per the authors.

- Packet length statistics (minimum, maximum, quartiles)
- Packet inter-arrival statistics (mean, variance)
- TCP/UDP port numbers
- Fourier transform of packet inter-arrival times
- Total packet count

In addition, Moore *et al.* [33] proposes that different variants of packet length statistics such as *Ethernet packet size, IP packet size* and *control header size* have a significant impact on flow-classification. Further it highlights the importance of defining features based on the direction of traffic flows (bidirectional or unidirectional). Our feature selection was influenced by the findings of aforementioned work.

We extract a set of 90 features from Ethernet, IP and TCP/ UDP packet headers to generate a fingerprint for a device-type as shown in Table 4.

We selected minimum, maximum, first quartile, median, third quartile, mean, variance and inter-quartile range as the summary statistics to be calculated for each feature type as they were the most frequently used statistics in Internet traffic classification. Assume the length of the sequence to be L. We measured the inter-arrival times (delay between consecutive packets) for the L packets and computed the summary statistics as well as the 10 highest magnitudes of the Fast Fourier transform to generate 18 features. We have divided the packet size measurement into three subs categories as *Ethernet frame size*, *IP packet header size* and *IP payload size* to gain additional information about the behavior of each header size used during the initial communication of different devices. Then we have computed summary statistics for each of the above feature types (header sizes) in order to grasp the aspects related to distribution of each feature type. Total number of packets transmitted during a

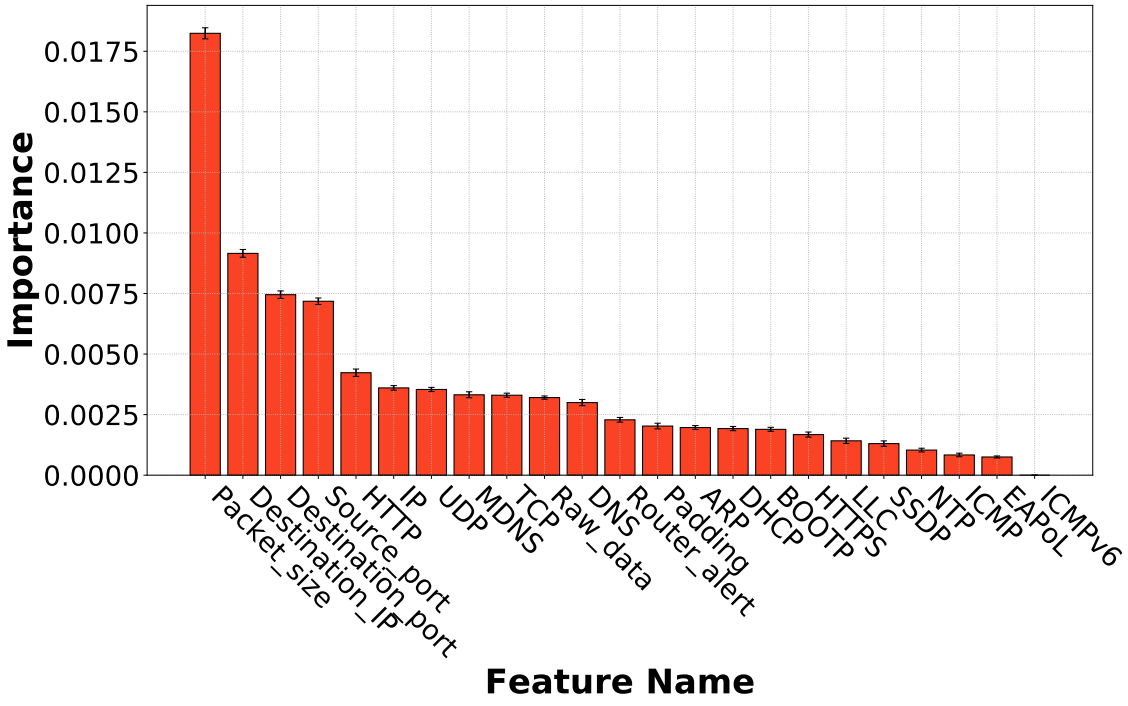


Figure 7: **Average feature importance values for packet-based device fingerprinting.** *Error bars showing the 95% confidence interval of the average importance values. The features are ordered in the decreasing order of importance and the results are obtained by performing a 10-fold cross validation for 10 iterations.*

device initiation is relatively higher compared to the normal operation of the device. Hence we introduced a *packet count* feature to measure the packet transmission during a defined time span. Furthermore, we also extract two variations of same feature set considering bidirectional sequences and source-originated sequences. For source-originated sequences we consider only packets having the source MAC address of the device being fingerprinted, while for the bidirectional sequences we consider packets where either the source or the destination has the MAC address of the device being fingerprinted. In order to represent the direction of each packet between the device and other entities during the startup, a new feature was defined. Each packet direction was represented by an integer as mentioned in Table 4 with respect to the device and the *packet direction* feature was created as a series of integers for the first L packets between a device and other entities. During the startup of a device, it communicates with various outside units to receive parameters. Thus we computed the *IP destinations* feature to denote the count and order of which the device communicates during its startup. A counter is incremented whenever the device communicates with a new IP address and the value of the counter is added

Feature ID	**Feature Type**	**Description**
0 - 7	Packet inter-arrival time (1) *Minimum, Maximum, Q1, Median, Mean, Q3, Variance, IQR*	Packet inter-arrival times for the bidirectional sequence
8 - 17	FFT frequency of inter-arrival times (2)	10 highest magnitudes of Fast Fourier transform of packet inter-arrival times for the bidirectional sequence
18 - 25	Ethernet packet size (3) *Minimum, Maximum, Q1, Median, Mean, Q3, Variance, IQR*	Size of Ethernet packets for the bidirectional sequence
26 - 33	IP payload size (4) *Minimum, Maximum, Q1, Median, Mean, Q3, Variance, IQR*	Size of IP payload for the bidirectional sequence
34 - 41	IP packet header size (5) *Minimum, Maximum, Q1, Median, Mean, Q3, Variance, IQR*	Size of the IP header or control bits for the bidirectional sequence
42	Packet count (6)	Total number of packets transmitted during the first 5 seconds
43	Packet direction (7)	Series of int values, each representing direction of a packet as, 0: Source -> destination 1: Destination -> source
44 - 86	Same above set of features (0-42) for packets originated from source	Filter packets from source and calculate the same set of features as above
87	IP destinations (8)	Number of different IP addresses the source establish connections
88	Source port class (9)	Series of values each representing the source/ destination port category as, A: *no port*, B: *well-known*, C: *registered*, D: *dynamic*
89	Destination port class (10)	

Table 4: **Feature set used in sequence-based device fingerprinting**

to a series of values for the first L packets from the source. Analyzing the TCP/ UDP ports of the startup traffic provide a vision of services/ protocols each device

uses to boot up. We assigned a value per packet depending on the port value as mentioned in Table 4 for the first L packets originated from source resulting in a series of values. Two features were created by inspecting the source and destination ports.

Feature types (1), (2), (3), (4), (5) and (6) are generated for bidirectional and source-originated sequences separately which results (2*(8+10+8+8+8+1)) = 86 features. Feature type (7) is generated only for bidirectional sequences while feature types (8), (9), and (10) are computed for source-originated sequences. The above (86+4) = 90 features are used to generate a fingerprint to uniquely represent a device-type.

A comparison of the feature extraction under packet-based and sequence-based approaches can be explained as below, considering the packet size parameter as an example. If X shows the list of packet sizes of the first n packets from a device.
$X = [x_1, x_2, x_3, x_4 ..., x_k, ..., x_n]$
Packet-based approach extracts feature from each individual packet for k packets ($k<n$), thus the resultant fingerprint would be,
$f_{pkt} = [x_1, x_2, x_3, x_4 ....., x_k]$
In contrast, in the sequence-based fingerprinting, summary statistics of the packet sizes of k packets are computed to create fingerprint. If first k packets are represented by X_k where $X_k = [x_1, x_2, x_3, x_4 ....., x_k]$, the resultant fingerprint would be,
f_{seq} = [maximum(X_k), minimum(X_k), mean(X_k), first quartile(X_k), median(X_k), third quartile(X_k), Variance(X_k), inter-quartile range(X_k)]

The fingerprint generated using the packet-based approach will be a subset of X, whereas under sequence-based approach it will contain summary statistics (min, max, mean etc.) of X.

When the setup traffic of a new device is captured, a fingerprint is generated following the above procedure. Then the fingerprint is subjected to a trained multi-class machine learning classifier which is discussed in Section 3.3.

3.3 Fingerprint classification

After extracting features and generating fingerprints, next step is to develop a fingerprinting algorithm that has the capability of identifying a fingerprint. Depending on the prior knowledge of device-types to the classification, fingerprinting algorithms can be divided into two categories: Supervised (White-listing) and unsupervised

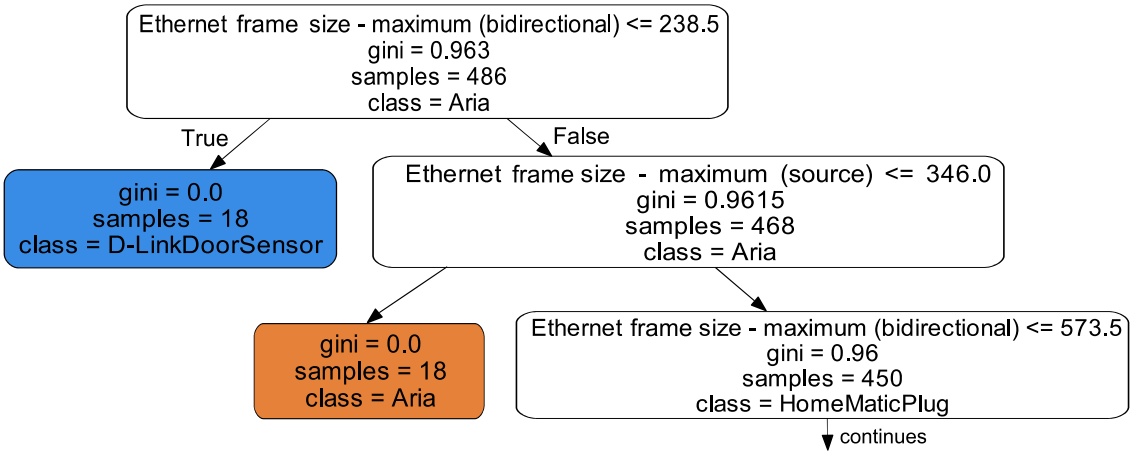


Figure 8: **A section of a decision tree from the Random Forest classifier starting from root node.** *Colored-boxes represent the end-nodes where a class label is assigned to each and transparent boxes represent decision-making nodes. First parameter in the decision-making node is the feature and the threshold. 'Gini' value denotes the impurity at each node and 'samples' represent the total number of training samples. The class name in the colored-boxes denotes the final predicted class.*

machine learning. In supervised learning approach knowledge of existing device-types should be known to the fingerprinting algorithm in advance and Unsupervised algorithms do not require prior knowledge. In our proposed approach a supervised machine learning technique was used to train a machine learning classification model with the extracted fingerprints. We selected `Random Forest` classification algorithm as the supervised learning model due to the higher accuracy of the results. The Random Forest classifier is an ensemble algorithm that uses a forest of decision trees and aggregates the decisions of each tree to derive on the final prediction. Aggregation of results reduce the effect of noise and over-fitting, thus provide accurate predictions. The basic parameters of the Random Forest algorithm are number of decision trees and the decision tree related parameters like maximum tree depth, minimum split, split criteria etc.

The first few branches of a decision tree with the Random Forest algorithm is presented in Figure 8, which indicates the process of decision making in the classifier (a complete decision tree can be seen in Figure 21 in Appendix 2). A decision tree is created by comparing the feature values based on *Gini impurity*. The *Gini impurity* is a measure of how often a randomly chosen fingerprint will be classified incorrectly if it is classified randomly based on the distribution of classes. The dataset is split

into training and testing set using k-fold cross validation mechanism and a classifier is generated using the training set with same number of items in each device class (balanced training set). When an *unknown* fingerprint is subjected to the classifier, it predicts a list of probabilities. Each probability value in the list represents the probability of the *unknown* fingerprint to be classified as a device-type the classifier is trained for. The device-type label with the highest probability is assigned as the predicted device type of the *unknown* fingerprint.

