Your Smart Home Can’t Keep a Secret: Towards Automated Fingerprinting of IoT Traffic

Shuaike Dong[*]
ds016@ie.cuhk.edu.hk
The Chinese University of Hong Kong

Zhou Li
zhou.li@uci.edu
University of California, Irvine

Di Tang
td016@ie.cuhk.edu.hk
The Chinese University of Hong Kong

Jiongyi Chen
jiongyi_chen@126.com
National University of Defense Technology

Menghan Sun
sm017@ie.cuhk.edu.hk
The Chinese University of Hong Kong

Kehuan Zhang
khzhang@ie.cuhk.edu.hk
The Chinese University of Hong Kong

ABSTRACT

The IoT (Internet of Things) technology has been widely adopted in recent years and has profoundly changed the people’s daily lives. However, in the meantime, such a fast-growing technology has also introduced new privacy issues, which need to be better understood and measured. In this work, we look into how private information can be leaked from network traffic generated in the smart home network. Although researchers have proposed techniques to infer IoT device types or user behaviors under clean experiment setup, the effectiveness of such approaches become questionable *in the complex but realistic network environment*, where common techniques like Network Address and Port Translation (NAPT) and Virtual Private Network (VPN) are enabled. To this aim, we propose a traffic analysis framework based on sequence-learning techniques like LSTM and leveraged the temporal relations between packets for the attack of device identification. We evaluated it under different environment settings (e.g., pure-IoT and noisy environment with multiple non-IoT devices). The results showed our framework was able to differentiate device types with a high accuracy. This result suggests IoT network communications pose prominent challenges to users’ privacy, even when they are protected by encryption and morphed by the network gateway. As such, new privacy protection methods on IoT traffic need to be developed towards mitigating this new issue.

CCS CONCEPTS

• **Security and Privacy** → Traffic Analysis.

KEYWORDS

IoT Security; Privacy; Neural Networks

ACM Reference Format:
Shuaike Dong, Zhou Li, Di Tang, Jiongyi Chen, Menghan Sun, and Kehuan Zhang. 2020. Your Smart Home Can’t Keep a Secret: Towards Automated Fingerprinting of IoT Traffic. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIA CCS ’20), October 5–9, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3320269.3384732

1 INTRODUCTION

The Internet of Things (IoT) has been gaining increased popularity in recent years, and continues to expand in areas such as smart homes, smart cities, industrial systems, connected health products, and so on. According to a report from Forbes, the global IoT market will grow from 157 billion US dollars in 2016 to 457 billion US dollars by 2020, attaining a Compound Annual Growth Rate (CAGR) of 28.5% [1].

The Smart Home is a prominent use case of IoT, where multiple IoT devices work together to facilitate all kinds of user activities by sensing surroundings, interpreting human commands and providing feedback. However, the Smart Home can introduce new threats to residents’ privacy. Since network packets between IoT devices and remote servers can be intercepted, a motivated attacker can leverage such data to infer private information about the residents, like what IoT devices are installed and whether they are active. Leaking such information would cause grave consequences to the residents: e.g., a theft can break into the home when no one is inside by learning the status of the installed smart switch.

This paper aims to assess the privacy threat to smart home residents by evaluating different traffic-analysis approaches on datasets carrying real IoT traffic. Though a few recent works also investigated the privacy issues related to IoT network communications [9, 11, 37], those works all assume a local adversary (only the traffic between IoT device and gateway can be sniffed) or relatively simple network environment (e.g., traffic from devices can be easily separated). Whether traffic analysis is effective under a remote adversary or a more complex network is not yet assessed. In particular, we assume that the gateway may enable configurations that are common but hamper traffic analysis, like Virtual Private Network (VPN) and Network Address and Port Translation (NAPT). Under those settings, traffic flows belonging to different IoT devices could be merged to a single flow and the valuable information from fields like destination ports could be erased. Still, our study shows by

[*] Part of the work was done when visiting University of California, Irvine.

ASIA CCS ’20, October 5–9, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-6750-9/20/10…$15.00
https://doi.org/10.1145/3320269.3384732

exploiting the temporal relations between packets of an individual device, the device can be reliably identified.

More specifically, we found such temporal relations can be modeled by sequence model LSTM-RNN when grouping consecutive packets into a traffic window. We carefully designed the structure of both models and evaluated on two datasets filled with traffic generated from off-the-shelf IoT devices and non-IoT devices. The evaluation results show our models can achieve better accuracy comparing to the models widely used by existing works, like Random Forest. To highlight a few, our bidirectional LSTM model can achieve an accuracy of 99.2% and 97.7% on IoT devices in NAPT and VPN configuration. Even when a large-amount of non-IoT traffic is generated at the same time, it can still achieve 95.3% and 80.9% accuracy in these two configurations.

Contributions. We summarize the technical contributions made by this work as below.

(1) We present a traffic-analysis system, HomeMole, to automatically infer the IoT devices behind a smart home network even when traffic fusion like NAPT and VPN are enabled. We designed a basic LSTM model and a bidirectional LSTM model that are able to identify IoT devices based on the sniffed packets.
(2) We evaluated our system under two types of network configuration (NAPT and VPN) and two scenarios (pure-IoT and noisy environment). The results indicate that our framework could achieve high accuracy under those settings. Our models outperform the baseline model, Random Forest, due to the ability to model the temporal relations between packets.

To facilitate further research in this domain, we release the datasets and the models on our Github repository[1]. We believe by releasing datasets and models, we could help other researchers to investigate new traffic-analysis methods and IoT community to build a better defense.

Paper organization. The paper is organized as follows: Section 2 presents the background about the relevant previous work, smart home network, our adversary model and neural networks used in our paper. Section 3 describes the design of our framework. Section 4 presents the experimental results of our framework in different scenarios. Section 5 discusses alternative scenarios and defense. Section 6 discusses the limitations of this work. In the end, we conclude this study in Section 7.

2 BACKGROUND

2.1 Related Work

Network traffic analysis. Network traffic analysis has been shown reasonably effective when applied to anomaly detection [25, 43], software identification [32] and device fingerprints [10, 33]. For example, Sakthi et al. [33] applied statistical techniques on network traffic to create unique fingerprints for multiple wireless devices. Their evaluation results show even with the interference from other wireless users in the proximity, the approach can still reach a satisfying accuracy. Aksu et al. [10] introduced a wearable fingerprinting framework based on Bluetooth traffic. Their evaluation shows that wearable devices can be accurately fingerprinted using Bluetooth classic protocols. Taylor et al. [41] proposed an identification framework for smartphone apps called AppScanner, which extracts statistical features from network flows for classification tasks. Trained by different learning algorithms, AppScanner can reach the highest accuracy of 99.8% on 110 apps. Chen et al. [17] proposed an online traffic classification framework which utilizes kernel methods and deep neural networks. They evaluated their approach on 5 different protocols and 5 mobile applications, which achieved an accuracy of 99.84% and an accuracy of 88.43% correspondingly.

Traffic analysis in IoT domain. Following the rapid development of the IoT ecosystem, how to characterize and fingerprint of IoT devices has become a trending topic. In recent years, there have been some works on IoT traffic analysis [10–12, 29, 30, 35, 37]. Apthorpe et al. [11] performed case studies on four IoT devices. The results showed distinctive features were found in the network traffic, which could enable device identification and behavior inference. Marcus et al. [30] proposed an IoT device identification system IoT Sentinel. They extracted 23 features from raw packets, which include IP options, IP addresses, ports, as well as some network protocols from the link layer to the application layer. However, some of the features could not be directly used in our settings due to the adversary capabilities and complex network configurations (NAPT, VPN). For example, ARP and LLC are only capturable in a local area network environment. Also, SSDP and MDNS protocols only account for a small amount of traffic and could not holistically represent the network status. Sivanathan et al. [37] collected network traces of more than 20 IoT devices in a campus environment over 3 weeks and characterized the profiles of those IoT devices according to their traffic patterns. It also relies on an extensive feature engineering to select the salient features and some of them become vain when a complex network setting is applied, like NAPT and VPN. Abbas et al. [9] proposed a multi-stage privacy attack in smart home environment. They used machine-learning approaches for revealing different types of privacy information. However, their research is performed under a different scenario from our work. In their work, an attacker is located physically within the wireless range of the targeted users' smart devices while our model assumes the attacker can acquire network traffic between the gateway and the remote server. Due to the different capabilities of adversaries, the raw network traffic obtained in their work is different from ours, therefore leading to different solutions.

