IoTMosaic: Inferring User Activities from IoT Network Traffic in Smart Homes

Yinxin Wan, Kuai Xu, Feng Wang, Guoliang Xue

***Abstract*—Recent advances in cyber-physical systems, artificial intelligence, and cloud computing have driven the wide deployment of Internet-of-things (IoT) in smart homes. As IoT devices often directly interact with the users and environments, this paper studies if and how we could explore the collective insights from multiple heterogeneous IoT devices to infer user activities for home safety monitoring and assisted living. Specifically, we develop a new system, namely IoTMosaic, to first profile diverse user activities with distinct IoT device event sequences, which are extracted from smart home network traffic based on their TCP/IP data packet signatures. Given the challenges of missing and out-of-order IoT device events due to device malfunctions or varying network and system latencies, IoTMosaic further develops simple yet effective approximate matching algorithms to identify user activities from real-world IoT network traffic. Our experimental results on thousands of user activities in the smart home environment over two months show that our proposed algorithms can infer different user activities from IoT network traffic in smart homes with the overall accuracy, precision, and recall of 0.99, 0.99, and 1.00, respectively.**

I. Introduction

Recent advances in cyber-physical systems, artificial intelligence, and cloud computing have driven the rapid growth and deployment of IoT devices in smart homes. Although there is a rich literature in studying traffic patterns [17, 34], security and privacy challenges [8, 9, 11, 14, 15, 20, 39], and device events and functions [6, 7, 10, 24, 31, 33, 38] of *individual* IoT devices, little effort has been devoted to exploring IoT network traffic for inferring user activities. The accurate knowledge and awareness of user activities in smart homes is crucial for home safety monitoring and assisted living, e.g., a motion sensor's `motion detection` event at the front door followed by a smart lock's `open` event indicating a person entering the home.

Towards filling this research gap, this paper introduces a new system, namely IoTMosaic, for inferring user activities from IoT network traffic in the smart home environment. IoTMosaic first recognizes and generates the signatures of user activities by characterizing each user activity with an ordered sequence of IoT device events via controlled experiments in smart homes. Based on the signatures of user activities, we could search and match them from IoT device event streams extracted from IoT network traffic for user activity inference. However, the sequence of IoT devices events for a given user activity sometimes is disrupted by missing or out-of-order device events due to device malfunctions, long end-to-end network latencies between IoT devices and cloud servers, and extended wake-up delays from the sleep mode of battery-powered devices.

Given the existence of missing or out-of-order device events, our paper formulates a new research problem of inferring user activities with approximate user activity signature matching. To address this problem, we develop simple yet effective approximate matching algorithms which can identify the exact or approximate matches with a varying number of missing device events. In addition, we devise a heuristic trimming strategy to resolve the conflicts caused by multiple matches of different user activities which share overlapping device events in their signatures.

To systematically evaluate the performance of our proposed IoTMosaic system, we set up a real-world smart home environment consisting of heterogeneous IoT devices with various functions in a two-bedroom apartment, and identify 21 user activities that are common and crucial to home safety and security. Our extensive experiments on the dataset which includes thousands of user activities collected during a two-month long period have shown that IoTMosaic is able to infer all of these 21 user activities from the IoT network traffic with high accuracy, precision, and recall.

The contributions of this paper are summarized as follows:

- We design and implement IoTMosaic to first profile diverse user activities in smart homes and identify the unique IoT device event sequences as their signatures. Subsequently, we develop deterministic approximate signature matching algorithms for inferring user activities from the device event sequences.
- We propose the approximate signature matching algorithms with rigorous theoretical analysis to effectively infer user activities with the flexibility of allowing missing or out-of-order IoT device events due to device malfunctions or varying network and system latencies.
- Our two-month long experiments based on a real-world smart home have demonstrated that our proposed algorithms can infer different user activities from IoT network traffic with the overall accuracy, precision, and recall of 0.99, 0.99, and 1.00, respectively.

The remainder of this paper is organized as follows. Section II introduces the architecture of our proposed IoTMosaic system. Section III describes the strategy of profiling and generating signatures for user activities. Section IV presents the approximate matching algorithms for user activity infer-

All authors are 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 2007469. The information reported here does not reflect the position or the policy of the funding agency.