3.4 Summary

Our solution for device identification is based on capturing the initial communication behavior of a device during its setup and extracting the features related to a sequence of packets. A set of summary statistics and Fast Fourier transform components of sequence-based feature types such as Ethernet frame size, packet inter-arrival times introduces the notion of distribution of feature values to distinguish different device-types. In addition we consider the directionality of sequences as bidirectional and source-originated to define two sets of same feature types resulting in 90 features in total. Using the extracted features, a fingerprint is generated for each device and a multi-class classification model is derived using Random Forest classification model that has the ability to distinguish different devices based on feature values.

4 Methodology

4.1 Dataset

The fingerprinting technique was evaluated using the same dataset that was used by the authors of IoT Sentinel [6]. The dataset contains traffic traces for 27 different off-the-shelf IoT devices related to smart lighting, health monitoring, home automation, security cameras and household appliances. The list of devices along with the device model information and supported connectivity technologies can be seen in Table 5.

Identifier	**Device Model**	**WiFi**	**ZigBee**	**Ethernet**	**Z-Wave**	**Other**
Aria	Fitbit Aria WiFi-enabled scale	◉	○	○	○	○
HomeMaticPlug	Homematic pluggable switch HMIP-PS	○	○	○	○	◉
Withings	Withings Wireless Scale WS-30	◉	○	○	○	○
MAXGateway	MAX! Cube LAN Gateway for MAX! Home automation sensors	○	○	◉	○	◉
HueBridge	Philips Hue Bridge model 3241312018	○	◉	◉	○	○
HueSwitch	Philips Hue Light Switch PTM 215Z	○	◉	○	○	○
EdnetGateway	Ednet.living Starter kit power Gateway	◉	○	○	○	◉
EdnetCam	Ednet Wireless indoor IP camera Cube	◉	○	◉	○	○
EdimaxCam	Edimax IC-3115W Smart HD WiFi Network Camera	◉	○	◉	○	○
Lightify	Osram Lightify Gateway	◉	◉	○	○	○
WeMoInsightSwitch	WeMo Insight Switch model F7C029de	◉	○	○	○	○
WeMoLink	WeMo Link Lighting Bridge model F7C031vf	◉	◉	○	○	○
WeMoSwitch	WeMo Switch model F7C027de	◉	○	○	○	○
D-LinkHomeHub	D-Link Connected Home Hub DCH-G020	◉	○	◉	◉	○
D-LinkDoorSensor	D-Link Door & Window sensor	○	○	○	◉	○
D-LinkDayCam	D-Link WiFi Day Camera DCS-930L	◉	○	◉	○	○
D-LinkCam	D-Link HD IP Camera DCH-935L	◉	○	○	○	○
D-LinkSwitch	D-Link Smart plug DSP-W215	◉	○	○	○	○
D-LinkWaterSensor	D-Link Water sensor DCH-S160	◉	○	○	○	○
D-LinkSiren	D-Link Siren DCH-S220	◉	○	○	○	○
D-LinkSensor	D-Link WiFi Motion sensor DCH-S150	◉	○	○	○	○
TP-LinkPlugHS110	TP-Link WiFi Smart plug HS110	◉	○	○	○	○
TP-LinkPlugHS100	TP-Link WiFi Smart plug HS100	◉	○	○	○	○
EdimaxPlug1101W	Edimax SP-1101W Smart Plug Switch	◉	○	○	○	○
EdimaxPlug2101W	Edimax SP-2101W Smart Plug Switch	◉	○	○	○	○
SmarterCoffee	Smarter SmarterCoffee coffee machine SMC10-EU	◉	○	○	○	○
iKettle2	Smarter iKettle 2.0 water kettle SMK20-EU	◉	○	○	○	○

Table 5: **Description of devices used in evaluating our fingerprinting technique.** *'Identifiers' are used to address each device-type throughout the thesis.*

The dataset consists of the traffic traces of devices during their setup procedure that includes activating the device, connecting to the device directly over WiFi

or Ethernet and transmitting the WiFi credentials. Most of the devices have been connected to the network using WiFi or Ethernet, but few devices used IoT protocols like ZigBee or Z-Wave to indirectly connect to network through WiFi or Ethernet hubs. For those devices, indirect traffic of the hub has been captured which acts as a gateway. A WiFi access point (AP) has been emulated with a laptop using `hostapd` to create WiFi interfaces of the AP and external Ethernet port was attached to the laptop to represent the Ethernet port of the AP. The packet capture module has been implemented using `tcpdump`. The setup process has been facilitated by a smartphone application or a PC application following a guided list of instructions. After each capture, the device has been hard reset to revert it back to factory settings. The setup procedure has been repeated for 20 times, to generate sufficient amount (20) of fingerprints for each device which accounts to 540 fingerprints in total. The average capture duration of traces was 68 seconds and average packet length was 350 packets.

4.2 Feature extraction

Packet-based fingerprinting technique extracts 23 features from each individual packet and constructs a feature vector. Then it generates a 176 dimensional fingerprint for each device using the first 12 unique feature vectors. In order to generate a fingerprint, packet-based fingerprinting technique used 21 packets in average. In sequence-based fingerprinting approach, we evaluated the performance of the system with sequence lengths (number of packets) in the range of 10 to 23. By analyzing the results we discovered that the best performance occurs when the sequence length is 21. Hence we selected 21 packets as the sequence length and then for generating these sequences, we included all packets intercepted till we obtained 21 packets originating from the device being fingerprinted. The packets were filtered using the device's MAC address to create the two sequences with 21 packets each. Once the filtered sequence is complete, Scapy[1] was used to extract features as described in Section 3.2 to produces a fingerprint. For each of the capture files, aforementioned feature extraction was performed and a fingerprint was generated each consisting of 90 features.

[1]Scapy - Interactive packet manipulation program http://www.secdev.org/projects/scapy/

4.3 Device classification

Generated fingerprints were used to train a multi-class classifier using Random Forest (RF) classification algorithm implemented with `Scikit-learn` [31] libraries. We used a Random Forest multi-class classifier with 50 decision trees where final predicted class is the mode of the predicted classes under all decision trees. Classification was performed with a 10-fold cross validation for 10 iterations to generalize the results. In addition we implemented the system proposed by IoT Sentinel for the purpose of comparison and evaluation.

4.4 Metrics

4.4.1 Precision

Precision is a measure of relevant instances among retrieved instances, which is also called as positive predictive value. It represents the probability that a randomly selected retrieved instance is actually relevant or the probability of it being a false instance. Precision can be mathematically represented as below.

$$Precision = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

We use Precision to measure the exactness of the predictions and the fraction of False Positives of each device-type.

4.4.2 Recall

Recall is the referred to as the True Positive rate or sensitivity. It shows the fraction of relevant instances that have been retrieved over the total relevant instances of a test or the probability of a randomly selected relevant instance is retrieved in search. We use Recall to measure the completeness of the prediction for each device-type. Mathematical representation of Recall is as follows.

$$Recall = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

4.4.3 *F_1-score*

F_β-score is a measure of effectiveness of retrieval, giving β times importance to Recall as Precision. The mathematical equation for F_β-score score is shown below.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

F_1-score is a variant of F_β-score score, attaching an equal weight to Precision and Recall, thus is balanced. Other commonly used F measures include *F_2-score* and *$F_{0.5}$-score* which weighs Recall higher than Precision and Recall lower than Precision respectively. We use the *F_1-score* as the performance metric to measure the accuracy of prediction for each device-type. The *F_1-score* is the harmonic mean of Precision and Recall, and it reaches its maximum value when both Precision and Recall reach their respective maximums. The mathematical representation of *F_1-score* is as follows.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

It summarizes the exactness and completeness of prediction by taking into account the impact of False Positives and False Negatives, and compared with the Accuracy metric, the *F_1-score* edges ahead in terms of the ability to distinguish between False Positives and False Negatives. In the rest of the paper, all figures showing the *F_1-score* include error bars showing the 95% confidence intervals, and these results were obtained by performing a 10-fold cross validation over 10 iterations.

4.4.4 Zero-one loss

In order to measure the performance of a classifier in terms of the degree of accurate classifications across all device-types we use Zero-one loss as the metric. Zero-one loss is a standard loss function used in classification and decision theory which evaluates the overall performance of a system. It assigns 0 to loss for a correct classification and 1 for an incorrect classification. Loss value for each prediction is accumulated and an average of the accumulated loss values is used as the to denote the Zero-one loss for a classifier in our case.

5 Performance evaluation

5.1 Selection of a classifier

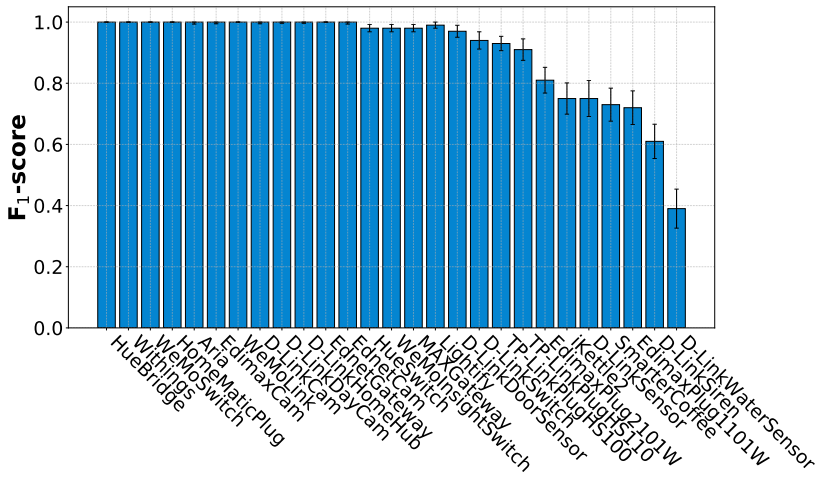
We first analyze the impact of the selected classification algorithm by comparing the average *F_1-score* among well established classification algorithms using the `scikit-learn` [31] library. Minimum, maximum and average *F_1-score* under each classification algorithm are shown in Table 6 and the classification was performed with a 10-fold cross-validation over 10 iterations to generalize the results.

Classification algorithm	***F_1-score***		
	Minimum	*Maximum*	*Average (C.I 95%)*
Random Forest	0.8199	0.9803	0.9121 (±0.0063)
Adaptive Boosting (Random Forest)	0.8161	0.9803	0.8980 (±0.0072)
Adaptive Boosting (Decision Tree)	0.7210	0.9432	0.8632 (±0.0081)
Decision Tree	0.7580	0.9605	0.8596 (±0.0085)
Linear SVM	0.7161	0.9210	0.8057 (±0.0090)
Multilayer perceptron (MLP)	0.6432	0.8938	0.7863 (±0.0092)
K-neighbors	0.6716	0.8914	0.7863 (±0.0081)
Gaussian Naive-bayes	0.6117	0.8395	0.7350 (±0.0088)
Radial Basis Function SVM	0.5429	0.8073	0.6873 (±0.0111)
Quadratic Discriminant Analysis	0.3557	0.6681	0.5210 (±0.0115)

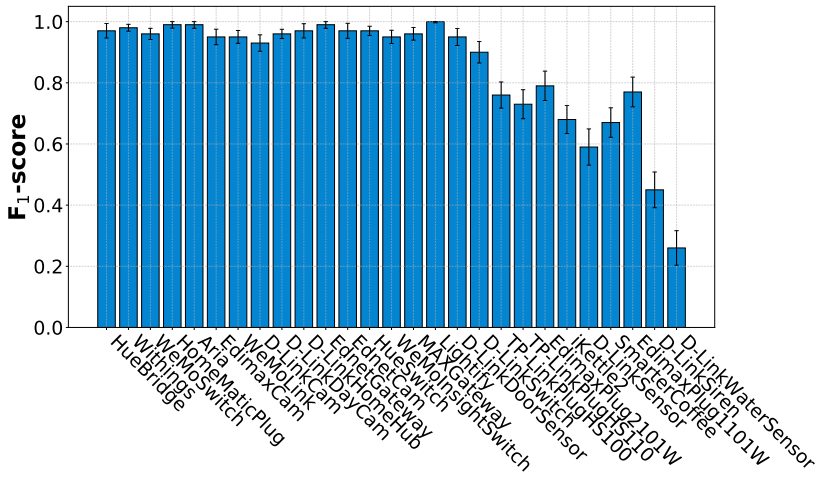
Table 6: **The performance comparison of classification algorithms.** *The minimum, maximum and average F_1-score with 95% confidence intervals are presented. The list of classifiers are ordered with the decreasing order of average F_1-score.*

Based on the results, it is observed that the Random Forest Classification exhibits the highest average prediction *F_1-score* with the least deviation. Even though Adaptive Boosting classifier in conjunction with Random Forest and decision tree classifiers classify devices faster than Random Forest, latter edges in terms of the prediction *F_1-score*. Hence we chose Random Forest as the classification algorithm to be used in the proposed sequence-based device-type fingerprinting technique to train the classification model and predict device-types.