Privacy leakage in IoT apps. The issue of privacy leakage in smart home apps also draws attention from the security community in recent years. Celik et al. [16] used taint analysis in tracking sensitive information in IoT apps. Their evaluation shows the approach can mark the sensitive data flow at high accuracy. Babun et al. [13] proposed an NLP-based dynamic analysis tool to inform users of potential privacy risks when using an IoT app. The evaluation shows that the tool can reach an average accuracy of 94.25% with a minimal execution overhead. Surbatovich et al. [39] analyzed the risk of privacy leakage in if-this-then-that(IFTTT), an IoT automation framework. Using an information flow model, they found that around 50% of the unique IFTTT recipes violate confidentiality or integrity. Different from their works, we focus on uncovering privacy leakage issues in IoT network traffic.

[1] https://github.com/DongShuaike/iot-traffic-dataset.git

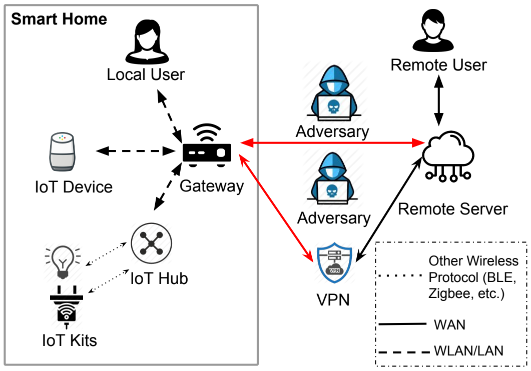

Figure 1: Network structure of a typical smart home.

2.2 Smart Home Network

We assume the network communication within a smart home involves four parties: IoT device, service provider, gateway and user. The communication schema is illustrated in Figure 1. Below we briefly overview it.

- The first type of IoT device senses the surrounding environment and sends notifications to its associated listeners, either periodically or immediately when the event takes place. For example, Samsung ST Motion Sensor detects when a person approaches in proximity [8] and notifies other IoT devices. Another type of IoT device is *IoT hub*, which acts as the "brain" (centroid controller) for other IoT devices (or *IoT kits*) in the close range. IoT hub is necessary to control IoT devices that use low-power protocols like BLE. Both regular IoT device and IoT hub can be controlled by user's commands sent remotely or locally.
- The IoT device interacts with the *service provider* operated by its manufacturer through Internet communications. The service provider is responsible for handling requests and relaying resources. To reduce the operational cost, many device vendors have moved their services to the public cloud infrastructure and leverage the cloud analytics, like AWS IoT Core [2], to process a massive amount of IoT data.
- A *gateway* is a bridge between in-home IoT devices and the remote service provider. A typical gateway (e.g., router) supplies two types of interfaces for in-home devices, WLAN and LAN. The communication with a service provider is through WAN interface.
- A *user* takes control of all smart devices in his/her home. There are usually two ways to interact with IoT devices, which are through mobile applications or human interactions (e.g., walking, talking, and touching).

Device identifier. IoT devices within the smart home network can be distinguished by device identifiers determined by specific network protocols. Each Wi-Fi device has a unique source IP address and MAC address. Though IoT kits might not get IP addresses in a smart home if they are not using WiFi, they can still obtain identifiers from IoT hub through other protocols (e.g., NwkAddr for Zigbee devices and Resolvable Private Address for BLE devices).

2.3 Insights into IoT Traffic

We carried out an exploratory analyse on the real-world IoT devices and public datasets to characterize their network communication patterns. We identify several insights which highlight the uniqueness of IoT traffic comparing to the desktop and mobile traffic.

1) *The devices belong to the same category have similar traffic patterns.* As one example, we show the traffic patterns of Amazon Echo Dot and Google Voice Assistant (both are voice assistants) when they are waken up in Figure 16 and Figure 17 of Appendix. As can be seen, when the voice command is recognized, both of them generate a traffic burst, followed by a period of continuous communication with the remote server.

2) *Same devices in different running statuses can have significantly different traffic patterns.* From Figure 18 of the Appendix we can see, Orvibo switch produces dense, but a smaller amount of traffic in standby mode compared with that in running mode. Another example is Figure 19, from it we know Xiaomi camera generates much more traffic in WAN shooting mode than in LAN shooting mode.

Device (%)	IPv4	UDP	TCP	TLS	HTTP	DNS	O
Google Home	100.0	1.5	98.1	26.6	0	0.6	0.4
Echo Dot	100.0	~0	~100.0	14.6	~0	~0	~0
Tmall Assist	99.6	0	99.6	21.0	5.5	~0	0.4
360 Cam (LAN mode)	99.7	78.6	21.1	0.4	0	0	0.4
360 Cam (WAN mode)	100.0	99.9	0.1	~0	0	0	0
Orvibo	99.6	0.2	99.4	~0	0	0	0.4
Broadlink	99.7	99.7	0	0	0	0	0.3
Tplink	99.4	0.1	99.3	50.9	0	0	0.6
Xiaomi Hub	99.5	99.5	0	0	0	0	0.5
Noise - mobile	87.6	5.5	80.1	4.5	1.7	2.2	12.4
Noise - tablet	87.0	0.5	86.4	0.6	0.1	0.1	13.0

Table 1: Protocol distribution (O means other protocols).

3) *The choice of protocol is diverse.* The communication between devices and remote services must follow certain conventions, which is reflected in the chosen protocols. In Table 1 we show such diversified choices in terms of packets ratio under different network protocols. Note that we only measure protocols at the network layer, the transmission layer and the application layer, discarding those from lower layers. According to our statistics, most traffic captured by an attacker is IPv4-based. For devices with large volumes of traffic, like network camera, UDP is usually adopted. Compared with non-IoT devices, IoT devices generate much less HTTP traffic for communication. Though DNS is usually used by previous works for device identification [11, 12, 37], our results show that its ratio is quite low compared to other protocols. As such, simply relying on DNS does not guarantee device identification especially when the traffic observed comes from an incomplete session.

2.4 NAPT and VPN

When the network traffic is observed between the gateway and the service provider, the original device identifiers may be obscured.

Firstly, the gateway could strip off the source MAC address from the packets [3]. Even the port information can be changed under NAPT (Network Address and Port Translation) or VPN (Virtual Private Network). Below, we describe how changes are made by NAPT and VPN.

NAPT. To conserve the limited global IPv4 resource, NAPT is developed to enable the sharing of one IP address among different devices. In particular, NAPT modifies the network-layer and transport-layer identifiers like destination IP address and destination port numbers of inbound packets[38]. For outbound packets, the source IP address and the source port are translated. In both cases, the IP address of a local device is replaced with the gateway's IP address. The gateway using NAPT holds a *translation table* which records the mapping of addresses and ports so that packets will be routed to the right destination.

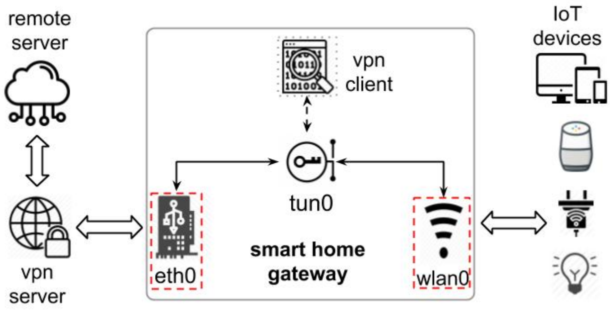

Figure 2: VPN-enabled gateway.

VPN. VPNs are often used to interconnect different networks to form a new network with a larger capacity [19]. Based on the IP tunneling mechanism, hosts in different subnets can communicate with each other and the delivered information can be kept secret with authentication and encryption.

Figure 2 shows the network structure after deploying a VPN-enabled gateway. Different from normal routers, a VPN-enabled gateway owns three network interfaces – `wlan0`, `eth0` and `tun0`. Among them, `wlan0` works as the entrance of LAN, collecting and delivering packets from local devices. The Ethernet interface `eth0` holds the connection between the gateway and WAN. `tun0` is created by VPN client process. For every packet from `wlan0` to `eth0`, the VPN client first encrypts the original packet into a payload and constructs a new packet. The new packet is then delivered to a VPN server and gets decrypted. The VPN server then forwards restored packets to their original destinations. From the viewpoint of destination remote server, the original metadata like the source IP address and the source port are completely hidden, which protects user's privacy against on-path eavesdroppers.

2.5 LSTM-RNN

Recently, deep neural network (DNN) has been gaining traction in the security domain and showing promising results, with its capability of feature representation learning. For example, Rimmer et al. demonstrated that websites visited by Tor users can be fingerprinted automatically with DNN [34].

