IoT Traffic Flow Identification using Locality Sensitive Hashes

Batyr Charyyev
School of Systems and Enterprises
Stevens Institute of Technology
bcharyye@stevens.edu

Mehmet Hadi Gunes
School of Systems and Enterprises
Stevens Institute of Technology
mgunes@stevens.edu

***Abstract*—Systems get smarter with computing capabilities, especially in the form of Internet of Things (IoT) devices. IoT devices are often resource-limited as they are optimized for a certain task. Hence, they are prone to be compromised and have become a target of malicious activities. Since IoT devices lack computing power for security software, network administrators need to isolate such devices and limit traffic to the device based on their communication needs. To this end, network administrators need to identify IoT devices when they join a network and detect anomalous traffic when they are compromised. In this paper, we introduce a novel approach to identify the IoT device based on the Nilsimsa hash of its traffic flow. Different from previous studies, the proposed approach does not require feature extraction from the data. In our evaluations, our approach has an average precision and recall of 93% and 90%, respectively.**

I. Introduction

The proliferation of IoT devices has attracted the interest of malicious entities due to their limited security capabilities. For instance, recently, Mirai botnet utilized over 400,000 IoT devices in DDoS attack with over 1 TB bandwidth, which was the highest ever attack traffic volume [6]. To mitigate the impact of such IoT-based attacks, it is vital to have knowledge of the devices in a network. Device fingerprinting [1] enables network administrators to identify which devices are attached to a network. Researchers have proposed different fingerprinting approaches of network traffic [1], [10] as well as hardware characteristics such as clock skewness or magnetic fields emitted from hardware components [8], [18]. Network traffic based methods rely on feature extraction from the traffic data, which might be challenging as new devices with new protocols are developed. On the other hand, hardware characteristics based approaches need to frequently update the model with recent data from devices under various physical conditions [17].

IoT devices have specific behavior, performing simple tasks with well-defined network traffic patterns and the use of computational resources [15]. It is possible to create a network traffic signature for each device to identify the device [1], device events [4] or detect anomalies [20] in its communications. For instance, N-BaIoT [20] uses autoencoders to generate a signature of the device from its benign network traffic and use it to identify botnet devices. Similarly, IoTDS [15] uses one-class classification algorithms such as Elliptic Envelope and Isolation Forest with the device's computational resource utilization data to generate a signature of the device. While the signature enables us to identify the device, the function generating signatures should be less sensitive to small changes in the input, i.e., network traffic in our case.

In this paper, we present a novel IoT device fingerprinting method LSIF (*Locality-Sensitive IoT Fingerprinting*). LSIF generates a hash output of a device from its network traffic flow using the Locality-Sensitive Hash (LSH) function, namely Nilsimsa. This hash output is labeled with the device name and stored in a database. New devices are identified by comparing the locality-sensitive hash of the traffic from the device to the hash values stored in the database. Note that locality-sensitive hash differs from cryptographic hashes as LSH is less sensitive to small changes in input to produce similar hash values with similar inputs. This enables LSIF to identify a device if a sample hash from a similar device is stored in the database. Compared to previous studies that rely on the network traffic fingerprinting [1], [5], [10], our approach does not require feature extraction from the data and tuning of parameters. We evaluated the proposed approach on data collected in our laboratory as well as another public IoT dataset from [10]. We also show the effectiveness of the proposed LSIF method in comparison with the IoT Sentinel [10] and the System Identifier (SysID) [1] as well as two machine learning approaches of One-class Support Vector Machine (OSVM) and Elliptic Envelope (EE) [15].

Contributions of this paper are as follows:

- We introduce *Locality Sensitive IoT Fingerprinting* to utilize network traffic flow for IoT device identification.
- LSIF does not require feature extraction from the data.
- We evaluate LSIF on real IoT device traffic data and compare the result with the previous approaches.
- We share the IoT device traffic data collected in our laboratory and the LSIF code with the research community at https://github.com/netlab-stevens/LSIF/

In the rest of the paper; Section II presents studies focusing on network traffic fingerprinting and locality sensitive hashing. Section III describes the data collection process. Section IV details the design principles of the proposed LSIF approach. Section V shows the evaluation results of the LSIF and presents comparison with the previous approaches. Finally, Section VI concludes the paper.