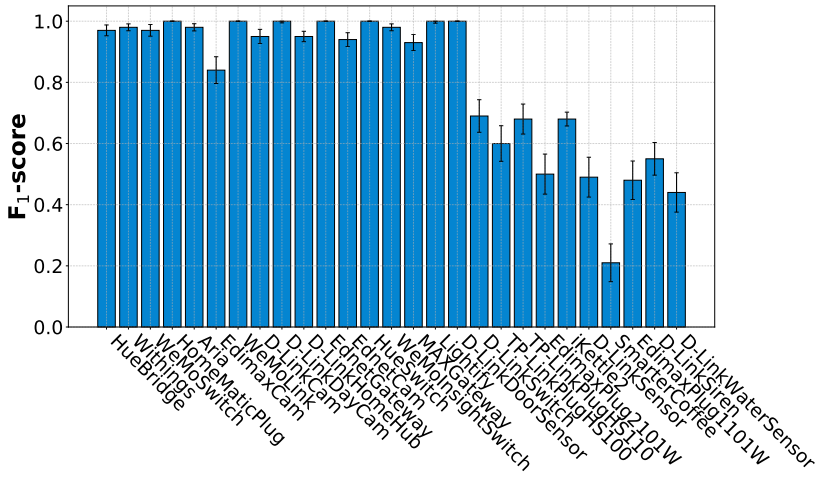
Further, Figure 9 shows the prediction *F_1-score* for each device-type, for classification algorithms Random Forest which was used in [6], Decision Tree in [14] and



(a) Random Forest



(b) Decision Tree



(c) Linear Support Vector Machine

Figure 9: **Comparison of classification algorithms.** *These 3 classification algorithms are selected as they have been used by existing device fingerprinting technique.*

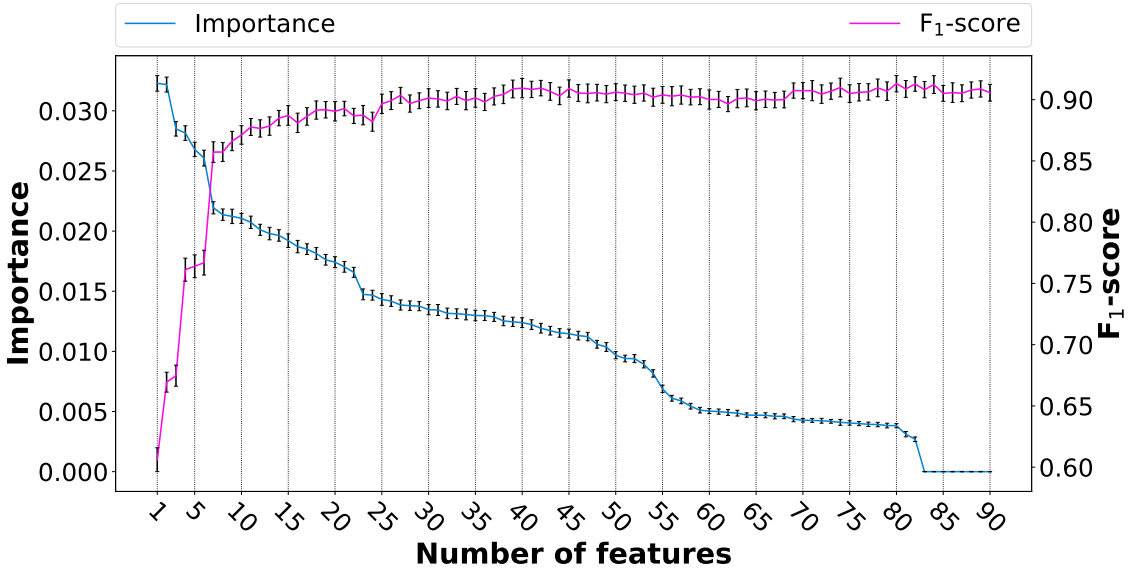


Figure 10: **Feature importance and *F1-score* variation with the number of features used to train the classifier.** *The error bars shows the 95% CI for the mean feature importance values and F_1-score. The results are obtained by performing a 10-fold cross validation for 10 iterations.*

Linear SVM in [12, 15] using the sequence-based featured introduced in our approach.

5.2 Analysis of feature importance

In the Random Forest algorithm, the importance of a feature in classification is measured as the decrease in Gini impurity (a measure of how often a randomly chosen fingerprint will be classified incorrectly if it is classified randomly based on the distribution of classes) averaged over whole set of trees. The average importance value of each feature used in the Random Forest classification under sequence-based device prediction technique are indicated in Figure 10.

The 10 most important features in packet-based and sequence-based approaches are listed down in Table 7. All of the top 10 important features in sequence-based fingerprinting technique are for sequences originated from source. In addition, among the top 10 important features, all of them are packet size related features and 80% features are for the source-originated sequences and rest for the bidirectional sequences. Apart from packet size related features, *source/destination port class* and *packet direction* features have an impact on prediction accuracy approximately around 0.18

	Packet-based	**Sequence-based**
1	Packet size	IP payload size - maximum (S)
2	Destination IP counter	Ethernet frame size - maximum (S)
3	Destination port class	Ethernet frame size - variance (S)
4	Source port class	IP payload size - variance (S)
5	HTTP feature	Ethernet frame size - median (S)
6	IP feature	IP payload size - maximum (B)
7	UDP feature	Ethernet frame size - maximum (B)
8	MDNS feature	Ethernet frame size - first quartile (S)
9	TCP feature	Ethernet frame size - mean (S)
10	Raw data feature	IP payload size - mean (S)

Table 7: **Top 5 important features in classification for packet-based and sequence-based approaches.** *Features are ordered in the decreasing order of importance for each approach. Sequence-based features marked with (S) are for source-originated sequences and (B) are for bidirectional sequences.*

and 0.14 respectively, which are among the top 25 influential features. As it is visible in the Figure 10, there are 8 features having an importance value of 0. These set of features are identified as the summary statistics (minimum, first quartile, median and third quartile) related to *IP packet header size* feature. Regardless, we decided to keep these features in the feature set as they might be useful when scaling the classification model with additional devices.

The average prediction F_1*-score* depends on the number of features selected to train the classifier. This scenario was analyzed by changing the number of features used to train the classifier and then recording the average F_1*-score* with each number of features. Figure 10 plots the change to the F_1*-score* against the number of features used by the classification algorithm to generate the classification model (Figure 22 in the Appendix 3 shows the variation of Precision and Recall with the number of features). It is noticed that there are few significant rises of the F_1*-score* values with the number of features added to the classification model specially at points 2, 4, 7, 9 and 25. The reason for those significant rises are the addition of features with high importance. The importance value of features after first 40, are well below 0.1, resulting in lower impact for differentiating device-types. Thus the first 40 features the F_1*-score* rises over 0.90 and the addition of features until 90, increases the accuracy from 0.90 to 0.92 gradually without any sharp rises. Hence in situations

Aria		HueBridge		EdnetGateway	
DHCP option	bytes	DHCP option	bytes	DHCP option	bytes
53	3	53	3	53	3
61	9	50	6	12	11
12	16	57	4	57	4
55	10	55	9	55	6
255	1	60	14	255	1
		12	13		
		255	1		
Padding	21	Padding	80	Padding	43
Total	300	*Total*	370	*Total*	308

Table 8: **Inspection of the different DHCP options and padding parameters for `Aria`, `HueBridge` and `EdnetGateway`.** *DHCP options are represented with the standard DHCP option IDs and the size of each option in bytes.*

of resource constraint IoT environments deploying a lesser number of features could be beneficial.

As *IP payload size - maximum* is the most important feature in sequence-based device fingerprinting, we analyzed the maximum sized IP payloads of each device. We noticed that among them, 56% of the packets are Dynamic Host Configuration Protocol (DHCP), 22% multicast Domain Name System (mDNS) and 15% Transmission Control Protocol (TCP). Further inspection of DHCP packets indicated that the cause for the difference of IP payload sizes among these DHCP packets is the DHCP header size, due to different DHCP options and padding values used in the communication. For an example DHCP header sizes for `Aria` consists of 300 bytes, for `EdnetGateway` it is 308 bytes and for `HueBridge` it is 370 bytes, thus differ the IP payload size maximums as a result.

A through inspection of the DHCP header fields, specifically DHCP options and padding fields for the aforementioned 3 device-types are shown in Table 8. As per the table, it is clear that the sizes of different DHCP header options vary and devices use various amount of padding which finally results in different packet sizes.

At certain occasions *maximum IP payload size* lies close to each other for different devices. In the case of `D-linkSiren` and `D-LinkSensor`, *maximum IP payload sizes* are 641 bytes and 640 bytes respectively, thus using just the *maximum IP payload size* as a feature is not sufficient. Summary statistics of a parameter provide addi-

tional insights based on the distribution of the parameter values. These information add value to the classification model providing means of distinguishing devices. For an example, using *Variance* of IP payload sizes along with the *maximum IP payload size* for a sequence of packets as features in classification, reduce the overlaps of devices as seen in Figure 14a. This case clarifies the importance of using a set of summary statistics of parameters under sequence-based device fingerprinting approach in order to discriminate different devices.

5.3 Comparison to the state-of-the-art

In order to compare the performance of the sequence-based device fingerprinting technique with state-of-the-art packet-based device fingerprinting (IoT Sentinel) approach we implemented the device prediction algorithm used by the packet-based approach. We managed to achieve similar results in comparison to the original implementation. IoT Sentinel has represented its results as accuracy of predictions for each device-type. Hence we measured the same metric under our implementation of the IoT Sentinel which will be referred to as packet-based device fingerprinting approach in order to compare the performance. The difference of average prediction accuracy of our implementation of packet-based approach, compared with the original implementation was 3.2%. Further for the 10 devices with the worst prediction accuracies under original version, the difference was 3.7% with our approach.

In comparing the performance of the proposed sequence-based device fingerprinting approach with the packet-based counterpart, we decided to employ F_1-*score* as the main and *Precision*, *Recall* as supporting performance metrics as described in Section 4.4.

A comparison between the packet-based and sequence-based device prediction F_1-*score* for 27 devices is shown in Figure 11. All devices except `D-LinkWaterSensor` have F_1-*score* over 0.60, which 18 of them are in fact over 0.95. Average F_1-*score* for our technique is 0.912 which is a 14% increase in the average F_1-*score* in comparison with the packet-based technique using the same number of packets from each device to extract the features. In packet-based approach 10 devices were identified with an average F_1-*score* of 0.44. Nevertheless in the sequence-based approach for the same 10 devices the average F_1-*score* is 0.78 and all the 10 devices have higher F_1-*score* compared with the respective scores under packet-based approach. Among the aforementioned 10 devices, `SmarterCoffee` exhibits the highest gain in terms of the F_1-*score* compared to the F_1-*score* under packet-based approach which is 0.60

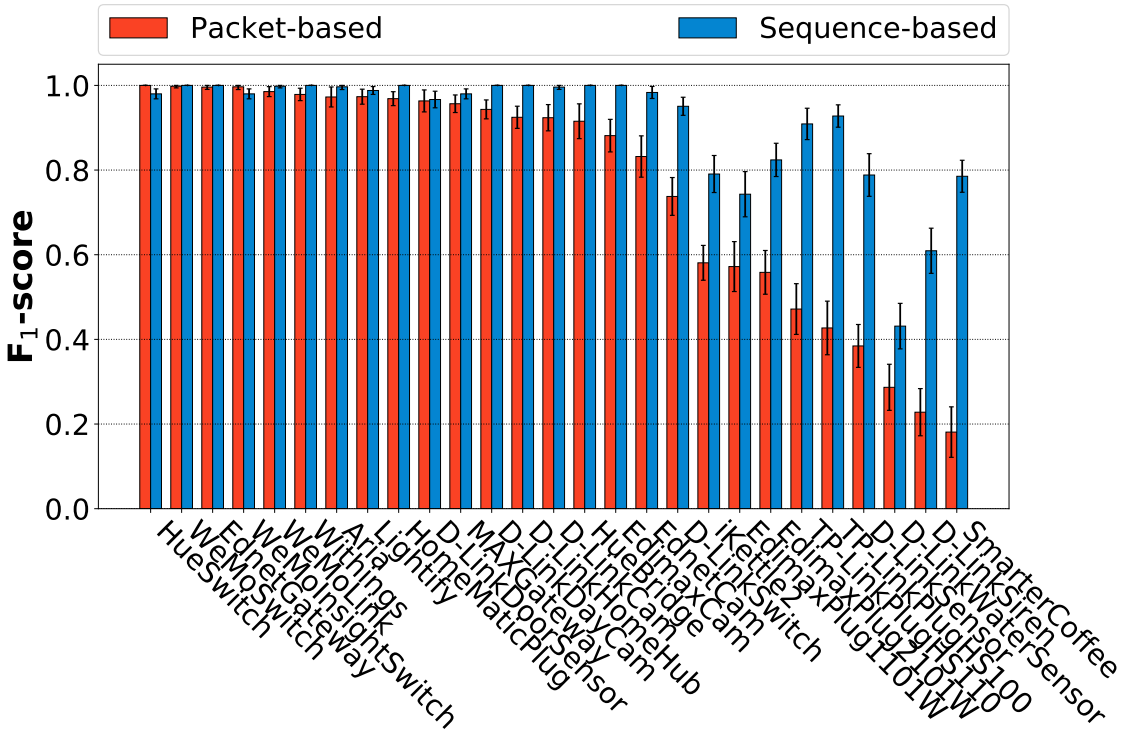


Figure 11: **Packet-based vs sequence-based average device prediction *F₁-score*.** *The Error bars show the 95% confidence interval of the average F_1-score. The devices are ordered in the decreasing order of F_1-score values based on the packet-based prediction results. The results are obtained by performing a 10-fold cross validation for 10 iterations.*

and a detailed description of *F_1-score* and percentage gain are listed in Table 9.

