# Peek-a-Boo: I see your smart home activities, even encrypted!

Abbas Acar*, Hossein Fereidooni†, Tigist Abera†, Amit Kumar Sikder*,
Markus Miettinen†, Hidayet Aksu*, Mauro Conti‡, Ahmad-Reza Sadeghi†, and A. Selcuk Uluagac*
*Florida International University, Email:{aacar001,asikd003,haksu,suluagac}@fiu.edu
†TU Darmstadt, Email: {hossein.fereidooni, tigist.abera, markus.miettinen, ahmad.sadeghi}@trust.tu-darmstadt.de
‡University of Padua, Email: conti@math.unipd.it

***Abstract*—A myriad of IoT devices such as bulbs, switches, speakers in a smart home environment allow users to easily control the physical world around them and facilitate their living styles. However, an attacker inside or near a smart home environment can potentially exploit the innate wireless medium used by these devices to exfiltrate sensitive information about the users and their activities, invading user privacy. With this in mind, in this work, we introduce a novel *multi-stage privacy attack* against user privacy in a smart environment. It is realized utilizing state-of-the-art machine-learning approaches for detecting and identifying particular types of IoT devices, their actions, states, and ongoing user activities in a cascading style by only observing passively the wireless traffic from smart home devices. The attack effectively works on both encrypted and unencrypted communications. We evaluate the efficiency of the attack with real measurements from an extensive set of popular off-the-shelf smart home IoT devices utilizing a set of diverse network protocols like WiFi, ZigBee, and BLE. Our results show that an adversary passively sniffing the network traffic can achieve very high accuracy (above 90%) in identifying the state and actions of targeted smart home devices and their users. In contrast to earlier straightforward approaches, our multi-stage privacy attack can perform activity detection and identification *automatically*, without extensive background knowledge or specifications of analyzed protocols. This allows an adversary to efficiently aggregate extensive behavior profiles of targeted users. To protect against this privacy leakage, we also propose a countermeasure based on generating spoofed network traffic to hide the real activities of the devices. We also demonstrate that the provided solution provides better protection than existing solutions.**

## I. Introduction

Previously, the Internet was mainly used for accessing and displaying content of web pages (i.e., web browsing). However, with the emergence of IoT devices in smart homes, users have now the ability to control their home's electronic systems (e.g., smart bulbs, smart locks, etc.) using appropriate smartphone apps and also from remote locations [29]. To realize this, smart home devices are continuously communicating with associated back-end system servers or other devices (e.g., smart hubs). On the other hand, as IoT devices usually are single-purpose devices, the capabilities of individual smart home devices are relatively limited, comprising only a few states or actions. For example, a smart lock can assume only one of two states, locked or unlocked. Given that the communications among the server, smart-hub, and the smart home devices are usually encrypted using standard protocols like WPA2, in the case of WiFi, the contents of the exchanged messages or commands are hidden. However, the encryption only hides the payload, related meta-data (e.g., packet lengths, traffic rate) of the network traffic still leaks some information about the messages exchanged [34], [38], [33], [23], [14], [13].

Identification of the encrypted network traffic is a well-studied problem. However, applying traditional identification methods such as statistical techniques [34] in the domain of smart home is not straightforward due to challenges arising from the inherent properties of IoT devices. First, unlike targets using a widely-deployed protocol to perform a well-known specific activity like web browsing, in the smart home context, the targeted device population is much more heterogeneous and uses various network protocols such as WiFi, ZigBee, BLE, etc. for supporting an even wider variety of device-type-specific, potentially proprietary application protocols. On the one hand, this naturally extends the potential attack surface, but also makes it even harder to devise *generic* attacks or countermeasures. Second, although some earlier works have shown [32], [8] that it is relatively easy to make some simple inferences (e.g., something happened at a particular time), combining such partial information from different smart home devices to get a more meaningful picture about a user's actions or his/her activity profile is not easy. This is because a successful attacker must aggregate information about actions over a longer period of time from a multitude of smart home devices, which is only feasible if activity detection and identification can be *automated* to a large degree to keep the required effort manageable.

In this paper, we demonstrate how machine learning methods based on traffic profiling of smart home IoT device communications can be used by an adversary to automatically identify actions and activities of the IoT devices and its users in a victim's smart home with very high accuracy, even if only encrypted data are available. Indeed, device types, daily mundane activities of the users (e.g., left home, walking from kitchen to bedroom), or states of the devices (e.g., door locked, unlocked) can all be easily identified even if the traffic is encrypted, posing greater threat to user privacy than ever before. We refer to this novel attack to user privacy as *multi-stage privacy attack*, which is achieved in a cascading style by only observing passively the wireless traffic from smart home devices. In this, a passive attacker can easily realize the multi-stage privacy attack to extract meaningful data from any smart environment equipped with smart devices including personal homes, residences, hotel rooms, offices of corporations or government agencies. Here, unlike earlier approaches, the presented attack is device-type and protocol-agnostic, making it easily applicable to a wide variety of different IoT device types without the need for tedious harvesting of device-type or protocol-specific knowledge about specifications for supporting the activity identification task.

We evaluate the effectiveness of the novel multi-stage privacy attack with 22 different off-the-shelf IoT devices utilizing the most popular wireless protocols for IoT. Our experimental results show that an attacker can achieve very high accuracy (above 90 %) in identification of the types, actions, states, activities of the devices. Moreover, to counter the identified privacy threats posed by the multi-stage privacy attack, we also propose a new effective countermeasure solution based on generating spoofed traffic to hide the real states of targeted IoT devices and thereby the real activities of the users. Our solution does not require modifications in targeted IoT devices and is, therefore, easier to deploy than previously proposed solutions for IoT devices, for which it is very difficult to implement client-based countermeasures due to the vast heterogeneity of smart devices and limited resources available on the IoT devices. Also, even if the user is not at home, a fake traffic-based solution for the user's presence will mask the user's absence, further improving privacy.

**Contributions & Organization:** The contributions of this work are as follows:

- After discussing the specific characteristics of smart home devices (Sec. III) and demonstrating their related sources of privacy leakage based on case studies (Sec. IV), we propose a *novel multi-stage privacy attack* on smart home users and devices which can leak sensitive information including types of devices, states of the devices, and on-going user activities (Sec. V). The attack includes several novel techniques for reducing the inference on the timing-based network traffic to Machine Learning (ML) problem and inferring user activities using the Hidden Markov Model (HMM).
- We evaluate our proposed novel attack with a dataset of popular commercial smart home devices (n=22) (Sec. V). We show that an attacker can automatically detect and identify device actions with high accuracy ($> 90\%$), allowing an adversary to infer potentially sensitive information about the smart home users.
- Finally, although the focus of this paper is on the novel attack, we also propose a new solution based on traffic spoofing to address this new privacy threat (Sec. VI).

## II. Adversary Model

One of the unique challenges in the domain of IoT, and particularly smart home, the attack surface is naturally extended and comprised of diverse set of devices deployed at home by the users. Figure 1 shows different data capturing points that an attacker can take advantage of when inferring user activities. In this work, we consider a passive adversary located physically within the wireless range of the targeted user's smart home devices similar to [20], [21], [22]. The adversary can eavesdrop on various wireless IoT network communications transmitted by the user's smart home devices. For example, as presented in Figure 1, the attacker can sniff all the network traffic transmitted over WiFi, BLE, and ZigBee protocols. The attacker only needs to passively sniff the network traffic and does not need to interrupt. Therefore, the attacker may stay active long enough without detected by the victim.

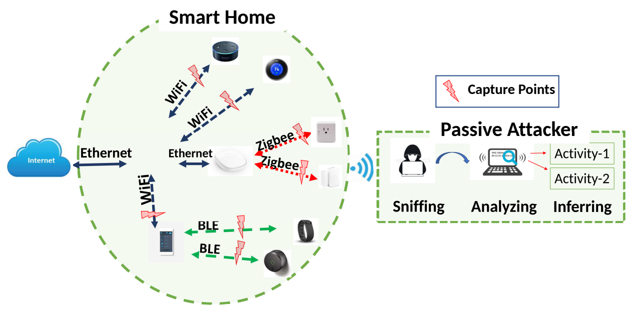


Figure 1: Passive attacker model considered in this paper.

**Assumptions.** We further make the following assumptions:

- The attacker has access to the same kind of smart home devices as the targeted user, s/he can analyze the devices by collecting the network traffic of these devices, and use the collected data to train its algorithms.
- The attacker has access to protocol headers data on all layers that are not protected by encryption. In particular, it can use Layer 2 information like MAC addresses, or BLE advertisement packets, to automatically identify additional information, the brand of individual devices, thereby narrowing down the search space of devices to guess the set of smart home devices that the targeted user is using.

**Attacker's goals.** We model the attacker's goals under four different categories:

- Goal-1: The attacker aims to infer the devices used in a smart home. (Section V-D)
- Goal-2: The attacker aims to infer the daily routine of the user. (Section V-E)
- Goal-3: The attacker aims to infer the state of a specific smart home device. (Section V-F)
- Goal-4: The attacker aims to infer specific user activities from the states of multiple devices. (Section V-G)

## III. Smart Home Devices

In this section, we describe the typical characteristics of smart home devices relevant to this paper. First, we classify the smart devices according to their capabilities. This capability-based classification can also be used to classify the device actions. Second, we present required background information about the communication protocols used by these devices.

