IoTAthena: Unveiling IoT Device Activities from Network Traffic

Yinxin Wan, *Student Member, IEEE,* Kuai Xu, *Senior Member, IEEE,* Feng Wang, *Member, IEEE,* Guoliang Xue, *Fellow, IEEE*

***Abstract*—The recent spate of cyber attacks towards Internet of Things (IoT) devices in smart homes calls for effective techniques to understand, characterize, and unveil IoT device activities. In this paper, we present a new system, named IoTAthena, to unveil IoT device activities from raw network traffic consisting of timestamped IP packets. IoTAthena characterizes each IoT device activity using an activity signature consisting of an ordered sequence of IP packets with inter-packet time intervals. IoTAthena has two novel polynomial time algorithms, `sigMatch` and `actExtract`. For any given signature, `sigMatch` can capture all matches of the signature in the raw network traffic. Using `sigMatch` as a subfunction, `actExtract` can accurately unveil the sequence of various IoT device activities from the raw network traffic. Using the network traffic of heterogeneous IoT devices collected at the router of a real-world smart home testbed and a public IoT dataset, we demonstrate that IoTAthena is able to characterize and generate activity signatures of IoT device activities and accurately unveil the sequence of IoT device activities from raw network traffic.**

***Index Terms*—Wireless networking, IP packets, network traffic collection and analysis, time-sensitive subsequence matching, polynomial time algorithms, unveiling IoT device activities.**

1. Introduction

In today's smart homes, various IoT devices can connect to the Internet via home routers with wired cable connections or wireless communications such as WiFi, Bluetooth, and ZigBee. The proliferating IoT devices in smart homes bring many innovative applications and services such as improved home automation and safety, efficient energy, and connected healthcare. However, the recent spate of cyber attacks and threats [3, 7, 14, 19, 25, 30–32, 36, 46] towards a wide range of IoT devices with flawed system designs and weak security management calls for effective techniques to understand, characterize, and unveil the detailed activities of heterogeneous IoT devices, e.g., *when* and *how* a smart lock is opened.

In the research literature, there have been some prior studies on characterizing behavioral patterns of IoT devices and identifying IoT device types and activities using supervised machine learning models or the simple request/reply pattern matching [1, 5, 18, 23, 27, 35, 37, 39–41, 44]. However, little effort has been devoted to the understanding and characterization of full and detailed signatures of IoT device activities which shed light on when and how IoT devices communicate with cloud servers and smartphones for carrying out what device activities.

This paper presents IoTAthena, a system to generate detailed signatures of IoT device activities from IoT network traffic and to unveil IoT device activities via time-sensitive subsequence matching. IoTAthena first collects the background traffic of IoT devices and the normal network traffic triggered by IoT device activities with packet capturing tools on programmable home routers. It then characterizes each IoT device activity using an activity signature consisting of an ordered sequence of IP packets with inter-packet time intervals[1].

Our IoT device activity signature generation is inspired by PingPong [37], which generates packet-level signatures of IoT device activities in the form of abstracted packet-pairs with ping/pong, i.e., request/reply, patterns. It has been demonstrated [37] that many IoT device activities can be efficiently captured with the use of ping/pong like signatures. However, such short signatures have their limitations. For example, our experiments of running the PingPong open source package were unable to generate signatures of the WiFi and Bluetooth locking or unlocking activities of August Lock, as well as the autolocking activity. In contrast, IoTAthena's detailed activity signature carries crucial information for characterizing more IoT device activities than [37] and differentiating IoT device activities with overlapping packet pairs or packet sequences.

To unveil IoT device activities from network traffic logs, IoTAthena relies on two novel algorithms, `sigMatch` and `actExtract`. The `sigMatch` algorithm can effectively capture all matches of a given activity signature from the network traffic log in polynomial time. Using `sigMatch` as a subfunction, the `actExtract` algorithm can accurately unveil the sequence of IoT device activities from raw network traffic logs, also in polynomial time. Our experimental evaluations of IoTAthena were based on 16 IoT devices in a real-world smart home environment, and a public IoT dataset of 25 IoT devices [29]. Our experimental results showed that IoTAthena can effectively generate the detailed signatures of IoT device activities and accurately unveil future activities of these IoT devices.

The main contributions of this paper are the following:

- We develop a systematic approach to programmatically generate detailed signatures of IoT device activities consisting of an ordered sequence of IP packets with inter-packet time intervals.

Wan, Xu, Wang, and Xue are all affiliated with Arizona State University. Emails: {ywan28; kuai.xu; fwang25; xue}@asu.edu. This research was supported in part by NSF grants 1816995, 1717197, and 1704092. The information reported here does not reflect the position or the policy of the funding agency.

[1] An inter-packet time interval is calculated from the timestamp difference of two consecutive IP packets in the packet sequence.

- We design two novel polynomial time algorithms, `sigMatch` and `actExtract`, for capturing all matches of a given IoT device activity signature and unveiling activity sequences of all IoT devices from the network traffic logs.
- We conduct extensive experiments using a smart home testbed and a public IoT dataset [29] to demonstrate that IoTAthena can accurately capture the activities of a wide range of heterogeneous IoT devices.

The remainder of this paper is organized as follows. We first discuss the related work in Section 2. In Section 3, we present an overview of the IoTAthena system. In Section 4, we describe how IoTAthena collects IoT network traffic, and analyzes and characterizes the background traffic of IoT devices. In Section 5, we formally define the IoT device activity signatures, and describe the process of generating the signature for each device activity. In Section 6, we present the `sigMatch` and `actExtract` algorithms with theoretical analysis. In Section 7, we present our experimental evaluation results. Section 8 concludes the paper and outlines our future work.

2. Related Work

The recent growth and deployment of IoT devices in smart homes have attracted the networking research community to study the traffic characterization and behavioral fingerprinting of IoT devices, and explore network traffic to discover IoT devices' types and activities. Most of the existing studies [11–13, 16, 17, 21, 23, 29, 34, 39, 41, 43, 47] in IoT traffic characterization and fingerprinting are interested in a wide range of traffic features from TCP/IP protocols as well as from IoT wireless communication channels. For example, [16] utilizes the wireless radio propagation patterns of IoT devices for secure authentication. [43, 47] explore the captured WiFi signals in home network for applications of localization and positioning. [39, 41], on the other hand, examine the home network traffic at flow level to model and profile IoT devices. These prior research provide critical insights for understanding traffic patterns of heterogeneous IoT devices and identifying IoT device models or types for IoT device discovery and management, IoT application performance monitoring, and vulnerability and security analysis.

In light of the recent IoT Botnets exploiting and control thousands of vulnerable IoT devices [2, 3, 9, 10, 14, 20, 24, 26, 42], some research efforts have proposed innovative methods of classifying IoT devices based on machine learning, statistical inference, or passive traffic measurement [5, 23, 35]. For example, the IoTSentinel system [23] first extracts 23 traffic features of IoT network traffic, and subsequently builds Random Forest classifiers to identify IoT device types. Similarly, IoTSense [5] fingerprints the behaviors of IoT device types with feature vectors from packet headers and payload, and builds several machine learning classifiers for effectively detecting IoT device types based on the trained behavioral fingerprinting. The research in [35] first monitors a smart IoT environment with various IoT devices for six months for extensive IoT network traffic analysis, and then builds a machine learning framework for classifying IoT device types.

As homeowners continue to deploy smart home IoT devices such as smart locks and security cameras for mission-critical applications, accurately identifying IoT device activities via supervised machine learning models [1, 27] and deterministic inference [37, 44] becomes an urgent research problem. For example, HomeSnitch [27] constructs bidirectional application data unit exchanges for representing IoT application behaviors and applies supervised machine learning classifiers to classify IoT application behaviors and identifying unknown behaviors. Similarly, Peek-a-Boo [1] demonstrates the feasibility of identifying the types, states, and IoT devices' activities via machine learning techniques from an attacker's perspective. The closest work to ours is PingPong [37], which explores the sequential and directional "ping/pong" behavioral patterns between cloud servers and IoT devices or between cloud servers and smartphones. The experiments in [37] have shown that the simple ping/pong packet-pairs with payload size and traffic directions can effectively detect many IoT devices' activities. HoMonit [44], another work of deterministically detecting IoT device activity, monitors encrypted wireless traffic of some home apps and infers smart app activities based on the deterministic finite automaton (DFA) model of smart app behavior and wireless side-channel analysis. IoTGaze [12] also builds up a system to identify IoT device activities using the sniffed wireless traffic.

Inspired and motivated by these studies on identifying IoT device types and/or activities, our proposed IoTAthena system is focused on understanding traffic signatures of IoT device activities and accurately extracting device activities from IoT network traffic. The insights from the unveiled IoT device activities have a broad range of applications such as anomaly detection, e.g., an unauthorized user is watching the video stream of the surveillance camera, IoT device malfunction detection, e.g., a smart plug shows two consecutive *on* activities, and smart home safety, e.g., the smart lock was unlocked remotely by an unauthorized user.

Note that our work is significantly different from [5, 23, 35] in the way that our objective is generating signatures for concrete IoT device activities such as *on* or *off* activities of a smart plug and unveiling these activities from network traffic, instead of identifying IoT device models or types. Different from machine learning based solutions [1, 27], IoTAthena adopts a *white-box* approach to programmatically generate activity signatures of IoT device activities consisting of ordered sequences of IP data packets with relative timestamps. IoTAthena's signature generation module is inspired by PingPong [37], but it generates a *full* signature for each IoT device activity and introduces a novel time-sensitive subsequence matching approach for unveiling IoT device activities from new IoT network traffic logs.

3. IoTAthena System Overview

Developing effective techniques to understand and report IoT device activities, *e.g., the smart lock of the home's main entrance is unlocked remotely with a smartphone app*, is crucial for ensuring the physical and property safety of these devices' homeowners. Our real-world experiments with August Lock and other IoT devices demonstrated the feasibility

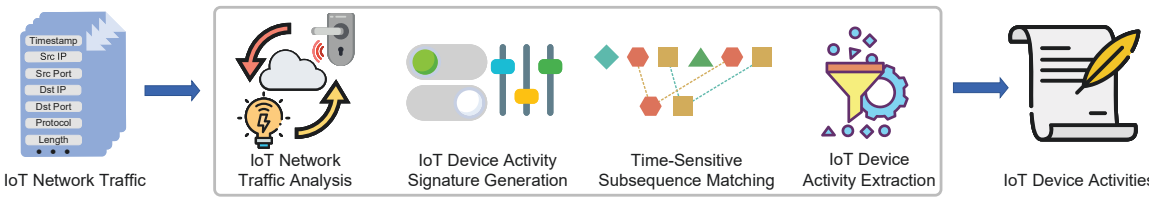


Fig. 1. Overall architecture of the IoTAthena system for unveiling IoT device activities from IoT network traffic.

of developing an automated system to learn and generate signatures of IoT device activities and use them for unveiling IoT device activities from network traffic logs. Such a system is urgently needed for understanding what is happening to IoT devices in millions of smart homes and for detecting suspicious and unauthorized behaviors towards critical home devices.