Furthermore, comparison of Precision and Recall of device prediction under packet-based and sequence-based fingerprinting approaches can be seen in Figure 12 and Figure 13 respectively. As per the two figures, in sequence-based approach, for all devices precision and recall remain approximately similar while in packet-based approach some devices exhibit a vast difference. `ikettle2` as an example, has precision and recall values as 0.467 and 0.80 which are significantly differ to each other resulting in a *F_1-score* of 0.581 with the packet-based approach. Global average Precision under the sequence-based approach is 0.925 compared to the 0.778 with packet-based approach which results in a 14.5% enhancement to the average Precision. Similarly, global average Recall gained a 13.5% increase from 0.781 to 0.918 with the sequence-based device fingerprinting approach. Altogether 20 devices has shown over 0.90 in average Precision, Recall and *F_1-score* with our approach where only `D-LinkWaterSensor` exhibit values less than 0.60. Nevertheless Precision and

Device Name	***F₁-score***		**Gain (%)**
	Packet-based	*Sequence-based*	
SmarterCoffee	0.1810	0.7853	60.4
TP-LinkPlugHS100	0.4270	0.9277	50.1
TP-LinkPlugHS110	0.4717	0.9090	43.7
D-LinkSensor	0.3845	0.7883	40.4
D-LinkSiren	0.2280	0.6093	38.1
EdimaxPlug2101W	0.5583	0.8240	26.6
D-LinkSwitch	0.7377	0.9507	21.3
iKettle2	0.5809	0.7907	21.0
EdimaxPlug1101W	0.5720	0.7430	17.1
D-LinkWaterSensor	0.2868	0.4313	14.5
Average *F₁-score*	**0.4428**	**0.7759**	**33.3**

Table 9: **Comparison of *F₁-score* of packet-based and sequence-based prediction for the 10 devices with lowest prediction accuracies with the packet-based approach.** *Gain of prediction is shown as a percentage with the sequence-based compared to packet-based approach. Devices are ordered in the decreasing order of the prediction F_1-score gain.*

Recall might not be the best metric to evaluate the performance of the system in general, but has its advantages in special occasions where the priority is to detect all relevant instances of a device-type or limited to identify a device-type accurately.

A timing analysis was performed to determine the amount of time required to complete each phase in device fingerprinting in sequence-based approach. As seen in Table 10 our technique is able to capture the required amount of packets, extract the features, generate the fingerprint and classify the device under 13.5 seconds in average. At best the identification could happen in 5.33 seconds and 82.63 at worst. Capturing packets required to generate the features takes at least 5 seconds, which is 94% of the time it takes to classify a device, while feature extraction and classification requires only 352 ms. All these timings are for capturing the full set of 90 features and when we reduce the size of feature set depending on the constraints of IoT devices the results could be generated in a lesser time.

Comparison of the impact of most important features in packet-based and sequence-based approaches on distinguishing different devices is shown in Figure 14. A set of 5 devices were selected to be analyzed based on the following criteria.

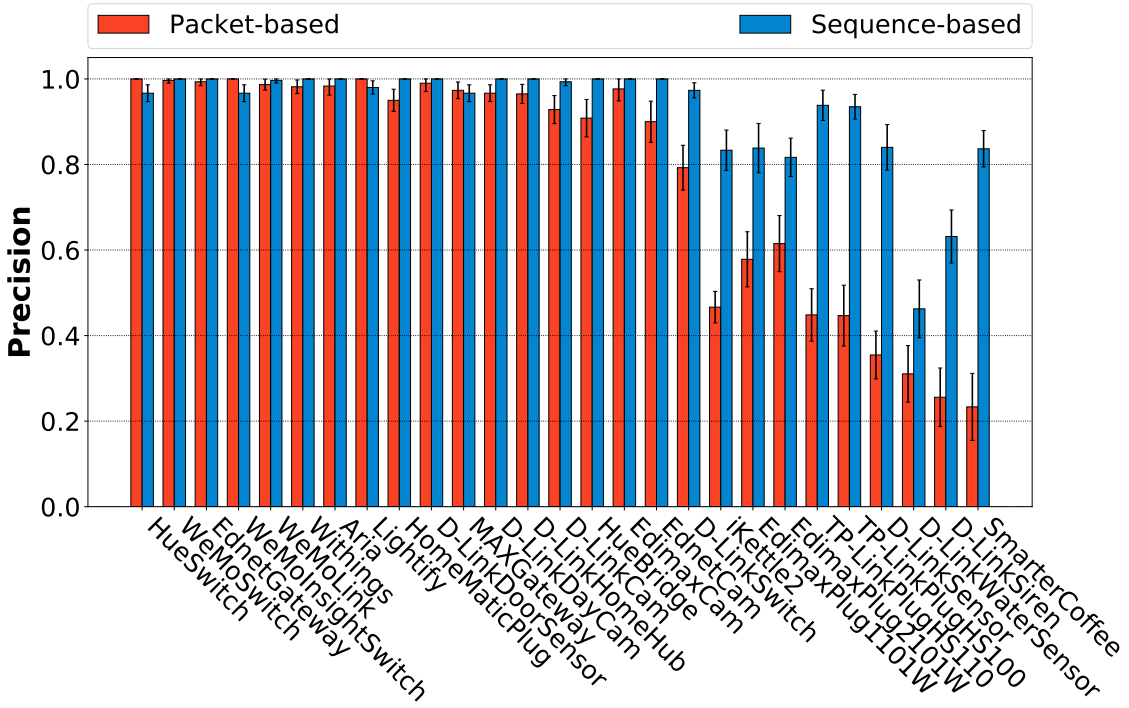


Figure 12: **Packet-based vs sequence-based average device prediction Precision.** *Error bars show the 95% confidence interval of the average Precision. The devices are ordered in the decreasing order of F1-score values based on the packet-based prediction results. The results are obtained by performing a 10-fold cross validation for 10 iterations.*

	Time (seconds)		
	Minimum	*Maximum*	*Average (95% C.I)*
Packet capture	5.0000	82.2475	12.7205 (±7.371)
Fingerprint extraction	0.3271	0.3779	0.3522 (±0.010)
Classification	0.0001	0.0004	0.0002 (±0.0001)
Total	**5.3272**	**82.6258**	**13.0729**

Table 10: **A timing analysis of sequence-based device fingerprinting.** *The timing results are averaged over 10 iteration for fingerprint extraction and 100 iterations for classification to generalize the results. Timings for packet capturing are averaged over 540 fingerprints.*

- `D-LinkSensor`, `D-LinkSiren`: Representing two device-types which were able to differentiate better than in the packet-based approach.

- `SmarterCoffee`, `iKettle2`: Representing two device-types which experience

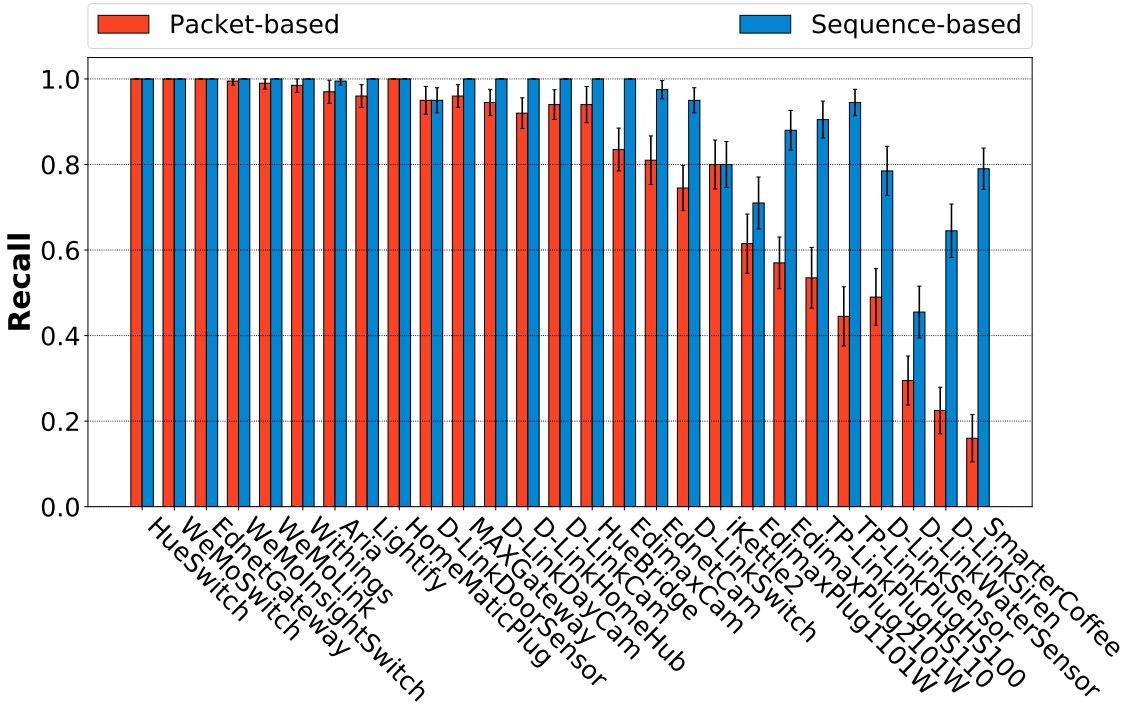


Figure 13: **Packet-based vs sequence-based average device prediction Recall.** *Error bars show the 95% confidence interval of the average Recall. The devices are ordered in the decreasing order of F1-score values based on the packet-based prediction results. The results are obtained by performing a 10-fold cross validation for 10 iterations.*

confusion in packet-based as well as sequence-based approaches.

- `WeMoSwitch`: Representing a device-type which was able to successfully identify under both approaches.