### A. Capabilities of Smart Devices

We categorize smart devices in our study into three categories in terms of their capabilities. The first category is the *Hub-like devices*. They are central communication hubs that

| | | Communication | | | Capabilities | | |
|---|---|---|---|---|---|---|---|
| ID | Device | WiFi | ZigBee | BLE | Type-I | Type-II | Type-III |
| 1 | ApexisCam | ● | ○ | ○ | ○ | ○ | ● |
| 2 | AirRouter | ● | ○ | ○ | ● | ○ | ○ |
| 3 | AugustSmartlock | ○ | ○ | ● | ○ | ● | ● |
| 4 | BelkinWemoLink | ○ | ○ | ○ | ○ | ● | ○ |
| 5 | DLinkCam | ● | ○ | ○ | ○ | ○ | ● |
| 6 | DLinkDoorSensor | ● | ○ | ○ | ○ | ○ | ● |
| 7 | DLinkMotionSensor | ● | ○ | ○ | ○ | ○ | ● |
| 8 | DLinkSiren | ● | ○ | ○ | ○ | ○ | ● |
| 9 | EdimaxCam | ● | ○ | ○ | ○ | ○ | ● |
| 10 | EdimaxSPlug1101 | ● | ○ | ○ | ○ | ● | ○ |
| 11 | EdinetCam1 | ● | ○ | ○ | ○ | ○ | ● |
| 12 | EdinetGateway | ● | ○ | ○ | ● | ○ | ○ |
| 13 | FitbitAria | ● | ○ | ○ | ○ | ● | ○ |
| 14 | Lightify2 | ● | ○ | ○ | ○ | ● | ○ |
| 15 | PhilipsHueBridge | ● | ○ | ○ | ● | ○ | ○ |
| 16 | SMCRouter | ● | ○ | ○ | ● | ○ | ○ |
| 17 | STMotionSensor | ○ | ● | ○ | ○ | ○ | ● |
| 18 | STOutlet | ○ | ● | ○ | ● | ● | ○ |
| 19 | STMultiSensor | ○ | ● | ○ | ○ | ○ | ● |
| 20 | TPLinkHS110 | ● | ○ | ○ | ○ | ● | ○ |
| 21 | WansviewCam | ● | ○ | ○ | ○ | ○ | ● |
| 22 | WemoInsightSwitch | ● | ○ | ○ | ○ | ● | ○ |

Type-I: Hub-like devices, Type-II: User-controlled devices, Type-III: Sensor-like devices

TABLE I: The communication protocols and capabilities of the smart home devices used.

connect other devices to both each other and to the Internet. They mostly do not provide a functionality of their own to users as their main purpose is to act as gateways connecting devices using other protocols than WiFi to the smart home network. In some cases, like the Samsung ST Hub, they serve as a centralized platform to install and run smart home apps for different smart devices. The second category of devices is the *User-controlled devices*. These devices can be controlled by their users either manually or via a controller device like a smartphone or tablet. Examples of such devices include Smart Lights, Smart Switches or Smart Locks. These devices can be controlled both remotely and locally by the user. The third category is the *Sensor-like devices*. These devices are the most primitive ones and have only the capability of sensing the environment via their built-in sensors. An example of this type of device is the Samsung ST Motion Sensor, which can detect persons moving in its proximity. These devices send notification messages to their associated services either when an event takes place, or periodically. All the devices studied in this paper are shown in Table I.

Apart from these devices, a typical smart home environment uses a smartphone or tablet as a controller device to control smart home devices. The smartphone or tablet can also be used as an interface to connect smart devices and smart home hubs and install different apps on the devices. We consider the smartphone or tablet as the controller device in the user activity inference.

### B. Communication Features

Both the smart home vendors and users mostly prefer wireless communication over wired communication as it is more convenient. However, compared to wired communication, the wireless network traffic from smart home devices is open to the eavesdropping attacks. In this work, we target three wireless protocols: WiFi, ZigBee, and Bluetooth Low Energy (BLE). Among these, WiFi is used in the wired or plugged-in devices, while other protocols, ZigBee and BLE, are implemented for short range communication tasks of battery-powered devices as they consume less power than WiFi.

*1) WiFi-enabled devices:* WiFi-enabled devices are connected to the Internet either through a Hub-like device or directly connected to an access point. In both cases, the adversary can track and capture the traffic through a specific device via MAC address. Even though MAC addresses may help the attacker to narrow down the device type, it can not precisely decide the device type from MAC address. It may want to use IP addresses of servers. However, the adversary can only see the traffic that is encrypted by both the network protocols (SSL/TLS) and WiFi encryption (WPA). Therefore, it cannot see the IP or transport layer headers encrypted by the WPA protocol. This prevents the attacker from using header-based features for the device identification. However, the traffic rates of the devices still cannot be hidden from the attacker.

*2) ZigBee-enabled devices:* ZigBee devices have two addresses: MAC address and Network Address (NwkAddr). The MAC address is exactly the same as the MAC used in WiFi-enabled devices, which is unique for every device in the world and never changes. On the other hand, NwkAddr is created and assigned when the device joins a network and changes when it leaves and re-joins another network. It is similar to IP, however, it is not encrypted and source and destination NwkAddr of the packets can be seen by the attacker. In addition, the network coordinator (i.e., hub) has the $0x0000$ address and each network has a unique identifier, called the Personal Area Network Identifier (PAN ID). This information may additionally help the attacker.

*3) BLE-enabled devices:* In a BLE network, a device can be either a master or a slave. A slave can connect to only one master node while a master can connect to multiple slave nodes. In all the smart home devices that we used, while the smartphone acts as a master, targeted smart device acted as a slave. Before establishing the connection, a slave device broadcasts advertising packets (ADV_IND) randomly on channel 37, 38, and 39. Once a connection starts, they agree on a channel map, where they follow in the rest of the communication. If an attacker wants to follow the BLE traffic through a smart device, it needs to capture the first packet so that it can learn the channel mapping. Once the attacker captures the access address, it can follow the rest of the communication.

## IV. Showing the Privacy Leakage

In this section, we show the feasibility and possibility of privacy leaks from encrypted network traffic of smart home devices. We show that an attacker who can sniff the network traffic of the devices can easily infer some simple information without using any advanced techniques. First, we explain our methodology and tools that we used for collecting the network traffic. Then, we show a case study for each WiFi, ZigBee, and BLE protocols captured from smart home devices.

## A. *Methodology*

*1) Tools:* For all of the protocols, the packet capture, packet field extraction analysis (e.g., timestamp, packet length) and analysis were done using *Wireshark* and *t-shark* tools [5]. However, we used different hardware and software tools to capture the network traffic of each protocol.

**WiFi:** In order to capture the network traffic of WiFi, we used the topology in Figure 2. We used *hostapd* [1] to create an WPA-enabled Access Point (AP) and recorded all the traffic through the specific device with *tcpdump* [3] and *t-shark*. For the hardware, we used Alfa AWUS036AC in monitor mode in order to create a rogue AP.

**ZigBee:** The ZigBee network traffic was captured using Atmel RZUSBStick, which we flashed with *killerbee* [2] firmware and open source killerbee tools were used to capture the traffic.

**BLE:** We used Ubertooth One and open source *Ubertooth* [6] tools to capture the BLE network traffic. Ubertooth One has a follow mode, where it will trace the channel hopping and follow the connection.

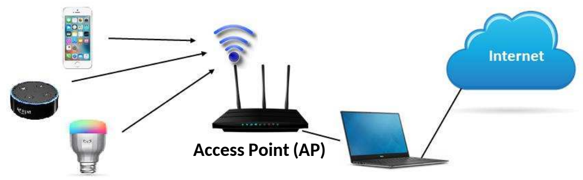


Figure 2: Wifi network traffic capture topology.

## B. *Uniquely Identifying Each Device*

While collecting network traffic from different devices, we installed and collected data from one device at a time[1]. Therefore, we needed to identify the network traffic of each device to analyze it more deeply. Below, we explain how we filter a device for each protocol:

**WiFi:** In WiFi, WPA encryption will normally prevent the attacker from seeing the inside of the packets exchanged between the access point and the client. As long as the key is not known by the attacker, the security of encrypted packets relies on the security of the underlying protocol. However, the MAC addresses of the source and destination are not encrypted. An attacker with a wireless network analyzer will be able to see the MAC addresses of every device in the smart home environment, which allows the attacker to record the network traffic of a specific smart home target device. Even though some methods to avoid this type of eavesdropping attacks such as MAC filtering or randomization were proposed, they are not enough against an attacker targeting a specific device [37].