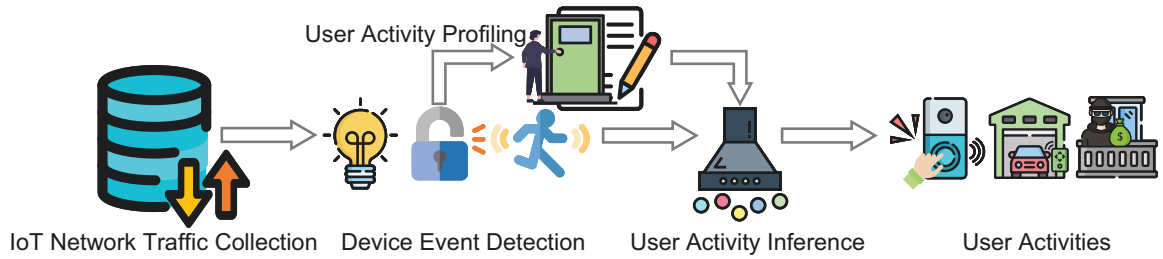

Fig. 1. System architecture of IoTMosaic for inferring user activity from IoT network traffic in smart homes

ence. Section V presents the performance evaluations of our proposed systems. Section VI discusses the related work, and Section VII concludes this paper.

II. System Overview and Architecture

A. Towards A User Activity Inference System

With the prevalence and popularity of IoT devices, a wide range of innovative services and applications such as home automation, remote healthcare, and voice assistants are available for homeowners. These dedicated IoT devices mostly work independently for specific functions, e.g., a smart camera starts recording once sensing sound or motions, and a smart lock is opened or closed remotely via a companion smartphone app.

Many real-world user activities trigger a sequence of temporally and spatially *correlated* events involving multiple IoT devices. The individual device event information of each of these IoT devices often lacks sufficient evidence to infer the user activities and tell homeowners what happened in their homes. For example, a delivery personnel dropping off a package on the front door and ringing the Ring Doorbell could trigger the `motion detection` event and the doorbell's `ringing` event. However, exploring correlated events and collective insights from heterogeneous IoT devices in the same home could reveal many important user activities. Such automated user activity monitoring and inference system could have many practical benefits in home safety and security, assisted living, and remote healthcare.

Towards this end, this paper proposes the IoTMosaic system for automatically and algorithmically inferring user activities based on the underlying network traffic of IoT devices in smart homes. Rather than inferring user activities directly from the IoT network traffic, IoTMosaic first detects IoT device events by analyzing network traffic in smart homes by adopting one of the existing solutions [1, 31, 33]. Based on these extracted IoT device events, IoTMosaic generates the signatures of diverse user activities which consist of IoT device event sequences. IoTMosaic then infers user activities using approximate matching algorithms to accommodate missing or out-of-order IoT device events due to device malfunctions or varying latencies.

B. System Architecture and Components

Fig. 1 illustrates the IoTMosaic's overall system architecture for inferring user activity from IoT network traffic in smart homes. IoTMosaic consists of four main components: i) IoT network traffic collection, ii) IoT device event detection, iii) user activity profiling, and vi) user activity inference.

The first key system component, *IoT network traffic collection*, leverages programmable home routers to continuously collect, process, and analyze outgoing and incoming timestamped TCP/IP data packets of smart home IoT devices. For the second component, *IoT device event detection*, our paper adopts one of the state-of-the-art solutions [1, 31, 33] for extracting IoT device events from network traffic collected by the home routers in the first component.

The primary focus of this paper is to design and implement the last two system components in Fig. 1, i.e., *user activity profiling* and *user activity inference*. For learning and profiling user activities, we first collect IoT network traffic in smart homes while repeatedly running and labeling each user activity as ground truth. Subsequently, we extract the sequence of IoT device events as the signature for each user activity. The *user activity profiling* component is responsible for recognizing and generating the signatures of different user activities which are the input of the next *user activity inference* component.

Towards developing the *user activity inference* system component, we propose simple yet effective algorithms for identifying user activities from IoT device events with the tolerance of missing events. Our proposed approximate matching algorithms effectively infer user activities with varying missing device events. In case of multiple matches to different user activities which share overlapping IoT device events, we devise a heuristic trimming step to strategically remove some inferred activities based on the number of missing device events and the dependency among the signatures of user activities.

To evaluate the performance of our proposed system, we set up an experimental smart home environment in a two-bedroom apartment with heterogeneous IoT devices. We systematically evaluate our system with different user activities, mostly related to home safety and security, in the smart home testbed. Our extensive experiments on the labeled user activities and network traffic data collected span over two months show that IoTMosaic is able to accurately infer diverse user activities in smart homes.