In Figure 14a the distribution of *IP payload size - maximum* and *Ethernet frame size - variance* which are two of the top 3 important features is displayed whereas in Figure 14b *Packet size* vs *Destination port class* has been shown. Figure 14b shows that `D-LinkSensor` and `D-LinkSiren` have overlapping feature values which get confused for each other in classification with packet-based features. However Figure 14a shows the contribution of the summary statistics *IP payload size - maximum* and *Ethernet packet size - variance* in reducing overlaps of feature values to successfully distinguish `D-LinkSensor` from `D-LinkSiren` when using sequence-based features. This further elaborates the importance of using a set of summary statistics of a parameter (in this case *IP payload size*), rather than a single values representing that parameter as in the case of packet-based features. Even in

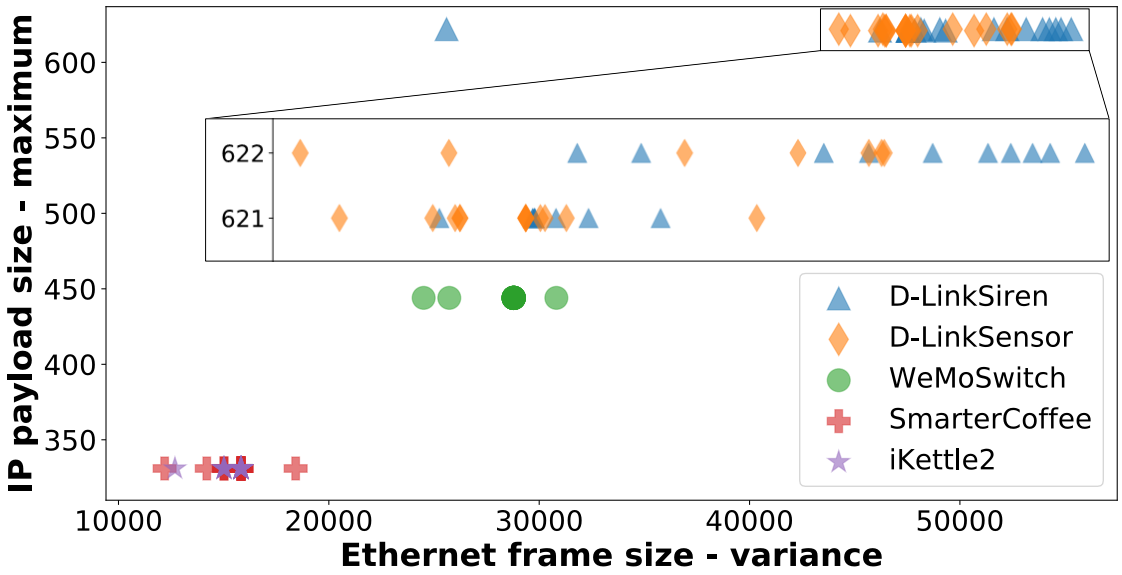
	Predicted class											
	Packet-based						Sequence-based					
Actual class	A	B	C	D	E	Θ	A	B	C	D	E	Θ
D-LinkSiren (A)	45	82	0	0	0	73	129	1	0	0	0	70
D-LinkSensor (B)	43	98	0	0	0	59	10	157	0	0	0	33
SmarterCoffee (C)	0	0	87	105	0	8	0	0	158	36	0	6
ikettle2 (D)	0	0	97	95	0	8	0	0	40	160	0	0
WeMoSwitch (E)	0	0	0	0	200	0	0	0	0	0	200	0
Other (Θ)	46	121	19	19	1	-	68	31	2	0	0	-

Table 11: **Comparison of confusion matrices for sequence-based and packet-based prediction.** *Each row represent the actual device and columns represent the predicted classes for 200 fingerprints. Other groups the results for the remaining devices.*

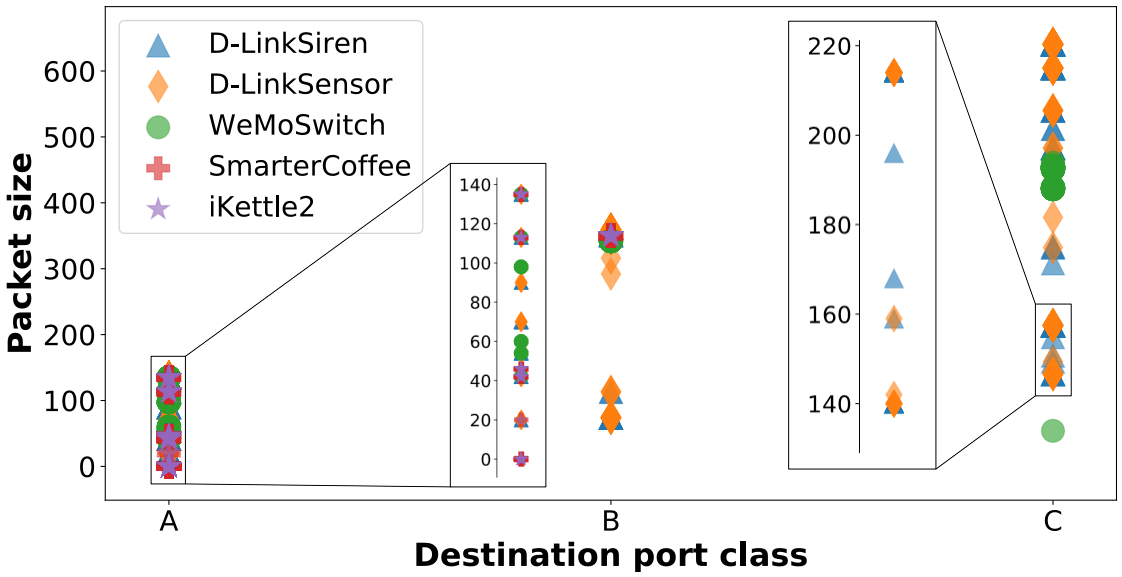
the sequence-based approach if we used only the *IP payload size - maximum* as a feature without using *Ethernet frame size - variance*, the separation of overlaps in `D-LinkSensor` and `D-LinkSiren` would have been difficult. Nevertheless using *IP payload size - maximum* in combination with *Ethernet frame size - variance* reduces the overlaps significantly, enabling the fingerprinting technique to differentiate the two device-types. As a result, we managed to increase the device prediction F_1*-score* for `D-LinkSensor` and `D-LinkSiren` to 0.79 and 0.61 using sequence-based features compared to the 0.38 and 0.23 with the packet-based features. The confusion matrix in Table 11 supports the above claim by providing quantitative evidence on the confusion with each approach. A more detailed version of confusion matrices are attached in Section 7, Figure 20 of the Appendix 1.

In the case of `SmarterCoffee` and `iKettle2` the average prediction F_1*-score* rise to 0.79 for both device-types with sequence-based features. However, Figure 14a visually implies that a considerable amount of overlaps are still present with sequence-based features between `SmarterCoffee` and `iKettle2` compared to the overlaps of `D-LinkSensor` and `D-LinkSiren`. Quantities in Table 11 validates this fact. As per the Table, while the confusion between `D-LinkSensor` and `D-LinkSiren` reduces to 0.064 and 0.008 respectively, confusion between `SmarterCoffee` and `iKettle2` only managed to achieve 0.23 and 0.25.

Distinctive features of `WeMoSwitch` enabled identifying it consistently resulting in zero confusion irrespective of the feature set (sequence-based or packet-based) used.



(a) Sequence-based approach



(b) Packet-based approach

Figure 14: **Impact of the important features.** *The important features of sequence-based solution are able to identify the devices which largely appear indistinguishable to the packet-based approach of IoT Sentinel.*

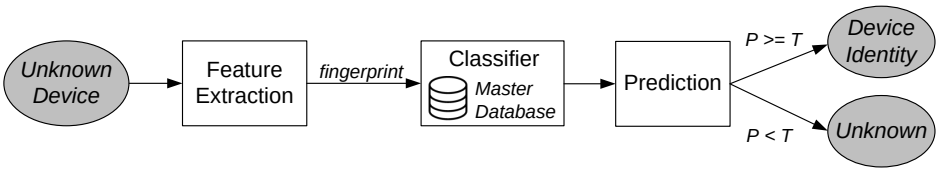


Figure 15: **Identifying a previously unseen device.** *Feature extraction generates a fingerprint with the packet sequence of the unknown device. Then it is subjected to the classifier which use a master database to provide a list of predictions. If at least one of the prediction probabilities P are higher than T, a device-type is predicted or else declared as unknown.*

5.4 Fingerprinting an unknown device

In general context there is a higher probability that the IoT device connecting to the network would be unknown to the classifier as there are vastly different IoT device-types currently in use. But the supervised machine learning approach used in our fingerprinting technique will fail to identify such a device due to the lack of knowledge of the device in the classifier. Since a fingerprinting technique is expected to fingerprint all devices connecting to a network, it should have a mechanism to identify previously unseen devices as *unknown*. In the IoT Sentinel [6], they have used a threshold based identification technique to detect previously unseen devices. They have used a threshold of 0.2 and if the prediction probability is less than the threshold they identify that as an unknown device. However the reasons for selecting such a threshold is not clearly expressed in their work. Therefore we investigated the impact of a threshold on the confidence of the classification, where devices with low confidence would be classified as unknown rather than misclassifying. On one hand, if a classification confidence threshold is not used (threshold = 0), there is a higher possibility of an unknown device being detected as some known device to the classifier. On the other hand an extremely high threshold will classify all devices as unknown.

In order to identify the effect of classification confidence threshold (T) on the device prediction performance, we modified the classification algorithm used. The modified Random Forest classifier outputs a list of probabilities which denotes the likelihood of the fingerprint being tested match each of the device-types the classifier is trained for. If at least one probability in that list is greater than a defined confidence threshold, the unknown device is identified with a device-type or else considered

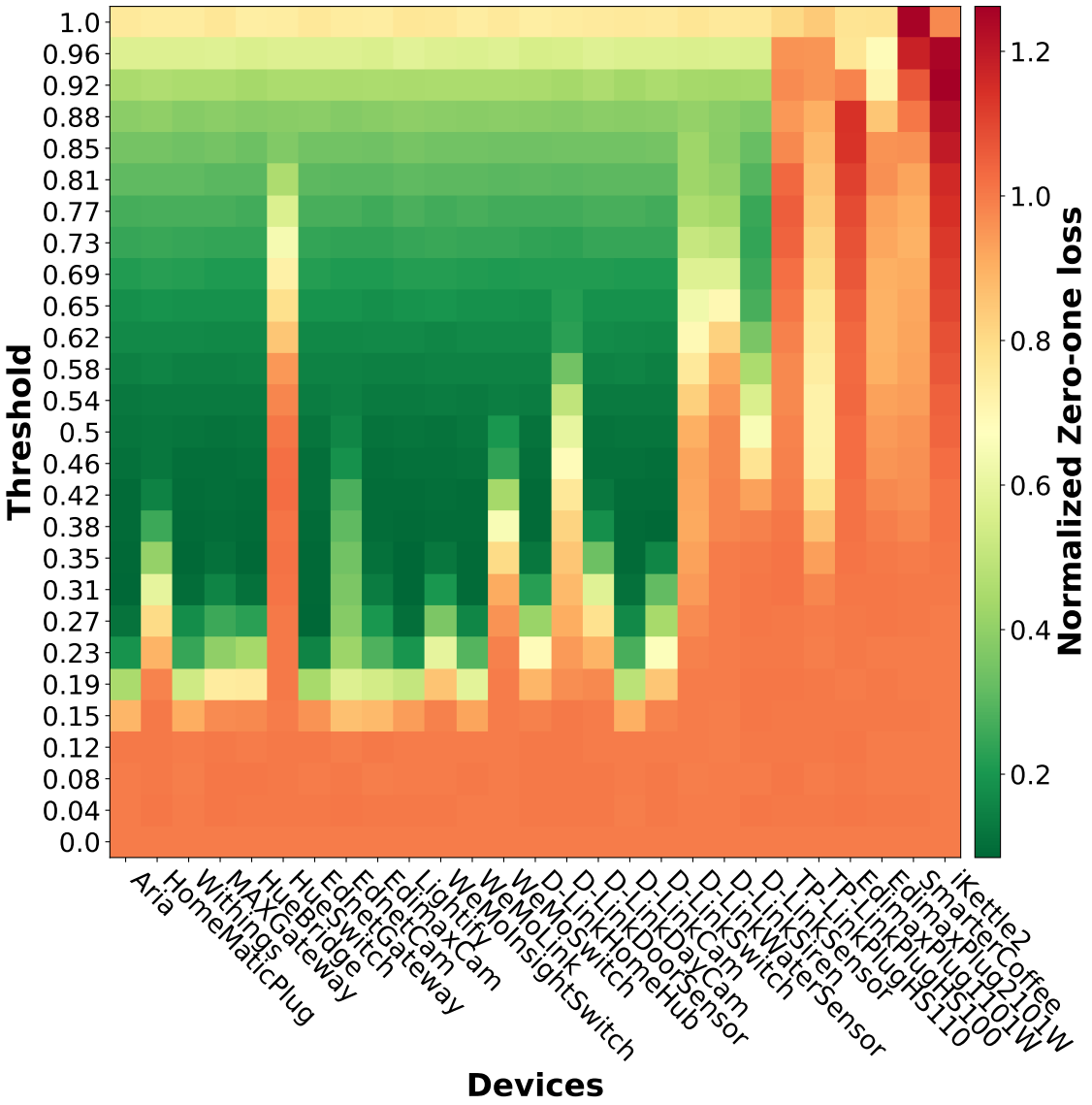


Figure 16: **Normalized Zero-one loss score with the variation of confidence threshold in the absence each device.** *Loss score is normalized based on loss values at no threshold (threshold = 0). Each pixel denotes the normalized Zero-one loss value for the threshold in y-axis when device-type in x-axis is unknown. The results are generalized with a 10-fold cross validation for 10 iterations.*

to be unknown. The classifier was trained using 26 device-types, omitting the set of fingerprints for a particular device-type. We then used this omitted device-type as the unknown device-type to find the confidence threshold at which it would be correctly classified as unknown, as well as classifying the other known device-types correctly.