Recurrent Neural Network (RNN) is one type of DNN that is good at handling temporal-related sequences. With multiple recurrent cells connected, the output of a previous cell can be passed to the current one. In this way, historical information is kept and forwarded. Among different implementations of RNN, LSTM (Long Short-Term Memory)-RNN has become a popular choice, as it is able to address the weakness of other RNNs like exploding and vanishing gradients. It provides a novel memory cell consisting of three different gates: input gate, forget gate and output gate. These gates are used to process the data transferred from the previous memory cell and manipulate the current cell state. LSTM-RNN has achieved many successes in different areas, such as speech recognition [21], medical diagnose [28], and system log analysis [18]. Figure 3 shows the structure of a basic LSTM-RNN.

Inspired by recent research, we find our problem is a natural fit for LSTM-RNN models. Similar to system logs, traffic generated by IoT devices can be organized in chronological order. There exists contextual dependency between packets based on the running states of the device and such dependency can be modeled by LSTM-RNN models. In Section 3, we describe our LSTM-RNN models in details.

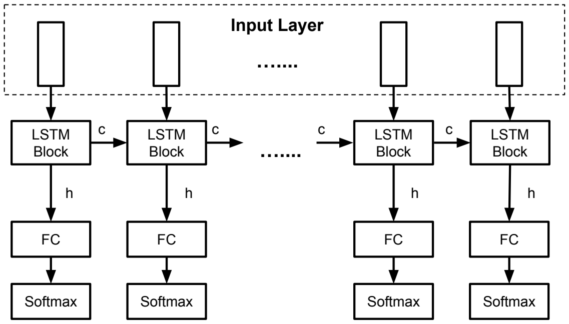

Figure 3: Structure of a basic LSTM used in our work (FC means fully-connected layer).

2.6 Adversary Model

The goal of our adversary is to identify *active* IoT devices in a targeted smart home. As previous work [9] shows, identifying the type of devices in a smart home not only can leak sensitive information like user preference for IoT products but also acts as the building blocks for further inference, like user behavior detection. Such privacy leakage can bring negative impacts to users' daily lives. For example, an ISP can infer the device information and sell it to advertisers who like to do targeted advertising. Or a theft can pick the time when the user is not at home by sniffing the outbound traffic and inferring the status of the installed surveillance camera.

In this work, we consider passive eavesdroppers who can observe the *encrypted* network traffic flowing *between the gateway and the remote service*. More importantly, there are two realistic settings that are not considered by prior works. On one hand, we assume that NAPTs or VPNs are enabled in the gateway so that the original device identifiers are replaced by the gateway's and the traffic belonging to different devices are *merged*, even the contacted remote server becomes opaque to the adversary. On the other hand, we assume multiple devices (including IoT devices and non-IoT devices such as mobile phones and tablets) may work *simultaneously* such that their packets are interleaving. As shown in [37], non-IoT devices usually have a higher rate in generating packets and their volume is larger than that of IoT devices, which means the existence of non-IoT devices can significantly distort the original statistical features learned on IoT devices.

Previous works [9, 12, 41] assumed that the adversary can sniff traffic within the smart home network (i.e., *local adversary*) or traffic fusion is not performed by the gateway. In their scenarios, the flows from different devices are clearly separated based on device identifiers. Unlike previous works, the remote adversary in our study is more realistic and the traffic analysis is much more challenging to perform.

3 SYSTEM DESIGN

3.1 System Overview

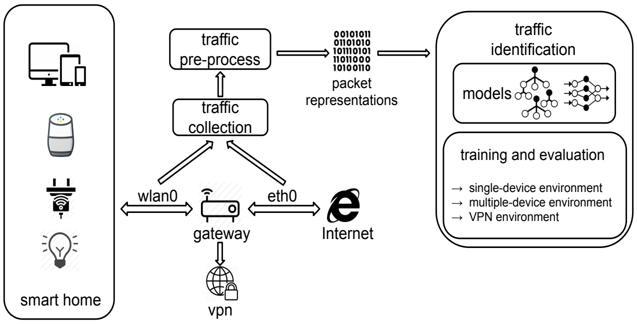

Figure 4: System design of HomeMole

The goal of HomeMole is to identify the active IoT devices in a smart-home environment based on their network traffic. To this end, HomeMole takes three steps – traffic collection, traffic pre-processing and traffic identification. As Figure 4 shows, we first set up the smart home environment and collect the raw traffic generated by IoT devices and non-IoT devices under different settings. After that, HomeMole pre-processes the traffic and convert it into the form that can be recognized by its identification module. Finally, the identification module uses the pre-processed data to train the models and perform the device identification task.

Different from those doing fingerprinting at flow level[17, 41], HomeMole works at *packet* level, which means a lable will be given to all packets after they are processed by our models. As such, HomeMole is able to work in *online* mode and give prompt results of current device status. Apart from that, packet-level identification can be easily applied to tasks with any other granularity, which is discussed in Section 5.

3.2 Traffic Collection

While some traffic datasets in prior works were released [37], we found that they could not be used in our study, as most of them have different target environments compared with ours. We focus on more complex network settings with NAPT and VPN enabled, where LAN data is also purged due to our adversary model out of the gateway. As such, we set up our own smart home environment and collected the traffic.

We set up the environment in a campus laboratory with 15 devices, including 10 IoT and 4 non-IoT devices. Table 2 shows the details of our devices. Our devices can be divided into six categories: voice assistant, IoT hub, IoT kits (smart plug), network camera, interactive machine, and non-IoT devices. The devices have different interaction modes and traffic patterns, which can be helpful to depict the overall picture of IoT device traffic.

We use a Raspberry Pi [7] as the gateway. A typical Raspberry Pi provides two network interface cards – `eth0` and `wlan0`. To simulate NAPT, we connect `eth0` to the Internet and then enable the Linux service `hostapd` to create an access point with `wlan0` network card of Raspberry Pi. Next, we create rules for `iptables` so that packets can be forwarded from `wlan0` to `eth0` and vice versa. For VPN, we established a virtual machine with DigitalOcean Droplets service [4] and used it as our VPN server, we then set up `openvpn` client on our Raspberry Pi to enable VPN tunneling.

device	MAC	type
echo dot	88:71:e5:ed:be:c7	voice assistant
google home	f4:f5:d8:db:61:84	voice assistant
tmall assist	18:bc:5a:19:eb:7d	voice assistant
xiaomi hub	78:11:dc:e1:f0:6b	hub
360 camera	b0:59:47:34:16:ff	network camera
xiaobai camera	78:11:dc:cf:c8:f1	network camera
tplink plug	30:20:10:fb:7c:05	smart plug
orvibo plug	b4:e6:2d:08:63:0c	smart plug
broadlink plug	78:0f:77:1b:00:8c	smart plug
mitu story teller	28:6c:07:87:54:b0	interactive
xiaomi mobile	a4:50:46:06:80:43	non-IoT
xiaomi tablet	20:a6:0c:5a:42:10	non-IoT
sony mobile	28:3f:69:05:2d:b0	non-IoT
motorola mobile	44:80:eb:21:cb:95	non-IoT

Table 2: Devices used in our experiment

We collect network traffic under three different settings:

- *Single-device environment.* We assume only one device is active and we connect one IoT device to the gateway at a time.
- *Multi-device and noisy environment.* In this case, all IoT and non-IoT devices are connected to the gateway which has NAPT enabled. Several devices may work simultaneously, leading to traffic fusion.
- *VPN environment.* In addition to the above settings, we assume a global VPN is enabled, which means the traffic passing through the gateway will be processed by a VPN client. Traffic before and after VPN processing are both collected from `wlan0` and `eth0` at the same time.

To generate traffic, we adopt two strategies: automatic triggering and manual triggering. Automation can relieve the burden of tedious operations and manual triggering can simulate human-machine interaction in real environments.

Automatic triggering. For devices like smart plugs and network cameras that can be controlled by mobile apps, we use MonkeyRunner [5] to interact with the UI of mobile apps and trigger different functions of IoT devices. For devices like voice assistants directly controlled by human input, we replay the voice commands near them. For example, Google Home plays songs when it hears the command "sing a song". We record a list of different commands and play them in a loop with a proper interval. As smart devices may have different responses even hearing the same command, though the total amount of recorded commands is fixed, the traffic we collect varies.

Manual triggering. Manual triggering is used in collecting traffic from a multi-device scenario. In this setting, devices are set up in a shared room (laboratory), people coming in and out of this room can interact with the devices as they want. The functions triggered in this scenario and their time intervals are irregular and comply with what may happen in a real environment. Compared with the automatic triggering approach, manual triggering introduces more randomness to the dataset, which is helpful to the generalization of our models.

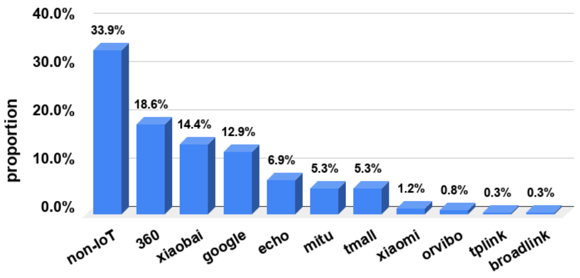

Figure 5: Proportion of collected packets.