**ZigBee:** In ZigBee, each node is identified by a unique identifier. There are two variants of this identifier: 1) IEEE/MAC address and 2) Network Address (NwkAddr). While 64-bit long IEEE/MAC address is used to uniquely identify every device in the world, the network address is 16-bit and locally identifies a device in the network. A unique network address is randomly assigned to each end device by the coordinator during the installation of the devices (i.e., joining the network). However, a dedicated network address is used by the device for the rest of the communication after the installation. In terms of the attacker, this allows the attacker to specify its target and filter the packets of the target device. Finally, a smart hub always gets $0x0000$ as the network address; therefore, it can be easily recognized by the attacker.

**BLE:** In BLE, there are two roles that a device can have: central and peripheral. While a peripheral device can only connect to one central device, a central device can connect to multiple peripheral devices simultaneously. A peripheral device broadcasts advertisement packets to announce its availability to be connected by the master devices. In order to filter down of a specific target smart home device, the attacker can use the advertisement packets of the target device by using Advertising Address. Even though it should change over the time, as other studies [17], in our experiments, we observed it does not change Advertising Address. Once the connection has been established, an access address can be assigned. We use this address to filter the connection of the target smart home device. The tools (i.e., Ubertooth) does this automatically by setting it in the follow mode.

## C. *Case Studies*

In this sub-section, we consider one device for each protocol: Wemo Insight Switch Samsung ST Outlet (ZigBee), and August Smart Lock (BLE). We analyze the raw network traffic of each device and see if it is really possible to extract information from the network traffic, specifically from data rate.

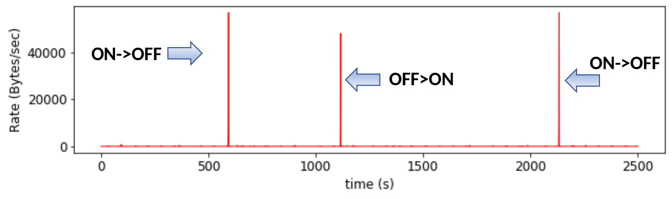
*1) Wemo Insight Switch (WiFi):* Wemo Insight Switch is a Wifi-enabled device and used to monitor and control other appliances (e.g., smart light) from a smartphone. It has only two capabilities: ON and OFF.

Figure 3a shows the data rate of the sample traffic collected from Wemo Insight Switch, where we illustrated a number of actions of the user to change the state of the device. As can be seen from the figure, the data rate shows a significant increase when the device state is changing. Therefore, the data rate clearly reveals the device state changes. In the first peak, the device's state is changed by the user, i.e., the device is turned on and in the second peak, the user turned off the device and so on.

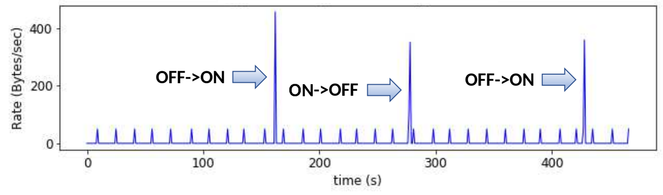
*2) Samsung ST Outlet (ZigBee):* Samsung SmartThings (ST) Outlet uses ZigBee protocol to communicate with Samsung ST Hub. It can also act as a repeater and repeats the broadcast packet of Hub for the smart devices, which is not in the range of Hub. This increase the range of Hub. Other than repeating Hub's broadcasting packets, it has only two capabilities: ON and OFF.

The traffic rate of a sample network capture of Samsung ST Outlet is plotted in Figure 3b. In the given sample network traffic, the device's activity has been changed by the user three times, which clearly corresponds to the three large peaks. On the other hand, small peaks correspond to the repeating of

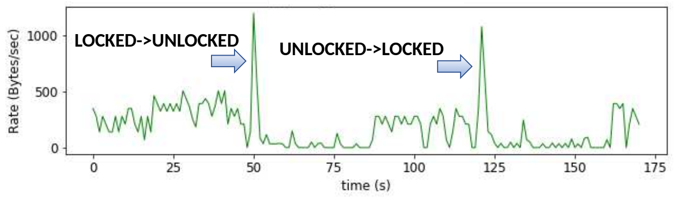
[1]For more detail about this assumption, we refer to multi-device and multi-user discussion in the Section VII



(a) Wemo Insight Switch



(b) Samsung SmartThings Outlet



(c) August Smart Lock

Figure 3: The traffic rates of (a) Wemo Insight Switch (b) Samsung ST outlet, and (c) August Smart Lock. Here, a number of actions are illustrated, with many signals easily discerned by the naked eye. For instance, when the lock is turned on, the significant amount of packets are transmitted and received, which creates a peak in the traffic rate for a certain duration.

the broadcast packets of the hub, which is periodic with 15 seconds.

*3) August Smart Lock (BLE):* The August Smart Lock communicates with the user's smartphone via BLE. In addition to locking and unlocking from the app on the smartphone, the owner (main user) can also give access to guest users through the web servers. The user can also enable the auto-unlock, where the lock is unlocked when the user is in range. However, only the lock does not have the remote control capability. For remote access, it needs other accessories (e.g., WiFi bridge). Here, we only consider the BLE communication between the lock and smartphone.

Figure 3c shows the plot of the sample packet capture of August Smart Lock. As in the previous case studies, the transition between the device's actions can be clearly identified by the attacker. The small increase in the traffic rate in the first part of the capture is because of the advertising packets.

## V. Multi-stage Privacy Attack

As shown in Figure 4, our novel multi-stage privacy attack consists of four stages connected in a cascaded manner. While the goal of the attack is to infer user activities at the final stage, every stage also leaks partial information about devices and their actions and can be independently used by the attacker for various purposes. In the following, we first outline the high-level overview of the attack and then present details of individual stages and related results.

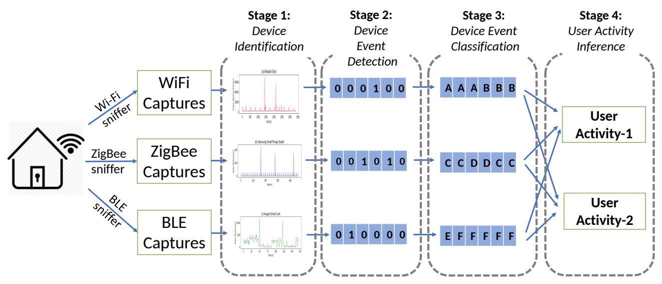


Figure 4: Overview of the multi-stage privacy attack.

### A. Attack Stages

**Stage-1:** In the first stage, the attacker's goal is to identify the type of each smart home device. Even though used protocols use unique identifiers for each device (e.g., MAC address, NwkAddr), the attacker does not know the device type a specific address corresponds to. By sniffing packets of individual protocols, the attacker will obtain network traffic profiles of all devices using that protocol. Identifying individual devices' device types becomes then a multi-class classification task based on the traffic profiles of individual devices.

**Stage-2:** After discovering the types of individual devices the, attacker's goal is to infer the state of individual devices. As shown in Figure 3, a state change typically results in a significant increase in network traffic related to the device, causing an increase in the data rate and decrease in the inter-arrival time of the packets. Therefore, the attacker can in most cases detect state changes of devices by observing changes in these metrics. At the end of this stage, as shown in Figure 4, the attacker converts the network packets into 1s and 0s, where the 1s show where the transition occurred.

**Stage-3:** After detecting transitions between device states, the attacker splits the network trace of a device into segments corresponding to different device states (e.g., ON, OFF). Identifying these states is then reduced to a multi-class classification problems, where classes represent possible device states.

**Stage-4:** In this stage, by using the results of the state classification in Stage-3, the attacker knows the predicted states of all devices. For example, at a particular moment, the attacker may know the smart lock is in the LOCKED state, no motion is detected in the motion sensor placed in the kitchen and so on. Using the state information of the devices, the attacker can guess that the user is sleeping. Any user activity in a smart home can be predicted by observing the predicted states of devices and sensors and using a Hidden Markov Model to predict the corresponding user activity.

In the next sections, we evaluate the efficiency of our multi-stage privacy attack on network traffic data collected from 22 different off-the-shelf IoT devices used in smart homes.

| Device | Period (mins) | Size (MB) | Packets |
|---|---|---|---|
| ApexisCam | 133 | 80 | 152220 |
| AirRouter | 85 | 49 | 115192 |
| AugustSmartLock | 25.8 | 0.66 | 8129 |
| BelkinWemoLink | 71 | 0.66 | 2039 |
| DLinkCam | 225 | 1.15 | 5389 |
| DLinkDoorSensor | 74 | 0.48 | 3519 |
| DLinkMotionSensor | 74 | 0.47 | 2849 |
| DLinkSiren | 71 | 0.41 | 3073 |
| EdimaxCam | 225 | 0.27 | 1798 |
| EdimaxSPlug1101 | 74 | 0.5 | 2823 |
| EdinetCam1 | 117 | 0.3 | 2779 |
| EdinetGateway | 225 | 0.34 | 3240 |
| FitbitAria | 213 | 0.043 | 257 |
| Lightify2 | 74 | 0.25 | 1022 |
| PhilipsHueBridge | 53 | 0.8 | 2680 |
| SMCRouter | 124 | 47 | 150768 |
| STOutlet | 6 | 0.04 | 1061 |
| STMotionSensor | 11 | 0.05 | 1291 |
| STMultiPurpose | 12 | 0.22 | 5255 |
| TPLinkHS110 | 71 | 0.14 | 473 |
| WansviewCam | 193 | 11 | 73759 |
| WemoInsightSwitch | 117 | 0.8 | 1675 |

TABLE II: The characteristics of the network traces used in the experiments.