We employed Zero-one loss to evaluate the performance of the classifier at different confidence thresholds. For each prediction made, Zero-one loss was measured and averaged to represent a loss value for the classifier at each confidence threshold. Further we normalized the Zero-one loss value at each threshold by the loss value observed when there is no threshold (threshold = 0). For an example, in the absence of device-type `Aria`, Zero-one loss at confidence thresholds 0 and 0.5 are 0.545 and 0.063 respectively. Thus the normalized Zero-one losses at thresholds 0 and 0.5 are 1 (0.545/0.545) and 0.116 (0.063/0.545). The normalized Zero-one loss at each threshold in the absence of a particular device-type is displayed in Figure 16.

It is visible in the Figure 16 that when the confidence threshold is in the range of 0 to 0.19 when any device is unknown, the loss remains closer to 1 because the unknown devices are mis-classified as known device-types by the classifier. Similarly, when the confidence threshold reaches 1.0, the loss increases again towards 1, as more devices are classified as unknown by the classifier. First column represents the behavior of Zero-one loss at each threshold when device-type `Aria` is unknown to the classifier. Until the confidence threshold reaches 0.19 the loss remains closer to 1 and then gradually decreases with the incrementing threshold. Confidence thresholds in the range of 0.23 - 0.65 results in a normalized Zero-one loss under 0.2 and reaches its minimum of 0.08 at a confidence threshold of 0.31. As seen in Figure 16, such a range exists when the unknown device is any device except `SmarterCoffee`, `ikettle2`, `EdimaxPlug2101W`, `EdimaxPlug1101W`, `TP-LinkPlugHS100` and `TP-LinkPlugHS110`. When the unknown device was `SmarterCoffee`, it was identified as `ikettle2` frequently by the classifier and vice-versa, resulting in a higher Zero-one loss around 0.98 irrespective of the confidence threshold been used. The same behavior is observed for device pairs (`EdimaxPlug2101W`, `EdimaxPlug1101W`) and (`TP-LinkPlugHS100`, `TP-LinkPlugHS110`) with a higher Zero-one loss round 0.95 throughout. Similarity of the features in these 3 device-type pairs caused the higher normalized Zero-one losses at any threshold.

Figure 17 analyses zero-one loss in a different perspective where it explores the distribution of combined zero-one loss for all devices at a defined confidence threshold. Until the confidence threshold reaches 0.19, the normalized mean and median zero-one losses stay closer to 1 and starts to reduce with increasing thresholds. As it depicts, best mean performance occurs in range $t \in [0.54, 0.69]$, while the minimum mean zero-one loss occurring at a confidence threshold of 0.62. However selection of a threshold between 0.31 and 0.35 ensures that the combined zero-one loss for all device-types stay within the IQR while maintaining the mean loss value under

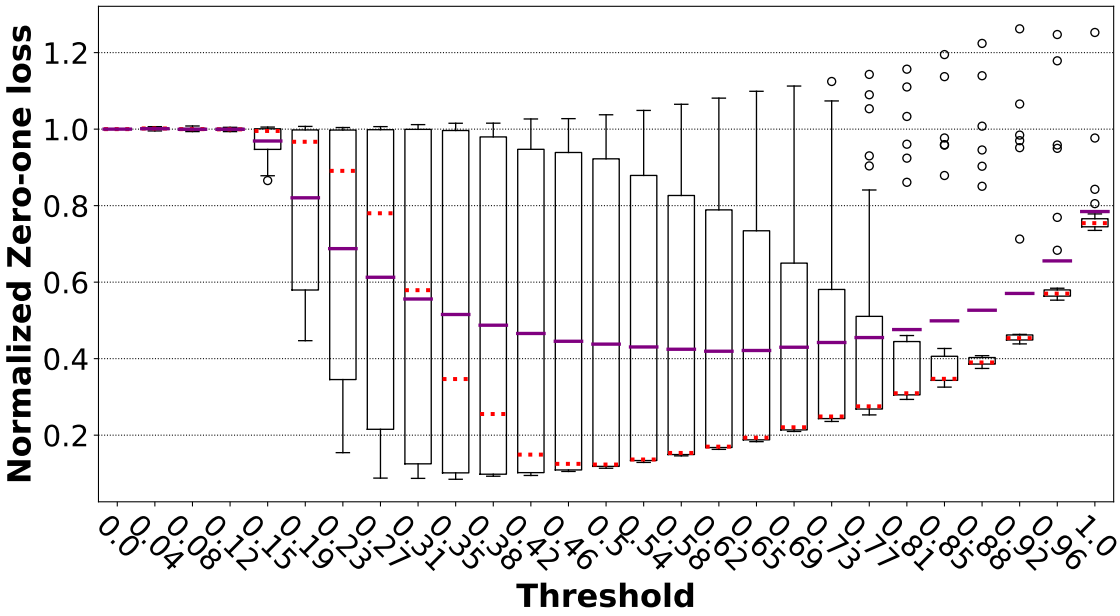


Figure 17: **Normalized Zero-one loss score distribution of all devices for a defined confidence threshold.** *The boxes show the inter-quartile range and whiskers range till the last datum within 1.5 * IQR. Purple solid lines and Red dotted lines represent the mean loss and median loss respectively. Outliers are marked with circles outlined in black. The results are generalized with a 10-fold cross validation for 10 iterations.*

0.6, thus is a safer approach. The aforementioned region of thresholds provides a higher recall but with higher false positives which fits in well when the priority is to identify all instances of a particular device-type. A confidence threshold over 0.77 results in a condensed normalized zero-one loss distribution with few outliers. These higher thresholds produce classifiers with a higher precision while sacrificing the recall, which suits the cases where accurate identification is critical with a low false-positive rate.

6 Future work: A novel feature-based anomaly detection

The rapid growth of wireless technologies such as the Internet of Things has caused a heavy burden on network security in terms of monitoring, troubleshooting and securing networks. In addition, devices are vulnerable for hacking and compromising which increase the risk of attacks. Existing network security solutions are mainly categorized as Signature-based and Anomaly-based techniques [34]. Signature-based threat detection mechanisms build a database of signatures or threat models to match with the suspicious activities. But their applications are limited due to their inability to detect new attack types, other than the attacks that are defined in the signature database. Further, these network security solutions mainly focus on preventing adversary attacks, but are proven to have security holes that can be exploited by the adversaries. E.g. Recovery of passphrases in Wi-Fi Protected Access (WPA) and Wi-Fi Protected Access II (WPA-II). Other than attacks devices could misbehave due to device configuration changes, driver failures, failed software/firmware updates, incompatibility of firmware and hardware, operational human errors. The effectiveness of traditional security solutions are limited in above cases due to the evolving nature of attacks and unpredictability of device misbehaviors.

Anomaly-based threat detection models define a normal behavior of a device through offline training and any activity that lies outside of the normal range is considered to be an anomaly. Device fingerprinting has emerged as an anomaly-based solution to identify devices present in the network and reducing the device vulnerabilities. Our future work focuses on analyzing long-term device behavior of devices and identify abnormalities by comparing the features extracted from packet headers. The proposed anomaly detection technique uses our sequence-based device fingerprinting approach to identify device-types and then a identifies anomalies based on a white-listing approach.

6.1 Traffic anomaly detection

Traffic anomalies can be defined as patterns in data that do not conform to well defined notion of normal behavior [35]. Therefore identifying the deviating patterns in data streams are considered as traffic anomaly detection which can be an attack or a misbehavior of a device. Selection of an anomaly detection technique depends

on few factors.

- Nature of input data
  - Different attribute types (feature, field, dimension etc.)
  - Relationship among data instances (sequential, spatial, graph etc.)
- Type of anomaly
  - Point anomaly: anomalous behavior in a single data point with respect to other data points.
  - Contextual anomaly: Data instance is considered anomalous in a specific context, but may be it is normal in another context. The data instances consist of contextual attributes and behavioral attributes.
  - Collective anomaly: When a set of data instances are anomalous with respect to the entire data set, but if we consider each individual data instance it might not be anomalous.

Recent researches have developed variety of techniques to detect anomalies such as offline log analysis, Statistical anomaly detection, Rule-based anomaly detection and feature-based behavior classification. Notable work under each of these techniques will be discussed in the next section.

6.1.1 Offline log analysis

Usually information generated by network devices in the case of an event/ activity occurrence are recorded in device logs. Due to the heterogeneity of devices in network, the logs differs widely, often incomplete, contradictory and very large in volume. The usual attempt of referring to logs manually takes time as well as a considerable amount of effort. Beehive [36] provides a mechanism of automatically mining and extracting information from logs produced by large amount of network devices. The concept is based on behavioral detection of suspicious activities of hosts which might lead to traffic anomalies. Beehive has been evaluated in a large enterprise and it has been able to detect suspicious activities which were not detected by the state-of-art security tools. Since Beehive does not depend of signature based traffic anomaly detection it is able to detect previously unknown anomalous behaviors. Further it used a customized white-list generated by monitoring the normal traffic pattern of the organization to reduce the raw log data by 74%. Cid [37]

propose a system that uses the logs collected by network devices to detect traffic anomalies with the support of user-defined rules. The logs are collected at the OSSEC server and they are pre-decoded to extract generic information from logs. Then the server decodes the logs to identify key information from logs such as source IP, username, id *etc.* Then the decoded information is checked against the user-defined rules reducing the processing time.

6.1.2 Statistical anomaly detection

A generic analytics engine is proposed in [38], a research by Intel to detect changes and anomalies in IoT sensor data streams. The framework comprises of multi-level analysis including time series classification, single/ multi sensor change detection, abnormal behavior detection and change classification using statistical testing techniques such as Kolmogorov Smirnov, YIN algorithm, *etc.*

6.1.3 Rule-based detection

NETAD [39] operates on rule-based traffic filtering. The filtering process removes uninterested traffic such as Non-IP packets, all outgoing traffic, UDP to high numbered ports *etc.* Thus reduce the amount of data processing as well as the time consumption. The basis for filtering process is that the first few packets of a connection request are sufficient for traffic anomaly detection. The NETAD models 48 attributes consisting of the first 48 bytes of the packet starting with the IP header. It computes a packet score depending on the time and frequency of each byte of packet. Rare and novel header values are assigned high scores. A threshold is applied on a packets score to find anomalous packets. One drawback of the approach is, since the technique is an anomaly detection it does not provide information about the nature of the fault/ attack. But unlike the usual security tools it will provide alerts on all the unknown traffic anomalies.

The research of Palanivel *et al.* [40] propose a system named *IDEA* for efficient maintenance of IoT sensors to ensure the minimal disruption of services in IoT enabled smart environments which monitors activities of daily living (ADLs). The identification of ADLs are performed by learning patterns of sensors that are initiated while performing an ADL, thus the focus of the study lies on behavioral monitoring of smart sensors rather than off-the-shelf IoT devices in our consideration. IDEA follows mainly three methods to identify failures in sensors which could be periodic

or event-driven. For periodic sensors the system raises a failure alert upon reaching a threshold amount of time since the last recorded state report. The system classifies sensors based on the importance a particular sensor in identifying ADLs. A *grave sensor* is such a type where an ADL detection depends solely on that sensor and the ADL cannot be detected without inputs from that sensor. The second failure detection is associated with grave sensors, where the system raises an failure alert upon reaching a threshold time since the last occurrence of ADL, based on the frequency of occurrences for that particular ADL. The third technique focuses the sensors which are redundant in detecting a particular ADL. In such case a *rarity score* is associated with a sensor which denotes the probability that the sensor is not triggered given that some ADLs which the sensor participates have been detected using other sensors. IDEA raises a failure alert when the rarity score exceeds a particular threshold.

6.1.4 Behavior classification using feature extraction

The main concept under this technique is to extract features from packets transmitted to/from a device and use machine learning to train a classification model that has the capability to classify the device behavior as normal/abnormal. A list of popular features used by recent studies on security attack classification are shown in Table 12.