In this paper, we propose a new system, named IoTAthena, to automatically and accurately unveil IoT device activities from smart home network traffic logs. Fig. 1 illustrates the overall architecture of IoTAthena, which includes four key system modules: i) IoT network traffic analysis, ii) IoT device activity signature generation, iii) time-sensitive subsequence matching, and iv) IoT device activity extraction.

The *IoT network traffic analysis* module takes IoT network traffic during the intentionally "silent" period, and characterizes background network traffic for each IoT device. The *IoT device activity signature generation* module collects the corresponding network traffic of each IoT device activity by intentionally triggering the activity and collecting the traffic. The collected IoT network traffic along with the labelled activity logs serve as the ground truth for generating the signature of each IoT device activity consisting of an ordered sequence of IP packets with inter-packet time intervals. The *time-sensitive subsequence matching* module relies on the `sigMatch` algorithm to capture all matches of each IoT device activity signature in the network traffic log, while the *IoT device activity extraction* module relies on `actExtract` to unveil the sequence of IoT device activities from the network traffic log.

In summary, IoTAthena adopts a white-box approach to first generate signatures of IoT device activities consisting of ordered sequences of IP packets with inter-packet time interval information. Subsequently, IoTAthena applies efficient matching algorithms for deterministically unveiling the sequence of IoT device activities from the network traffic log, unlike black-box machine learning classification models [1, 5, 23, 35].

4. Network Traffic Collection and Analysis

Network traffic of IoT devices embeds rich information on device types and their behavioral patterns [29, 35]. In this section, we describe how to collect and analyze IoT network traffic in order to characterize and generate signatures of IoT device activities.

A. IoT Network Traffic Collection

Fig. 2 illustrates the data flows initiated from an IoT device or destined to an IoT device in a smart home environment. For clarity, we use two IoT devices as examples: a smart lock and a security camera. A user usually interacts with an IoT device using the device's companion app on the smartphone in the home or outside the home, e.g., in the office or on the road. The app first communicates with the cloud server which in turn generates traffic between the cloud server and the device, as illustrated by the solid red line between the smart lock and the cloud server. Sometimes, the smartphone directly communicates with the device without involving the cloud server, such as streaming request on the security camera, illustrated by the solid green line between the smart phone and the security camera.

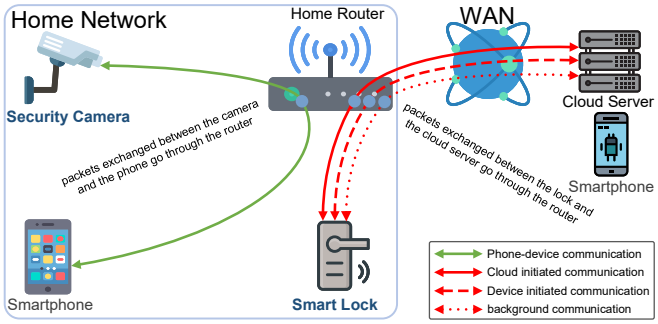


Fig. 2. Illustration of data flows initiated from or destined to IoT devices, using smart lock and security camera as examples.

The user can also manually operate the device in the traditional way, such as locking the smart lock manually. This action causes the device to update its status to the cloud server immediately following the action. In addition, the user can communicate directly with the device locally through a non-WiFi communication channel, such as Bluetooth or ultrawideband (UWB) when the user is in the vicinity of the device. This action also causes device initiated status updates. Furthermore, automatic device operations such as the smart lock's autolocking function also introduce status update traffic. These types of device initiated communications are illustrated by the dashed red line. There also exists traffic introduced by device background operations such as device firmware update checks. We use the dotted red line between the smart lock and the cloud server to illustrate these data flows.

IoTAthena collects the network traffic at the programmable home router, which enables the capture of incoming and outgoing packets of all above mentioned device-related operations. The smart home router is a desirable centralized location for data collection, considering its switching and routing function, sufficient computational and processing capacities, and the design transparency to IoT devices and apps. In our experiments,

we used Linksys WRT1900AC WiFi home routers which run the open-source Linux-based OpenWrt operating system for network traffic collection. The data processing and analysis were performed offline in this study. One of our future work is designing and implementing a prototype system on commodity home routers to evaluate the real-time feasibility of IoTAthena for unveiling IoT device activities on the fly.

B. IoT Network Traffic Analysis

The network traffic collected at the home router can be classified into two parts: the first part consists of traffic between the IoT devices and the cloud servers, while the second part consists of internal LAN traffic such as address resolution protocol (ARP) requests and simple service discovery protocol (SSDP) broadcast packets. In order to separate the logs of an individual IoT device from the mixed home network traffic, IoTAthena first identifies each device's unique IP address via the mapping of its media access control (MAC) address and host name in the dynamic host configuration protocol (DHCP) packets. It subsequently uses the device IP address as the unique *cluster* key to separate IoT network traffic into individual traffic clusters to simplify further analysis.

We carefully studied the network traffic within each individual traffic cluster of an IoT device. We can clearly observe the network traffic one would anticipate for normal IoT device activities, e.g., users issuing locking or unlocking commands for August Lock via the smartphone app. Surprisingly, we also discovered a significant amount of network traffic when there is no human-triggered or environment-triggered activity. We use the term *background traffic* to denote such network traffic, i.e., network traffic not triggered by human or environment. In order to gain a thorough understanding of IoT background traffic, we left the devices in our controlled smart home environment without any human interactions for one week and consider the network traffic cluster of each IoT device during this "silent" period as background traffic. By separating IP data packets based on the destination (and source) ports of the outgoing (and incoming) traffic, we observed that these IoT devices typically exchange messages with the remote cloud servers on the well-known application ports such as 22/TCP (SSH), 53/UDP (DNS), 80/TCP (HTTP), 123/UDP (NTP), 5353/UDP (mDNS). This observation leads us to classify IoT background traffic into three categories: management and service, signal and update, and random noise.

The management and service traffic is mainly used to manage and maintain the devices, e.g., periodical time synchronizations with NTP servers. The signal and update traffic corresponds to keep-alive signals and regular firewall update checks between IoT devices and cloud servers. The random noise traffic is mostly generated by other IoT or non-IoT devices in the local home network for a variety of reasons, e.g., ARP requests, SSDP broadcasts, and multicast DNS (mDNS) traffic from Apple Bonjour protocol for automatic device and service discovery.

The background traffic analysis not only removes unnecessary noise for characterizing and generating signatures of IoT device activities, but also sheds light on the potential vulnerabilities of the protocol stacks of mission-critical IoT devices in millions of smart homes. For example, our analysis discovered the usage of non-encrypted and insecure Telnet and HTTP sessions between some camera devices and cloud servers, for logins and firmware update checks. Discovering and mitigating security weaknesses of IoT devices is beyond the scope of this paper.

5. IoT Device Activity Signatures

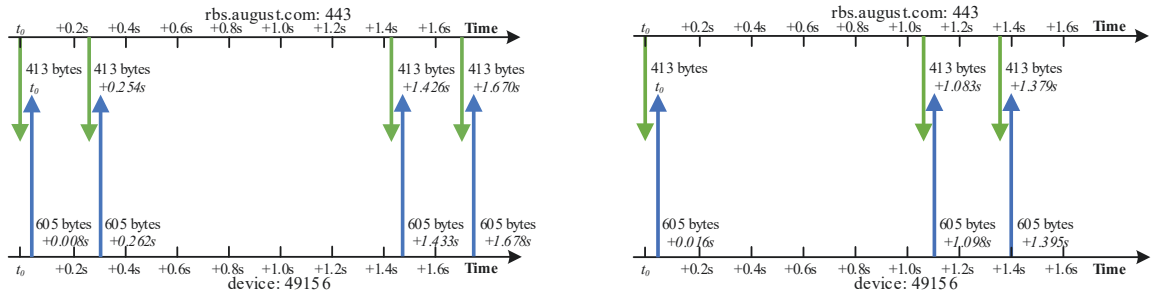
Consistent with the findings of PingPong [37], we observed repetitive network packet sequences that correspond to repeated device activities in the network traffic collected at the router of the smart home network. We also observed certain August Lock activities resulting packet sequences that are challenging for PingPong to recognize. Fig. 3 illustrates such an example. The Bluetooth (un)locking[2] activity's packet sequence (3 pairs as illustrated in Fig. 3(b)) is a subset of the WiFi (un)locking activity's packet sequence (4 pairs as illustrated in Fig. 3(a)). The clustering of re-occurring packet pairs approach in PingPong cannot distinguish WiFi (un)locking from Bluetooth (un)locking. In fact, it is difficult to distinguish these two activities in the network traffic solely based on request/reply patterns, which leads us to consider more information (the full detailed packet sequence) and inter-packet time intervals to characterize IoT device activities. The time intervals between consecutive packets provide critical information to effectively and accurately differentiate IoT device activities such as those in Fig. 3 that share overlapping packet sequences and happen very closely in time.

A. Inter-Packet Time Interval Measurement

Because IoTAthena collects network traffic at the home router, the inter-packet time interval is essentially the round-trip time (RTT) between the home router and IoT devices in the smart home plus the processing time at the device (LAN). The time interval could also be the RTT between the home router and cloud servers across the Internet plus the processing time at the cloud server (WAN).

Fig. 4 illustrates the time interval between the first and second packets (left plot), and the time interval between the second and third packets (right plot), of $1,200$ repeated *on* activities of TP-Link Plug over a 24-hour span. We observe that *the inter-packet time intervals exhibit stable and consistent patterns, with small variances*. However, *the time interval between one pair of consecutive packets may significantly differ from that between another pair of consecutive packets*. Specifically, the interval between the first and second packets has a mean (μ) of $76.73ms$ and a standard deviation (σ) of $0.003995ms$, while the interval between the second and third packets has a mean (μ) of $0.02ms$ and a standard deviation (σ)

[2]The locking activity and the unlocking activity exhibit the same packet sequences and inter-packet time intervals because of the simple lock/unlock state transitions. The encrypted application data prevents us for further differentiating these two activities with network traffic only. We generate a unique signature for each *indistinguishable activity group*. For example, we use `(un)locking` for short to denote either the `locking` activity or the `unlocking` activity. Similarly, we use `on or off` to denote either the `on` activity or the `off` activity.



(a) August Lock *WiFi (un)locking*.

(b) August Lock *Bluetooth (un)locking*.

Fig. 3. Packet sequences of August Lock's activities: the pattern in (b) *seems like* a subsequence of the pattern in (a).