II. Related Work

A. IoT Device Identification

Researchers have developed approaches to identify a device by analyzing the network traffic generated from the device. Christoph et al. propose a passive 802.11 based device identification approach [5]. Authors focus on transmission rate, frame size, medium access time, transmission time, and frame inter-arrival time of the packets from the wireless interface. Evaluation results show that frame inter-arrival time is the most promising parameter for device identification. Kennedy et al. show that the network traffic can be used in voice command fingerprinting attack [14]. Miettinen et al. propose IoT Sentinel for network traffic based identification [10]. This study collects network traffic of the devices when they are being set up, and utilizes packet headers for identification and security enforcement of IoT devices. Similarly, Ahmet et al. propose SysID, which classifies IoT devices using machine learning and genetic algorithms [1]. They use a genetic algorithm to determine relevant features of TCP/IP packet headers and use machine learning algorithms such as DecisionTable, J48 Decision Trees, and OneR to classify devices from the analysis of the protocol features selected by the genetic algorithm.

Hardware related characteristics are also used for device identification. Fabian et al. identify remote devices based on their clock skewness as each quartz has a unique skew pattern [8]. Yushi et al. propose DeMiCPU, which relies on the magnetic induction signals emitted from the CPU of a device [18]. Ya et al. propose a radio frequency based identification technique which combines the principal component analysis and support vector machines [22]. The problem with this approach is that the model performance gets worse over time and hence requires a frequent update to the model. Xinyu et al. address this issue with the long-term stacking of repetitive symbols [17].

B. Locality Sensitive Hashing

In this paper, we use *locality-sensitive hashing* (LSH) of traffic flow for device fingerprinting. Locality-sensitive hashing has been used for spam detection [7], [11], anomaly detection [12], malware classification [9], and electronic identification [16]. Muhammad et al. use Nilsimsa, which is a locality-sensitive hashing algorithm, for spam fingerprinting at the packet level [11]. Authors evaluate their approach on the SpamAssassin email dataset [13] and show the effectiveness of Nilsimsa on spam fingerprinting under different constraints such as packet reordering, fragmentation, and packet sizes. Alburan et al. leverage locality-sensitive hashing to recover biometric information of individuals in a corporate environment to detect incidents such as information leakage [2]. They acquire biometric information of individuals as they interact with the system and link them to the objects that they interact. In the case of incidents, locality-sensitive hashing digest of similar objects is utilized to recover biometric information of individuals even if the original object is modified.

There are a couple of studies using the similarity of network traffic akin to the locality sensitive hashing. Meidan et al. propose N-BaIoT, a network-based approach for IoT identification, and anomaly detection [20]. From the statistical features of the IoT traffic, they train a deep learning autoencoder to learn the normal behavior of a device. Then, when the autoencoder fails to reconstruct a snapshot, it indicates that the observed behavior is anomalous. Similarly, Vitor et al. propose IoTDS [15]. This study utilizes a one-class classification approach to detect botnets in IoT devices. Authors use a device's CPU utilization, temperature, memory consumption, and the number of running tasks as features for benign device identification. They compare four one-class algorithms (i.e., Elliptic Envelope, Isolation Forest, Local Outlier Factor, and One-class Support Vector Machine) in detail. These studies are similar to ours in the sense that they generate signatures of devices and use those signatures to identify devices or to check if device behavior deviates from the expected norm.

III. IoT Data Collection and Synthesis

In this section, we describe the data collection and synthesis of this study. We collected IoT device traffic in our laboratory from 22 IoT devices, consisting of various illumination devices, smart plugs, cameras, doorbells, and home appliances such as coffee makers and radio. Our experimental testbed with the 22 IoT devices, shown in Figure 1, was designed similar to the setup in [3]. Linksys router is flashed with the OpenWrt firmware, and tcpdump is installed for capturing the traffic. Traffic passing through the router was captured with tcpdump and stored as a pcap file to the local server for further processing. To extract the traffic of each device from the captured traffic, we use the MAC address of the device with tcpreplay.

Data collection lasted for 20 days (i.e., measurements) where each measurement is divided into 24 hours starting from midnight. Devices such as light bulbs, strips, smart plugs, and sockets were scheduled to turn on every 30 minutes and turn