Study	**Time frame**	**Features**		**Output**
		Basis	*Type/ Name*	
Moskovitch *et al.* [41]	Every 1 second	Time based	323 features: Internet Control Message Protocol (27) Internet Protocol (17) Transport Control Protocol (9) User Datagram Protocol (5) Features of device hardware and processor	Detection of known/ unknown worm activities

Study	**Time frame**	**Features**		**Output**
		Basis	*Type/ Name*	
Lakhina *et al.* [42]	Every 5 minutes	Statistical	4 features: Source IP address, destination IP address, source port and destination port	Classification of port scans, outage events, worms, flash crowds, alpha flows
Brutlag *et al.* [43]	Every 1 second	Time based	1 feature: Outgoing bandwidth rate	Abnormal behavior is detected by time series analysis
Lakhina *et al.* [44]	Every 5 minutes	Time based	3 features: Number of bytes, number of packets, number of IP-flows (origin to destination)	Detect alpha crowds, outages, scans, ingress-shift and flash crowds
Kim *et al.* [45]	-	Time based	12 features: ICMP/TCP/UDP packet counts, TCP/UDP source and destination ports, TCP/UDP source and destination IPs, Total packet count	ICMP/TCP/UDP flooding, ping-pong, port scanning
Nychis *et al.* [46]	Every 5 minutes	Time based	7 features: Source and destination addresses Source and destination ports Flow size distribution (FSD) In and Out-degree distributions	Identifies alpha flows, scans, spoofed DoS activities in the network

Study	**Time frame**	**Features**		**Output**
		Basis	*Type/ Name*	
King *et al.* [47]	-	Time based	8 features: Source and destination addresses Source and destination ports TCP flags Protocol number Packet size Flow duration	Detects port misconfigurations, server failures, abnormal protocol behaviors, configuration anomalies, port scanning, DoS attacks, SYN flooding

Table 12: **List of features used by feature-based anomaly detection techniques**

The extracted features are then treated differently in each of the aforementioned studies to classify device behaviors. Anomalies cause changes in traffic feature distributions and this has been used to detect attacks in [42]. They use sample entropy as the summary statistic to denote the degree of dispersal or concentration of the distribution. [46] use the feature entropy distributions and correlations between the time series of entropy data to detect abnormal behavior of devices. In [43], they generate a time series from the collected features and model the normal behavior. Then they use Holt-Winters forecasting method to predict the feature values and if the observed feature value falls outside the confidence band of the predicted value it is declared as an anomaly. A similar time series analysis has been used in [44] where a time series is generated for each feature and they are analyzed using subspace method. For each feature, the calculation of squared prediction error for residual vector and t-squared (t^2) statistic reveal the possible anomalies. Kind et al. [47] propose a technique that constructs histograms, model normal behavior and identify deviations as anomalies.

CISCO proposes [48] an anomaly detection technique to identify worm viruses in the network and infected hosts by training a threshold-based detection mechanism. The system first conducts a learning phase (default 24 hours) assuming no attacks

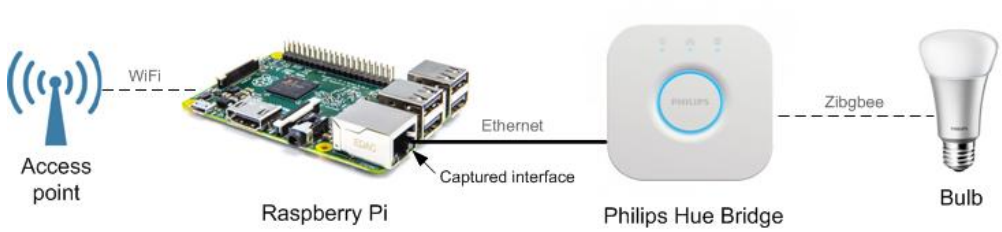


Figure 18: **Experimental setup used to check the behavior of Philips HueBridge.** *Raspberry Pi was used as an access point connecting to the HueBridge via Ethernet port and to the Internet via Wi-Fi.*

happen during that time and derives on policy thresholds which fits the network. During the learning phase histograms are developed for each TCP/ UDP ports to generate a baseline for the normal behavior of the network. For an example the thresholds for the number of source addresses, based on the amount of destination addresses reached by each source address are derived. In the detection phase the system monitors the network traffic flows that violates the thresholds and generate alerts based on the alert configuration.

6.2 Anomaly detection based on long-term feature analysis

Our focus is to develop an anomaly detection technique by analyzing the features of packet headers. As per the Table 12 a huge set of features are at the disposal to detect suspicious device behaviors. After analyzing the aforementioned related work on feature-based anomaly detection we have identified a set of features to represent the behavior of a device as indicated in Table 13.

We set up a `Philips HueBridge` to monitor its normal behavior as well as the behavioral changes when some interruption was created. The experimental setup is displayed in Figure 18. All the packets to/ from the `Philips HueBridge` was captured (all traffic of Ethernet port with the MAC address of HueBridge) with a Raspberry Pi III using *tcpdump* as a capturing module. The capturing process was started by turning on the device after a reset, while *tcpdump* capture module was running.

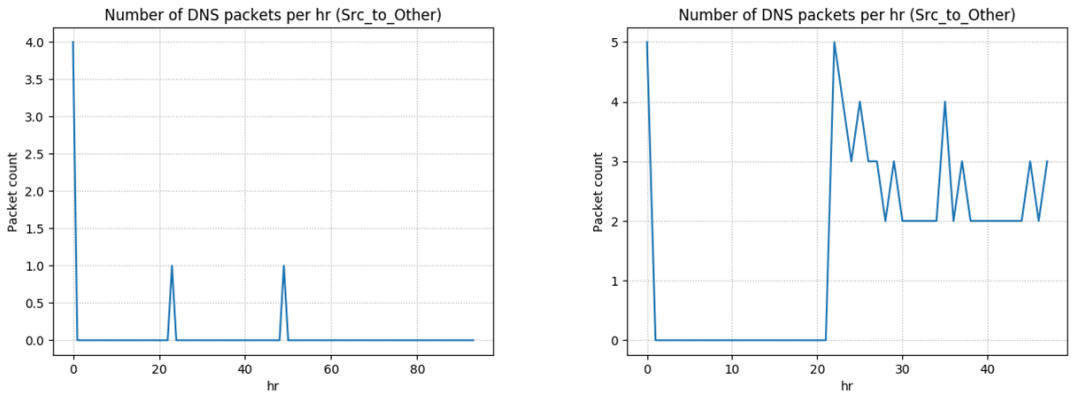
As an initial test we checked the changes to the traffic patterns of the `HueBridge` when the Internet connection was interrupted intentionally. First we captured the network traffic during the normal operation of the device for 90 hours. We assumed that during the capture of normal operation the device behaves as it is expected

Feature	**Explanation**
Periodic statistics of packet inter arrival times *Minimum, Maximum, Q1, Median, Mean, Q3, Variance, IQR*	Summary statistics of packet inter arrival times measured in a defined time period
Fast Fourier transform components of packet inter-arrival time	
Periodic counts of protocol occurrences *TCP, UDP, HTTP, DHCP, DNS*	Number of packets for each protocol in a defined time period
Rate of packets	Number of packets transmitted in a defined period
Order of Protocols	Order of packet transmission based on protocol type (TCP, UDP)
Periodic Ethernet, IP header, IP payload sizes *Minimum, Maximum, Q1, Median, Mean, Q3, Variance, IQR*	Summary statistics of header sizes in bytes
Packet direction	Direction of the packet flow represented by a series of integers
IP destinations	Number of different IP addresses a device connects to
Source/ Destination port class	Series of values representing the source/ destination port classes

Table 13: **List of features which can be used to detect anomalies.**

to without any abnormalities. Then we restarted the capturing process, however after 22 hours we intentionally disconnected the Internet connection while capturing packets. We observed that the Domain Name Server (DNS) traffic was increased significantly as a result of the interruption as seen in Figure 19.

It is seen in the Figure 19a, that during the normal operation initially the rate of DNS traffic was higher during the device setup time. After that periodic spikes are observed in the rate of DNS packets while it remains at zero mostly. But once the Internet connection is interrupted, the rate of DNS traffic increased significantly and exhibit an offset from the DNS traffic rate at normal operation. Hence, monitoring



(a) Normal traffic pattern (b) Interrupted Internet connection

Figure 19: **Comparison of DNS traffic behavior**

the rate of DNS traffic enables the identification of interruptions to the Internet connection.

As clarified in this simple example, the set of features introduced in Table 12 has the ability to detect abnormalities in traffic flows and support locating the issues in the devices. Our next step is to capture the traffic from different IoT devices and generate traffic profiles for them to denote the normal behavior. Then using those traffic profiles we plan to synthetically generate anomalies and try to identify them using machine learning classification models similar to the models used in device identification.

7 Conclusion

We propose a novel device fingerprinting technique based on features extracted from a sequence of packets representing the summary statistics of packet sizes, inter-arrival times, Fast Fourier Transform of inter-arrival times, packet directions, *etc.* A 14 % increase in the average prediction F_1-*score* was achieved in our approach compared to the state-of-the-art technique. Furthermore, we were able to increase the average prediction F_1-*score* of the 10 devices with the lowest prediction F_1-*score* from 0.44 to 0.76. We pointed out that the summary statistics related to packet size measurements are the most important in classifying different devices due to the differences in the DHCP header options used by each devices. Then we analyzed the impact of the confidence threshold on identifying previously unseen devices by the classifier maintaining the prediction loss at minimum and we propose possible ranges of thresholds that can be used under different circumstances. As our future work we plan to extend this approach to detect anomalous behaviors of devices by comparing long-term device behavior with the featured extracted from packet headers similar to the fingerprinting approach.

Acknowledgment

I would like to thank my supervisors, Professor Sasu Tarkoma, Dr.Julien Mineraud and Dr.Ashwin Rao for the immense support and guidance provided throughout my thesis. Furthermore I would like to thank the authors of the IoT sentinel project [6] for their support in providing the dataset and additional information.

References

1 S. Lucero, "IoT platforms: enabling the Internet of Things." IHS Technology Whitepaper (2016).

2 A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys Tutorials*, vol. 17, pp. 2347–2376, Fourthquarter 2015.

3 M. Zorzi, A. Gluhak, S. Lange, and A. Bassi, "From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view," *IEEE Wireless Communications*, vol. 17, pp. 44–51, December 2010.

4 R. Khan, S. U. Khan, R. Zaheer, and S. Khan, "Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges," in *2012 10th International Conference on Frontiers of Information Technology*, pp. 257–260, Dec 2012.

5 Z. Hu, "The Research of Several Key Question of Internet of Things," in *2011 International Conference on Intelligence Science and Information Engineering*, pp. 362–365, Aug 2011.

6 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 *37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA, June 5-8, 2017*, pp. 2177–2184, 2017.

7 G. F. Lyon, *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. USA: Insecure, 2009.

8 F. Yarochkin, M. Kydyraliev, and O. Arkin, "Xprobe project," 2014.

9 M. Zalewski, "p0f: Passive OS Fingerprinting tool," 2006.

10 P. Auffret, "SinFP, unification of active and passive operating system fingerprinting," *Journal in computer virology*, vol. 6, no. 3, pp. 197–205, 2010.

11 S. Bratus, C. Cornelius, D. Kotz, and D. Peebles, "Active Behavioral Fingerprinting of Wireless Devices," in *Proceedings of the First ACM Conference on*

Wireless Network Security, WiSec '08, (New York, NY, USA), pp. 56–61, ACM, 2008.

12 B. Sieka, "Active fingerprinting of 802.11 devices by timing analysis," in *CCNC 2006. 2006 3rd IEEE Consumer Communications and Networking Conference, 2006.*, vol. 1, pp. 15–19, Jan 2006.

13 K. Gao, C. Corbett, and R. Beyah, "A passive approach to wireless device fingerprinting," in *2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN)*, pp. 383–392, June 2010.

14 L. Letaw, J. Pletcher, and K. Butler, "Host Identification via USB Fingerprinting," in *2011 Sixth IEEE International Workshop on Systematic Approaches to Digital Forensic Engineering*, pp. 1–9, May 2011.

15 J. François, H. Abdelnur, R. State, and O. Festor, "PTF: Passive Temporal Fingerprinting," in *12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops*, pp. 289–296, May 2011.

16 L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee, "Identifying Unique Devices Through Wireless Fingerprinting," in *Proceedings of the First ACM Conference on Wireless Network Security*, WiSec '08, (New York, NY, USA), pp. 46–55, ACM, 2008.

17 T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, pp. 93–108, April 2005.

18 S. Jana and S. K. Kasera, "On Fast and Accurate Detection of Unauthorized Wireless Access Points Using Clock Skews," *IEEE Transactions on Mobile Computing*, vol. 9, pp. 449–462, March 2010.

19 F. Lanze, A. Panchenko, B. Braatz, and A. Zinnen, "Clock skew based remote device fingerprinting demystified," in *2012 IEEE Global Communications Conference (GLOBECOM)*, pp. 813–819, Dec 2012.