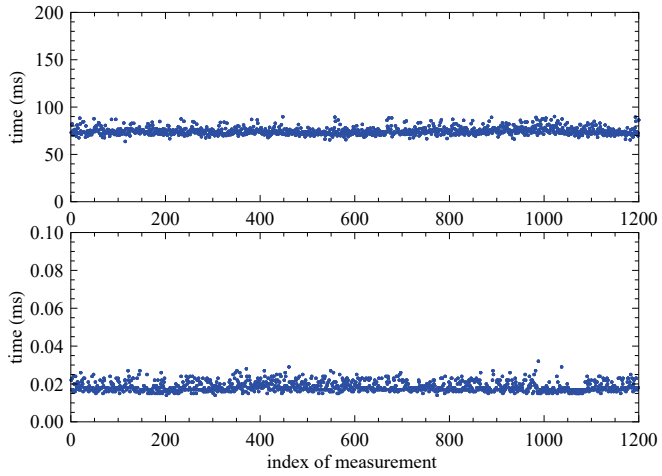


Fig. 4. Inter-packet time intervals: large values in WAN (upper plot) vs small values in LAN (lower plot).

of 0.000003ms for TP-Link Plug's *on* activity. The unstable wireless channel between the IoT devices and the home router could result in packet loss and retransmission which contribute to the fluctuation in the LAN inter-packet time intervals. The uncertain number of retransmissions in the MAC layer affects the inter-packet time intervals in our collected traffic log. However, compared with the short wireless transmission delay, the local processing time at the IoT device still dominates the LAN inter-packet time intervals, as we observe from the right plot in Fig. 4. These key observations inspire us to include inter-packet time intervals as an important component in characterizing the signatures of IoT device activities.

B. IoT Device Activity Signature Definition

A network data packet p collected at the home router is an 8-tuple, where the first through eighth fields are *timestamp*, *IoT device internal IP address*, *canonical remote cloud server name*, *remote application port*, *protocol*, *traffic direction*, *packet length*, and *application-layer data*, respectively. It is important to note that each TCP/IP data packet carries a variety of traffic features including those in the 8-tuple. However, this study only selects the features that provide additional information on identifying and differentiating IoT device activities, while skipping the features, e.g., Time to Live (TTL), sequence and acknowledgement numbers with redundant or little contributions towards device activity identification.

We use $p.t$ to denote the timestamp of packet p, and use $\widehat{p}$ to denote the 7-tuple obtained by deleting the first field (timestamp) in p. We call $\widehat{p}$ the *base packet* of packet p.

Definition 1: The signature of an IoT device activity is given by an ordered sequence of n base packets $(\widehat{q_1}, \widehat{q_2}, \ldots, \widehat{q_n})$, together with an ordered sequence of $n-1$ inter-packet time intervals $(\tau_1, \tau_2, \ldots, \tau_{n-1})$, where $\tau_j > 0$ is the time interval between the jth packet and the $j+1$th packet, $j = 1, 2, \ldots, n-1$. The number of base packets, n, in each device activity signature is determined by the observed TCP/IP data packets triggered by the activity minus the protocol-specific packets, e.g., TCP three-way handshake, and the regular heart-beat signals between the device and the remote cloud server. □

Instead of using $(\widehat{q_1}, \widehat{q_2}, \ldots, \widehat{q_n})$ and $(\tau_1, \tau_2, \ldots, \tau_{n-1})$ to represent a signature, we can equivalently represent the same signature using a sequence of n packets $(\rho_1, \rho_2, \ldots, \rho_n)$, where $\widehat{\rho_j} = \widehat{q_j}$ for $j = 1, 2, \ldots, n$ and $\rho_{j+1}.t - \rho_j.t = \tau_j$ for $j = 1, 2, \ldots, n-1$. In this representation, the time interval between the jth packet and the $j+1$th packet can be uniquely computed by $\tau_j = \rho_{j+1}.t - \rho_j.t$.

The signature of an IoT device activity is a constant, as defined in **Definition** 1. The above *alternative* representation of the signature, however, does not look like a constant *in format*. For example, for any given real number c, $(p_1, p_2, \ldots, p_n)$ and $(\rho_1, \rho_2, \ldots, \rho_n)$ denote exactly the same signature, provided that $\widehat{p_j} = \widehat{\rho_j}$, $p_j.t = \rho_j.t + c$, for $j = 1, 2, \ldots, n$. Since $(p_1, p_2, \ldots, p_n)$ and $(\rho_1, \rho_2, \ldots, \rho_n)$ define exactly the same sequence of n base packets $(\widehat{p_1}, \widehat{p_2}, \ldots, \widehat{p_n}) = (\widehat{\rho_1}, \widehat{\rho_2}, \ldots, \widehat{\rho_n})$ and exactly the same sequence of $n-1$ inter-packet time intervals $(p_2.t - p_1.t, p_3.t - p_2.t, \ldots, p_n.t - p_{n-1}.t) = (\rho_2.t - \rho_1.t, \rho_3.t - \rho_2.t, \ldots, \rho_n.t - \rho_{n-1}.t)$, *we can use this alternative representation without losing any accuracy.*

Given the above discussions, we will denote a signature of an IoT device activity using an ordered sequence of packets $(q_1, q_2, \ldots, q_n)$, where the timestamp fields are only used to compute the inter-packet time intervals $\tau_j = q_{j+1}.t - q_j.t$. For this reason, we also call the timestamp fields in a signature *relative timestamps*. We set $q_1.t$ to 0 for simplicity.

C. Automated Signature Generation

Towards automatically generating activity signatures of IoT device activities, we first follow the same practice as [45] to compile a complete list of any given IoT device's activities from the AndroidManifest.xml file of the device's companion app. We then write scripts using command-line tool and scripting feature in Android Debug Bridge (ADB) to automate the user interactions with IoT devices such as turning on/off Philips Hue and (un)locking of August Lock. For all IoT devices in our lab, we trigger each of their activities 100 times[3] in order to remove randomness and gain statistically meaningful understanding of the activity packet sequence. The time interval between two consecutive triggers of the same activity was a random number in the range $[3s, 60s]$. Here we set the minimum 3-seconds time interval between two triggers to prevent the smartphone apps from freezing, i.e., becoming unresponsive, due to rapid back-to-back activity triggerings. The network traffic captured by IoTAthena during these "active" period provides the ground truth of activity signatures of IoT device activity.

Filtering the background traffic described in Section 4-B, which happens in parallel with the device activity, leads to an ordered sequence of timestamped IP packets exchanged between IoT devices and the cloud servers. Each packet in the sequence carries a variety of traffic features such as the timestamp of each packet, local IP address and port number of the IoT device, remote IP address and port number of the cloud server, protocol, packet length, and the actual application payload of IoT applications which are mostly encrypted for security and privacy reasons. For each packet, we continue to remove features with random and dynamic values due to the protocol designs, e.g., the random local port number at IoT devices in TCP connections with cloud servers and TCP sequence and acknowledge numbers. In addition, we transform certain traffic features to retain the stable values, e.g., converting dynamic IP addresses of load-balanced cloud servers to the canonical remote cloud server names.

The inter-packet time interval τ_j between the jth packet and the $j+1$th packet in the signature is set to the mean (over the 100 tries) of the inter-packet time intervals. To simplify notations, we set $q_1.t$ to 0, and set $q_{j+1}.t = q_j.t + \tau_j$, $j = 1, 2, \ldots, n$.

TABLE 1 illustrates the signatures for various activities of the August Lock[4]. Note that we have used the alternative representation of signatures. The signature shown in Fig. 3(a) corresponds to the lower-right box in TABLE 1. The signature shown in Fig. 3(b) corresponds to the lower-left box in TABLE 1. While the sequence of base packets in the signature for *Bluetooth (un)locking* (illustrated in Fig. 3(b)) is a subset of the sequence of base packets in the signature for *WiFi (un)locking* (illustrated in Fig. 3(a)), the additional information embedded in the inter-packet time intervals makes it possible to distinguish these two activities. With the aid of additional information on inter-packet time intervals, we can distinguish these two activities from network traffic logs, which are difficult to distinguish using the base packets only.

[3]This could be replaced by any reasonably large number.

[4]In our experiments we noticed firmware updates of IoT devices might cause slight changes on the activity signatures.

6. Algorithms for Unveiling IoT Device Activities from Network Traffic

Having discussed network traffic in Section 4 and device activity signatures in Section 5, we are now ready to present our algorithms for unveiling IoT device activities from network traffic logs. In Section 6-A, we formally define the IoT activity signature matching problem and the IoT activity extraction problem. In Section 6-B, we present the `sigMatch` algorithm for identifying all matches of a given signature in the network traffic log. In Section 6-C, we present the `actExtract` algorithm for unveiling the sequence of activities of an IoT device from the network traffic log. In Section 6-D, we discuss the limitations and extensions of our algorithms.

A. Problem Formulation

As discussed in Section 4, an IoT *network traffic log* (denoted by $\mathbb{L}$) is an ordered sequence of packets $(p_1, p_2, \ldots, p_m)$ with increasing timestamps (i.e., $p_{i'}.t < p_{i''}.t$ for $i' < i''$). As discussed in Section 5, a *signature* of an IoT device activity (denoted by $\mathbb{S}$) is an ordered sequence of packets $(q_1, q_2, \ldots, q_n)$ with increasing relative timestamps (i.e., $q_{j'}.t < q_{j''}.t$ for $j' < j''$). Recall that $\widehat{p}$ and $\widehat{q}$ denote the 7-tuple obtained by deleting the timestamp in p and the relative timestamp in q, respectively. A *signature set* of an IoT device (denoted by $\mathbb{SS}$) is a set of distinct signatures $\{\mathbb{S}^1, \mathbb{S}^2, \ldots, \mathbb{S}^K\}$, one signature per activity of the device. For ease of presentation, we will use *activity* and *signature* interchangeably in the rest of the paper.

In light of the end-to-end network latency variations on the Internet [8, 15, 28], we allow an inter-packet time interval tolerance $\epsilon_j > 0$ as the "safety margin" for the measurement of $q_{j+1}.t - q_j.t$ when trying to find a match of a signature in the network log.

Let j satisfy $1 < j \leq n$ and $\delta > 0$ be a given tolerance. Let i' and i'' satisfy $1 \leq i' < i'' \leq m$. We say that $(p_{i'}, p_{i''})$ is a δ-valid match of (q_{j-1}, q_j), if

1) $\widehat{p_{i'}} = \widehat{q_{j-1}}$, $\widehat{p_{i''}} = \widehat{q_j}$;
2) $|(p_{i''}.t - p_{i'}.t) - (q_j.t - q_{j-1}.t)| \leq \delta$.