III. User Activities Profiling

A. Detecting IoT Device Events with IoT Network Traffic

The network traffic of IoT devices plays a crucial role in classifying the IoT device types, e.g., `LG smart TV`, and

detecting IoT device events, e.g., switching Philips Hue smart lighting `on` or `off`. These individual and distinguished events generated by IoT devices are referred to as IoT device events. Several recent parallel studies [1, 6, 31, 33] have proposed practical solutions to generate the traffic signatures of IoT device events and extract IoT device events via matching such signatures from the IoT network traffic.

In this study, we adopt the approach in [33] to first generate the *unique* signature of each IoT device event with diverse traffic features including the crucial inter-packet time interval information in IP packets. Then we use the deterministic algorithms developed in [33] to extract IoT device events in our smart home environments. Table I lists the smart home IoT devices deployed in this study and their respective device events for learning and inferring a wide range of user activities related to home safety and security, e.g., a person with smart lock app access entering the home from the front door.

TABLE I
SMART HOME IOT DEVICES DEPLOYED IN THIS STUDY AND THEIR RESPECTIVE DEVICE EVENTS

Device Name	Device Event	Abbreviation
Arlo Q Camera (AQ)	motion detection	AQ_{mot}
August Lock (AL)	WiFi (un)locking	AL_{wlk}
	manual (Un)locking	AL_{mlk}
	auto locking	AL_{alk}
D-Link Water Sensor (DW)	water detected	DW_{wtr}
	water not detected	DW_{nwtr}
Kangaroo Motion Sensor (KM)	motion detection	KM_{mot}
Reolink Camera (RC)	motion detection	RC_{mot}
	stream on / off	RC_{on} / RC_{off}
Ring Doorbell (RD)	motion detection	RD_{mot}
	stream on / off	RD_{on} / RD_{off}
	ringing	RD_{ring}
Ring Spotlight (RS)	motion detection	RS_{mot}
	motion light on / off	RS_{on} / RS_{off}
Smart Life Contact Sensor (SC)	open / close	SC_{open} / SC_{close}
Tessan Contact Sensor (TC)	open / close	TC_{open} / TC_{close}
TP-Link Bulb (TB)	on / off	TB_{on} / TB_{off}
TP-Link Plug (TP)	on / off	TP_{on} / TP_{off}

As IoT devices continue to be deployed in smart home applications such as smart locks and surveillance cameras, the knowledge and awareness of IoT device events have become increasingly important for understanding the statuses of IoT devices and detecting anomalous behaviors and attacks towards them. More importantly, multiple IoT device events, happening very closely in time and space, could *collectively* provide valuable knowledge for inferring user activities in smart homes. Inspired by this critical insight, IoTMosaic explores IoT device events for understanding, profiling, and inferring user activities in smart homes.

B. Profiling User Activities with IoT Device Event Sequences

Recognizing and tracking user activities and behaviors in smart environments have been a long-standing research problem [25] due to its importance in assisted living, remote healthcare, and home safety. In this study, we consider a user activity as the interaction between a person and the smart home environment, e.g., an e-commerce delivery personnel pushing the smart doorbell and leaving a package on the porch. The direct interaction between users and IoT devices, e.g., a user opening or closing the smart lock via the companion smartphone App, is only considered as IoT device events (or actions) instead of user activities. In other words, a user activity, consisting of several human actions, could trigger one or more IoT device events, determined by the availability and deployment of IoT devices as well as the layout of the smart home.

Our real-world experiments with heterogeneous IoT devices deployed in the smart home testbed have discovered that many user activities trigger an ordered sequence of device events from several adjacent IoT devices. For example, a person with the physical key entering our smart home environment from the front door triggers a series of IoT device events related to August smart lock, Tessan Contact Sensor, Ring doorbell, Ring Spotlight, and Alro Q Camera. It should be noted that a single IoT device event is often unable to infer the underlying user activities independently. However, combining and correlating the time and space of multiple events from adjacent IoT devices that are deployed at the nearby locations where user activities happen potentially provide sufficient information for extracting these user activities.