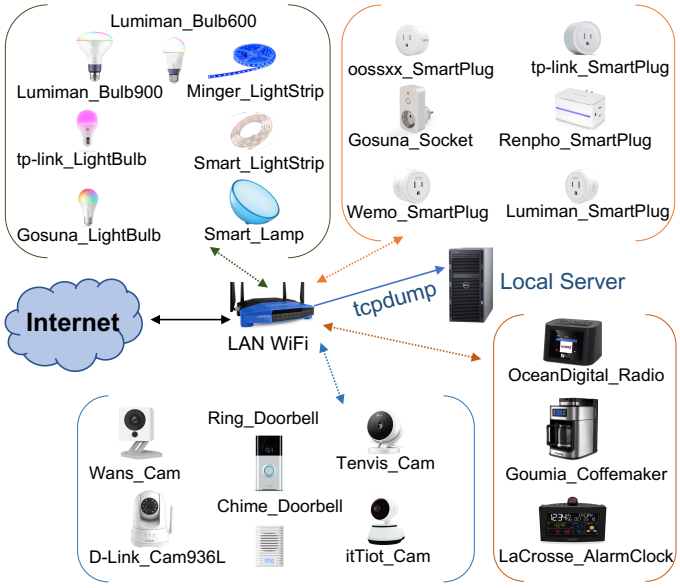

Fig. 1: Testbed for traffic collection from the 22 IoT devices

off after 5 minutes. Devices such as doorbells and cameras were configured to send notifications when they detect motion in the lab or light change, which was prompted by automated switching. After partitioning the traffic between devices, we remove IP address and MAC address associated with the device, to remove any bias that may occur in classifying the devices. Over the 20 days, we collected around 16 GB of network traffic.

IV. Device Fingerprinting using Network Traffic

In this section, we present details of the Locality-Sensitive IoT Fingerprinting (LSIF) system. We first present a short background on the Nilsimsa hash. Then, we present the LSIF system to identify network devices without a need for feature extraction or packet inspection.

A. Locality Sensitive Hash

Previous studies employed locality-sensitive hashing for spam detection [7], [11], and malware classification [9]. In this study, we use a locality-sensitive hash function, namely Nilsimsa [7], for device identification. Small changes in input produce entirely different hash values for the cryptographic hash algorithms, Nilsimsa produces similar hash values with a small difference in the input. Nilsimsa generates a 32-byte hash output representing the distribution of the trigrams (i.e., group of three bytes) in an input. The similarity score of two Nilsimsa hash values ranges from -128 (i.e., completely different) to 128 (i.e., completely similar) and is calculated by subtracting 128 from the number of similar bits in the hashes.

If we compute the Nilsimsa hashes of two different traffic flows from similar devices, we will expect the similarity score of these two hashes to be high compared to hashes of traffic from different devices. Figure 2 shows the average hash value similarity score as a color-coded matrix for five illumination devices and four smart plugs. The similarity score is mostly higher between the same kind of devices and between devices from the same manufacturer. The maximum similarity score is achieved when two different traffic flows from the same device are compared (i.e., cells on the diagonal).

	Gosuna_LightBulb	Lumiman_Bulb600	Lumiman_Bulb900	tp-link_LightBulb	Minger_LightStrip	Lumiman_SmartPlug	Renpho_SmartPlug	oossxx_SmartPlug	tp-link_SmartPlug
Gosuna_LightBulb	127	65	61	28	25	71	60	65	23
Lumiman_Bulb600	65	126	74	30	30	74	78	79	24
Lumiman_Bulb900	61	74	127	29	28	74	67	70	25
tp-link_LightBulb	28	30	29	111	73	37	32	30	75
Minger_LightStrip	25	19	23	78	124	31	27	23	61
Lumiman_SmartPlug	71	74	74	37	33	128	79	74	35
Renpho_SmartPlug	60	78	67	32	27	79	127	74	33
oossxx_SmartPlug	65	79	70	30	31	74	74	127	22
tp-link_SmartPlug	23	24	25	75	75	35	33	22	117

-130 -110 -90 -70 -50 -30 -10 10 30 50 70 90 110 130

Fig. 2: Nilsimsa hash similarity score for sample devices

B. Locality - Sensitive IoT Fingerprinting (LSIF)

Most of the IoT devices perform simple tasks, with well-defined traffic patterns [15]. Hence, it is possible to create a set of signatures to identify the device or detect anomalies in the behavior of the device. Network traffic of the device can slightly vary based on different factors, such as environmental conditions or configuration parameters. Thus, traffic patterns might slightly change from the same IoT device and need to be reflected in the signature mechanism. This makes Nilsimsa be an excellent option to generate signatures for IoT devices as it generates hash outputs where similar inputs produce similar hashes. LSIF employs Nilsimsa to generate a set of signatures for each device it wants to profile and uses the signature of traffic flow from a new device to identify the device.

The overall operation of LSIF is shown in Figure 3. For each device, LSIF generates Nilsimsa hashes from the network traffic flows of the device. We define a flow as a sequence of network packets sent from/to a given source MAC address (IoT device). Then, it stores the hashes of the device in a signature database. Note that this operation is performed only once, thus, it does not impose future overheads. When a new device joins the network, Nilsimsa hash of the device is computed from the network traffic flow and compared to the hash values in the database. The new device is labeled with the label of the device with the highest average hash similarity score.