Let $\mathbb{S} = (q_1, q_2, \ldots, q_n)$ be a signature. Let $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_{n-1})$ be the matching tolerance vector, where ϵ_j is the tolerance for the matching of (q_j, q_{j+1}). Let $(l[1], l[2], \ldots, l[n])$ be an increasing sequence of integers indicating the index of the location of a packet in the network log. We say that $(p_{l[1]}, p_{l[2]}, \ldots, p_{l[n]})$ is an ϵ-valid match of signature $\mathbb{S}$ in log $\mathbb{L}$, if $(p_{l[j]}, p_{l[j+1]})$ is an ϵ_j-valid match of (q_j, q_{j+1}), for $j = 1, 2, \ldots, n-1$.

We study the following two related problems:

IoT activity signature matching: Given network traffic log $\mathbb{L}$, signature $\mathbb{S}$, and tolerance vector ϵ for $\mathbb{S}$, identify all ϵ-valid matches of signature $\mathbb{S}$ in log $\mathbb{L}$.

IoT device activity extraction: Given network traffic log $\mathbb{L}$ and signature set $\mathbb{SS}$, find a sequence of IoT activities $\mathbb{A}_1, \mathbb{A}_2, \ldots$, whose execution leads to the network traffic log $\mathbb{L}$.

TABLE 1
ACTIVITY SIGNATURES OF AUGUST LOCK ACTIVITIES.

Activity	Signature
Opening Mobile App (12 packets)	0.000s lock:49157 → rbs.august.com:443 637B
	0.132s lock:49157 → rbs.august.com:443 221B
	0.204s rbs.august.com:443 → lock:49157 237B
	0.209s lock:49157 → rbs.august.com:443 637B
	0.327s lock:49157 → rbs.august.com:443 237B
	0.526s rbs.august.com:443 → lock:49157 237B
	0.602s lock:49157 → rbs.august.com:443 637B
	0.723s lock:49157 → rbs.august.com:443 237B
	0.823s rbs.august.com:443 → lock:49157 237B
	1.116s lock:49157 → rbs.august.com:443 637B
	1.205s lock:49157 → rbs.august.com:443 221B
	1.251s rbs.august.com:443 → lock:49157 237B
Autolocking (15 packets)	0.000s lock:49157 → rbs.august.com:443 637B
	0.086s lock:49157 → rbs.august.com:443 221B
	0.129s rbs.august.com:443 → lock:49157 237B
	0.240s lock:49157 → rbs.august.com:443 637B
	0.329s lock:49157 → rbs.august.com:443 221B
	0.373s rbs.august.com:443 → lock:49157 237B
	0.990s lock:49157 → rbs.august.com:443 637B
	1.084s lock:49157 → rbs.august.com:443 221B
	1.127s rbs.august.com:443 → lock:49157 237B
	1.277s lock:49157 → rbs.august.com:443 637B
	1.366s lock:49157 → rbs.august.com:443 221B
	1.410s rbs.august.com:443 → lock:49157 237B
	1.549s lock:49157 → rbs.august.com:443 637B
	1.640s lock:49157 → rbs.august.com:443 221B
	1.679s rbs.august.com:443 → lock:49157 237B
Bluetooth (Un)Locking	0.000s rbs.august.com:443 → lock:49156 413B
	0.016s lock:49156 → rbs.august.com:443 605B
	1.083s rbs.august.com:443 → lock:49156 413B
	1.098s lock:49156 → rbs.august.com:443 605B
	1.379s rbs.august.com:443 → lock:49156 413B
	1.395s lock:49156 → rbs.august.com:443 605B

Activity	Signature
Manual (Un)Locking (21 packets)	0.000s lock:49157 → rbs.august.com:443 637B
	0.088s lock:49157 → rbs.august.com:443 205B
	0.134s rbs.august.com:443 → lock:49157 237B
	0.441s lock:49157 → rbs.august.com:443 637B
	0.526s lock:49157 → rbs.august.com:443 221B
	0.571s rbs.august.com:443 → lock:49157 237B
	0.581s lock:49157 → rbs.august.com:443 637B
	0.666s lock:49157 → rbs.august.com:443 237B
	0.712s rbs.august.com:443 → lock:49157 237B
	0.870s lock:49157 → rbs.august.com:443 637B
	0.954s lock:49157 → rbs.august.com:443 237B
	1.001s rbs.august.com:443 → lock:49157 237B
	1.078s lock:49157 → rbs.august.com:443 637B
	1.169s lock:49157 → rbs.august.com:443 221B
	1.214s rbs.august.com:443 → lock:49157 237B
	1.321s lock:49157 → rbs.august.com:443 637B
	1.410s lock:49157 → rbs.august.com:443 221B
	1.473s rbs.august.com:443 → lock:49157 237B
	1.559s lock:49157 → rbs.august.com:443 637B
	1.659s lock:49157 → rbs.august.com:443 221B
	1.707s rbs.august.com:443 → lock:49157 237B
WiFi (Un)Locking (8 packets)	0.000s rbs.august.com:443 → lock:49156 413B
	0.008s lock:49156 → rbs.august.com:443 605B
	0.254s rbs.august.com:443 → lock:49156 413B
	0.262s lock:49156 → rbs.august.com:443 605B
	1.426s rbs.august.com:443 → lock:49156 413B
	1.433s lock:49156 → rbs.august.com:443 605B
	1.670s rbs.august.com:443 → lock:49156 413B
	1.678s lock:49156 → rbs.august.com:443 605B

B. Signature Matching via Time Sensitive Subsequence Matching

The IoT activity signature matching problem is different from the traditional subsequence matching problem [22] and the longest common subsequence problem [4, 6] due to the inter-packet time interval constraint. The matching problem with such constraints cannot be solved via the simple adjustment of existing algorithms. We solve the signature matching problem using a time-sensitive subsequence matching approach, called `sigMatch`, as presented in Algorithm 1.

For a given network traffic log $\mathbb{L} = (p_1, p_2, \ldots, p_m)$ and signature $\mathbb{S} = (q_1, q_2, \ldots, q_n)$, together with a inter-packet time interval tolerance vector ϵ, we compute a DAG $G_{\mathbb{LS}} = (V_{\mathbb{LS}}, E_{\mathbb{LS}})$ that captures all ϵ-valid matches of signature $\mathbb{S}$ in log $\mathbb{L}$. The vertex set $V_{\mathbb{LS}}$ contains vertices in the form of $v_{i,j}$, where p_i is a *potential match* of q_j. The edge set $E_{\mathbb{LS}}$ contains directed edges in the form of $(v_{i,j}, v_{k,j-1})$, where (p_k, p_i) is an ϵ_{j-1}-valid match of (q_{j-1}, q_j) for $1 \le k \le i-1$, and there is a directed path from vertex $v_{i,j}$ to a vertex $v_{i',1} \in V_{\mathbb{LS}}$ (for some $i' \le i - j + 1$).

If $\widehat{p_i} \neq \widehat{q_j}$, vertex $v_{i,j}$ *does not* exist. If $\widehat{p_i} = \widehat{q_j}$, vertex $v_{i,j}$ *may* exist. Each edge has the form $(v_{i,j}, v_{k,j-1})$ for some $k < i$. Hence we have $|V_{\mathbb{LS}}| \le mn$ and $|E_{\mathbb{LS}}| \le \frac{m(m-1)(n-1)}{2}$.

In Line 1 of Algorithm 1, both $V_{\mathbb{LS}}$ and $E_{\mathbb{LS}}$ are initialized to $\emptyset$. The algorithm then populates the vertex set and the edge set while looping over the packets $p_1, p_2, \ldots, p_m$. For each i, the algorithm loops over the packets $q_1, q_2, \ldots, q_n$. When $\widehat{p_i} = \widehat{q_1}$, $v_{i,1}$ is a vertex in the DAG. For $j = 2, 3, \ldots, n$, $v_{i,j}$ is a vertex if and only if $\widehat{p_i} = \widehat{q_j}$ **and** (p_k, p_i) is an ϵ_{j-1}-valid match of (q_{j-1}, q_j) for some $k < i$. In this case, $(v_{i,j}, v_{k,j-1})$ is an edge in the DAG.

We use Fig. 5 to illustrate a running example of `sigMatch`. The goal is to identify all ϵ-valid matches of signature

Algorithm 1: sigMatch$(\mathbb{L}, \mathbb{S}, \epsilon)$

Input: Network traffic log $\mathbb{L} = (p_1, p_2, \ldots, p_m)$, Signature $\mathbb{S} = (q_1, q_2, \ldots, q_n)$, tolerance vector $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_{n-1})$.

Output: A DAG $G_{\mathbb{LS}} = (V_{\mathbb{LS}}, E_{\mathbb{LS}})$ that captures all ϵ-valid matches of signature $\mathbb{S}$ in $\mathbb{L}$.

1 $V_{\mathbb{LS}} \leftarrow \emptyset$; $E_{\mathbb{LS}} \leftarrow \emptyset$;
2 **for** $i := 1$ **to** m **do**
3 **if** $\widehat{p_i} == \widehat{q_1}$ **then**
4 $V_{\mathbb{LS}} \leftarrow V_{\mathbb{LS}} \cup \{v_{i,1}\}$;
5 **for** $j := 2$ **to** n **do**
6 **for** $k := 1$ **to** $i - 1$ **do**
7 **if** $v_{k,j-1} \in V_{\mathbb{LS}}$ **and** (p_k, p_i) *is an* ϵ_{j-1}*-valid match of* (q_{j-1}, q_j) **then**
8 $V_{\mathbb{LS}} \leftarrow V_{\mathbb{LS}} \cup \{v_{i,j}\}$;
9 $E_{\mathbb{LS}} \leftarrow E_{\mathbb{LS}} \cup \{(v_{i,j}, v_{k,j-1})\}$;
10 **output** DAG $G_{\mathbb{LS}}$.

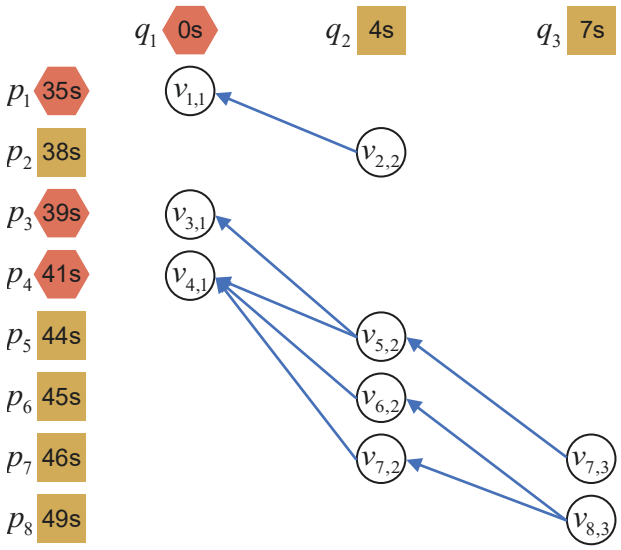


Fig. 5. Running example of `sigMatch`: row index corresponds to the traffic log, column index corresponds to the signature.