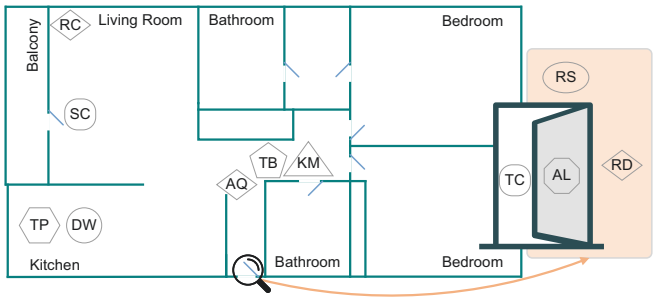

Fig. 2. Layout of IoT device deployment in a real-world smart home. Each IoT device is represented with its abbreviation name, cf. Table I

Fig. 2 illustrates the layout of our smart home experimental environment in a two-bedroom apartment. Given the list of IoT devices in Table I and the deployment of devices in Fig. 2, we use repeated and controlled experiments to discover and generate distinct signatures of user activities with the sequences of IoT device events. It is important to point out that the smart home environment setup and user activity experiments have received official approval from our university institutional review board (IRB).

Table II summarizes the list of 21 user activities in our smart home testbed which are common and essential to home safety and security. For each user activity, we repeatedly trigger it 20 times, while simultaneously collecting TCP/IP packets using the programmable home router. To only include IoT device events that are actually triggered by the user activity, we run all of the experiments at this stage in a controlled environment and ensure no other interfering IoT device events or user activities happen at the same time. To avoid the mutual inference of user activities, we separate the experiments of two consecutive user activities for at least 10 minutes.

We first extract all device events from the IoT network traffic during the experiment period where user activity U

TABLE II
USER ACTIVITIES AND THEIR SIGNATURES OF IOT DEVICE EVENT SEQUENCES

No.	User Activity	Device Event Sequence
1	A person without key entering the home from the front door (day)	$RD_{mot}, RS_{mot}, RD_{ring}, RD_{on}, RD_{off}, AL_{mlk}, TC_{open}, TC_{close}, AL_{mlk}, AQ_{mot}$
2	A person without key entering the home from the front door (night)	$RD_{mot}, RS_{on}, RD_{ring}, RD_{on}, RD_{off}, AL_{mlk}, TC_{open}, TC_{close}, AL_{mlk}, AQ_{mot}, RS_{off}$
3	A person with app access entering the home from the front door (day)	$RD_{mot}, RS_{mot}, AL_{wlk}, TC_{open}, TC_{close}, AL_{alk}, AQ_{mot}$
4	A person with app access entering the home from the front door (night)	$RD_{mot}, RS_{on}, AL_{wlk}, TC_{open}, TC_{close}, AL_{alk}, AQ_{mot}, RS_{off}$
5	A person with key entering the home from thefront door (day)	$RD_{mot}, RS_{mot}, AL_{mlk}, TC_{open}, TC_{close}, AL_{alk}, AQ_{mot}$
6	A person with key entering the home from thefront door (night)	$RD_{mot}, RS_{on}, AL_{mlk}, TC_{open}, TC_{close}, AL_{alk}, AQ_{mot}, RS_{off}$
7	A person ring the doorbell and leave (day)	$RD_{mot}, RS_{mot}, RD_{ring}$
8	A person ring the doorbell and leave (night)	$RD_{mot}, RS_{on}, RD_{ring}, RS_{off}$
9	A person checking the front door of the home (day)	RD_{mot}, RS_{mot}
10	A person checking the front door of the home (night)	$RD_{mot}, RS_{on}, , RS_{off}$
11	A person with key leaving the home from the front door (lock the door) (day)	$AQ_{mot}, AL_{mlk}, TC_{open}, TC_{close}, RD_{mot}, RS_{mot}, AL_{mlk}$
12	A person with key leaving the home from the front door (lock the door) (night)	$AQ_{mot}, AL_{mlk}, TC_{open}, TC_{close}, RD_{mot}, RS_{on}, AL_{mlk}, RS_{off}$
13	A person with key leaving the home from the front door (do not lock the door) (day)	$AQ_{mot}, AL_{mlk}, TC_{open}, TC_{close}, RD_{mot}, RS_{mot}$
14	A person with key leaving the home from the front door (do not lock the door) (night)	$AQ_{mot}, AL_{mlk}, TC_{open}, TC_{close}, RD_{mot}, RS_{on}, RS_{off}$
15	A person appearing in the hallway of the home	KM_{mot}, TB_{on}
16	A person leaving the hallway of the home	KM_{mot}, TB_{off}
17	A person checking the living room's camera streaming	RC_{on}, RC_{off}
18	A person entering the balcony from living room or entering the living from balcony (contact sensor alarm off)	RC_{mot}
19	An person entering the home from the balcony (contact sensor alarm on)	SC_{open}, RC_{mot}
20	An person leaving the home to enter the balcony ((contact sensor alarm on)	RC_{mot}, SC_{close}
21	A person checking water leakage	$DW_{wtr}, TP_{off}, DW_{nwtr}, TP_{on}$