## B. Dataset and Evaluation Metrics

In order to evaluate the attacks in the stages above, we collected the network data from 22 different smart home devices. We installed all devices in a laboratory network and emulated user inputs triggering device state changes. We captured all network traffic of a device and performed the analysis offline. The duration and the total size of captures and the number of the packets are given in Table II.

For evaluating the efficiency of our attacks, we use different metrics. First, we use accuracy, which is the ratio of correctly predicted observations to total observations. In some cases, as in real deployments, the collected network data may have imbalanced data, where the duration of the active state is much less than the inactive one. In those cases, we use additional metrics such as Precision, Recall, F1 score, and Support. In the cases that the dataset includes a lot more label 0 (no activity) rows than label 1 (activity) rows, we observed that F1 score is a better performance measurement than accuracy although accuracy is a more intuitive performance measurement, in general. The detailed calculation of these evaluation metrics is given in Appendix A.

## C. Calculating Features from Network traffic

In this sub-section, we explain how we use the traffic flow for the classification task. Particularly, we take advantage of the fact that while the encryption layer in the protocol protects the payload of a packet, it fails to hide other information revealed by network traffic patterns, for instance, sequence of packet lengths (SPL) and direction (incoming/outgoing). We consider each network traffic flow as a time ordered sequence of packets exchanged between two peers during a session. Before processing the network traffic for classification, we converted packet in traffic flow into a Sequence of Packet Lengths and Times (SPLT) as in following format:

$$pkt = [timestamp,\ direction,\ packet\ length] \quad (1)$$

where the direction is 1(0) if it is an incoming (outgoing) packet. This transformation is done for each packet in the captured trace, where each result is written to a new row. In the end, we obtained a matrix with three columns. Then, in the feature extraction of each attack, we calculated the features from this matrix.

## D. Stage-1: Device Identification

Several different identification approaches for IoT devices have been proposed in literature. Numerous works have shown that IoT devices can be identified with high accuracy for both WiFi-enabled [26], [16], [10], [25], [28] and BLE-enabled [17] devices. Therefore, in this paper we focus only on ZigBee-enabled smart home devices.

In our dataset, each device can be uniquely identified by the $< brand, device-type >$ pair. We did not consider the different models of devices as different devices. On the other hand, a hub in ZigBee always uses the network address $0x0000$, so it can be easily recognized by the attacker. Therefore, we did not include the hub in the identification of ZigBee devices.

After collecting ZigBee network traffic, the second step involves extracting the features to identify the devices. In this step, the features we used include *mean packet length*, *mean inter-arrival time*, and *standard deviation in packet lengths*. We split each individual network traffic trace of a device into equal time intervals (e.g., 5 sec, 10 sec). Then, we calculated these features for each interval.

For the classification, we used the kNN classification algorithm. The classifier could correctly identify devices with an overall accuracy of 93% for ZigBee devices. This shows that as for WiFi and BLE, also devices using ZigBee can be identified with high accuracy.

## E. Stage-2: Device State Detection

When an interaction between the device and the user occurs, a significant amount of data is transmitted, which leads to a significant increase in the traffic rate. After this data exchange, the data transmission drops to the minimum until a new interaction starts. When there is no activity, only the minimum amount of continuation packets like heartbeat messages are sent to minimize the device's power and bandwidth consumption. We also observed that almost the same amount of data transfer occurs for the same activities. All this information allows us to detect transitions between the activities or states of the device. For further validation, we do the following experiments.

*1) Feature Extraction:* Our goal is to transform a sequence of packets into a supervised learning dataset. To achieve this, we divided the sequence of packets into windows of size $W$. For a given time interval length $W$, we extracted a feature vector comprised of three variables: *mean packet length*, *mean inter-arrival time* and *median absolute deviation of packet size*. Based on timestamped labels telling whether an activity was ongoing or not, we labeled the given vector with 1 for an ongoing activity or 0 for no activity. Window size has the

| Device | Random Forest | | kNN | |
|---|---|---|---|---|
| | F1 Score | Accuracy | F1 Score | Accuracy |
| ApexisCam | 93 | 97 | 94 | 98 |
| AirRouter | 98 | 97 | 98 | 97 |
| AugustSmartLock | 100 | 100 | 100 | 100 |
| BelkinWemoLink | 80 | 79 | 85 | 83 |
| DLinkCam | 85 | 80 | 85 | 80 |
| DLinkDoorSensor | 94 | 98 | 92 | 97 |
| DlinkMotionSensor | 74 | 96 | 69 | 95 |
| DlinkSiren | 89 | 99 | 91 | 99 |
| EdimaxCam | 84 | 82 | 82 | 81 |
| EdimaxSPlug1101 | 91 | 97 | 92 | 97 |
| EdinetCam1 | 76 | 96 | 76 | 96 |
| EdinetGateway | 80 | 99 | 66 | 99 |
| FitbitAria | 100 | 100 | 100 | 100 |
| Lightify2 | 86 | 99 | 81 | 98 |
| PhilipsHueBridge | 74 | 98 | 76 | 98 |
| SMCRouter | 94 | 91 | 100 | 100 |
| STOutlet | 83 | 99 | 92 | 99 |
| STMotionSensor | 91 | 97 | 92 | 97 |
| STMultiSensor | 86 | 99 | 92 | 99 |
| TPLinkPlug1101 | 98 | 99 | 92 | 99 |
| WansviewCam | 91 | 87 | 91 | 86 |
| WemoInsightSwitch | 86 | 98 | 88 | 98 |
| **Avg** | **88** | **99** | **91** | **95** |

TABLE III: Evaluation results of device activity detection stage.

significant influence on the performance of our model. The window size for the best performance depends on adjusting the size according to the duration of the activity. In general, selecting a smaller window size improves the performance until some level, but any further reduction results in decline of the performance. From our observation, better performance was observed when the window size is about a quarter of the duration of an activity.

*2) Results:* After obtaining feature vectors with labels from the sequence of packets, any supervised learning algorithm can be applied on the dataset. We have evaluated two supervised learning algorithms, namely Random Forest classifier (RF) and k-Nearest Neighbors classifier (kNN). As shown in Table III both RF and kNN have similar performance with RF averaging 88% and kNN with 91% average of correctly detecting activities. F1 Score of each device in Table III differs slightly. DlinkMotionSensor has the worst F1 score 74% using RF and 69% using kNN and the best F1 score is 100% for the Aria Fitbit and AugustSmartLock.

### F. Stage-3: Device State Classification

In the device state classification experiments, the attacker's goal is to decide the state of the device (e.g., deciding if it is ON or OFF). When looking at the device's exchanged network packets, unlike previous steps, this is more difficult to determine. However, each state has a unique pattern which helps us to differentiate them from each other. In order to see if it is possible to differentiate the states, we did the following experiments:

*1) Feature extraction:* To conduct device state classification, informative and distinctive features must be extracted from time-series generated in the preprocessing steps. We used the *tsfresh* [4] tool that automatically calculates a large number of time series characteristics and features and then constructed our feature vector. Examples of the features extracted from time-series are as follows:

- Absolute Energy of time-series
- Length of time-series
- Mean of time-series
- Median of time-series
- Skewness of time-series
- Entropy of time-series
- Standard deviation of time-series
- Variance of time-series
- Continuous wavelet transform coefficients
- Fast Fourier Transform Coefficients
- Coefficients of polynomial fitted to time-series

*2) Feature selection:* The output of the feature extraction phase is a set of feature vectors including 795 binary features. A large number of features, some of which redundant or irrelevant might present several problems such as misleading the learning algorithm, and increasing model complexity. A feature selection technique was therefore used to mitigate these problems and also to reduce over-fitting, training time and improve accuracy. We use a technique leveraging ensembles of randomized decision trees (i.e., Extra Trees-Classifier) for determining the importance of individual features. We exploited Extra-Trees Classifier to compute the relative importance of each attribute to inform feature selection. The features considered unimportant were discarded. The feature selection phase reduced the feature vector size from 795 binary features to 197 features.

*3) Results:* Our objective was to build a performant model to correctly classify IoT devices' states even if their traffic is encrypted. To this end, we employed several machine learning algorithms for the classification such as *XGBoost*, *Adaboost*, *Random Forest*, *SVM with RBF kernel*, *kNN*, *Logistic Regression*, *Naïve Bayes*, and *Decision Tree*. In order to ensure that our machine learning model has gotten the most of the patterns from the training data correctly, and its not picking up too much noise, we shuffled and split the data-points to conduct the following experiments: *(i)* we performed 5-fold Cross Validation (CV) on a training set of 377 samples (75% of data) for assessing the effectiveness of the machine learning model and *(ii)* we carried out Hold-out Validation on 126 samples (25% of data) to test the machine learning model performance against unseen data.