$\mathbb{S} = (q_1, q_2, q_3)$ in log $\mathbb{L} = (p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8)$ with tolerance vector $\epsilon = (1, 1)$. In this example, we have $\widehat{p_1} = \widehat{p_3} = \widehat{p_4} = \widehat{q_1}$, denoted by the *hexagon* shape; we also have $\widehat{p_2} = \widehat{p_5} = \widehat{p_6} = \widehat{p_7} = \widehat{p_8} = \widehat{q_2} = \widehat{p_3}$, denoted by the *square* shape. The timestamps (for traffic log) and relative timestamps (for signature) are inside the corresponding shape.

We start from p_1. Since $\widehat{p_1} = \widehat{q_1}$, vertex $v_{1,1}$ is added to $V_{\mathbb{LS}}$; Then we move to p_2. Since $\widehat{p_2} = \widehat{q_2}$, $v_{1,1} \in V_{LS}$, and $|(p_2.t - p_1.t) - (q_2.t - q_1.t)| = |(38-35)-(4-0)| \leq 1$, vertex $v_{2,2}$ is added to $V_{\mathbb{LS}}$ and directed edge $(v_{2,2}, v_{1,1})$ is added to $E_{\mathbb{LS}}$. Similarly, vertices $v_{3,1}$ and $v_{4,1}$ are added to $V_{\mathbb{LS}}$. Next, we pay attention to the row corresponding to p_5. We found $\widehat{p_5} = \widehat{q_2}$. For $k = 1$, we found $v_{1,1} \in V_{\mathbb{LS}}$, but the time interval does not match. For $k = 3$, we found $v_{3,1} \in V_{\mathbb{LS}}$, and the time interval matches. Hence vertex $v_{5,2}$ is added to $V_{\mathbb{LS}}$, and edge $(v_{5,2}, v_{3,1})$ is added to $E_{\mathbb{LS}}$. For $k = 4$, we found $v_{4,1} \in V_{\mathbb{LS}}$, and the time interval matches. At this moment, vertex $v_{5,2}$ is already in $V_{\mathbb{LS}}$, and edge $(v_{5,2}, v_{4,1})$ is added to $E_{\mathbb{LS}}$. We obtain the DAG as shown in Fig. 5 by continuing the above process.

Theorem 1: Algorithm `sigMatch` has a worst-case time complexity of $O(m^2 n)$, where n is the number of packets in the signature $\mathbb{S}$, and m is the number of packets in the network traffic log $\mathbb{L}$. Furthermore,

(a) If $(p_{l[1]}, p_{l[2]}, \ldots, p_{l[n]})$ is an ϵ-valid match of $\mathbb{S}$ in $\mathbb{L}$, then $(v_{l[n],n}, v_{l[n-1],n-1}, \ldots, v_{l[1],1})$ is a directed path in $G_{\mathbb{LS}}$, and $l[1] < l[2] < \cdots < l[n]$.

(b) If $(v_{l[n],n}, v_{l[n-1],n-1}, \ldots, v_{l[1],1})$ is a directed path in $G_{\mathbb{LS}}$, then $(p_{l[1]}, p_{l[2]}, \ldots, p_{l[n]})$ is an ϵ-valid match of $\mathbb{S}$ in $\mathbb{L}$, and $l[1] < l[2] < \cdots < l[n]$.

Proof. The loop over i runs m times. The loop over j runs n times. The loop over k runs $i-1$ times, for each i. This leads to the worst-case time complexity of $O(m^2 n)$.

From the condition in Line 7 of the algorithm, we notice that *there is an edge in the form* $(v_{i,j}, v_{k,j-1})$ **if and only if** *there is an* $\epsilon_{[j]}$*-valid match of* $(q_1, q_2, \ldots, q_j)$ *in* $\mathbb{L}$ *that matches* (q_{j-1}, q_j) *to* (p_k, p_i), where $\epsilon_{[j]} = (\epsilon_1, \epsilon_2, \ldots, \epsilon_{j-1})$. This leads to claims (a) and (b). □

We point out that the total number of ϵ-valid matches of signature $\mathbb{S}$ in $\mathbb{L}$ may be exponential. However, all of them are captured by a polynomial sized DAG $G_{\mathbb{LS}}$, which can be computed in polynomial time.

C. Unveiling IoT Activities from Network Traffic Log

We investigate how to unveil the activities of an IoT device using `sigMatch` in Algorithm 1 as a building block. Note that we can separate the traffic of a specific IoT device from all network traffic using the IoT device's distinct IP address. For a given IoT device, we first extract its signature set $\mathbb{SS} = \{\mathbb{S}^1, \mathbb{S}^2, \ldots, \mathbb{S}^K\}$. For each signature $\mathbb{S}^k$, using its corresponding tolerance vector ϵ^k, we can apply `sigMatch` to construct the corresponding DAG $G_{\mathbb{LS}^k}$ in $O(m^2 n_k)$ worst-case time, where n_k is the number of packets in $\mathbb{S}^k$. We can compute all K DAGs in $O(Km^2 n_{\max})$ worst-case time, where $n_{\max} = \max\{n_1, n_2, \ldots, n_K\}$.

For each $k = 1, 2, \ldots, K$, there may be zero or more ϵ^k-valid matches of signature $\mathbb{S}^k$. Making use of $G_{\mathbb{LS}^k}$, we can either confirm that there is no ϵ^k-valid match (when there is no vertex v_{i,n_k} in $V_{\mathbb{LS}^k}$) or compute *the earliest* ϵ^k-valid match $(p_{l[1]}, p_{l[2]}, \ldots, p_{l[n_k]})$, in the sense that $(p_{l[1]}, p_{l[2]}, \ldots, p_{l[n_k]})$ is *lexicographically smallest*, in $O(m + n_k)$ worst-case time.

Given the network traffic $\mathbb{L}$, and the valid matches of signatures in $\mathbb{SS}$, how do we decide which IoT activity happened first? Through extensive experiments, we found that *in normal situations, each network packet corresponding to an earlier IoT activity proceeds every network packet corresponding to a later IoT activity*. Therefore the signature that has the earliest match happens first. Once this decision is made, we can delete each packet with a timestamp no later than that of the last packet in the match of the found signature from the network traffic. Repeating the above process, we can unveil the sequence of IoT activities from the given network traffic. We formally describe this process called `actExtract` in Algorithm 2.

Algorithm 2: actExtract$(\mathbb{L}, \mathbb{SS}, \varepsilon)$

Input: Network traffic $\mathbb{L} = (p_1, p_2, \ldots, p_m)$, signature set $\mathbb{SS} = \{\mathbb{S}^1, \mathbb{S}^2, \ldots, \mathbb{S}^K\}$, $\varepsilon = (\epsilon^1, \epsilon^2, \ldots, \epsilon^K)$ where ϵ^k is the match tolerance vector for $\mathbb{S}^k$.

Output: A sequence of IoT activities $\mathbb{A}_1, \mathbb{A}_2, \ldots$.

1 **for** $k := 1$ **to** K **do**
2 $G_{\mathbb{S}^k} \leftarrow sigMatch(\mathbb{L}, \mathbb{S}^k, \epsilon^k)$;
3 **while** *some signature* $\mathbb{S}^k$ *has a match in* $G_{\mathbb{S}^k}$ **do**
4 Let $\mathbb{S}^{k'}$ have the earliest match;
5 **output** Activity corresponding to $\mathbb{S}^{k'}$;
6 Remove all packets in $\mathbb{L}$ with timestamp no later than that of the last matched packet for $\mathbb{S}^{k'}$.
7 **for** $k := 1$ **to** K **do**
8 $G_{\mathbb{S}^k} \leftarrow sigMatch(\mathbb{L}, \mathbb{S}^k, \epsilon^k)$;

Theorem 2: The worst-case time complexity of Algorithm 2 is $O(Km^3 n_{\max})$, where K is the number of signatures, $n_{\max}$ is the maximum number of packets in any of the signatures, and m is the number of packets in network traffic log $\mathbb{L}$. In normal situations (i.e., each packet for an earlier activity precedes every network packet of a later activity), `actExtract` correctly outputs a sequence of IoT activities $\mathbb{A}_1, \mathbb{A}_2, \ldots$ whose sequential execution will generate a network traffic log that may be different from $\mathbb{L}$ only in the timestamp fields.

Proof. Initially, the K DAGs can be computed in $O(Km^2 n_{\max})$ time. The earliest match of $\mathbb{S}^k$ can be computed in $O(m + n_k)$ time, $\forall k$. Selecting the signature with the earliest match requires $O(Kn_{\max})$ time. This process is repeated for no more than m times, hence the time complexity.

Next, we prove the correctness of the algorithm. Assuming that the sequence of IoT device activities that generated the network traffic $\mathbb{L}$ is $\mathbb{A}_1, \mathbb{A}_2, \ldots, \mathbb{A}_x$. By our *normal* assumption, each network packet of $\mathbb{A}_1$ must happen earlier than every network packet of $\mathbb{A}_\lambda$, for any $\lambda > 1$. Since `actExtract` uses the earliest match, it will output $\mathbb{A}_1$ as the first activity, and all of the packets in the computed match for $\mathbb{A}_1$ have timestamps earlier than the timestamp of any packet in other IoT activity $\mathbb{A}_\lambda$, with $\lambda > 1$. Hence, when we delete the packets matched for $\mathbb{A}_1$, we delete all of the packets generated for $\mathbb{A}_1$, but none of the packets generated by $\mathbb{A}_\lambda$ with $\lambda > 1$. Therefore `actExtract` will next output $\mathbb{A}_2$, then $\mathbb{A}_3$, and so on. This proves the correctness of the algorithm. □

When we execute the computed sequence of IoT device activities, the network traffic observed may be different from $\mathbb{L}$, but only in the timestamp field. The sequence of packets will have increasing timestamps. Ignoring the timestamp field, two sequences of packets will be identical with $\mathbb{L}$. Note that we can divide the network traffic log into multiple sublogs where each sublog corresponds to a unique IoT device. We can apply `actExtract` to each sublog in parallel to *unveil the IoT activities for all IoT devices*.

D. Discussions

Our proposed `actExtract` algorithm can unveil the activity sequence of an IoT device with no ambiguity and guarantee correctness, assuming there is no ongoing attack and the device can only carry out one activity at a time, which is true for most devices. For devices that allow two or more concurrent activities, such as IP cameras, we can modify `sigMatch` algorithm to record only non-overlapping matches in network traffic for a signature in a DAG. We can then build the DAG for the same device's signatures independently and output all the identified activities. In case when there is attacking traffic, it is possible that one packet is matched to two different signatures. We can add a variable in `sigMatch` to record all the signatures that a packet is matched to. If such a conflict happens, our algorithm can report it as an anomaly and raise an alarm.