is triggered. For each IoT device event e, we compare its timestamp, i.e., the timestamp of the first packet captured by the smart home router which is matched to e, with the manually-recorded timestamp of the user activity U. Given the existence of clock synchronizations, varying end-to-end network latencies, and automatic events, e.g., August smart lock automatically closes the door if the door remains open for 30 seconds after an `unlock` event, we consider the device event e is actually generated by the user activity U if and only if $|e.t - U.t| \leq \Omega$. In our experiments, we choose Ω as 60 seconds as the values from 60 seconds to 5 minutes achieve similar results in mapping the relevant device event e to U.

Sorting all IoT device events associated with each user activity U based on their timestamps leads to an ordered sequence of IoT device events $(e_1, e_2, \ldots, e_n)$. If two or more sequences of IoT device events happen for the same user activity during the 20 experiments, we select the sequence with the most occurrences. Such disparity is not observed in our experiments, as the same user activity always triggers the same sequences of IoT device events. We refer to the sequence of IoT device events, $(e_1, e_2, \ldots, e_n)$, as the *signature* of the user activity U, as well as the user activity itself.

Table II summarizes the signatures of all 21 user activities with the corresponding IoT device event sequences generated by following the above process. These signatures confirm the feasibility of inferring user activities related to physical safety and security in smart homes from the device events extracted from the IoT network traffic.

IV. USER ACTIVITY INFERENCE

A. User Activity Inference Problem

Given a sequence of IoT device events $\mathbb{S} = (s_1, s_2, \ldots, s_m)$ and a set of user activity signatures $\mathbb{U} = \{U_1, U_2, \ldots, U_r\}$ where U_i is the signature of the user activity i, we define the **user activity inference problem** as inferring all user activities that generate events in $\mathbb{S}$.

The unpredictability of user activities and the missing and out-of-order device events have created substantial challenges for solving the user activity inference problem. In this paper, we present the first attempt to give a heuristic solution of this problem via finding approximate matches of the user activities' signatures from $\mathbb{S}$. Specifically, we formulate a problem called $k_{\leq}$*approximate signature matching* and develop an optimal solution for user activity inference based on algorithms designed for solving $k_{\leq}$approximate signature matching problem.

B. $k_{\leq}$Approximate Signature Matching Problem

The *edit distance* (ED) between a user activity signature U and a sequence of device events $\mathbb{S}$ is defined as the minimum cost of changing $\mathbb{S}$ into U where 1) only the operation of deletion from $\mathbb{S}$ and U is allowed, 2) deletion of an event from U has cost 1, and 3) deletion of an event from $\mathbb{S}$ has cost 0. This definition of ED is different from the classic definition in literature in that the substitution is not a valid operation here because a device event could be missing or out-of-order but should not change into another event. In addition,

the costs of deleting different events from U are different due to the varying importance of these events to U. For ease of presentation, we set the cost of deleting any event from U uniformly as 1.

Given a sequence of device events $\mathbb{S} = (s_1, s_2, \ldots, s_m)$, a user activity signature $U = (e_1, e_2, \ldots, e_n)$, and an integer $k \geq 0$, we define a $k_{=}$**single-approximate signature match** as a minimal-length subsequence of $\mathbb{S}$ with the start index i and end index j and $ED((s_i, \ldots, s_j), U) = k$, where k is the approximation parameter.

The $k_{=}$**approximate signature matching problem** is to reveal the *maximum* number of non-overlapping $k_{=}$single-approximate matches of U in $\mathbb{S}$, which is referred to as $k_{=}$approximate match.

The $k_{\leq}$**approximate signature matching problem** is to find all the approximate matches that:
1) exist in an $i_{=}$approximate match of U in $\mathbb{S}$ where $i \leq k$;
2) if there exists a match M in $i_{=}$approximate match, then there does not exist a match M' in $j_{=}$approximate match where $j > i$ and the start and end indices overlap with those of M.

For example, if $\mathbb{S} = (a, a, b, a, c, a, b, a, a, b, a, b, c)$ and $U = (a, b, a, c)$, a $1_{\leq}$approximate match of U in $\mathbb{S}$ includes two $0_{=}$single-approximate signature matches (s_2, s_3, s_4, s_5) and $(s_9, s_{10}, s_{11}, s_{13})$ and one $1_{=}$single-approximate signature match (s_6, s_7, s_8).