**5-fold Cross Validation**: To avoid the risk of missing important patterns or trends in the dataset, we applied cross validation, as it provides ample data for training the model and also leaves ample data for validation. Thus, we conducted a 5-fold cross validation experiment. In 5-fold CV the data are randomly partitioned into 5 equal-sized sub-samples. Of the 5 sub-samples, a single sub-sample is retained as the validation data for testing the model, and the remaining 4 sub-samples are used as training data. The process is then repeated 5 times, with each of the 5 sub-samples used exactly once as the validation data. The 5 results from the folds can then be averaged to

| **Classifier** | **5-fold CV** (75% of data) | **Hold-out data** (25% of data) | | |
|---|---|---|---|---|
| | | Precision | Recall | F1 Score |
| SVC RBF Kernel | 86 | 89 | 87 | 87 |
| Logistic Reg. | 87 | 90 | 89 | 88 |
| **Random Forest** | **92** | 96 | 94 | **94** |
| Naive Bayes | 87 | 92 | 87 | 88 |
| Decision Tree | 66 | 62 | 63 | 61 |
| K-NN | 84 | 91 | 87 | 87 |
| Adaboost | 86 | 89 | 87 | 87 |
| XGBoost | 85 | 91 | 87 | 87 |

TABLE IV: Cross-validation and hold-out validation results for device state classification.

| **Device name** | **Action** | **Pre.** | **Recall** | **F1** | **Supp.** |
|---|---|---|---|---|---|
| ApexisCamera | live view | 100 | 100 | 100 | 4 |
| AirRouter | surfing on amazon | 80 | 100 | 89 | 4 |
| AugustSmartLock | off | 100 | 67 | 80 | 3 |
| AugustSmartLock | on | 67 | 100 | 80 | 2 |
| BelkinWemoLink | off | 80 | 100 | 89 | 8 |
| BelkinWemoLink | on | 100 | 50 | 67 | 4 |
| DLinkCamera | live view | 100 | 100 | 100 | 3 |
| DLinkDoorSensor | open | 100 | 100 | 100 | 5 |
| DLinkSensor | motion detection | 100 | 100 | 100 | 6 |
| DLinkSiren | turn on | 100 | 100 | 100 | 1 |
| EdimaxCam | live view | 100 | 100 | 100 | 1 |
| EdimaxSPlug1101 | on | 100 | 100 | 100 | 5 |
| EdinetCam1 | live view | 100 | 100 | 100 | 2 |
| EdinetGateway | on | 100 | 100 | 100 | 3 |
| FitbitAria | measure weight | 100 | 100 | 100 | 4 |
| Lightify2 | change light type | 100 | 100 | 100 | 6 |
| PhilipsHueBridge | turn scene off | 100 | 100 | 100 | 3 |
| PhilipsHueBridge | turn scene on | 100 | 100 | 100 | 5 |
| SMCRouter | surfing on amazon | 100 | 80 | 89 | 5 |
| STOutlet | on | 100 | 89 | 94 | 9 |
| STMotion | active | 88 | 100 | 93 | 7 |
| STMotion | inactive | 100 | 71 | 83 | 7 |
| STMultiSensor | acceleration active | 100 | 100 | 100 | 8 |
| STMultiSensor | acceleration inactive | 71 | 100 | 83 | 5 |
| TPLinkPlugHS110 | turn off | 100 | 100 | 100 | 5 |
| WansviewCam | reboot | 100 | 100 | 100 | 9 |
| WemoInsightSwitch | on | 100 | 100 | 100 | 2 |
| **Avg./Total** | ——— | **96** | **94** | **94** | **126** |

TABLE V: Hold-out validation results of RF classifier for all IoT devices.

produce a single estimation. We obtained 92% accuracy in terms of F1 Score in the detection of devices' states using Random Forest classifier, as shown in Table IV.

**Hold-out Validation**: To make sure that our classifier can generalize well and is not over-fitted, we tested the classifiers' performance in terms of Precision, Recall, and F1 Score against unseen data (the data was removed from the training set and is only used for this purpose). Table V shows the detailed results obtained by Random Forest classification algorithm when conducting the device state classification over 126 unseen samples. As can be seen, the F1 Score of each device used in the experiment differs slightly. We obtained an average performance measurement of 0.94 (94%) of correctly classifying activities. This shows that an attacker can easily differentiate the devices' states.

### *G. Stage-4: User Activity Inference*

Modern smart home environments comprise several sensors and devices that are connected with each other and

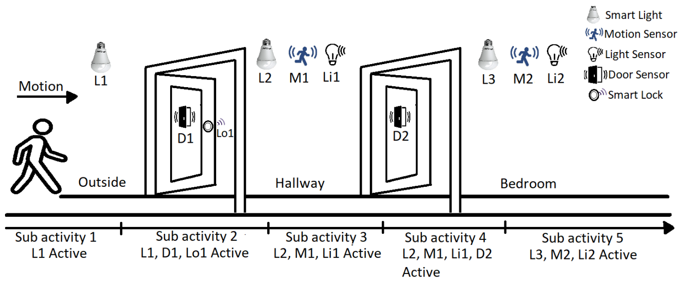


Figure 5: User walking scenario in a smart home environment.

share information. These devices and sensors are configured as independent entities but work co-dependently to provide an autonomous system. Any user activity in a smart home can be predicted by observing the states of the devices and sensors [31].

*1) Modelling User Activities:* In Figure 5, we demonstrate a simple walking scenario of a user. Here, a user is entering the smart home from outside to the bedroom through the hallway. The scenario consists of five different devices with lights both inside and outside the home controlled by the motion sensor (M) and light sensor (L). This simple activity can be illustrated as a sequential pattern: Sub-activity 1- moving towards the door from outside (L1 is active), sub-activity 2- user opens the front door (L1, D1, Lo1 are active), sub-activity 3- user enters the hallway (L2, M1, Li1 are active), sub-activity 4- user enters the room (Li2, L2, M2, D1, Lo1 are active), sub-activity 5- user inside the home (L2, M2, Li2 are active). To complete the activity, a user must follow the same sequence of sub-activities and complete each step. As discussed earlier, the devices' states (active/inactive) for a specific time can be determined from the network traffic captured from the devices. These device states can be used to infer an on-going activity in a smart home setting.

*2) Feature Extraction:* To infer user activities, different device features must be extracted from network traffic data. Network traffic data contain several features including timing information, sensor information, device states, location, etc. Based on the data-type, the extracted features from the network traffic for user activity inference can be represented as follow.

$$Data\ array, E_T = \{S, D, M, L\}, \tag{2}$$

where T is the set of timing features extracted from the network traffic, S is the set of sensors' features, D is the set of device features, M is the features extracted from the controlling device (smartphone/tablet), and L is the set of location features extracted from the network traffic. We describe the characteristics of these features below.

- *Timing features (T):* Smart home devices change their state according to user activities and commands. Some devices perform time-independent tasks (e.g., switching lights with motion), while some devices perform a task in a certain pattern with different user activities (e.g., walking from one point to another) based on smart home settings. We extract the time of an event from the network traffic captured from different devices to build the overall state of the smart home at the time of the user activity.

- *Sensor State features (S):* Smart home environment consists of different sensors (e.g., motion sensor, light sensor, door sensor, etc.) which act as a bridge between devices and the peripheral. Sensors in a smart home can sense different environment parameters which can trigger different pre-defined tasks in multiple devices. Moreover, sensors can sense any change occurred because of a user interaction and forward this information as an input to the associated devices. These sensor data can be both logical (motion sensor) and numerical (temperature sensor) depending on the nature of the sensor. We observe the changes in both logical and numerical value of a sensor from the captured network traffic and use as a feature to infer user activities. We represent the changes in sensor data as binary output: 1 for active state and 0 for inactive state.

- *Device State features (D):* In a smart home environment, multiple devices such as smart light, smart thermostat, etc. can be connected with each other and with a central hub to perform different tasks. These devices can be configured to change their states (active/inactive) to perform a pre-defined task or to perform a task based on user activities. We consider the state information of all the connected devices as features and extract this information from captured network traffic to infer the on-going user activity. The active and inactive states of the devices are illustrated as 1 and 0 respectively in the data array.

- *Controller State features (M):* Smart home devices can be controlled in an autonomous way and also by using a controller device (smartphone/tablet). To understand the changes in states of the sensors and devices, one should consider the control commands generated by the controller devices. We consider the state of controller device as active (represented as 1 in data array) when a user interacts with smart home devices via controller device and inactive otherwise (represented as 0 in data array). This state information of the controller devices can be extracted from the captured network traffic to build the data array.

- *Controller Location features (L):* The devices connected in a smart environment can be controlled from a different location and this location information can be collected from the captured network traffic. We consider the location of the controller device as a feature to understand any activities on smart home. We consider the home location of the controller device as 1 and the away location of the controller as 0 to represent the location feature as a binary number in the data array.