LSIF is simple, at the same time, achieves high accuracy in identifying the devices. Compared to previous studies [1], [10], [15], our approach does not require feature extraction, parameter tuning, and frequent update of the model, most of which are typically computationally intensive tasks or requires expert knowledge. Also, LSIF does not require/assume the operating states of the device and works well when the new device is being set up, used by the user or just in idle state. Furthermore, LSIF can be extended to detect anomalies in a device's traffic behavior. If there is an anomaly in the traffic flow of a device, the Nilsimsa signature generated from the anomalous traffic flow will deviate considerably from the signature of the benign traffic stored in a system. As anomaly detection is not the focus of this paper, we leave it for further research and analyses.

V. Evaluation

In this section, we present the evaluation results of LSIF on traffic flow data of IoT devices collected in our laboratory environment. The dataset contains 20 day measurements for each device. Each measurement is split into flows of 10 minutes. In our evaluation for each device, we used random 100 flows from one day as signatures for the device, and all flows from rest of the nineteen days are used for testing. Note that, signatures are created only once and stored in a database thus, they do not pose any future overhead. We present the average of all 20 possible selection of days from which signatures are generated.

A. Analysis of LSIF Performance

Average evaluation results are provided in Table I, in terms of precision (i.e., $TP/(TP+FP)$), recall ($TP/(TP+FN)$),

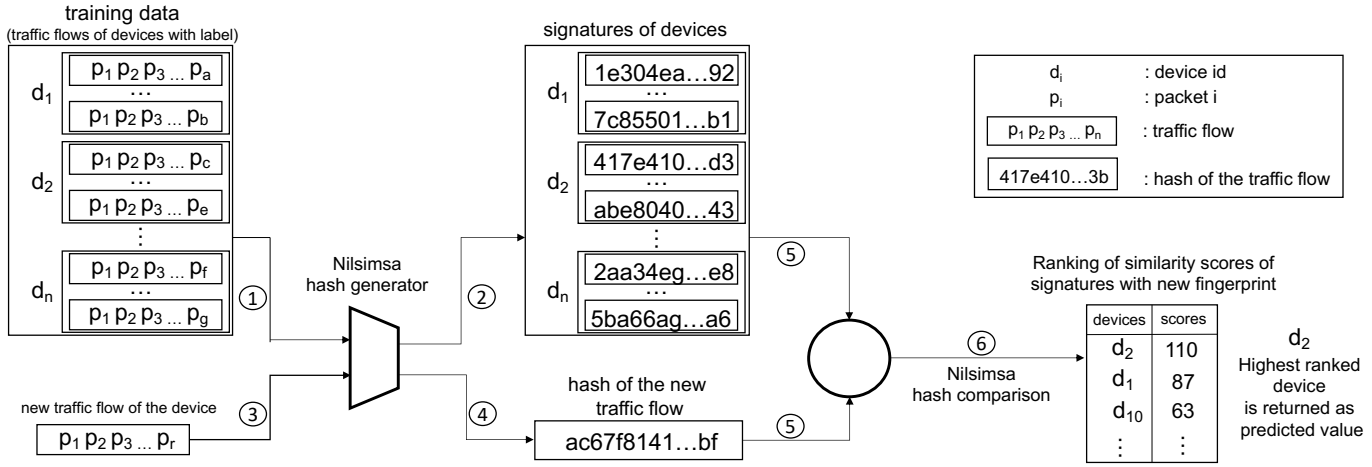

Fig. 3: Overall LSIF system operation where set of signatures for n devices are calculated with Nilsima hash generator and stored in a database, and a new instance of traffic flow from device $d2$ is identified by comparing the hash of its traffic flow to signatures stored in the system

F1-score (i.e., $2/(1/precision + 1/recall)$), accuracy (i.e., $(TP + TN)/(TP + FP + TN + FN)$), specificity (i.e., $TN/(FP+TN)$) and, AUC (i.e., $(1/2)\{TP/(TP+FN)+TN/(TN+FP)\}$) where TP, TN, FP, and FN stand for true positives, true negatives, false positives and false negatives, respectively. For eleven devices out of 22, both precision and recall are equal to 100%, meaning that LSIF was able to identify all devices correctly. The rest of the eleven devices without the precision or recall being perfect is shown in Figure 4. We see that both precision and recall are higher than 80% for eight of the devices in the figure. However, for Ring_Doorbell recall is less than 10%.