We adopt the popular network analysis tool `tshark` to monitor `wlan0` and `eth0` simultaneously. The traffic collected is dumped into files with extension ".pcapng" and is then pre-processed before classification.

The collection lasts for 49.4 hours. In total, we collected **4.05 GB** traffic with **7,223,282** packets (those that cannot be obtained by our adversary model are excluded, like packets only being transmitted inside the LAN). The distribution of packets is shown in Fig 5. Due to the variant functionalities of devices and users' different habits, the collected traffic does not comply with a uniform distribution. Note that there are differences between the raw traffic collected at `wlan0` and what an attacker obtains due to identifiers of packets are modified by NAPT and VPN. To simulate the real adversary scenario, we then pre-process the traffic as Section 3.3 shows.

3.3 Traffic Pre-processing

Since the raw traffic cannot be directly used by `HomeMole`'s identification module, we pre-process the packets and convert them into datasets that can be used to train the model. We utilize a multi-platform packet parsing framework called `PcapPlus Plus` to do the task. The goal is to extract low-level but useful features from the packet and compose a numerical vector that can be processed by our models.

Feature extraction. Due to the encryption enforced by the communication, we extract features from the metadata of packet header. We select features from different layers – `frame length` and `epoch time` from `physical layer`, and `destination port number` from `transport layer`. In addition, we use a binary sequence to represent the protocols in packet transmission. We select 6 most common protocol types including `IP`, `TCP`, `UDP`, `TLS/SSL`, `HTTP` and `DNS`, according to our measurement (see Table 1). If a packet involves one of the protocols, the corresponding bit will be set to 1, otherwise 0. We set the last position of the 'binary string' to be `others` for the protocols beyond the previous 6 protocols. For example, a UDP-based DNS request is represented as `<1010010>` and a NTP packet is represented as `<1010001>`. The only feature we consider beyond metadata is the `direction` of the packet. We use 0 and 1 for inbound and outbound packets respectively.

Note that we do not use the domain name in DNS request/response like previous works [11, 12] for two reasons. Firstly, plaintext DNS information is unavailable under VPN or DNS encryption (e.g., DNS-over-TLS and DNS-over-HTTPS) [23, 24]. Secondly, as our result shown in evaluation, even without DNS information, `HomeMole` can achieve good accuracy. The destination `IP` is not used because it is periodically changed when the IoT vendors run the remote server on the public cloud, which has become a popular choice [2].

In the end, we concatenate all the selected 11 features and compose a one-dimensional vector as the representation of a packet `<dport, protocols, direction, frame length, time interval>`, as shown in Figure 6. Note that we compute `time interval` from the `epoch time` between two adjacent packets and use it as the feature to model the temporal relations between packets.

Packet labeling. After transforming each packet into a vector with eleven features, we label them to facilitate the training and evaluation of our identification module.

For NAPT environment, we extract the source MAC address and destination MAC address of each packet. Since the collection under NAPT is done at `wlan0`, the original information of MAC addresses is well kept. We compare them to MAC addresses of our devices and decide the label of that packet and its direction (inbound or outbound). Note that MAC addresses are only used in packet labeling. The physical location of our adversary model determines that he/she cannot take advantage of MAC addresses of an original packet, as they have been masqueraded by NAPT.

For VPN environment, as Section 2.4 shows, packets collected outside smart home (or between `eth0` and VPN server) are all merged into a single flow (packets with the same destination IP and port [41]), without any original identifier of sender/receiver. To identify the VPN packets and label those with their corresponding devices, we develop a mapping technique based on three observations obtained through our empirical analysis: (1) The size of a packet increases after being processed by VPN; (2) Multiple packets with different sizes can have the same size after the encryption performed by VPN; (3) There is a delay of packet transmission caused by VPN, which is usually shorter than 0.02 second.

Observation (1) and (2) can be reasoned through the cryptography algorithms used by `openvpn` servers. It provides three symmetric encryption algorithms – BF-CBC, AES-128-CBC and DES-EDE3-CBC. All of them are block ciphers through which encryption increases the size of packets. Observation (3) helps us reduce the scope for linking packets before and after the VPN. As a result, for each VPN-processed packet with timestamp t, we first check its `direction`. If it is inbound, we search its counterpart with smaller packet size in the time window $(t, t + 0.02]$. If it is outbound, the time window becomes $[t - 0.02, t)$. To evaluate the effectiveness of such a pairing method, we deploy a VPN-enabled gateway with only one device connected to it and collected the packets reaching and leaving the gateway simultaneously. We then apply the pairing method to all packets to obtain packet pairs and measure the rate of correct packet pairs. The overall accuracy reaches **98.8%**.

	dport	protocol							dir	packet size	time interval
packet 0	54087	1	0	1	0	0	0	0	0	0.384	0
packet 1	443	1	1	0	1	0	0	0	1	0.967	0.5e-2
packet 2	54087	1	0	1	0	0	0	0	0	0.384	2.3e-4
	… …										
packet n	53	1	0	1	0	0	1	0	1	0.684	5.1e-3

Figure 6: An example of traffic window.

3.4 Traffic Identification

In this section, we describe how we perform traffic identification with the pre-processed traffic datasets and the two types of models we use. The baseline model is targeted at single packet. Its goal is to learn whether the features extracted from an individual packet are enough to fingerprint the device it belongs to. We then present our sequence-based model, which takes a group of consecutive packets as input, to emphasize the gain of mining the hidden information between packets.

3.4.1 Single-packet Model. We consider Random Forest as our baseline model as it has been widely used in previous works on device fingerprinting [9, 37, 41]. A typical Random Forest is comprised of multiple single decision trees. During the training phase, inner decision trees are trained with different parts of the dataset and a final result is given based on the voting of those separate trees. In our work, we train a Random Forest model with the labeled packet vectors and the model predicts the device associated with each packet of the testing dataset.

Among all the 5 features, `dport` needs to be processed before being used by the baseline model, since it is a discrete value with a wide range (0∼65536) that cannot be directly learned by a machine-learning model. We first encode `dport` value into a one-hot binary string. Since most of the ports are rarely used, we use principal component analysis (PCA) to reduce the string into 50 principal components. The total variance of them is around 98.9% according to our statistics, which means the 50 components keep most of information involved in ports.

For the hyper-parameters of Random Forest, we set the number of individual trees to 100 to balance training speed and performance.

3.4.2 Sequence-based Model. In Section 2.5, we overview the LSTM-RNN and describe its advantage when being used to solve our problem. Below we describe the construction of our LSTM-based models, including a basic version and a bidirectional version.

Embedding dimension	**LSTM hidden dimension**	**LSTM layer**	**Dropout rate**
30	64	1	0.5
Learning rate	**Activation function**	**Optimizer**	**Training epochs**
0.001	ReLU	Adam	15

Table 3: Parameters of a LSTM-RNN model.

We show the impact of window size on evaluation result in Section 4.2.2.

Basic LSTM. A basic version contains multiple blocks and each block contains 4 layers:

-`Embedding layer`. Embedding has been widely used in the domain of Natural Language Processing (NLP) [31, 40] which transforms discrete values into continuous vectors. In our LSTM models, we transform `dport` information into embeddings instead of the one-hot encoding like the baseline model, mainly because this embedding layer can be seamless connected to other layers. In addition, it is a dynamic "mini" neural network gradually updating during the training phase. Therefore, this representation can be optimized during the training phase.

-`LSTM layer`. After the processing of the embedding layer, the input will be fed into the LSTM layer. At each step, a packet is assigned to an LSTM cell. The output of LSTM cells can be stacked into a matrix as the input of the next layer.

-`Fully-connected layer`. We put a hidden fully-connected layer between the LSTM layer and softmax layer with the size equal to the number of total categories.

-`Softmax layer`. The hidden dense layer output is then fed into the softmax layer for normalization. The output of the softmax layer is the probability distribution indicating how likely a sample belongs to a category, which sums to one. For our task of multi-class classification, we select the category with the highest probability as the final output.

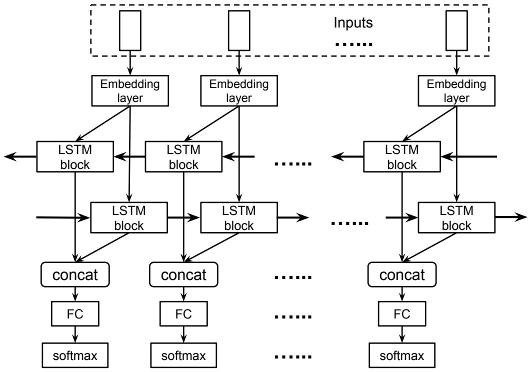

Figure 7: Structure of the bidirectional LSTM used in our work.