For Stage 4, we captured the network traffic from a smart home environment and create the feature array explained in Equation 2. We captured the network traffic for a specific time to correctly portray user activities from the network data. Each element of the data array represents the operating conditions of different smart devices, sensors, and controller devices. These data were then used to train a Hidden Markov Model (HMM) to detect user activities in a smart home environment.

HMM is a statistical Markov model, where each state of the model contains unobserved states. In traditional Markov model, all the states of an ongoing process are observable while in Hidden Markov model the states are not directly visible. Here, only the output depending on the states is visible. The main assumptions of HMM are similar to the Markov Chain model which are as follows: (1) The probability of occurring a particular state depends only on the previous state. (2) The transition between two consecutive states is independent of time. (3) Hidden states are not visible, but each hidden state randomly generates one of the defined observations or visible states. We use these properties of HMM to detect different user activities from the captured network traffic in a smart home environment. The probabilistic condition of HMM is shown in Equation 3, where $X_t$ denotes the state at time $t$ for a user activity in a smart home [30].

$$\begin{aligned} P(X_{t+1} = x | X_1 = x_1, X_2 = x_2 ..., X_t = x_t) = \\ P(X_{t+1} = x | X_t = x_t), \\ when,\ P(X_1 = x_1, X_2 = x_2 ..., X_t = x_t) > 0 \end{aligned} \quad (3)$$

For each activity in the smart home environment, multiple feature arrays were created and these arrays maintain different, but specific sequences for different user activities. For a specific time, $t$, the state of the smart home can be represented by the data array $E_T$ where each element of this data array illustrates the conditions of smart home devices' and sensors' as binary output (1 for active status of an entity and 0 for inactive status). Thus, each state can be represented as an n-bit binary number, where n is the total number of devices in the smart home. Let assume the smart home environment is in state $i$ at time $t$ and changing to state $j$ at time $t + 1$. The transition probability from state $i$ to state $j$ can be noted as $P_{ij}$. If the smart home environment comprises of n number of devices and $m = 2^n$ states in the system, the transition matrix of HMM is given as follows:

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \dots & \dots & P_{1m} \\ P_{21} & P_{22} & P_{23} & \dots & \dots & P_{2m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ P_{m1} & P_{m2} & P_{m3} & \dots & \dots & P_{mm} \end{bmatrix} \quad (4)$$

If the smart home environment has $X_t$ number of states where $t = 0, 1, \dots, T$, the elements of the transition matrix can be shown as follows [27]:

$$P_{ij} = \frac{N_{ij}}{N_i}, \quad (5)$$

where $N_{ij}$ denotes the number of transition from $X_t$ to $X_{t+1}$, where $X_t$ is the state at time $t$ and $X_{t+1}$ is the state at time $t + 1$.

To build the observation probability matrix, we consider different user activities as hidden states of the smart home environment and correlates with the system's states build from the data arrays. Let assume the smart home environment has k number hidden states (H) in the system. The observation

| Task Category | Task Name |
|---|---|
| Time-independent | 1. Controlling device within smart home. |
| | 2. Controlling device from outside of the home. |
| | 3. Presence in a specific point at home. |
| Time-dependent | 4. Walking in the smart home. |
| | 5. Opening/ closing doors/windows. |
| | 6. Entering/ exiting from smart home |

TABLE VI: Typical activities of users in a smart home environment.

probability matrix of HMM is given as follows:

$$B = \begin{bmatrix} X_1(H_1) & X_2(H_1) & X_3(H_1) & \dots & \dots & X_m(H_1) \\ X_1(H_2) & X_2(X_2) & P_3(X_2) & \dots & \dots & X_m(H_2) \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ X_1(H_k) & X_2(H_k) & X_3(H_k) & \dots & \dots & X_m(H_k) \end{bmatrix} \quad (6)$$

where $X_m(H_k)$ is the probability of observing $H_k$ from state $X_m$. $X_m(H_k)$ can be represented by Equation :

$$X_m(H_k) = P(H_k|X_m), \quad (7)$$

For our work, we want to detect the hidden state (user activity) from a given state sequence. To calculate the probability of user activity, we use the Forward-Backward (FB) algorithm to decode HMM. The FB algorithm can be expressed by the following equations.

$$Forward\ recursion, P_m(t+1) = B_{mH_{t+1}} \sum_{a=0}^{m} P_a(t) P_{am} \quad (8)$$

$$Backward\ recursion, B_i(t) = \sum_{b=1}^{k} P_{ij} B_{jH_{t+1}} B_{j(t+1)}, \quad (9)$$

where, t= 0,1, ..., T-1. The probability of occurring a hidden state (user activity) from the sequence of observable states (device states) can be calculated from the following equation.

$$P(H_1, H_2, \dots, H_k) = \sum_{l=1}^{K} P_k(t) B_k(t). \quad (10)$$

To train this HMM, we collected data from a smart home environment with real smart devices. We consider common smart home devices to build our training environment [18]. Our test smart home environment included Samsung SmartThings hub, Samsung multipurpose sensor, Samsung motion sensor, Netgear Arlo security camera, Philips Hue smart light, Ecobee Smart Thermostat, and August Smart Lock. We collected network traffic data from 10 different users for different user activities.

*3) Activity Types:* User activities in a smart home environment can be instantaneous (e.g., switching on a device) or sequential over time (e.g., walking from one place to another). We categorized user activities in a smart home environment in two categories - time-independent and time-dependent user activities.

- *Time-independent Activities:* These user activities are instantaneous, non-sequential activities which do not depend on time. For example, a user can switch on/off a device in the smart home environment at a specific time instance. This activity will show changes in different features for only one time.
- *Time-dependent Activities:* These user activities are time-dependent, sequential activities. For example, a user can move from one point to another point. This activity will show changes in different features over time in a specific sequence.

We test our HMM model with data collected from six different user activities. Our user activity model is explained below.

- *User Activity- 1.* A user is controlling a device from inside of the smart home environment.
- *User Activity- 2.* A user is controlling a device from outside of the smart home environment.
- *User Activity- 3.* A user is performing tasks from a specific point of a smart home environment.
- *User Activity- 4.* A user is walking from one point to another inside the smart home environment.
- *User Activity- 5.* A user is entering/ exiting from the smart home environment.
- *User Activity- 6.* A user is opening/ closing a window/ door in smart home environment.

| Smart Home User Activity | TPR | FNR | TNR | FPR | Accuracy | F-score |
|---|---|---|---|---|---|---|
| Activity-1 | 1 | 0 | 1 | 0 | 1 | 1 |
| Activity-2 | 1 | 0 | 1 | 0 | 1 | 1 |
| Activity-3 | 1 | 0 | 1 | 0 | 1 | 1 |
| Activity-4 | 0.96 | 0.03 | 0.94 | 0.05 | 0.95 | 0.95 |
| Activity-5 | 0.95 | 0.04 | 0.87 | 0.12 | 0.93 | 0.91 |
| Activity-6 | 0.97 | 0.02 | 0.91 | 0.08 | 0.94 | 0.94 |

TABLE VII: User activity inference from network traffic data in a smart home environment.

*4) Results:* To train our proposed HMM for user activity inference, we collected user activity data for a week from 15 different people (total 30 datasets) in an emulated smart home environment. We asked the users to perform their daily activities in a timely manner (from morning to night) and performed the same activities in defined sequences in a real-life smart home setting. We considered single authorized smart home user interacting with smart devices at a time for data collection. We trained our HMM model with these data. We also collected data for this activity model to test our proposed method. We collected two datasets for each activity (12 in total) to test the efficacy of the activity inference model.

In Table VII, the evaluation results of our activity inference model are shown. For time-independent activities (Activity-1, Activity-2, and Activity-3), one can infer with 100% accuracy and F-score from the captured network traffic data in a smart home environment. On the contrary, accuracy and F-score decreases slightly for time-dependent activities as these activities introduce FP and FN instances in the activity inference model. For activity-4, our proposed stage 4 activity inference HMM can achieve both accuracy and F-score over 95%. The false positive rate (FPR) and false negative rate (FNR) are over

5% and 3% respectively for Activity-4. For Activity-4 and Activity-5, the accuracy of user activity inference decreases (93% and 94% respectively) while FPR and FNR increases. The reason for the increment of FPR and FNR is that different time-dependent user activities can have similar patterns over time with small changes in specific time instances. This affects the probability of occurring an activity calculated from HMM. *In summary, an attacker can infer time-independent activities more accurately (with 100% accuracy and F-score) than the time-dependent activities (with over 95% accuracy and F-score).*

Finally, note that an accurate user activity inference means that all the stages in the multi-stage attack have to be correctly guessed, which may lower the end-to-end successful inference rate of the attacker. For example, if the stage 1, 2, 3, and 4 are $X$, $Y$, $Z$, and $T$, respectively, for an attacker, the probability of correctly guessing the activity 4 of the user is $X \times Y \times Z \times T$. However, we also note that independently inferred information in every stage is also valuable as it may also include sensitive information (e.g., inferring the device type of a connected medical device may reveal the health status of the subject [35]).