Figure 5 depicts the confusion matrix for devices with imperfect precision or recall. Rows show the actual values, and columns show predicted values in percentage. We see that Ring_Doorbell with the lowest recall is mostly confused

TABLE I: Average evaluation results

Precision	Recall	F1-score	Accuracy	Specificity	AUC
93%	90%	90%	100%	100%	95%

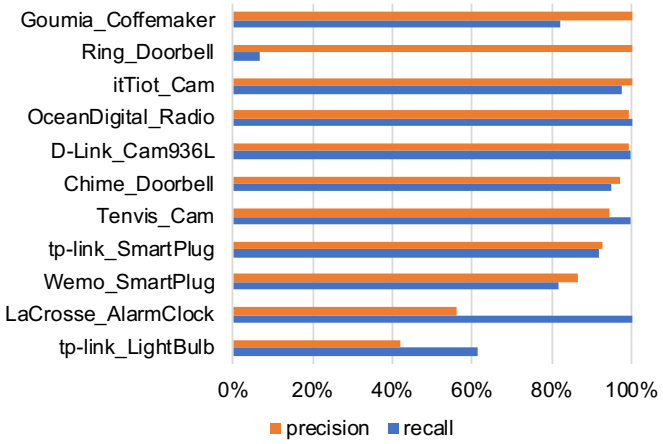

Fig. 4: Devices with imperfect precision or recall

TABLE II: Average time overhead to generate the hash with average precision and recall when only a fraction of the traffic flow is used for the identification

Traffic Proportion	100%	80%	60%	40%	20%
Precision	93%	93%	93%	92%	92%
Recall	90%	89%	89%	88%	87%
Time (ms)	1002	814	611	417	223

with LaCrosse_AlarmClock and tp-link_SmartPlug. We also see that LaCrosse_AlarmClock lead to confusion in most of the devices, tp-link_LightBulb being the most affected device after Ring_Doorbell. On the other hand, tp-link_LightBulb and tp-link_SmartPlug were distinguishable even if they belong to the same vendor.

Since LSIF identifies the device utilizing the hash of its traffic flow, the overhead of the device identification increases linearly with the size of the traffic from the device. We analyze how much traffic is needed to identify the device with acceptable accuracy. Table II shows the result of evaluation when a fraction of the 10 minute traffic flow is used for device identification. Results are an average of all possible cases considering a sliding window with a fraction of the flow. We see that there is not much performance loss with 80% or 60% of the traffic flow. Even when 20% of the data is used, the performance drops by only 3%, indicating that LSIF can identify the device from any point of the flow using only a fifth of the data. Table II also shows a average time overhead to generate the hash of the flow. Time overhead analyses were conducted on *Dell Latitude 3480 computers with Intel Core i5-7200U CPU @2.50 GHz*. LSIF functionality mainly consist of generating the hash of the flow that needs to be identified and comparison of the hash to the signatures stored in a database for identification. Time overhead mainly dominated by the time required to generate the hash of the flow because time for hash comparison and identification is negligible 4.4 ms for network consisting of 22 different

Actual \ Predicted	Goumia_Coffemaker	Ring_Doorbell	itTiot_Cam	OceanDigital_Radio	D-Link_Cam936L	Chime_Doorbell	Tenvis_Cam	tp-link_SmartPlug	Wemo_SmartPlug	LaCrosse_AlarmClock	tp-link_LightBulb
Goumia_Coffemaker	82.1	0.0	0.0	0.0	0.0	0.1	2.2	0.3	1.2	10.7	3.4
Ring_Doorbell	0.0	6.6	0.0	0.0	0.0	6.0	0.2	26.7	3.6	52.8	3.0
itTiot_Cam	0.0	0.0	97.3	0.0	0.0	0.0	1.3	0.0	0.9	0.3	0.0
OceanDigital_Radio	0.0	0.0	0.0	100.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
D-Link_Cam936L	0.0	0.0	0.0	0.0	99.8	0.0	0.0	0.0	0.0	0.2	0.0
Chime_Doorbell	0.0	0.0	0.0	0.0	0.0	95.0	0.3	0.0	0.0	4.2	0.3
Tenvis_Cam	0.0	0.0	0.0	0.0	0.0	0.0	99.7	0.0	0.0	0.2	0.0
tp-link_SmartPlug	0.0	0.0	0.0	0.0	0.0	0.5	0.3	91.8	0.0	4.4	2.8
Wemo_SmartPlug	0.0	0.0	0.2	0.1	1.9	0.0	8.7	0.3	81.4	6.2	0.9
LaCrosse_AlarmClock	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	100.0	0.0
tp-link_LightBulb	0.0	0.0	0.0	0.0	0.0	6.9	1.2	0.4	0.2	27.4	61.3

Fig. 5: Confusion matrix in percentage for misidentified devices (i.e., devices with either precision or recall less than 100%)

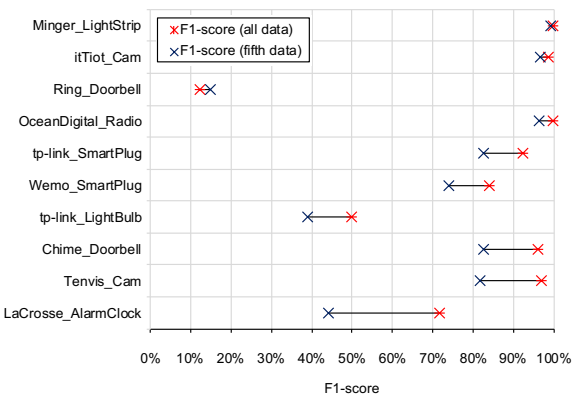

Fig. 6: Devices with F1-score affected by identification with a fifth of sample data