C. $k_{\leq}$Approximate Signature Matching Algorithm

In this section, we first present Algorithm 1 to solve the $k_{=}$approximate match problem which aims at identifying all $k_{=}$single-approximate matches of U in $\mathbb{S}$. In Algorithm 1, *start* records the starting index for the chunk of $\mathbb{S}$ that U is matched against. C is a two-dimensional cost matrix of size $(m+1) \times (n+1)$ and C records the edit distance between two sequences $\mathbb{S}_{start,i}$ and $U_{0,j}$. The first row of C is initialized as 0 to n since the cost of matching U to an empty device event sequence equals to cost of deleting events from U. The first column of C is initialized as all 0s because the match of U can start from anywhere in $\mathbb{S}$. The last column of C indicates the edit distance between U and $\mathbb{S}_{start,i}$ where $i = start, \ldots, m-1$.

The main idea of this algorithm is that the cost of matching sequence $U_{0,j}$ in $\mathbb{S}_{start,i}$ is the same as the cost of matching $U_{0,j-1}$ in $\mathbb{S}_{start,i-1}$ if $s_i = e_j$. Otherwise, the cost either equals to the cost of matching $U_{0,j}$ in $\mathbb{S}_{start,i-1}$ due to the cost 0 of deleting s_i, or equals to the cost of matching $U_{0,j-1}$ in $\mathbb{S}_{start,i}$ plus 1 due to the cost 1 of deleting e_j.

If $C_{i,n} = k$, we find a match of U in $\mathbb{S}$ that ends at s_i with the approximate parameter k . We then output the indices of the current match by backtracking the costs saved in C. Next, we reset the cost matrix by setting the i-th row of C from 0 to n and then continue to find matches starting from $\mathbb{S}_{i+1}$.

Fig. 3 shows a running example of Algorithm 1 with $\mathbb{S} = (a, a, b, a, c, a, b, a, a, b, a, b, c)$, $U = (a, b, a, c)$, and $k = 0$, respectively. In this example, s_1 equals to e_1

Algorithm 1: $k_{=}$appxMatch($\mathbb{S}, U, k$)

Input: Device event sequence $\mathbb{S} = (s_1, s_2, \ldots, s_m)$, a user activity signature $U = (e_1, e_2, \ldots, e_n)$, approximation parameter k
Output: A list L_k of all $k_{=}$single-approximate match U in $\mathbb{S}$. Each match is represented by an ordered sequence of the indices of device events in $\mathbb{S}$ matched to U

```
L_k ← ∅; start ← 1;
for i := 0 to m do
    C_{i,0} ← 0;
for j := 1 to n do
    C_{0,j} ← j;
for i := 1 to m do
    for j := 1 to n do
        if s_i == e_j then
            C_{i,j} ← C_{i-1,j-1};
        else if C_{i-1,j} < C_{i,j-1} + 1 then
            C_{i,j} ← C_{i-1,j};
        else
            C_{i,j} ← C_{i,j-1} + 1;
    if j == n and C_{i,j} == k then
        M ← ∅; p ← i; q ← n;
        while q > 0 do
            if C_{p,q} == C_{p-1,q-1} and s_p == e_q then
                p ← p − 1; q ← q − 1;
                M.insert(p);
            else if C_{p-1,q} < C_{p,q-1} + 1 then
                p ← p − 1;
            else
                q ← q − 1;
        L_k.insert(M);
        for j := 1 to n do
            C_{i,j} ← j; start ← i;
output List L_k.
```

		1. a	2. b	3. a	4. c
	0	1	2	3	4
1. a	0	0	1	2	3
2. a	0	0	1	1	2
3. b	0	0	0	1	2
4. a	0	0	0	0	1
5. c	0	0	0	0	0
6. a	0	0	1	2	3
7. b	0	0	0	1	2
8. a	0	0	0	0	1
9. a	0	0	0	0	1
10. b	0	0	0	0	1
11. a	0	0	0	0	1
12. b	0	0	0	0	1
13. c	0	0	0	0	0

Fig. 3. A running example of Algorithm 1

which are both a, thus $C_{1,1} = 0$. s_1, i.e., a, does not equal to e_2, i.e., b, therefore, $C_{1,2} = 1$ because $C_{1,1} + 1 = 1$ and $C_{0,2} = 2$. Similarly, we determine the other values in the cost matrix C until comparing s_5 with e_4 which are both c, so $C_{5,4} = C_{4,3} = 0$. As we reach the last column of the cost matrix, and the cost value equals to $k = 0$, we find a match of U in $\mathbb{S}$ with the approximation parameter 0. Then we start from $C_{5,4}$ and revisit the cost matrix to output this match as $(5, 4, 3, 2)$.