## VI. Mitigating the Privacy Leaks

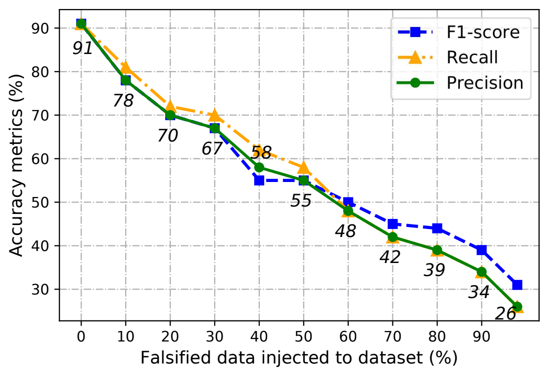
Despite all the security vulnerabilities exploited, as these privacy concerns are *inherent* and *insidious*, it is too hard to detect and avoid these types of threats associated with smart home devices. An attacker can passively listen to the wireless medium and record all the network traffic from a smart home environment without interrupting the normal activities of devices and their users.

### A. Straightforward Solutions

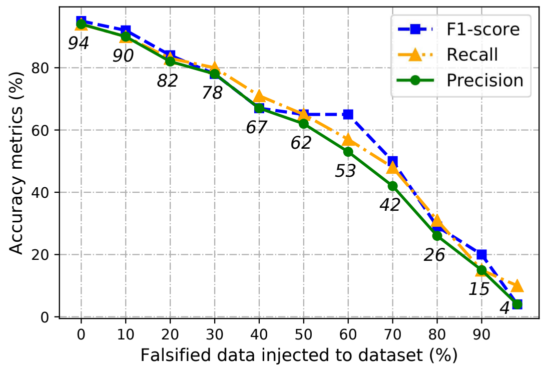
*1) Using VPN or Tor-like Tools:* The use of VPN will prevent an attacker from recording the victim's traffic after the gateway as it is going to be encrypted by the VPN provider. Even an ISP cannot record the network traffic of the user. On the other hand, Tor will make the source and destination IP impossible to determine for the ISP. Both methods will prevent the attacker to follow the communication between the home AP and the server of the hub. However, it provides no protection against an attacker within range (e.g., outside, near home) and sniffing the internal traffic.

*2) Signal Attenuation:* A signal attenuator can be used in theory to protect from an attacker sniffing the internal network traffic of the smart home. This can be realized via a wired connection or using Faraday cages [32]. Nonetheless, forcing all the devices to such a modification in the hardware level and a Faraday cage could be too unrealistic and very expensive to set up for the smart home users.

*3) Traffic Shaping:* The traffic shaping solutions have been widely studied in the literature of website classifiers. Padding to proper MTU, exponential padding, or random padding are some of the countermeasures with the traffic shaping methods. Indeed, not only padding, but also constant or random delays can be applied to the packets transmitted to protect from inference attacks. In all these solutions, the underlying protocol, which needs to provide a real-time accurate values from the devices, is modified in a way that unfortunately lowers the efficiency and accuracy of the devices.



(a) Impact of false data injection on device state detection



(b) Impact of false data injection on device state classification

Figure 6: (a) Device state detection results when injecting false data, (b) Device state classification results when injecting false data.

### B. Proposed Approach

In this sub-section, we propose a solution based on generating spoofed traffic. In this way, even if the user is not at home, generating false activity for the user's presence traffic will mask the user's absence.

In order to measure the efficacy of our proposed spoofed traffic, we simulated the injection of false packets by modifying the feature vectors and evaluated how the performance measurements would change. Then, we applied it to the device state detection and device activity classification attacks. Since the user activity inference is based on the results of the device state detection and device activity classification attacks, if we can falsify their results, the attacker will not able to infer the activities correctly. Particularly, we conducted a set of experiments where we injected falsified data into the training set to observe how the previously shown detection and classification algorithms would behave in such a situation. The results are shown in Figure 6.

**Impact of False Data Injection on Device State Detection.** Figure 6a shows the average of the accuracy measures for the KNN algorithm after increasingly injecting false packets. When there is no injected false packet, all of the devices have 91% F1 score, then it linearly decreases with the increase of false packets. For example, injecting false data equivalent to

10% of packets exchanged during the observation time resulted in a decrease by 13%. For 90% false traffic addition, the accuracy of device state detection declined by about 57%. This shows that traffic injection can be efficiently used for hiding the state of devices from the adversary.

**Impact of False Data Injection on Device State Classification.** We injected the falsified data into the training data and computed the accuracy metrics in terms of F1 Score, Precision, and Recall. We injected 10% falsified data and continued injecting until 90% of the dataset contained false data. As can be seen in Figure 6b, the F1 Score plunges dramatically when injecting 90% false data and reaches 15%. This is due to the fact that randomly falsified features deteriorate traffic patterns used for classifying the devices' states. Also here we can see that by injecting increasing amounts of fabricated traffic, the adversary can effectively be prevented from making inferences about the types of device events occurring.

## VII. DISCUSSION

**Multi-user vs. single user:** Smart home devices support multiple authorized users. In a multi-user smart home scenario, more than one user can control and change the settings of smart devices. Additionally, different users can perform different activities within the smart environment at a time. This can create some false positive and false negative cases in user activity inference using our proposed method. Nonetheless, an attacker can still infer the device type and devices states from the network traffic. Additionally, the attacker can also infer the presence of multiple users and the specific point of ongoing activities in multi-user smart home environment using the network traffic. Compared to a multi-user scenario, a single user smart home environment is more vulnerable to our proposed threat as it is easier to infer a single on-going user activity in the smart home.

**Local vs. remote control:** To improve the user control over smart devices and increase convenience, smart homes offer remote access control in addition to traditional local access. Our proposed threat model can guess both local and remote access from location feature of the captured network traffic. This is a serious threat to user privacy as attackers can detect when user is changing the state of a specific device remotely and perform malicious activities. For example, an attacker can infer when a user is accessing the smart lock remotely and acquire physical access to the home environment.

**Smart device diversity:** Smart devices have no common network protocols. Indeed, some of them such as WiFi, ZigBee, and BLE are more popular than others. This makes it harder to sniff all the devices that the smart home user is using. In addition to the diversity of network protocols, smart home devices come with different computational resources, hardware types, capabilities, exchanged data format etc. All of these differences in smart devices make it very challenging to build a generic solution as well as an attack. However, with our automated multi-stage privacy-attack, we showed the feasibility of the attack with the most popular network protocols, which covers the most of the commercial devices.

**Limitations of defense:** As our results show, injecting false data to the communication clearly decreases the accuracy of the attacks. However, even though it is an effective method and it has the advantage of not affecting the efficiency of real traffic on the devices, but it requires an extension to the protocols to put a flag on the fake activities, which will be known by both devices and server. Here, we propose two different ways to implement this solution with trade-offs on the power consumption and security.

1) *Only on the Hub*: This countermeasure can be implemented only on the smart hub devices and does not require relatively-constrained smart home devices to be part of the countermeasure. Even though this type of solution is effective and better for battery-powered devices, it can be discovered by the attacker; after a while, an attacker can use the network traffic from the device(s) to hub only, but the attacker's success and accuracy will decrease.
2) *On the Hub and Device*: This countermeasure requires to modify the communication protocol both on the smart home device and the hub. This will generate a more realistic interaction between the device and the hub, but this may cause slightly more power consumption (depending on the size of the extra field for the flag) in the device and requires to modify the devices to send the false data.

## VIII. RELATED WORK

**Identification using the encrypted network traffic .** The meta-data (e.g. MAC, traffic rate) of encrypted network traffic triggers possible threats including unintentional disclosure of the content or user. There is an extensive literature in the identification of the content from the encrypted network traffic. For example, web page identification [34], web user identification [24], protocol identification [39] are some of the research on the identification using the encrypted traffic. Not only identification attacks, but also the countermeasures have been studied in several studies [19], [12].

**Smartphone Fingerprinting.** Recently, this research has been extended to smartphone users. For example, Conti et al. [14] showed a way of identifying user action on Android apps and Taylor et al. [36] presented their work the fingerprinting of apps from an encrypted network of the smartphone. In addition, [33] fingerprints the smartphones using the network traffic captured generated from the popular applications such as Facebook, WhatsApp. Finally, in [9], Ateniese et al. showed a new adversary model that can infer the location of the user from the encrypted network traffic.

**Fingerprinting Methods.** In all the aforementioned studies, either statistical techniques [38] or machine learning methods [14] were used to infer different sensitive information about the user and the context. Even ML has been used for the task of identification such as user, device, or website identification, in none of these studies, the attacks are timing-based as we have in our work.