7. Experimental Evaluations

We evaluate the performance of IoTAthena using two different settings: 1) our own smart home testbed, and 2) a large public IoT network traffic dataset [29]. We first describe these settings in Section 7-A. In Section 7-B, we present experimental results on the sensitivity of IoTAthena's accuracy on the matching tolerance. In Section 7-C, we present experimental results for homogeneous device activities. In Section 7-D, we present experimental results for mixed device activities, together with a case study.

A. Experiment Setting

Our smart home testbed has 16 widely-used IoT devices, including multiple models of *IP cameras, smart bulbs, smart doorbells, smart locks, and smart plugs*. These IoT devices are all ranked as popular by Smart Home DB [33]. Our experiments have identified a total of 44 different device activities by using these 16 devices. The numbers of devices and activities in this study are comparable to existing studies on understanding IoT device activities in smart home network environments [1, 37]. This controlled smart home environment was used to create the "silent" week for collecting, analyzing, and characterizing background network traffic, as discussed in Section 4. For each IoT device activity, we repeatedly generate the activity while collecting the associated network traffic as well as recording the activity logs which are used for establishing the ground truth at the same time. Using the signature extraction technique introduced in Section 5, we were able to extract signatures for most of the device activities of all 16 representative IoT devices except the *stream off* activity of Amcrest ProHD camera, which does not have the deterministic traffic pattern to form a signature.

In addition to network traffic and activity logs collected from our own smart home testbed, we also evaluated IoTAthena's performance using a large public IoT network traffic dataset [29], known as the MON(IOT)R dataset. The MON(IOT)R dataset includes raw IP data traffic and the labeled activity logs of 25 IoT devices[5]. Among these devices, 6 of them are also included in our smart home testbed, while the other 19 devices are unique to the dataset. The IoT network traffic and labeled activities in the dataset allow us to evaluate the performance of IoTAthena.

B. Sensitivity Analysis on the Tolerance Parameter

Our time-sensitive subsequence matching algorithm `sigMatch` uses the tolerance vector $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_{n-1})$ for accommodating inter-packet time intervals' variations. In our experiment, ϵ_j is set to $r \times \sigma_j$ for $j \in [1, n-1]$, where σ_j is the standard deviation of the inter-packet time interval between two consecutive packets q_j and q_{j+1} and $r \geq 1$ is a tunable parameter.

The accuracy of IoTAthena depends on the matching tolerance parameter. Intuitively, when the matching tolerance is very small, IoTAthena tends to unveil fewer activities due to the strict checking of inter-packet time intervals, leading to low accuracy. On the other hand, when the matching tolerance is very large, IoTAthena tends to have more false negatives due to

[5]We evaluated IoTAthena on the IoT device activities with at least 30 samples in the MON(IOT)R dataset in order to have statistically meaningful results.

the loose checking of inter-packet time intervals, also leading to low accuracy. In order to have a deeper understanding of this dependency, we carried out sensitivity analysis. For each device activity, we repeatedly triggered it 120 times with random delays between two consecutive experiments. We then ran 6-fold cross validation using the data collected. In each of the 6 rounds, we observe the accuracy of IoTAthena on 20 of the experiments, while using the remaining 100 to generate the signature. Fig. 6 illustrates some representative results.

Figs. 6(a)-(d) show the accuracy changes for unveiling the *on* and *off* activities of TP-Link Bulb and TP-Link Plug as r increases from 1 to 30, while Figs. 6(e)-(h) show the accuracy dynamics of unveiling August Lock's app opening, Wifi (un)locking, autolocking, and Bluetooth (un)locking activities. In Figs. 6(a)-(d), we observe that the accuracy of IoTAthena exhibits a non-decreasing trend for the *on* and *off* activities of both TP-Link Bulb and TP-Link Plug, when r increases from 1 to 30. These observations are not surprising since the increasing value of r leads to a higher tolerance value to allow larger inter-packet time intervals.

However, Figs. 6(e)-(g) contradict such conjectures as increasing r to a particular value leads to decreasing accuracy in matching August Lock's app opening, WiFi (un)locking, and autolocking activities. Our in-depth investigation discovered that the accuracy decrease for the larger r values is due to the interference of August Lock's background traffic noise. The background traffic happens to shares overlapping packets with the signatures of app opening, WiFi (un)locking, and autolocking activities when we allow bigger tolerance of inter-packet intervals. Unlike Figs. 6(e)-(g), the accuracy of unveiling August Lock's Bluetooth (un)locking activities changes in a non-decreasing fashion in Fig. 6(h) as r increases from 1 to 30. The underlying reason of this distinct observation is the unique traffic patterns of the network traffic collected when triggering August Lock's Bluetooth (un)locking activities, where no background noise that cannot be filtered out has been observed.

In summary, our sensitivity analysis on the tolerance parameter confirmed the importance and impact of choosing appropriate tolerance values during the IoT device activity extraction process. More importantly, the observations in Fig. 6 highlight the rationale and necessity of our *full* packet sequences with inter-packet time intervals as the IoT device activity signature and our time-sensitive subsequence matching algorithm for unveiling IoT device activities.

C. Performance of IoTAthena on Homogeneous Device Activities

Having done the sensitivity analysis, we focused on evaluating IoTAthena's performance of unveiling homogeneous device activities. We first present experimental results on our smart home testbed. We then present results on the MON(IOT)R dataset [29].

For each IoT device activity, we repeated it 120 times and collected the network traffic on the router in our smart home testbed. We again ran the 6-fold cross validation. TABLE 2 shows the accuracy ($\mathcal{A}$), precision ($\mathcal{P}$), and recall ($\mathcal{R}$) measures of IoTAthena for various activities of the 16 devices in our smart home testbed, with r set to 3, 11, and 23, respectively.

From TABLE 2, we observe that the performance of IoTAthena depends on r. For $r = 3$, IoTAthena achieves a minimum accuracy of 0.78, a minimum precision of 0.98, and a minimum recall of 0.78. When r is increased to 11, the performance of IoTAthena improves, with precision of 1.00, accuracy of 0.99 or better, and recall of 0.99 or better, across all activities in our experiments. When r is further increased to 23, the performance of IoTAthena drops, with a minimum accuracy of 0.75, a minimum precision of 0.81, and a minimum recall of 0.88. Based on this empirical evidence, we choose $r = 11$ as the "optimal" value for the tolerance parameter.

We also evaluated the performance of IoTAthena using the MON(IOT)R dataset [29]. Due to the relatively small sample size (between 30 and 40), we ran 4-fold cross validation instead of 6-fold cross validation. TABLE 3 illustrates the accuracy, precision, and recall measures of running IoTAthena against 25 IoT devices in the MON(IOT)R dataset with r set to 11. We observe that IoTAthena achieve a minimum accuracy of 0.95, a minimum precision of 0.98, and a minimum recall of 0.95.

The prior study [37] also evaluates the algorithm with the same the MON(IOT)R dataset [29]. TABLE VI in [37] reports an average accuracy of 99.12% on 3 IoT device activities for WAN Sniffer, and an average accuracy of 99.06% on 4 IoT device activities for WiFi sniffer. As shown in TABLE 3, our approach achieves an average accuracy of 99.57% on 33 IoT device activities on the same dataset. Therefore, our proposed IoTAthena system is able to generate signatures for more IoT device activities than [37] while achieving slightly better accuracy in identifying the signatures with the same public dataset.

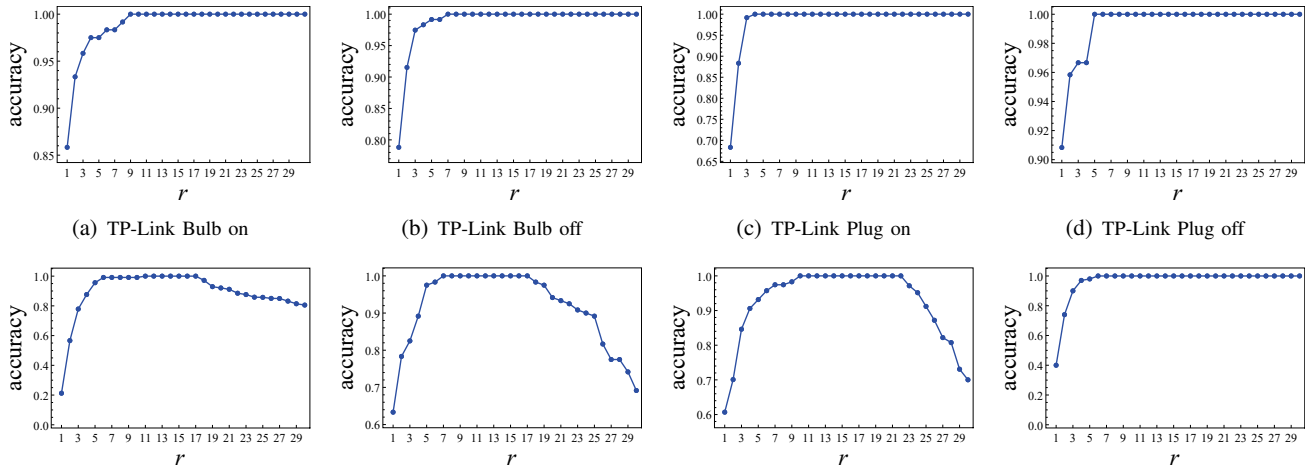
In summary, experimental evaluations with our smart home testbed and the MON(IOT)R dataset demonstrate that IoTAthena can successfully unveil homogeneous IoT device activities from network traffic logs.

D. Performance of IoTAthena on Mixed Device Activities

A significant benefit of our IoTAthena system lies in the realtime security monitoring of IoT devices in smart homes, which has become an increasingly important research topic. Given the IoT network traffic logs, IoTAthena can accurately unveil the sequence of IoT device activities over time and potentially detect anomalous traffic patterns and behaviors towards IoT devices.

As a case study, we applied IoTAthena to unveil the activities of 5 IoT devices in our smart home during a 24-hour span. The 5 devices in this case study consist of Arlo Ultra Camera, August Lock, Ring Doorbell, TP-Link Bulb, and TP-Link Plug. Fig. 7 visualizes the time-series activities of these 5 IoT devices discovered by IoTAthena during a 24-hour time span in our smart home environment.

The two activities highlighted by the light blue box near the left end of Fig. 7 capture two consecutive user-triggered events at around 12:45pm: i) (the homeowner) unlocked the



(a) TP-Link Bulb on (b) TP-Link Bulb off (c) TP-Link Plug on (d) TP-Link Plug off

(e) August Lock app opening (f) August Lock WiFi (un)locking (g) August Lock autolocking (h) August Lock BLE (un)locking

Fig. 1Fig. 6. The impact of r in the inter-packet time interval tolerance parameter on the accuracy of IoT activity signature matching.