Bidirectional LSTM. The basic LSTM model only looks into the "past" of a packet when learning contextual information. Bidirectional LSTM (BLSTM) is an extension to the basic LSTM, which utilizes the information from the "future", by combining another

LSTM layer moving from the end of a sequence to its beginning [20]. In areas like phoneme classification [22] and sequence tagging [26], BLSTM significantly improves the performance compared to a traditional one. Since our model works on a sequence of consecutive packets, we can utilize the information of the packets transmitted *after* the current packet to classify it. The main change we apply on the LSTM layer is to concatenate cell states of backward and forward LSTM layers and feed them to the dense layer. Figure 7 shows the structure of the bidirectional LSTM used in our work.

4 EVALUATION

In the evaluation, we want to understand how our models perform under different network scenarios. In this section, we first introduce the datasets we used for evaluation and our evaluation metrics. Then we describe our two scenes and the corresponding results. Finally, we show several case studies.

As a quick overview of our results, we found LSTM-RNN models can well handle packet identification tasks with an overall accuracy of over 92.0% in NAPT and VPN configurations on IoT traffic. Compared to basic LSTM, bidirectional LSTM performs better, suggesting the packet dependency indeed reveals the patterns unique to each individual IoT device.

4.1 Experiment Settings

Scenarios. We evaluate HomeMole in two scenarios – pure-IoT (only one active IoT device) and noisy (multiple active IoT and non-IoT devices). In each scenario, we evaluate HomeMole with two different gateway configurations – NAPT and VPN.

Traffic window. After pre-processing, each packet is transformed into a feature vector. To facilitate the training and testing of sequence-based models, we group every n consecutive vectors to form a traffic window. Figure 6 shows an example of a traffic window. With traffic windows, we are able to model the temporal relations of adjacent packets.

Datasets. We constructed two datasets for the two scenarios. Each dataset has a NAPT version and a VPN version. We split each dataset with the training and testing ratio of 8:2 and conduct 5-fold cross-validations on it. Below are the details of each dataset.

(1) Dataset-Ind. This dataset contains traffic representations from 10 individual IoT devices. To facilitate the training process of LSTM-RNN models, the dataset is organized into collections of traffic windows, each traffic window only contains packets from one certain device. To make the dataset more balanced, we set a maximum threshold of 5,000 windows for each device. All IoT devices own 5,000 randomly-selected samples except Xiaomi hub, tplink plug, orvibo plug and broadlink plug, due to that they generate much fewer packets than others. In total, Dataset-Ind contains 32,760 traffic windows.

(2) Dataset-Noise. The traffic used to generate this dataset comes from multiple active IoT and non-IoT devices. As a result, the traffic windows in this dataset are composed of packets from *more than* one device. In total, Dataset-Noise includes 114,989 traffic windows. Figure 8 shows the distribution of device combinations in traffic windows. From it we can see, 2-device and 3-device combinations are most common.

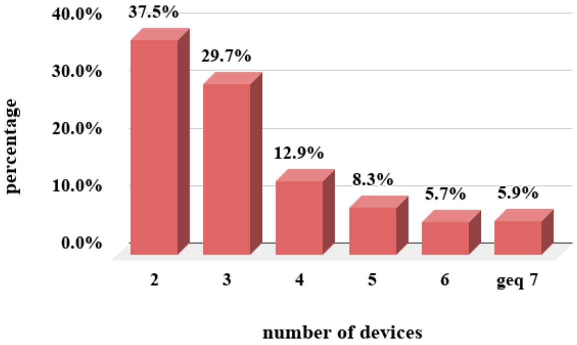

Figure 8: Proportion of device combinations (geq: greater than or equal to 7).

Metrics. Since HomeMole is able to classify individual packets, we measure the effectiveness of HomeMole based on the probability that the device is correctly identified *per packet*. We use *overall accuracy* (similar to [36]) and *category accuracy* for our case. For overall accuracy, we count N as all the packets and $P_{correct}$ as the total number of correctly classified packets, and compute $\frac{P_{correct}}{N}$. For category accuracy, we assess how HomeMole performs on each device. For device A, we count NA as all packets belonging to A and $PA_{correct}$ as A's packets correctly classified under A. The category accuracy for A is $\frac{PA_{correct}}{NA}$. As an example, the diagonal cells on the confusion matrix shown in Figure 10 describe the category accuracy.

4.2 Pure-IoT Scenario

In this setting, there is only one active IoT device working during the data collection period. In practice, such a scenario happens when the rest of IoT devices enter hibernation mode.

4.2.1 *Single-packet Model.*

We first evaluate the performance of our baseline model, Random Forest. The purposes are two-fold: (1) To explore the feasibility of classifying individual packets without context; (2) To evaluate the effectiveness of features we selected from the packet's metadata.

Experiment results. Table 4 shows the accuracy of Random Forest in NAPT and VPN configuration. As can be seen, with NAPT configuration, Random Forest can reach a high identification accuracy on most IoT devices. Among them, smart plugs and network cameras have the highest accuracy while voice assistants have a lower accuracy of ∼ 87%. Compared with NAPT, the result is worse in VPN configuration with a 9.0% decline in overall accuracy. Accuracy on voice assistant is affected most.

Result analysis. We first use the built-in API provided by scikit-learn library to obtain feature importance. The results show that in NAPT, dport, frame length, time interval and protocol hold an importance factor of 55.5%, 22.8%, 12.8% and 8.0% separately. In VPN, frame length and time interval take up around 54.9% and 43.0% separately.

The obvious accuracy decline in VPN mainly comes from the change of dport and protocol (dport and protocol information are not preserved in the packets between gateway and VPN server) and partial loss of frame length due to the padding by VPN client.

config	model	average	echo dot	google home	tmall assistant	xiaomi hub	360 cam	tplink plug	orvibo plug	mitu story	xiaobai camera	broadlink plug
NAPT	RF	92.2	89.0	85.9	86.9	89.6	99.0	**99.9**	**99.9**	93.3	98.5	99.3
	LSTM	97.3	**98.5**	91.6	93.9	98.6	**99.9**	**99.9**	**99.9**	98.7	**99.9**	**99.9**
	BLSTM	**99.2**	97.0	**99.2**	**99.8**	**99.9**	**99.9**	**99.9**	**99.9**	**99.3**	**99.9**	**99.9**
VPN	RF	83.2	76.1	81.2	74.7	94.0	83.2	89.1	93.1	87.5	90.5	**99.0**
	LSTM	92.4	89.7	89.7	75.4	96.1	95.9	92.2	95.5	96.8	94.7	95.7
	BLSTM	**97.7**	**96.6**	**96.8**	**94.7**	**99.4**	**98.5**	**98.0**	**99.5**	**98.9**	**99.7**	96.7

Table 4: Accuracy of baseline model under pure-IoT scenario (RF, LSTM and BLSTM stand for Random Forest, basic LSTM and bidirectional LSTM respectively).

4.2.2 Sequence-based Model. We conjecture that the dependency of packets can be used for device identification, and we model it through LSTM-RNN models. Below we evaluate the two proposed LSTM-RNN models on `Dataset-Ind`.

Experiment results. Table 4 also shows the performance of `basic LSTM` and `bidirectional LSTM` when the input traffic window contains 100 consecutive packets. We can see that compared to the baseline, both of the models have seen an increase in accuracy on most devices. The result also shows LSTM-RNN models can well handle IoT devices producing a large volume of traffic like voice assistant.

Impact of traffic window size. We compare the accuracy of LSTM-RNN models with different window sizes: 20, 40 and 100. The result is shown in Figure 9. From it we can see for both NAPT and VPN configurations, LSTM-RNN models perform better when the traffic window size grows. This result indicates the relation between packets with long timing gaps can still provide useful information for our models. In the following sections, we take 100 as the default size of our traffic window.

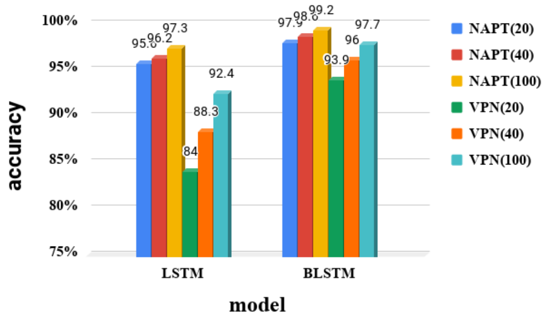

Figure 9: The impact of traffic window size (pure-IoT).

4.3 Noisy Scenario

When there are multiple active devices functioning in the smart home, the situation becomes more complicated, since the mixture of traffic from different devices introduce confusion to the network observers (attackers).