**IoT Fingerprinting.** So far, in all the aforementioned studies the results showed that the used methods are efficient and the threat is real, but the threat was limited to the web and online privacy of the user. Now with the emergence of IoT, it has been extended every part of our daily lives and, with this, threats and countermeasures have also evolved [7], [30]. When we look at the IoT context, there have been only a few works on the fingerprinting research. Miettinen et al. [26] presented a device type identification method using the packet header-based features (e.g., protocol, port number) and Bihl et al. [11] proposes an optimization framework that can be applied to Z-Wave device fingerprinting. These works are on device fingerprinting. On the other hand, the most closest works to ours are presented in the user action identification from the network traffic of IoT devices. In [15], Copos et al. shows their work on inferring the user activity from Nest Thermostat and Nest CO Detector, where they are only doing with a simple correlation analysis. The most similar studies to ours are [32], [8], where in [8], the similar privacy analysis has been done only on 7 devices with WiFi and in [32], the analysis has been done with motion and contact sensors only. In both [32], [8], the inference attack covers only device activity classification.

**Difference from existing work.** However, in our work, we included a more comprehensive analysis of the ecosystem. Specifically, we are working on 22 devices supporting a wide range of protocols (i.e, Wifi, ZigBee, and BLE) where earlier works only evaluate their attack models on the Wi-Fi traffic. We are extending the evaluation of privacy attacks more in-depth by using ML methods to automatize the detection and classification of device states and we use HMM in order to model and guess the on-going user actions and activities, which is another novel contribution of this work. We also discuss the limitations of their proposed countermeasures, which is traffic shaping, and propose another approach to defend against the novel multi-stage privacy attack proposed in this paper.

## IX. Conclusion

In this paper, we explored how encrypted network traffic from a smart home environment can be used to infer sensitive information about smart devices. Specifically, we introduced a novel multi-stage privacy attack, which an attacker can exploit to automatically detect and identify particular types of devices, their actions, states, and related user activities by passively monitoring the wireless traffic of smart home devices. Our evaluation on an extensive list of off-the-shelf smart home devices and real users showed that an attacker can achieve very high accuracy (above %90) in all the attack types. As opposed to to earlier straightforward activity identification approaches, the novel multi-stage privacy attack can perform detection and identification automatically, is device-type and protocol-agnostic, and does not require extensive background knowledge or specifications of analyzed protocols. Finally, we propose a new yet effective mitigation mechanism to hide the real activities of the users. The effectiveness of the multi-stage privacy attack raises serious privacy concerns for any smart environment equipped with smart devices including personal homes, residences, hotel rooms, offices of corporations or government agencies. Hence, the security community should further investigate this novel attack surface and develop solutions to ensure the privacy of smart home users.

## References

[1] "hostapd," https://w1.fi/hostapd/, 2018.

[2] "killerbee," https://github.com/riverloopsec/killerbee, 2018.

[3] "tcpdump," https://www.tcpdump.org/tcpdump_man.html, 2018.

[4] "tsfresh," https://github.com/blue-yonder/tsfresh, 2018.

[5] "tshark - the wireshark network analyzer 2.6.0," https://www.wireshark.org/docs/man-pages/tshark.html, 2018.

[6] "Ubertooth," https://github.com/greatscottgadgets/ubertooth, 2018.

[7] A. Acar, H. Aksu, A. S. Uluagac, and K. Akkaya, "Waca: Wearable-assisted continuous authentication," in *Security and Privacy Workshops (SPW), 2018 IEEE*, 2018.

[8] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster, "Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic," *arXiv preprint arXiv:1708.05044*, 2017.

[9] G. Ateniese, B. Hitaj, L. V. Mancini, N. V. Verde, and A. Villani, "No place to hide that bytes wont reveal: Sniffing location-based encrypted traffic to track a users position," in *International Conference on Network and System Security*. Springer, 2015, pp. 46–59.

[10] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Iotsense: Behavioral fingerprinting of iot devices," *arXiv preprint arXiv:1804.03852*, 2018.

[11] T. Bihl, M. Temple, and K. Bauer, "An optimization framework for generalized relevance learning vector quantization with application to z-wave device fingerprinting," in *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[12] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 605–616.

[13] Z. B. Celik, L. Babun, A. K. Sikder, H. Aksu, G. Tan, P. McDaniel, and A. S. Uluagac, "Sensitive information tracking in commodity iot," in *USENIX Security Symposium*, Baltimore, MD, August 2018.

[14] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing android encrypted network traffic to identify user actions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 114–125, Jan 2016.

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

[16] A. K. Dalai and S. K. Jena, "Wdtf: A technique for wireless device type fingerprinting," *Wireless Personal Communications*, vol. 97, no. 2, pp. 1911–1928, 2017.

[17] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, "Uncovering privacy leakage in ble network traffic of wearable fitness trackers," in *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*. ACM, 2016, pp. 99–104.

[18] C. de Looper. (2017) The 12 best smart home devices you need to live like the jetsons. [Online]. Available: http://www.businessinsider.com/best-smart-home

[19] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *Security and Privacy (SP), 2012 IEEE Symposium on*. IEEE, 2012, pp. 332–346.

[20] K. Fawaz and et al., "Protecting Privacy of BLE Device Users." in *USENIX Security Symposium*, 2016.

[21] D. Formby and et al., "Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems." in *NDSS*, 2016.

[22] J. Han, A. J. Chung, M. K. Sinha, M. Harishankar, S. Pan, H. Y. Noh, P. Zhang, and P. Tague, "Do you feel what i hear? enabling autonomous iot device pairing using different sensor types," in *Do You Feel What I Hear? Enabling Autonomous IoT Device Pairing using Different Sensor Types*. IEEE, 2018, p. 0.

[23] H. Li, Z. Xu, H. Zhu, D. Ma, S. Li, and K. Xing, "Demographics inference through wi-fi network traffic analysis," in *Computer Communications, IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on*. IEEE, 2016, pp. 1–9.

[24] M. Liberatore and B. N. Levine, "Inferring the source of encrypted http connections," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 255–263.

[25] Y. Meidan, M. Bohadana, A. Shabtai, M. Ochoa, N. O. Tippenhauer, J. D. Guarnizo, and Y. Elovici, "Detection of unauthorized iot devices

using machine learning techniques," *arXiv preprint arXiv:1709.04647*, 2017.

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

[27] N. T. Nguyen, D. Q. Phung, S. Venkatesh, and H. Bui, "Learning and detecting activities from movement trajectories using the hierarchical hidden markov model," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 955–960.

[28] T. D. Nguyen, S. Marchal, M. Miettinen, M. H. Dang, N. Asokan, and A.-R. Sadeghi, "D\" iot: A crowdsourced self-learning approach for detecting compromised iot devices," *arXiv preprint arXiv:1804.07474*, 2018.

[29] A. K. Sikder, A. Acar, H. Aksu, A. S. Uluagac, K. Akkaya, and M. Conti, "Iot-enabled smart lighting systems for smart cities," in *Computing and Communication Workshop and Conference (CCWC), 2018 IEEE 8th Annual*. IEEE, 2018, pp. 639–645.

[30] A. K. Sikder, H. Aksu, and A. S. Uluagac, "6thsense: A context-aware sensor-based attack detector for smart devices," in *USENIX Security*, 2017.

[31] A. K. Sikder, G. Petracca, H. Aksu, T. Jaeger, and A. S. Uluagac, "A survey on sensor-based threats to internet-of-things (iot) devices and applications," *arXiv preprint arXiv:1802.02041*, 2018.

[32] V. Srinivasan, J. Stankovic, and K. Whitehouse, "Protecting your daily in-home activity information from a wireless snooping attack," in *Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008, pp. 202–211.

[33] T. Stöber, M. Frank, J. Schmitt, and I. Martinovic, "Who do you sync you are?: smartphone fingerprinting via application behaviour," in *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*. ACM, 2013, pp. 7–12.

[34] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted web browsing traffic," in *Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on*. IEEE, 2002, pp. 19–30.

[35] T. Sderholm. (2017) Eu gdpr: Privacy for connected medical devices. [Online]. Available: https://blog.nordicsemi.com/getconnected/eu-gdpr-privacy-for-connected-medical-devices

[36] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 439–454.

[37] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why mac address randomization is not enough: An analysis of wi-fi network discovery mechanisms," in *AsiaCCS*, 2016.

[38] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.

[39] C. V. Wright, F. Monrose, and G. M. Masson, "On inferring application protocol behaviors in encrypted network traffic," *Journal of Machine Learning Research*, vol. 7, no. Dec, pp. 2745–2769, 2006.

## Appendix

To evaluate our proposed novel attack, we used seven different performance metrics: True Positive Rate (TPR), False Negative Rate (FNR), True Negative Rate (TNR), False Positive Rate (FPR), Precision, Accuracy, and F1-score. These can be calculated using following equations:

$$TPR\,(Recall) = \frac{TP}{TP+FN} \tag{11}$$

$$FNR = \frac{FN}{TP+FN} \tag{12}$$

$$TNR = \frac{TN}{TN+FP} \tag{13}$$

$$FPR = \frac{FP}{TN+FP} \tag{14}$$

$$Precision = TP/(TP+FP) \tag{15}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{16}$$

$$F1-score = \frac{2*TP*TN}{TP+TN} \tag{17}$$

where $TP = True\ Positive$, $FP = False\ Positive$, $TN = True\ Negative$ and $FN = False\ Negative$.