IoT devices with 100 signatures stored for each device in a database. Furthermore, time for identification does not change with size of the flow. On the other hand, time to generate the hash depends on the size of the flow. While decreasing the flow size may decrease the accuracy of the LSIF a little, it will also decrease the time overhead to generate the hash. Thus, it is up to network administrators to set the flow size to reasonable value considering the accuracy and time overhead of the LSIF. Figure 6 shows all devices with F1-score affected by using a fifth of the sample flow. We see that for devices such as Minger_LightStrip, itTiot_Cam, and Ring_Doorbell, there is a negligible impact on the F-1 score. On the other hand, for devices such as Tenvis_Cam or LaCrosse_AlarmClock, F-1 score decreases considerably.

LSIF correctly identified most of the devices using only a fifth of its traffic flow, which could begin at any point in the flow, without a need for updating the signature of the device. Previous approaches require the full traffic flow to extract relevant features, depend on the initialization traffic to identify the device, or need to update the model based on the amount of observed flow.

B. Comparison with Prior Work

In this section, we compare LSIF with SysID [1], IoT Sentinel [10], and to one-class machine learning classifier algorithms. IoT Sentinel uses a set of 23 packet header features, with a random forest classification algorithm to classify IoT devices. SysID uses a genetic algorithm to select relevant features of TCP/IP packet headers and uses machine learning algorithms such as Decision Table, J48 Decision Trees, and OneR. One-class machine learning algorithms can also be used by training the model on the traffic of the device and checking if the new traffic belongs to the same class or not. We used the One-class Support Vector Machine (OSVM) and Elliptic Envelope (EE) as they were used by [20] with the purpose of comparison with their traffic fingerprinting method. OSVM is a supervised machine learning algorithm used for classification problems. The algorithm creates a hyperplane (i.e., an n-dimensional plane) to separate two different classes. The elliptic envelope is a supervised machine learning algorithm that models the data as a high dimensional Gaussian distribution with possible covariances between the features. EE tries to find an ellipse that represents data the best.

The comparison was performed on the dataset that was used in the IoT Sentinel study as SysID also employs the same data for analysis. The dataset consists of the traffic flow of 23 IoT devices with different vendors (i.e., D-Link, Edimax Plug, Hue, TP-Link Plug, and Smarter). For each device, there exist 20 measurements, and each measurement is collected when the device initiates its setup phase. We evaluated LSIF by assigning only one measurement as device signature and use the rest of nineteen measurements for testing. Figure 7 shows the comparison of results for all IoT devices grouped by the manufacturer using Elliptic Envelope (EE), One-class Support Vector Machine (OSVM), System Identifier (SysID), IoT Sentinel, and Locality-Sensitive IoT Fingerprinting (LSIF).

For devices such as D-Link Day Cam, D-Link Door Sensor, D-Link Home Hub, and D-Link Cam, all classifiers achieve high classification performance where IoT Sentinel, SysID and LSIF outperforming machine learning techniques. For other devices, EE and OSVM perform better, especially in D-Link Siren, achieving 75% and 81% performance, respectively. For three devices out of the eight, LSIF has the highest performance; for one, it has the second-best; and for two, it has the third-best. All classifiers were able to classify Hue devices with high accuracy with LSIF outperforming other classifiers and followed by IoT Sentinel. On smarter devices, SysID achieves the highest accuracy with 92% for Smarter Coffee and 100% for Smarter iKettle2 and is followed by IoT Sentinel and LSIF.

Both Smarter Coffee and Smarter iKettle2 have a simple and similar setup phase, exchanging around 10-15 packets. Since SysID automatically searches for unique features within the dataset, it adjusts to the Smarter devices. While the simple nature of the traffic flow degrades the performance of other classifiers, SysID could capture uniqueness to achieve higher performance than others.