We continue on searching for matches from s_6 until we find another $0_{=}$single-approximate signature match ending

Algorithm 2: $k_{\leq}$appxMatch($\mathbb{S}, U, k$)

Input: Device event sequence $\mathbb{S}$, a user activity signature U, approximation parameter k
Output: $\mathbb{L} = (L_0, L_1, \ldots, L_k)$ where L_k is the $k_{=}$approximate match of U in $\mathbb{S}$

```
Pre-process S to remove all the events in S but not in U;
L_0, L_1, ..., L_k ← ∅;
Initialize Q with (0, m − 1) as its only element;
for i := 0 to k do
    T ← null;
    while Q do
        H ← Q.head(); D ← S[H.start, H.end];
        L_k.append(k_=appxMatch(D, U, i));
        for each k_=approximate match in L_k do
            B ← the event sequence before M in D;
            D ← the event sequence after M in D;
            if B is not empty then
                start ← starting index of B;
                end ← ending index of B;
                T.append(start, end);
        if D is not empty then
            start ← starting index of D;
            end ← ending index of D;
            T.append(start, end);
    Q ← T;
return L.
```

at s_{13} as $(13, 11, 10, 9)$. As a result, Algorithm 1 outputs $0_{=}$approximate match $L_0 = ((5, 4, 3, 2), (13, 11, 10, 9))$.

Algorithm 2 presents the solution for the $k_{\leq}$approximate signature matching problem. It outputs $\mathbb{L} = (L_0, L_1, \ldots, L_k)$, where L_i records the $i_{=}$approximate match of U in $\mathbb{S}$. Each element on L_i stores the indices of the device events in $\mathbb{S}$ that form the $i_{=}$single-approximate match of U.

The main idea of Algorithm 2 is to start from $0_{=}$approximate match, repeatedly discover the $i_{=}$approximate match of U in $\mathbb{S}$, and remove all the device events in the $i_{=}$approximate match from $\mathbb{S}$ until $i = k$. Indices of events in $\mathbb{S}$ are carefully recorded that only portions of device events in $\mathbb{S}$ that do not overlap with the starting and ending indices of matches to U in the current round will be considered in the next round. In this way, only non-overlapping matches are selected for output and approximate matches with smaller k are discovered before approximate matches with bigger k. List Q is designed to save a list of a pair of starting and ending indices for a continuous portion of $\mathbb{S}$ that have not been matched in the approximate match with smaller k. The output of Algorithm 2 with $k = 3$ on the running example in Fig. 3 is $L_0 = ((2, 5), (9, 13))$, $L_1 = ((6, 8))$, $L_2 = \emptyset$, and $L_3 = ((1, 1))$.

Theorem 1: The time complexity of $k_{=}$appxMatch algorithm is $O(mn)$ and the time complexity of $k_{\leq}$appxMatch algorithm is $O(kmn)$ where m is the number of events in $\mathbb{S}$ and n is the number of events in U.

Proof Sketch. This can be derived from the execution of the algorithms. Details are omitted due to space limitations. □

Theorem 2: The $k_{=}$appxMatch algorithm outputs the maximum number of non-overlapping $k_{=}$single-approximate signature matches of U in $\mathbb{S}$.

Proof Sketch. It has been proved in literature [16] that given a sequence of overlapped time intervals, the greedy algorithm which chooses an interval with the earliest finish time and excludes all other overlapping intervals, outputs the maximum number of non-overlapping intervals. The $k_{=}$appxMatch algorithm indeed outputs the match with the earliest finish time first and it only considers non-overlapping matches. □

D. Inferring User Activities via Approximate Matching

With the $k_{\leq}$appxMatch algorithm, we can discover approximate matches of each user activity independently. However, it is possible that two matches of different user activities share overlapped device events which cause collisions and ambiguity. Therefore, we further develop a trimming step to remove the collisions in the device event sequences that are matched to different user activities.