To better illustrate the influence brought by the noisy environment, we consider the results under two conditions, which we call `session-clear` condition and `session-obscure` condition.

The `session-clear` case refers to the NAPT environment, in which the original source IP of outbound traffic and the destination IP of inbound traffic are masked by the router. However, in most cases, each connection between two parties owns an unchanged connection 5-tuple (*sip*, *dip*, *sport*, *dport*, *protocol*) [14]. Packets under the same tuple are associated with the same TCP/UDP session, naturally coming from/going to the same device.

However, the session becomes obscure when it comes to global VPN. Since source IP, destination IP, source port and destination port are all masked by VPN, and the original session information is completely removed. In this way, packets from different sessions are merged into the same session and thus a traffic window may contain packets from different devices.

To better simulate the real conditions met by a network observer, we perform an extra processing step on `Dataset-Noise`. As traffic in NAPT (session-clear) environment can be differentiated through the connection 5-tuple, we use `PcapPlusPlus` to split the traffic into sessions. Packets in each session are determined to be from the same device, therefore avoiding the impact of traffic fusion. Since the split traffic has a different length, we concatenate or cut them into traffic windows of the same size and control the number of traffic windows in the same setting as `Dataset-Ind`. However, as for the VPN (session-obscure) environment, we leave `Dataset-Noise` as unchanged, which means a traffic window may contain packets from different devices.

4.3.1 Single-packet Model. We first tested Random Forest using `Dataset- Noise` to understand the impact of non-IoT traffic and traffic fusion. Given the different traffic volume among devices, the packets we collect are imbalanced, as Figure 5 shows.

Experiment results. Compared with pure-IoT scenario, Random Forest has a prominent decline in the overall accuracy, reaching 85.6% in NAPT and 67.6% in VPN. We also use confusion matrix across devices to show the classification results by category with non-IoT traffic in NAPT and VPN configuration (Figure 10 and Figure 11). From there we can see, `voice assistants`, like Echo Dot, Google Home and Tmall Assistant, see larger performance drop compared to other IoTs, with a ~ 8% average decline in NAPT and a ~ 50% average decline in VPN configurations.

Result analysis. We manually check the misclassified packets and find that most of them are transmitted through ports 443 and 80. The primary reason is that those ports are likely to be used by different IoT and non-IoT at the same time, so our model is more likely to be confused.

4.3.2 Sequence-based Model. **Experiment results.** Figure 14 and Figure 15 give the comparison results between three models in NAPT and VPN configurations. The last group of columns ("average") shows the overall accuracy. Bidirectional LSTM achieves the

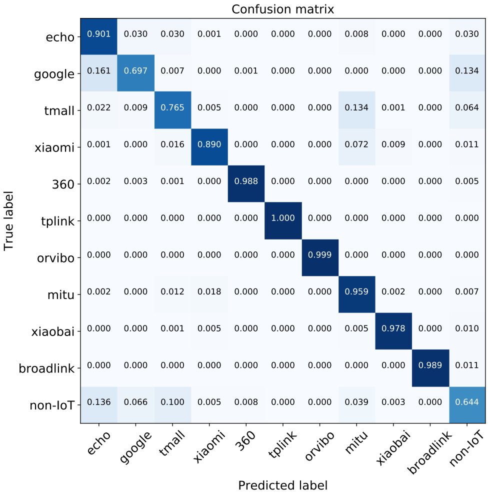

Figure 10: Confusion matrix of RF (noisy+NAPT).

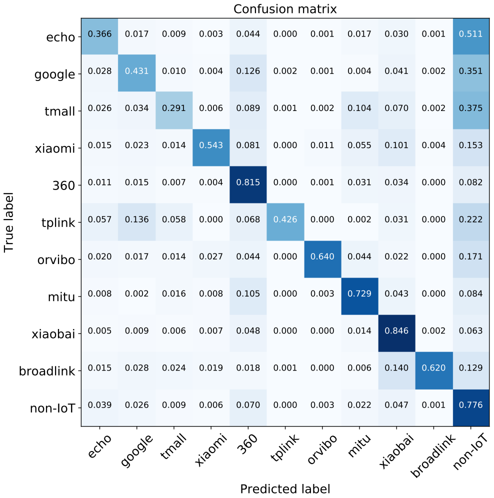

Figure 11: Confusion matrix of RF (noisy+VPN).

highest accuracy of 95.3% in NAPT and 80.9% in VPN. Basic LSTM reaches 91.9% and 74.1%. Figure 12 and Figure 13 show the concrete classification results of bidirectional LSTM by device categories in two configurations. From them we know: (1) LSTM-RNN models are good at recognizing traffic in NAPT configuration; (2) LSTM-RNN performs much worse in VPN configuration, especially on IoT devices like smart plugs (BLSTM: 13.4% for Orvibo, 20.2% for Tplink and 14.9% for Broadlink in 5 cross-validations). (3) Non-IoT traffic has a negative impact on the identification of most IoT devices.

Result analysis. From the above results, we can see LSTM-RNN models fail to classify traffic generated by smart plugs and Xiaomi hub in VPN configuration. We find this observation can be ascribed to the sparse traffic generated by the devices. Due to the relatively long time intervals between packets and the low packet amount, the traffic generated by these devices can be easily "overwhelmed" by traffic from others, leading to the original relation between packets being impaired. In extreme cases, packets generated by smart plugs can be "diluted" to less than 3% in a traffic window (3 in 100 packets). The situation becomes worse when distinctive features like `dport` and `frame length` are more likely to be confused. In Section 4.4 we show a case of `orvibo plug`.

On the other hand, we find the following characteristics of non-IoT traffic causes misclassification under VPN configuration: (1) Large volume ratio (33.9% of all packets) (2) Persistent communication (non-IoT devices usually continuously contact remote server while IoT devices send out packets sparsely) (3) Irregular communication patterns (non-IoT devices are activated or used frequently but their activation and usage time are quite variable).

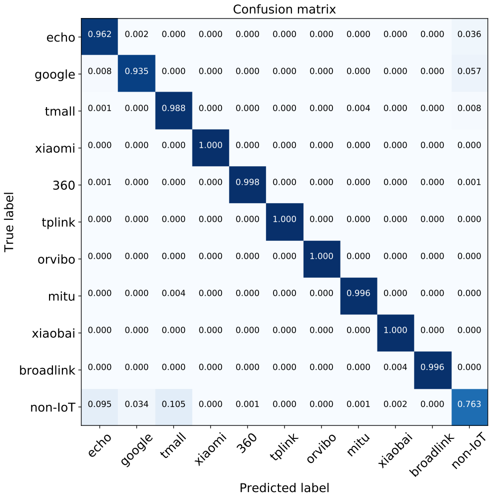

Figure 12: Confusion matrix of BLSTM(noisy+NAPT).

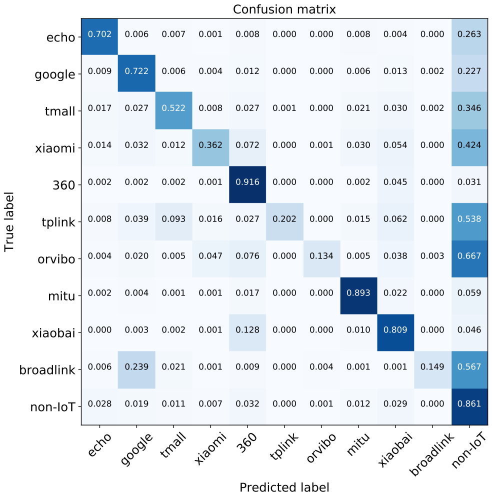

Figure 13: Confusion matrix of BLSTM(noisy+VPN).

4.4 Case Studies

Scarce traffic. Several IoT devices have a low traffic volume, such as `orvibo plug` and `tplink plug`. Therefore, the packets generated by them may be "overwhelmed" by other devices in the same traffic window. Take `orvibo plug` as an example. It has a three-TCP-packet sequence appearing multiple times. Their `frame lengths` are 224, 54, 240 bytes in two seconds correspondingly. But those

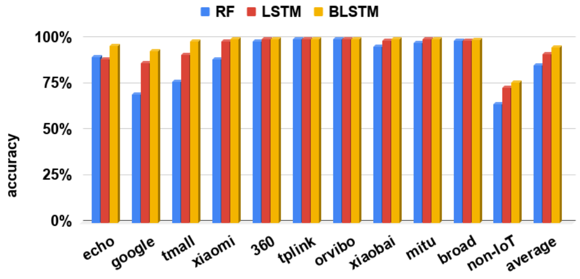

Figure 14: Performance of three models in NAPT setting.