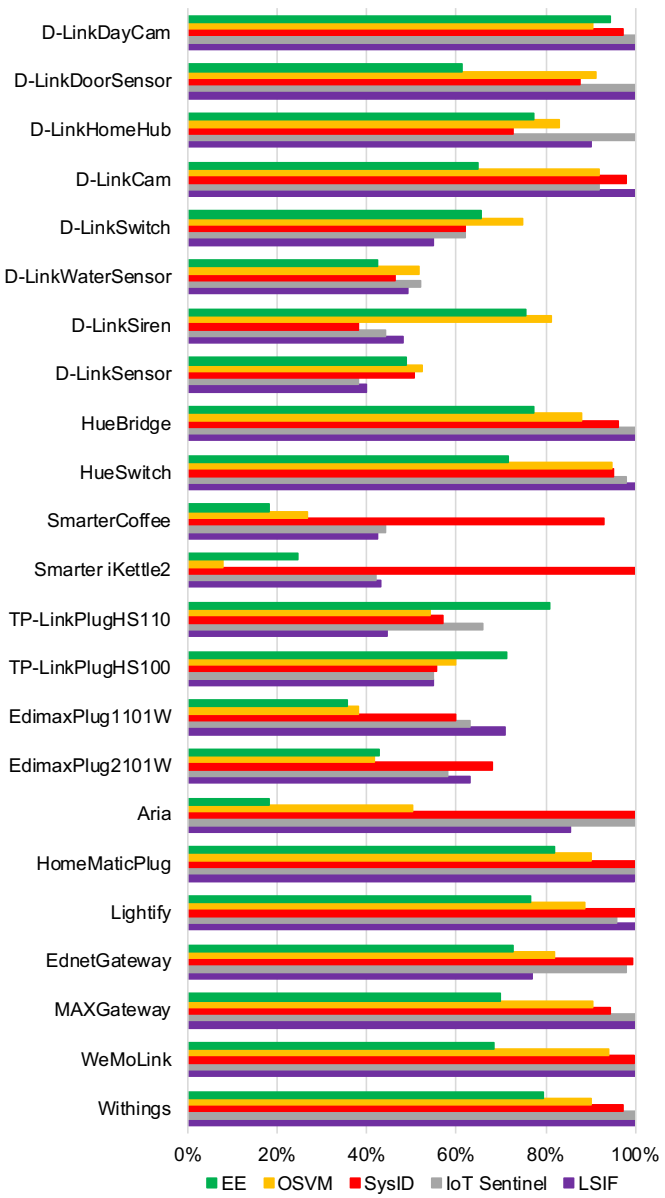

Fig. 7: IoT classification comparison

Additionally, while EE and OSVM achieve higher classification performance on TP-Link plugs, they fall behind for the Edimax plugs. TP-Link plugs have a more complex setup phase compared to Smarter devices, thus enabling machine learning classifiers to perform better than the SysID which probably stuck on a local optima while searching for unique features. LSIF performs better on Edimax plugs compared to TP-Link plugs, probably due to the more unique set of protocols in the Edimax plugs' setup phase. On average, there exist 5 to 7 different protocols in the traffic flow of the TP-Link plugs, whereas there are 8 to 12 protocols for Edimax plugs. The impact of the diverse set of protocols also increases the classification performance of the IoT Sentinel.

For the rest of the devices from different manufacturers, EE and OSVM perform worse than other classifiers, and LSIF achieves high accuracy except for Aria and EdnetGateway. LSIF confused Aria with Smarter Coffee and Smarter iKettle2 since they all have a simple setup phase, and Ednet Gateway is confused with Edimax and TP-Link Plugs.

LSIF achieves comparable results with previously proposed approaches IoT Sentinel and SysID as well as machine learning techniques EE and OSVM. Approaches perform similarly except for the Smarter devices where SysID considerably outperforms others. LSIF performs the best or equal to the best with 10 of the 23 IoT devices.

The weaknesses of LSIF compared to IoT Sentinel and SysID are as follows: Simple traffic flows (observed in SmarterCoffee and Smarter iKettle2) may hurt the classification performance of LSIF. Increasing flow size may increase the overhead of generating the hash. The flow size does not impact IoT Sentinel much, as it only focuses on some portion of the flow from the beginning. On the other hand, larger data increases the overhead of SysID because of expanded search space for relevant features.

Overall, LSIF has numerous strengths compared to the IoT Sentinel and SysID. Mainly, LSIF does not require feature extraction or tuning the parameters to obtain globally unique features. We showed that LSIF could obtain acceptable classification performance even if only a percentage of the flow is used starting from any position of the flow. On the other hand, IoT Sentinel focuses on device initialization traffic. Also, SysID may suffer in classification performance as unique feature extraction with genetic algorithms may obtain local optima results with a more extensive search space.

VI. Conclusion

In this paper, we present Locality-Sensitive IoT Fingerprinting (LSIF) for the identification of IoT devices based on Nilsimsa hash of their traffic flow. Different from previous studies, LSIF does not require feature extraction from the traffic, a lengthy process needing fine-tuning by an expert. We showed the effectiveness of the proposed method on two IoT traffic datasets and compared LSIF to recent studies. LSIF performed better than the prior work with most of the devices using significantly fewer computations.