TABLE 2
ACCURACY, PRECISION, AND RECALL OF IOTATHENA FOR 16 DEVICES WITH r SET AS 3, 11, 23 RESPECTIVELY.

Type	Device Name	Activity	$r=3$			$r=11$			$r=23$		
			$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$
bulb	Philips Hue	on or off	0.98	1.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00
		brightness	0.95	1.00	0.95	1.00	1.00	1.00	1.00	1.00	1.00
	Sengled SmartLED	on	0.88	1.00	0.88	1.00	1.00	1.00	1.00	1.00	1.00
		off	0.89	1.00	0.89	1.00	1.00	1.00	1.00	1.00	1.00
		brightness	0.92	1.00	0.92	1.00	1.00	1.00	1.00	1.00	1.00
	TP-Link Bulb	on	0.96	1.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00
		off	0.97	1.00	0.97	1.00	1.00	1.00	1.00	1.00	1.00
		color	0.99	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00
		brightness	0.95	1.00	0.95	1.00	1.00	1.00	1.00	1.00	1.00
camera	Amcrest ProHD	stream on	0.86	1.00	0.86	1.00	1.00	1.00	1.00	1.00	1.00
	Arlo - Q Indoor	stream on	0.96	1.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00
		stream off	0.94	1.00	0.94	1.00	1.00	1.00	1.00	1.00	1.00
		motion detection	0.98	1.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00
	Arlo Ultra	stream on	0.88	1.00	0.88	1.00	1.00	1.00	1.00	1.00	1.00
		stream off	0.92	1.00	0.92	1.00	1.00	1.00	1.00	1.00	1.00
		motion detection	0.95	1.00	0.95	1.00	1.00	1.00	1.00	1.00	1.00
	Blink XT2	stream on	0.92	1.00	0.92	0.99	1.00	0.99	0.99	1.00	0.99
		stream off	0.95	1.00	0.95	1.00	1.00	1.00	1.00	1.00	1.00
		motion detection	0.96	1.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00
	Reolink Camera	stream on	0.98	1.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00
		stream off	0.96	1.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00
		motion detection	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
doorbell	August Doorbell Cam Pro	stream on	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		stream off	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		ringing	0.94	1.00	0.94	1.00	1.00	1.00	1.00	1.00	1.00
		motion detection	0.95	1.00	0.95	1.00	1.00	1.00	1.00	1.00	1.00
	Ring VideoDoorbell	stream on	0.98	1.00	0.98	1.00	1.00	1.00	1.00	1.00	1.00
		stream off	0.96	1.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00
		ringing	0.96	1.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00
		motion detection	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
lock	August Lock Pro	app opening	0.78	1.00	0.78	1.00	1.00	1.00	0.77	0.86	0.88
		WiFi (un)locking	0.80	0.98	0.82	1.00	1.00	1.00	0.75	0.81	0.91
		Bluetooth (un)locking	0.90	1.00	0.90	1.00	1.00	1.00	0.99	0.99	1.00
		autolocking	0.85	1.00	0.85	1.00	1.00	1.00	0.97	1.00	0.97
		manual (un)locking	0.93	1.00	0.93	1.00	1.00	1.00	1.00	1.00	1.00
	Schlage WiFi Deadbolt	WiFi (un)locking	0.89	1.00	0.89	1.00	1.00	1.00	1.00	1.00	1.00
		autolocking	0.92	1.00	0.92	1.00	1.00	1.00	1.00	1.00	1.00
		manual (un)locking	0.90	1.00	0.90	1.00	1.00	1.00	1.00	1.00	1.00
plug	Amazon Smart Plug	on	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
		off	0.96	1.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00
	Gosund WiFi Smart Socket	on or off	0.98	1.00	0.98	1.00	1.00	1.00	0.99	0.99	1.00
	TP-Link Plug	on	0.99	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00
		off	0.97	1.00	0.97	1.00	1.00	1.00	1.00	1.00	1.00
	WeMo Plug	on or off	0.98	1.00	0.98	1.00	1.00	1.00	0.99	0.99	1.00

TABLE 3
ACCURACY, PRECISION, AND RECALL OF IOTATHENA IN THE EXTERNAL MON(IOT)R DATASET WITH r SET AS 11.

Device Name	Activity	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$
Amcrest Camera Wired	watch	1.00	1.00	1.00
Blink Camera	watch	1.00	1.00	1.00
Blink Security Hub	watch or photo	1.00	1.00	1.00
Bulb1	on or off	1.00	1.00	1.00
Fire TV	menu	0.95	1.00	0.95
Google Home Mini	volume or voice	1.00	1.00	1.00
Insteon Hub	on or off	1.00	1.00	1.00
Invoke	volume or voice	1.00	1.00	1.00
Lefun Camera Wired	watch or photo	1.00	1.00	1.00
LG TV Wired	menu	1.00	1.00	1.00
Lightify Hub	on or off	1.00	1.00	1.00
	color	1.00	1.00	1.00
Luohe Spycam	watch	1.00	1.00	1.00
Magichome Strip	on	0.98	0.98	1.00
	off	1.00	1.00	1.00
Microseven Camera	watch	1.00	1.00	1.00
Philips Bulb	on or off	0.97	1.00	0.97
Samsungtv Wired	menu	1.00	1.00	1.00
Sengled Hub	on or off	0.98	1.00	0.98
Smartthings Hub	on or off	1.00	1.00	1.00
T-philips Hub	on or off	1.00	1.00	1.00
TP Link Bulb	on	1.00	1.00	1.00
	off	1.00	1.00	1.00
	color	1.00	1.00	1.00
	dim	1.00	1.00	1.00
TP Link Plug	on	1.00	1.00	1.00
	off	1.00	1.00	1.00
Wink Hub2	on	1.00	1.00	1.00
	off	1.00	1.00	1.00
Xiaomi Hub	on or off	0.98	0.98	1.00
Xiaomi Strip	on or off	1.00	1.00	1.00
Zmodo Doorbell	watch	1.00	1.00	1.00

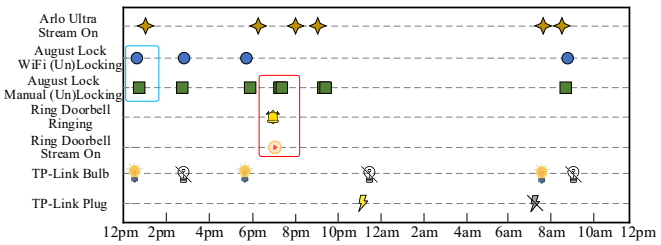


Fig. 7. IoT device activities discovered by IoTAthena in the small home environment during a 24-hour span.

August Lock with app (from outside), indicated by the dark blue disk inside the light blue box, and ii) manually locked the August Lock (after entering home), indicated by the green square inside the light blue box. Similarly, the four activities in highlighted by the red box at around 7:20pm in Fig. 7 reflect four consecutive events: i) (a visitor) pressed the button on the ring doorbell, which generated a push notification to the homeowner's smartphone; ii) (the homeowner) watched the video streaming feed on the ring doorbell to check the visitor's identity; iii) (the homeowner) manually unlocked the August Lock to let the visitor in; iv) the August Lock was manually locked (from inside).

To evaluate IoTAthena's ability in unveiling sequences of mixed IoT device activities, we used IoTAthena to unveil the mixed IoT device activities of all 16 devices in our smart home from the network traffic, in a 24-hour span, from 12:00pm to 11:59am. Fig. 8 illustrates IoTAthena's performance by comparing the ground truth activity sequences with the unveiled activity sequences of 16 IoT devices in the smart home testbed, where a blue dot represents a successful match, while a red cross represents a failed match. The actual dates for running different IoT device activity experiments might vary, so the x-axis only denotes the time of the day from 12:00pm to 11:59am. As can be seen from the figure, IoTAthena correctly unveiled all but one of the activities. The only missed activity occurs with the Blink XT2 Camera. Our root cause analysis revealed that IoTAthena missed one streaming activity due to the unseen variation in packet length.

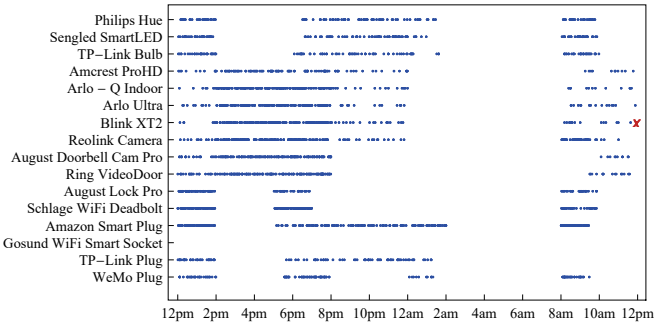


Fig. 8. Activity sequence extraction results of 16 IoT devices in the smart home environment. The actual dates for running different IoT device activity experiments might vary, so the x-axis only denotes the time of the day from 12:00pm to 11:59am.

In summary, our experimental evaluations based on a variety of heterogeneous IoT devices demonstrated that IoTAthena can effectively and accurately unveil individual IoT device activities as well as unveil IoT device activity sequences over time. We note that our single smart home environment in the experiments has its own limitation in performing large scale experimental evaluations. One of our future work is to deploy the IoTAthena system in a large number of smart homes to evaluate its performance and overhead.

8. CONCLUSIONS AND FUTURE WORK

This paper introduces *IoTAthena* to effectively and accurately unveil IoT device activities from network traffic in smart homes. We first recognize and generate activity signatures of IoT device activities consisting of ordered sequences of IP data packets by repeated and controlled experiments. Subsequently, we design two polynomial time algorithms, `sigMatch` and `actExtract`. The `sigMatch` algorithm captures all matches of any given IoT device activity signature from real network traffic logs. The `actExtract` algorithm unveils the full activity sequences of all IoT devices from the network traffic log. Through experimental evaluations based on a wide range of heterogeneous IoT devices from a real smart home environment and a public IoT dataset, we demonstrated that IoTAthena is able to accurately unveil IoT device activities from raw network traffic logs. We are in the process of designing and implementing a prototype system on commodity home routers to evaluate the real-time feasibility of IoTAthena for unveiling IoT device activities on the fly. Another possible future work is to explore the benefits of IoTAthena in detecting and mitigating security threats towards vulnerable IoT devices.

REFERENCES

[1] A. Acar, H. Fereidooni, T. Abera, A.-K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-Z. Sadeghi, and A.-S. Uluagac, "Peek-a-boo: I see your smart home activities, even encrypted!" in *Proc. of ACM WiSec*, 2020.