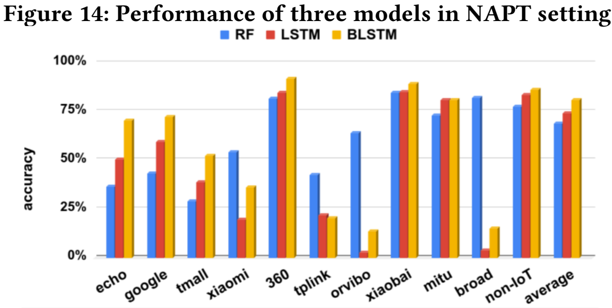

Figure 15: Performances of three models in VPN setting.

numbers are not unique. According to our analysis, packets with the same sizes are generated by other devices so our classification models could be confused. In particular, `360 camera` produces UDP packet of 224 bytes very frequently. `xiaomi hub` and `xiaomi tablet` generate TCP packets of 54 bytes consistently at high speed (15 packets in 0.1 second). As such, the three consecutive packets of `orvibo plug` are likely to be separated by those packets from other devices, leading to wrong classification results, especially in VPN configuration when protocol information is missing. This could explain why LSTM-RNN models perform worse in identifying IoT devices of small traffic volume in the VPN environment.

Effectiveness of bidirectional LSTM. Section 4 shows the accuracy of the bidirectional LSTM model is better than the basic LSTM model in most cases. Below we use one example to explain this performance difference. Through our manual check of traffic from different devices, we find packets of the size 66 bytes are commonly sent to the server in a sequence by `Echo Dot` and `Google Home`. However, the responses from the server are different between the two devices. For `Echo Dot`, most of the responses are of a size 1388 bytes while for `Google Home`, most of them are TLS packets with the size of 108 bytes and 105 bytes. Compared to a basic LSTM, a bidirectional LSTM can utilize the later packet (1388-byte or 105-byte) to help classify the previous packets (66-byte). As such, adding the information from "future" could improve the chance of correctly classifying packets.

5 DISCUSSION

Beyond packet identification. Different from previous work that assigns binary labels to *network flows* or *traffic windows* [9, 11, 15, 17, 41], our model classifies *all packets* within a time period. There are two reasons for this design decision:

(1) Packet-level identification allows the adversary to learn the running status of an IoT device in real time.
(2) The result of packet-level identification can be used to label objects of any other granularity, like traffic window or network flow.

To the first point, after packets of an IoT device are identified, the adversary can compute certain statistics from the packet labels to infer the device status. For example, if packets are observed sparsely, the device could be in sleeping or standby mode. Otherwise, the device could be busy running a task. Actions can be taken more promptly by the adversary and further threatens users' privacy and safety. For IoT camera, if a thief knows when a network camera is transmitting bulk data continuously, she can infer whether the homeowner is around and pick the best time to break into the house.

To the second point, we evaluate the accuracy of classifying traffic windows using the data from `Dataset-Noise` in a VPN environment. The label of a window consists of *all* devices with active network activities in the time period. For example, if a traffic window contains packets from three different devices A,B and C, then the label for this window is $[A, B, C]$. Since the problem becomes a multi-label classification problem, we select Hamming Loss as the metric. Hamming Loss [42] indicates the fraction of wrongly-classified samples to the whole dataset and can be calculated by $\frac{1}{|N|\cdot|L|}\sum_{i=1}^{|N|}\sum_{j=1}^{|L|} xor(y_{i,j}, z_{i,j})$, where N stands for the total amount of samples and L is the number of classes. Each target y and prediction z are represented by a 0-1 vector with L elements. To notice, both false positive (predicting an IoT device not existed in smart home) and false negative (missing an active IoT device) will introduce Hamming Loss. Our model reaches a 0.12 hamming loss, which means **88%** of the labels are correctly matched.

Defense against traffic fingerprinting attack. Experimental results in section 4 show that the fingerprinting attack based on network traffic introduces considerable privacy threat to smart-home users when the traffic from different devices can be separated. This observation aligns with previous works [9, 11, 12]. When traffic is merged across IoT/non-IoT devices under NAPT and VPN scenarios, accurately pinpointing IoT networks would be more difficult. In particular, from Section 4.3.1 and Section 4.3.2, it can be seen that accuracy drops ranging from 8.2% to 87.4% under the VPN scenario, especially for the devices generating small traffic volume. However, we believe a more principled approach that can achieve sufficient protection under all network configurations and device combinations is still necessary. To this end, we plan to explore whether local differential privacy (LDP) [27], which adds noise based on the randomized response, can be applied to obfuscate IoT traffic.

6 LIMITATION

VPN connections. In the experiment, we only establish our VPN connection using UDP protocols. Another option – TCP (TLS) is not tested. The main reason is that UDP is the default protocol adopted by the `openVPN` service and is the most-widely protocol used by VPN providers due to its low latency [6] compared with TCP. VPN using TCP requires modification of our packet labeling algorithm in Section 3.3 and we decide to leave it as our future work.

Behavior identification. We focused on device identification during the evaluation, while previous works also explored user behavior identification [9]. We did not experiment with behavior identification because the labeling cost is high given that we have large datasets with millions of IoT packets. On the other hand, we believe our models can be applied to this scenario if we have enough training samples. We will explore approaches that can generate labeled behavioral datasets efficiently.

7 CONCLUSION

In this paper, we systematically evaluated the effectiveness of traffic analysis in a smart home environment, even when traffic fusion like NAPT and VPN are enabled and non-IoT and IoT devices are both active. By exploiting the dependency between packets through DNN models like LSTM-RNN, we show it achieves high accuracy in device identification, even under the complex network environment as described above.

Our result suggests the network communications of IoT devices do have serious privacy implications, even under encryption and traffic fusion. We believe more research should be done to better understand the privacy issues in smart home network and mitigate such issues. To facilitate the research in this domain, we also release the data and our models.

ACKNOWLEDGEMENTS

We thank our shepherd Selcuk Uluagac and anonymous reviewers for their insightful comments. This work was partially supported by the General Research Funds (Project No. 14208818 and 14208019) established under the University Grant Committee of the Hong Kong Special Administrative Region.

REFERENCES

[1] Accessed: March 2019. 2017 Roundup Of Internet Of Things Forecasts. https://www.forbes.com/sites/louiscolumbus/2017/12/10/2017-roundup-of-internet-of-things-forecasts/.

[2] Accessed: March 2019. AWS IoT Core. https://aws.amazon.com/cn/iot-core/.

[3] Accessed: March 2019. Can a MAC address be traced? https://askleo.com/can_a_mac_address_be_traced/.

[4] Accessed: March 2019. Digital Ocean Droplets. https://www.digitalocean.com/products/droplets.

[5] Accessed: March 2019. Monkeyrunner. https://developer.android.com/studio/test/monkeyrunner.

[6] Accessed: March 2019. OpenVPN over TCP vs. UDP: Which should I choose? https://www.bestvpn.com/guides/openvpn-tcp-vs-udp-difference-choose/.

[7] Accessed: March 2019. Raspberry Pi. https://www.raspberrypi.org/documentation/.

[8] Accessed: March 2019. SmartThings Motion Sensor (2015 model). https://support.smartthings.com/hc/en-us/articles/205957580-SmartThings-Motion-Sensor-2015-model-.

[9] Abbas Acar, Hossein Fereidooni, Tigist Abera, Amit Kumar Sikder, Markus Miettinen, Hidayet Aksu, Mauro Conti, Ahmad-Reza Sadeghi, and A. Selcuk Uluagac. 2018. Peek-a-Boo: I see your smart home activities, even encrypted! *CoRR* abs/1808.02741 (2018).