In the future, we would like to extend the analysis of LSIF to larger datasets, inspect minimal packet stream length for identification, and examine other locality-sensitive hash algorithms. We will also inspect efficient mechanisms to filter input data to improve classification accuracy. Additionally, we would like to analyze how locality-sensitive hashing can be utilized to detect anomalies in the behavior of IoT devices and prevent attacks such as botnets or DDoS.

References

[1] A. Aksoy and M. H. Gunes. "Automated IoT Device Identification using Network Traffic" IEEE ICC, pp. 1-7, 2019

[2] A. Alruban, N. Clarke, F. Li, and S. Furnell. "Biometrically Linking Document Leakage to the Individuals Responsible" Springer Cham, pp 135-149, 2018.

[3] A. Sivanathan, H.H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics" IEEE Transactions on Mobile Computing 18 (8), 2019

[4] B. Charyyev, M.H. Gunes "IoT Event Classification Based on Network Traffic", IEEE International Conference on Computer Communications (INFOCOM) Workshops July 2020 Toronto, Canada

[5] C. Neumann, O. Heen and S. Onno. "An Empirical Study of Passive 802.11 Device Fingerprinting" IEEE ICDCSW, pp. 593-602, 2012

[6] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. "Ddos in the iot: Mirai and other botnets" IEEE Computer, 50(7), 80–84, 2017

[7] E. Damiani, S.D.C. di Vimercati, S. Paraboschi, P. Samarati. "An Open Digest-based Technique for Spam Detection" ISCA PDCS, 2004

[8] F. Lanze, A. Panchenko, B. Braatz and A. Zinnen. "Clock skew based remote device fingerprinting demystified" IEEE Globecom, 2012

[9] H. M. Kim, H. M. Song, J. W. Seo and H. K. Kim. "Andro-Simnet: Android Malware Family Classification using Social Network Analysis" IEEE PST, pp. 1-8, 2018

[10] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. Sadeghi and S. Tarkoma. "IoT SENTINEL: Automated Device-Type Identification for Security Enforcement in IoT" IEEE ICDCS, pp. 2177-2184, 2017

[11] Muhammad N. Marsono. "Packet-level open-digest fingerprinting for spam detection on middleboxes" Network Management, (22)12-26, 2012

[12] M. Pirker, P. Kochberger, and S. Schwandter. "Behavioural Comparison of Systems for Anomaly Detection" ACM ARES. Article 14, 2018

[13] "SpamAssasin email datasset" http://spamassassin.apache.org/.

[14] S. Kennedy, H. Li, C. Wang, H. Liu, B. Wang, and W. Sun. "I Can Hear Your Alexa: Voice Command Fingerprinting on Smart Home Speakers." IEEE CNS, pp. 232-240, 2019

[15] V. H. Bezerra, V.G. Turrisi da Costa, S.B. Junior, R. S. Miani and B. B. Zarpelao. "IoTDS: A One-Class Classification Approach to Detect Botnets in Internet of Things Devices" Sensors 2019, 19(14), 3188

[16] W. P. Filho, C. Ribeiro and T. Zefferer. "An Ontology-Based Interoperability Solution for Electronic-Identity Systems" IEEE SCC, 2016

[17] X. Zhou, A. Hu, G. Li, L. Peng, Y. Xing and J. Yu. "Design of a Robust RF Fingerprint Generation and Classification Scheme for Practical Device Identification" IEEE CNS, pp. 196-204, 2019

[18] Y. Cheng, X. Ji, W. Xu, Y.C. Chen. "DeMiCPU: Device Fingerprinting with Magnetic Signals Radiated by CPU" ACM CCS, 2019

[19] Y. Meidan, V. Sachidananda, Y. Elovici, A. Shabtai. "Privacy-Preserving Detection of IoT Devices Connected Behind a NAT in a Smart Home Setup" 2019. arXiv:1905.13430v1

[20] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, Y. Elovici. "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders" IEEE Pervasive Computing, 17(3), 12-22, Jul.-Sep. 2018

[21] Y. Song, Q. Huang, J. Yang, M. Fan, A. Hu, and Y. Jiang. "IoT device fingerprinting for relieving pressure in the access control" ACM TURC, Article 143, 2019.

[22] Y. Tu, Z. Zhang, Y. Li, C. Wang and Y. Xiao. "Research on the Internet of Things Device Recognition Based on RF-Fingerprinting" IEEE Access, (7), 37426-37431, 2019

[23] Y Zeng, Z Qi, W Chen, Y Huang, X Zheng. "TEST: an End-to-End Network Traffic Examination and Identification Framework Based on Spatio-Temporal Features Extraction." 2019. arXiv:1908.10271