[2] O. Alrawi, C. Lever, M. Antonakakis, and F. Monrose, "SoK: Security evaluation of home-based IoT deployments," in *Proc. of IEEE S&P*, 2019.

[3] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J.-A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai botnet," in *Proc. of USENIX Security*, 2017.

[4] A. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms," in *Proc. of IEEE SPIRE*, 2000.

[5] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral fingerprinting of IoT devices," in *Proc. of ACM ASHES*, 2018.

[6] T.-H. Cormen, C.-E. Leiserson, R.-L. Rivest, and C. Stein, "Introduction to algorithms," MIT Press, 2009.

[7] S. Demetriou, N. Zhang, Y. Lee, X. Wang, C. Gunter, X. Zhou, and M. Grace, "HanGuard: SDN-driven protection of smart home WiFi devices from malicious mobile apps," in *Proc. of ACM WiSec*, 2017.

[8] A. Dhamdhere, D. Clark, A. Gamero-Garrido, M. Luckie, R. Mok, G. Akiwate, K. Gogia, V. Bajpai, A. Snoeren, and K. Claffy, "Inferring persistent interdomain congestion," in *Proc. of ACM SIGCOMM*, 2018.

[9] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *Proc. of IEEE S&P*, 2016.

[10] E. Fernandes, A. Rahmati, K. Eykholt, and A. Prakash, "Internet of things security research: A rehash of old ideas or new intellectual challenges?" *IEEE Security & Privacy*, vol. 15, pp. 79–84, 2017.

[11] K. Gao, C. Corbett, and R. Beyah, "A passive approach to wireless device fingerprinting," in *Proc. of IEEE/IFIP DSN*, 2010.

[12] T. Gu, Z. Fang, A. Abhishek, H. Fu, P. Hu, and P. Mohapatra, "IoTGaze: IoT security enforcement via wireless context analysis," in *Proc. of IEEE INFOCOM*, 2020.

[13] T. Gu and P. Mohapatra, "BF-IoT: Securing the IoT networks via fingerprinting-based device authentication," in *Proc. of IEEE MASS*, 2018.

[14] S. Herwig, H. Harvey, G. Hughey, R. Roberts, and D. Levin, "Measurement and analysis of Hajime, a peer-to-peer IoT botnet," in *Proc. of NDSS*, 2019.

[15] T. Høiland-Jørgensen, B. Ahlgren, P. Hurtig, and A. Brunstrom, "Measuring latency variation in the Internet," in *Proc. of ACM CoNEX*, 2016.

[16] Y. Huang, W. Wang, H. Wang, T. Jiang, and Q. Zhang, "Authenticating on-body IoT devices: An adversarial learning approach," *IEEE Transactions on Wireless Communications (TWC)*, vol. 19, pp. 5234 – 5245, 2020.

[17] H. Jafari, O. Omotere, D. Adesina, H.-H. Wu, and L. Qian, "IoT devices fingerprinting using deep learning," in *Proc. of IEEE MILCOM*, 2018.

[18] Y. Jia, Y. Xiao, J. Yu, X. Cheng, Z. Liang, and Z. Wan., "A novel graph-based mechanism for identifying traffic vulnerabilities in smart home IoT," in *Proc. of IEEE INFOCOM*, 2018.

[19] G. Kambourakis, C. Kolias, and A. Stavrou, "The Mirai botnet and the IoT zombie armies," in *Proc. of IEEE MILCOM*, 2017.

[20] D. Kumar, K. Shen, B. Case, D. Garg, G. Alperovich, D. Kuznetsov, R. Gupta, and Z. Durumeric, "All things considered: An analysis of IoT devices on home networks," in *Proc. of USENIX Security*, 2019.

[21] X. Ma, J. Qu, J. Li, J. Lui, Z. Li, and X. Guan, "Pinpointing hidden IoT devices via spatial-temporal traffic fingerprinting," in *Proc. of IEEE INFOCOM*, 2020.

[22] D. Maier, "The complexity of some problems on subsequences and supersequences," *Journal of the ACM (JACM)*, vol. 25, pp. 322–336, 1978.

[23] 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 *Proc. of IEEE ICDCS*, 2017.

[24] P. Morgner, C. Mai, N. Koschate-Fischer, F. Freiling, and Z. Benenson, "Security update labels: Establishing economic incentives for security patching of IoT consumer products," in *Proc. of IEEE S&P*, 2020.

[25] P. Morgner, S. Mattejat, Z. Benenson, C. Muller, and F. Armknecht, "Insecure to the touch: Attacking ZigBee 3.0 via touchlink commissioning," in *Proc. of ACM WiSec*, 2017.

[26] A. Mosenia and M. Jha, "A comprehensive study of security of internet-of-things," *IEEE Transactions on Emerging Topics in Computing (TETC)*, vol. 5, pp. 586 – 602, 2016.

[27] T. J. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.-R. Sadeghi, "HomeSnitch: behavior transparency and control for smart home IoT devices," in *Proc. of ACM WiSec*, 2019.

[28] V. Paxson and M. Allman, "RFC2988: Computing TCP's retransmission timer," *Internet Engineering Task Force (IETF) Request for Comments*, November, 2000.

[29] J. Ren, D.-J. Dubois, D. Choffnes, A.-M. Mandalari, R. Kolcun, and H. Haddadi, "Information exposure from consumer IoT devices: A multidimensional, network-informed measurement approach," in *ACM IMC'2019*.

[30] E. Ronen, C. O'Flynn, A. Shamir, and A.-O. Weingarten, "IoT goes nuclear: creating a ZigBee chain reaction," in *Proc. of IEEE S&P*, 2017.

[31] E. Ronen and A. Shamir, "Extended functionality attacks on IoT devices: The case of smart lights," in *Proc. of IEEE EuroS&P*, 2016.

[32] M. Ryan, "Bluetooth smart: The good, the bad, the ugly, and the fix," https://lacklustre.net/bluetooth/bluetooth_smart_good_bad_ugly_fix-mikeryan-blackhat_2013.pdf, 2013.

[33] SmartHomeDB, "Smart home DB - the smart home database," https://www.smarthomedb.com/.

[34] S. Siby, R. Maiti, and N. Tippenhauer, "IoTScanner: Detecting privacy threats in IoT neighborhoods," in *Proc. of ACM IoTPTS*, 2017.

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

[36] V. Sivaraman, D. Chan, D. Earl, and R. Boreli, "Smart-phones attacking smart-homes," in *Proc. of ACM WiSec*, 2016.

[37] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "PingPong: Packet-level signatures for smart home device events," in *Proc. of NDSS*, 2019.

[38] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proc. of IEEE ICDE*, 2002.

[39] Y. Wan, K. Xu, F. Wang, and G. Xue "Characterizing and mining traffic patterns of IoT devices in edge networks," *IEEE Transactions on Network Science and Engineering (TNSE)*, vol. 8, pp. 89 – 101, 2020.

[40] Y. Wan, K. Xu, G. Xue, and F. Wang, "IoTArgos: A multi-layer security monitoring system for internet-of-things in smart homes," in *Proc. of IEEE INFOCOM*, 2020.

[41] K. Xu, Y. Wan, G. Xue, and F. Wang, "Multidimensional behavioral profiling of internet-of-things in edge networks," in *Proc. of IEEE/ACM IWQoS*, 2019.

[42] L. Yu, B. Luo, J. Ma, Z. Zhou, and Q. Liu, "You are what you broadcast: Identification of mobile and IoT devices from

(public) WiFi," in *Proc. of USENIX Security*, 2020.
[43] S. Zhang, W. Wang, S. Tang, S. Jin, and T. Jiang, "Robot-assisted backscatter localization for IoT applications," *IEEE Transactions on Wireless Communications (TWC)*, vol. 19, pp. 5807 – 5818, 2020.
[44] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "HoMonit: Monitoring smart home apps from encrypted traffic," in *Proc. of ACM CCS*, 2018.
[45] H. Zhu, Y. Li, R. Li, J. Li, Z.-H. You, and H. Song, "Sedmdroid: An enhanced stacking ensemble of deep learning framework for Android malware detection," *IEEE Transactions on Network Science and Engineering (TNSE)*, May 2020. URL https://ieeexplore.ieee.org/document/9099045/.
[46] T. Zillner and S. Strobl, "ZigBee exploited: The good, the bad and the ugly," in *Proc. of the DeepSec Conferences In Depth Security*, 2015.
[47] H. Zou, M. Jin, H. Jiang, L. Xie, and C.-J. Spanos, "WinIPS: WiFi-based non-intrusive indoor positioning system with online radio map construction and adaptation," *IEEE Transactions on Wireless Communications (TWC)*, vol. 16, pp. 8118 – 8130, 2017.

Yinxin Wan (Student Member 2020) received his B.E degree in Information Security from University of Science and Technology of China in 2018. He is currently a Ph.D. student of Computer Science at Arizona State University. His research interests include cyber security, the Internet of Things, and data-driven networked system.

Kuai Xu (Member 2008, Senior Member 2015) is an Associate Professor at Arizona State University. He received B.S. and M.S. degrees in Computer Science from Peking University, China in 1998 and 2001, and received Ph.D. degree in Computer Science from the University of Minnesota in 2006. His research interests include network security, Internet measurement, big data, data mining, and machine learning. He is a member of ACM and a senior member of IEEE.

Feng Wang (Member 2007) received the B.S. degree from Wuhan University in 1996, M.S. degree from Peking University in 1999, and the Ph.D. degree from University of Minnesota, Twin Cities in 2005, all in computer science. She is currently a Professor with School of Mathematical and Natural Sciences, Arizona State University. Her research interests focus on network science, social media analysis, network optimization, network security, and wireless sensor networks. She has published over 60 journal and conference papers and book chapters. Her research is supported by National Science Foundation.

Guoliang Xue (Member 1996, Senior Member 1999, Fellow, 2011) is a professor of Computer Science and Engineering at Arizona State University. He received a Ph.D degree in Computer Science from the University of Minnesota in 1991, an MS degree in Operations Research (in 1984) and a BS degree in Mathematics (in 1981) from Qufu Normal University. His research interests span the areas of wireless networking, network security and privacy, and IoT. He has received the IEEE Communications Society William R. Bennett Prize in 2019. He served as a TPC Chair of IEEE INFOCOM'2010 and IEEE Globecom'2020, and a General Chair of IEEE CNS'2014 and IEEE/ACM IWQoS'2019. He served on the editorial board of IEEE/ACM Transactions on Networking and the Area Editor of IEEE Transactions on Wireless Communications, overseeing 12 editors in the Wireless Networking area. He was the Vice President for Conferences of the IEEE Communications Society from January 2016 to December 2017. He is an IEEE Fellow, and the Steering Committee Chair of IEEE INFOCOM.