[10] Hidayet Aksu, A. Selcuk Uluagac, and Elizabeth S. Bentley. 2018. Identification of Wearable Devices with Bluetooth. *CoRR* abs/1809.10387 (2018). arXiv:1809.10387 http://arxiv.org/abs/1809.10387

[11] Noah Apthorpe, Dillon Reisman, and Nick Feamster. 2017. A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic. *CoRR* abs/1705.06805 (2017). arXiv:1705.06805 http://arxiv.org/abs/1705.06805

[12] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. 2017. Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic. *CoRR* abs/1708.05044 (2017). arXiv:1708.05044 http://arxiv.org/abs/1708.05044

[13] Leonardo Babun, Z. Berkay Celik, Patrick D. McDaniel, and A. Selcuk Uluagac. 2019. Real-time Analysis of Privacy-(un)aware IoT Applications. *CoRR* abs/1911.10461 (2019). arXiv:1911.10461 http://arxiv.org/abs/1911.10461

[14] Marcelo Bagnulo, Philip Matthews, and Iljitsch van Beijnum. 2011. Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. *RFC* 6146 (2011), 1–45. https://doi.org/10.17487/RFC6146

[15] George Dean Bissias, Marc Liberatore, David D. Jensen, and Brian Neil Levine. 2005. Privacy Vulnerabilities in Encrypted HTTP Streams. In *Privacy Enhancing Technologies, 5th International Workshop, PET 2005, Cavtat, Croatia, May 30-June 1, 2005, Revised Selected Papers*. 1–11. https://doi.org/10.1007/11767831_1

[16] Z. Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick D. McDaniel, and A. Selcuk Uluagac. 2018. Sensitive Information Tracking in Commodity IoT. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, William Enck and Adrienne Porter Felt (Eds.). USENIX Association, 1687–1704. https://www.usenix.org/conference/usenixsecurity18/presentation/celik

[17] Zhitang Chen, Ke He, Jian Li, and Yanhui Geng. 2017. Seq2Img: A sequence-to-image based approach towards IP traffic classification using convolutional neural networks. In *2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017*. 1271–1276. https://doi.org/10.1109/BigData.2017.8258054

[18] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. 1285–1298. https://doi.org/10.1145/3133956.3134015

[19] Bryan Gleeson, Arthur Lin, Juha Heinänen, Grenville Armitage, and Andrew G. Malis. 2000. A Framework for IP Based Virtual Private Networks. *RFC* 2764 (2000), 1–62. https://doi.org/10.17487/RFC2764

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[21] Alex Graves, Navdeep Jaitly, and Abdel rahman Mohamed. 2013. Hybrid speech recognition with Deep Bidirectional LSTM. *2013 IEEE Workshop on Automatic Speech Recognition and Understanding* (2013), 273–278.

[22] Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural networks : the official journal of the International Neural Network Society* 18 5-6 (2005), 602–10.

[23] Paul E. Hoffman and Patrick McManus. 2018. DNS Queries over HTTPS (DoH). *RFC* 8484 (2018), 1–21. https://doi.org/10.17487/RFC8484

[24] Zi Hu, Liang Zhu, John S. Heidemann, Allison Mankin, Duane Wessels, and Paul E. Hoffman. 2016. Specification for DNS over Transport Layer Security (TLS). *RFC* 7858 (2016), 1–19.

[25] Hong Huang, Hussein Al-Azzawi, and Hajar Brani. 2014. Network Traffic Anomaly Detection. *CoRR* abs/1402.0856 (2014). arXiv:1402.0856 http://arxiv.org/abs/1402.0856

[26] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *CoRR* abs/1508.01991 (2015).

[27] J. W. Kim, D. Kim, and B. Jang. 2018. Application of Local Differential Privacy to Collection of Indoor Positioning Data. *IEEE Access* 6 (2018), 4276–4286. https://doi.org/10.1109/ACCESS.2018.2791588

[28] Zachary Chase Lipton, David C. Kale, Charles Elkan, and Randall C. Wetzel. 2016. Learning to Diagnose with LSTM Recurrent Neural Networks. *CoRR* abs/1511.03677 (2016).

[29] Manuel López Martín, Belén Carro, Antonio Sánchez-Esguevillas, and Jaime Lloret. 2017. Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things. *IEEE Access* 5 (2017), 18042–18050. https://doi.org/10.1109/ACCESS.2017.2747560

[30] Markus Miettinen, Samuel Marchal, Ibbad Hafeez, N Asokan, Ahmad-Reza Sadeghi, and Sasu Tarkoma. 2017. IoT Sentinel: Automated device-type identification for security enforcement in IoT. In *Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on*. IEEE, 2177–2184.

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[32] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. 2016. Website Fingerprinting at Internet Scale. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/website-fingerprinting-internet-scale.pdf

[33] Sakthi Vignesh Radhakrishnan, A. Selcuk Uluagac, and Raheem A. Beyah. 2015. GTID: A Technique for Physical Device and Device Type Fingerprinting. *IEEE Trans. Dependable Sec. Comput.* 12, 5 (2015), 519–532. https://doi.org/10.1109/TDSC.2014.2369033

[34] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom van Goethem, and Wouter Joosen. 2018. Automated Website Fingerprinting through Deep Learning. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. https://arxiv.org/abs/1708.06376

[35] Sandra Siby, Rajib Ranjan Maiti, and Nils Ole Tippenhauer. 2017. IoTScanner: Detecting and Classifying Privacy Threats in IoT Neighborhoods. *CoRR* abs/1701.05007 (2017). arXiv:1701.05007 http://arxiv.org/abs/1701.05007

[36] Payap Sirinam, Mohsen Imani, Marc Juárez, and Matthew Wright. 2018. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. 1928–1943.

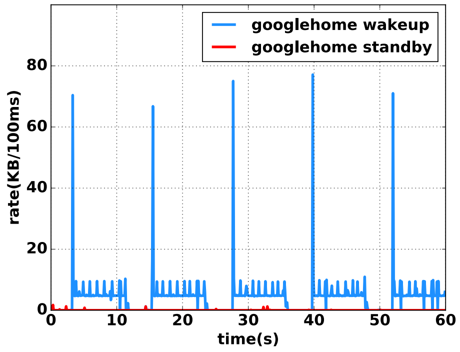

Figure 17: Google Voice Assistant

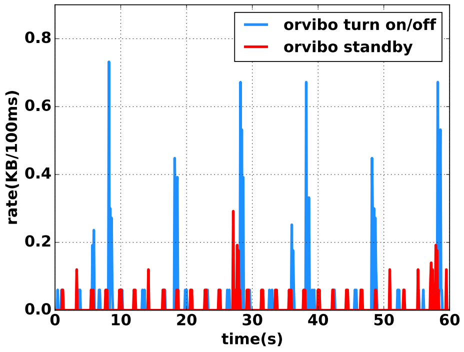

Figure 18: Echo dot

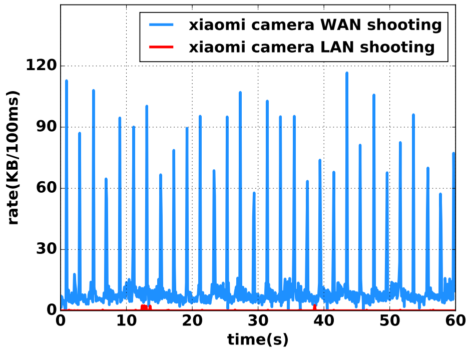

Figure 19: Google Voice Assistant

https://doi.org/10.1145/3243734.3243768

[37] Arunan Sivanathan, Daniel Sherratt, Hassan Habibi Gharakheili, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. 2017. Characterizing and classifying IoT traffic in smart cities and campuses. In *2017 IEEE Conference on Computer Communications Workshops, INFOCOM Workshops, Atlanta, GA, USA, May 1-4, 2017.* 559–564. https://doi.org/10.1109/INFCOMW.2017.8116438

[38] Pyda Srisuresh and Matt Holdrege. 1999. IP Network Address Translator (NAT) Terminology and Considerations. *RFC* 2663 (1999), 1–30. https://doi.org/10.17487/RFC2663

[39] Milijana Surbatovich, Jassim Aljuraidan, Lujo Bauer, Anupam Das, and Limin Jia. 2017. Some Recipes Can Do More Than Spoil Your Appetite: Analyzing the Security and Privacy Risks of IFTTT Recipes. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich (Eds.). ACM, 1501–1510. https://doi.org/10.1145/3038912.3052709

[40] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[41] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2016. AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016.* 439–454. https://doi.org/10.1109/EuroSP.2016.40

[42] Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)* 3, 3 (2007), 1–13.

[43] Jishen Yu, Feng Liu, Wenli Zhou, and Hua Yu. 2014. Hadoop-based network traffic anomaly detection in backbone. In *IEEE 3rd International Conference on Cloud Computing and Intelligence Systems, CCIS 2014, Shenzhen, China, November 27-29, 2014.* 140–145. https://doi.org/10.1109/CCIS.2014.7175718

A APPENDICES

Figure 16 and Figure 17 show the traffic patterns when echo dot and google voice assistant are waked up respectively. Figure 18 uses two different colors to depict the traffic patterns of Orvibo switch when it is turned on/off and in standby mode. In Figure 19 we compare the traffic patterns of Xiaomi camera in WAN mode and LAN mode.

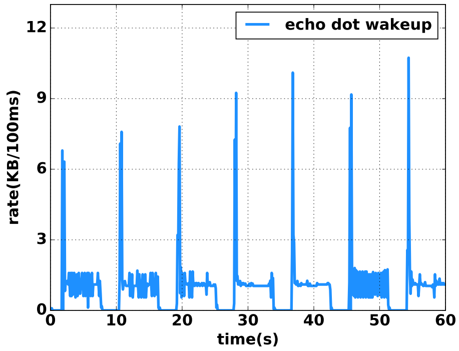

Figure 16: Echo dot