20 A. S. Uluagac, S. V. Radhakrishnan, C. Corbett, A. Baca, and R. Beyah, "A passive technique for fingerprinting wireless devices with Wired-side Observations," in *2013 IEEE Conference on Communications and Network Security (CNS)*, pp. 305–313, Oct 2013.

21 B. Li and L. Han, "Distance Weighted Cosine Similarity Measure for Text Classification," in *IDEAL*, vol. 8206 of *Lecture Notes in Computer Science*, pp. 611–618, Springer, 2013.

22 G. Sidorov, A. F. Gelbukh, H. Gómez-Adorno, and D. Pinto, "Soft Similarity and Soft Cosine Measure: Similarity of Features in Vector Space Model," *Computación y Sistemas*, vol. 18, no. 3, 2014.

23 J. P. Ellch, "Fingerprinting 802.11 Devices," tech. rep., NAVAL POSTGRADUATE SCHOOL MONTEREY CA, 2006.

24 V. Paxson, "On Calibrating Measurements of Packet Transit Times," *SIGMETRICS Perform. Eval. Rev.*, vol. 26, pp. 11–21, June 1998.

25 S. B. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1, pp. 227–234 vol.1, Mar 1999.

26 J. V. Tu, "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes," *Journal of Clinical Epidemiology*, vol. 49, no. 11, pp. 1225 – 1231, 1996.

27 C. Neumann, O. Heen, and S. Onno, "An Empirical Study of Passive 802.11 Device Fingerprinting," in *2012 32nd International Conference on Distributed Computing Systems Workshops*, pp. 593–602, June 2012.

28 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 *USENIX Security Symposium*, vol. 3, pp. 16–89, 2006.

29 C. L. Corbett, R. A. Beyah, and J. A. Copeland, "Using Active Scanning to Identify Wireless NICs," in *2006 IEEE Information Assurance Workshop*, pp. 239–246, June 2006.

30 C. L. Corbett, R. A. Beyah, and J. A. Copeland, "Passive classification of wireless nics during rate switching," *EURASIP J. Wireless Comm. and Networking*, vol. 2008, 2008.

31 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*, vol. 12, pp. 2825–2830, 2011.

32 T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, pp. 56–76, Fourth 2008.

33 A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," tech. rep., 2013.

34 J. Pacheco and S. Hariri, "Anomaly behavior analysis for IoT sensors," *Transactions on Emerging Telecommunications Technologies*, pp. e3188–n/a, 2017. e3188 ETT-16-0338.R1.

35 V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 15:1–15:58, 2009.

36 T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda, "Beehive: Large-scale Log Analysis for Detecting Suspicious Activity in Enterprise Networks," in *Proceedings of the 29th Annual Computer Security Applications Conference*, ACSAC '13, (New York, NY, USA), pp. 199–208, ACM, 2013.

37 D. B. Cid, "Log analysis using OSSEC," *URL: http://www. ossec. net/ossec-docs/auscert-2007-dcid. pdf*, 2007.

38 A. Amitai, W. Gilad, and F. Lev, "Change and Anomaly Detection Framework for Internet of Things Data Streams," 2016. Online; accessed 05-June-2017.

39 M. V. Mahoney, "Network Traffic Anomaly Detection Based on Packet Bytes," in *Proceedings of the 2003 ACM Symposium on Applied Computing*, SAC '03, (New York, NY, USA), pp. 346–350, ACM, 2003.

40 P. A. Kodeswaran, R. Kokku, S. Sen, and M. Srivatsa, "Idea: A System for Efficient Failure Management in Smart IoT Environments," in *MobiSys*, pp. 43–56, ACM, 2016.

41 R. Moskovitch, Y. Elovici, and L. Rokach, "Detection of unknown computer worms based on behavioral classification of the host," *Computational Statistics & Data Analysis*, vol. 52, no. 9, pp. 4544–4566, 2008.

42 A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *SIGCOMM*, pp. 217–228, ACM, 2005.

43 J. D. Brutlag, "Aberrant Behavior Detection in Time Series for Network Monitoring," in *LISA*, pp. 139–146, 2000.

44 A. Lakhina, M. Crovella, and C. Diot, "Characterization of Network-wide Anomalies in Traffic Flows," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, IMC '04, (New York, NY, USA), pp. 201–206, ACM, 2004.

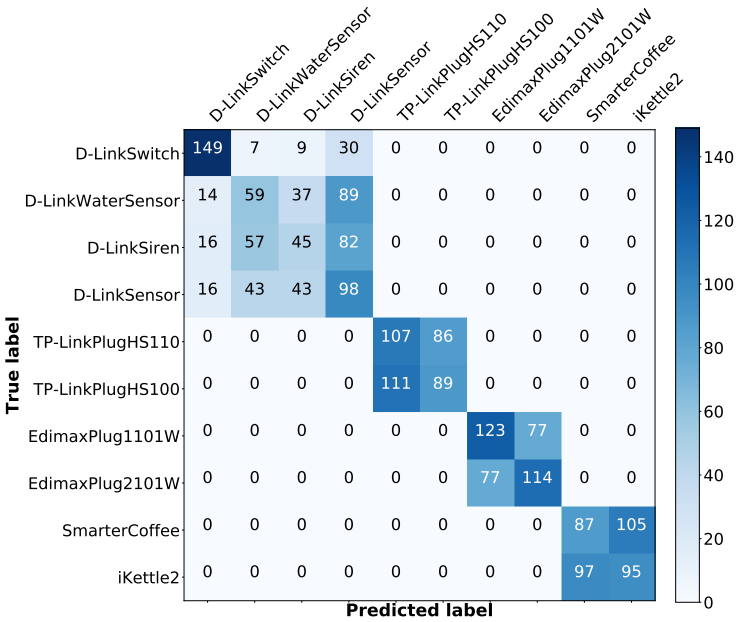
45 M. Kim, H. Kang, S. Hong, S. Chung, and J. W. Hong, "A flow-based method for abnormal network traffic detection," in *NOMS (1)*, pp. 599–612, IEEE, 2004.

46 G. Nychis, V. Sekar, D. G. Andersen, H. Kim, and H. Zhang, "An empirical evaluation of entropy-based traffic anomaly detection," in *Internet Measurement Conference*, pp. 151–156, ACM, 2008.

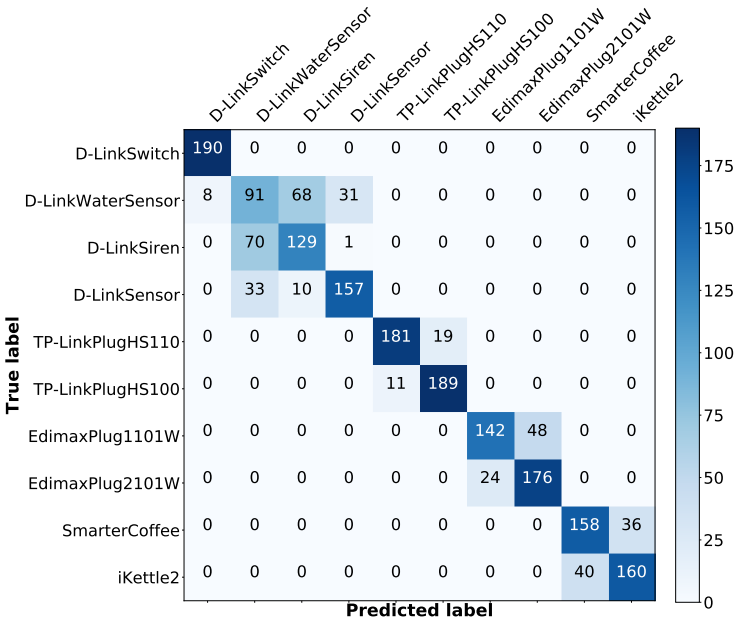
47 A. Kind, M. P. Stoecklin, and X. A. Dimitropoulos, "Histogram-based traffic anomaly detection," *IEEE Trans. Network and Service Management*, vol. 6, no. 2, pp. 110–121, 2009.

48 "User Guide for Cisco Security Manager 4.0." `https://www.cisco.com/c/en/us/td/docs/security/security_management/cisco_security_manager/security_manager/40/user/guide/CSMUserGuide_wrapper/ipsanom.html`. Online; accessed 20-Sep-2017.

Appendix 1. Confusion matrices comparison



(a) Confusion matrix for Packet-based approach



(b) Confusion matrix for Sequence-based approach

Figure 20: **Confusion matrix for the 10 devices with lowest prediction accuracies under packet-based fingerprinting technique.** *As clearly seen in the figures, confusion is much resolved in Sequence-based approach.*

Appendix 2. Complete decision tree for our sequence-based features classifier

A complete decision tree generated by the Random Forest algorithm is displayed in Figure 21. A pdf version of the full decision tree is available at `https://version.helsinki.fi/prana/device-fingerprinting/nishadh-thesis-manuscript/blob/master/appendix/decision_tree_V_full.pdf`

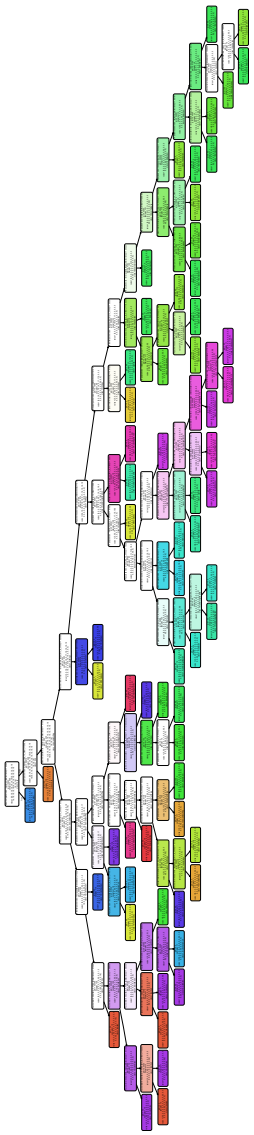


Figure 21: **Sample decision tree which was generated in the Random Forest classification algorithm.**

Appendix 3. Variation of feature importance, precision and recall

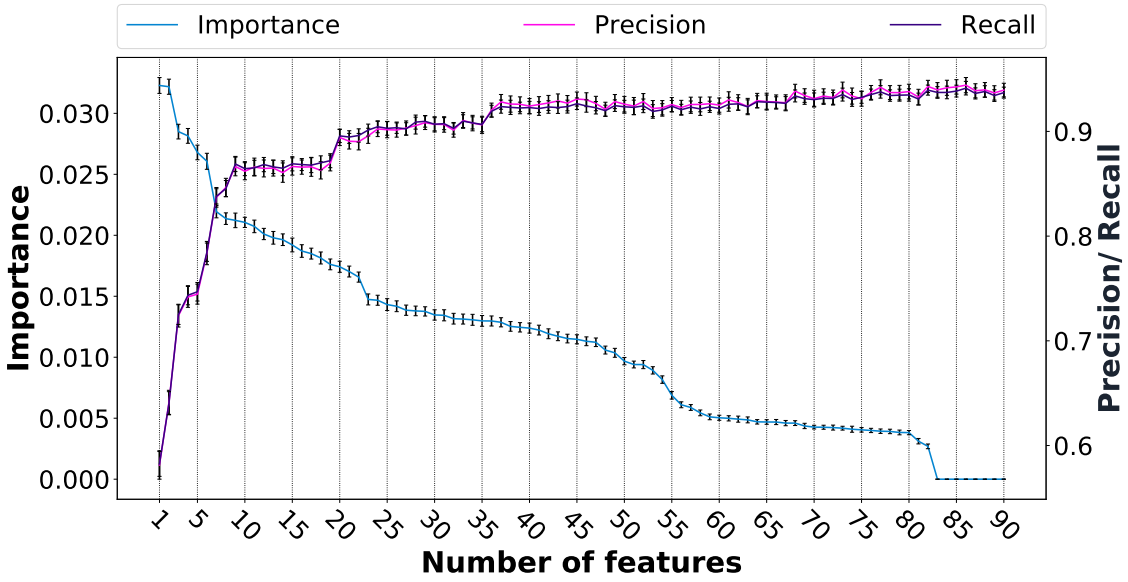


Figure 22: **Feature importance, Precision and Recall variation with the number of features used to train the classifier.** *The error bars shows the 95% CI for the mean feature importance values, Precision and Recall. The results are obtained by performing a 10-fold cross validation for 10 iterations.*

Resources:

We have shared the sequence-based device fingerprinting code modules at `https://version.helsinki.fi/prana/device-fingerprinting/fingerprinting-code`.

The manuscript and figures related to this thesis can be found at `https://version.helsinki.fi/prana/device-fingerprinting/nishadh-thesis-manuscript`.