Algorithm 3: actInfer($\mathbb{S}, \mathbb{U}, k$)

Input: A device event sequence $\mathbb{S}$, an ordered sequence of signatures of all user activities by their lengths $\mathbb{U} = (U_1, U_2, \ldots, U_r), k$
Output: $L_{U_1}, L_{U_2}, \ldots, L_{U_r}$ where collisions in approximate matches have been removed

```
for i := 0 to r do
    L_{U_i} ← k_≤appxMatch(S, U_i, k);
for i := 0 to k do
    for j := 1 to r do
        for M ∈ L^i_{U_j} do
            if ∃M' ∈ L^{i'}_{U_{j'}} (i' < i) and M ∩ M' ≠ ∅ then
                remove M from L^i_{U_j};
            else if ∃M' ∈ L^i_{U_{j'}} (j ≠ j') and M ⊆ M' then
                remove M from L^i_{U_j};
output L_{U_1}, L_{U_2}, ..., L_{U_r}.
```

The trimming heuristics in the last step of inferring user activities prefer a match with a smaller approximate parameter k or a longer sequence of IoT device events in its signature. This trimming step is designed based on the following empirical observations from our real-world experiments:

1) The chance of missing device events in the signatures of user activities during the matching is marginal. Thus we always prefer matches with smaller values of k.
2) If the signature of one user activity is a subset of another one and both of them are matched independently, the user activity with the superset signature is preferred if both activities are inferred at the same time.
3) If two user activities from different users have overlapping device events, and these two activities happen at the same time, then it is possible that the overlapped events are shared by both of them. For example, if one user is entering the home from the hallway while another

user is heading towards the kitchen through the hallway, these two user activities will only generate one light `on` event since the light will be turned on only once. Thus this device event is shared by these two user activities and can be mapped to both of them during the matching.

Algorithm 3 presents the pseudocode of the final algorithm of user activity inference and outputs the set of inferred user activities from the IoT device event streams with the simple yet effective trimming heuristic step.

V. Performance Evaluations

A. Experiment Setup of Smart Home Environments

To evaluate our proposed algorithms of inferring user activities from IoT network traffic, we have designed and set up a real-world smart home environment, as illustrated in Fig. 2, where we deployed a number of heterogeneous IoT devices. In this smart home environment, each user activity enumerated in Table II will trigger at least one IoT device event, thus leading us to observe and collect IoT network traffic via the programmable home router.

In our experiment, we use the Linksys WRT1900AC router running the OpenWrt operating system to collect TCP/IP packets for all the outgoing and incoming network traffic between IoT devices and remote hosts as well as internal traffic in the local area network (LAN) traffic between IoT devices. For each IoT device event, we adopt the method in [33] for generating its signature consisting of an ordered sequence of IP packets with inter-packet time intervals. Based on these signatures, we run the signature matching and event extraction algorithms in [33] on the collected traffic to detect all of the IoT device events that happened in the smart home.

For each of the 21 user activities in Table II, we first learn and build its signature via capturing and studying the underlying IoT network traffic while intentionally repeating the activity with time logs, which serve as the labeled ground truth. Extracting IoT device events from the network traffic and correlating them with the labeled and repeated user activities allow us to recognize and generate the signatures for all 21 user activities. To evaluate the performance of our proposed user activity inference algorithms, we also repeatedly run thousands of user activities at different times and days with time logs over a two-month evaluation period. In addition, our smart home environment continuously collects IoT network traffic, even during the time that we are not actively running the experiments of triggering the 21 user activities. The labeled user activities and collected IoT network traffic provide the valuable dataset for evaluating the correctness and accuracy of our algorithms for inferring user activities.

Although a number of existing studies have shared smart home IoT network traffic datasets [26, 31], none of them include the labeled mapping between individual user activity and the corresponding network traffic. Thus, our experiment setup and the collected dataset will potentially provide the IoT research community a unique dataset for understanding user behaviors from network traffic.

B. Evaluation Metrics

To evaluate and quantify the performance of our proposed algorithms, we use the widely used metrics, i.e., true positives ($\mathcal{TP}$), false positive ($\mathcal{FP}$), false negative ($\mathcal{FN}$), and true negative ($\mathcal{TN}$). Specifically, for a given *inferred* user activity α', if a matching *real* user activity α is found at the same time from the ground truth, we consider this activity as a true positive. However, if there is no matching user activity from the ground truth, we consider it as a false positive. For a given *real* user activity α, if the user activity inference algorithms report a different *inferred* user activity, e.g., β', or simply fail to report an *inferred* user activity, we consider it as a false negative. When the user inference algorithms do *not* report an *inferred* user activity α' for any *real* user activity other than α in the ground truth, it is a true negative. In our experiments, we always observe perfect true negative results, thus we leave the true negative measures out of the experiment results.

In addition to these four measures from the confusion matrix, we also use precision ($\mathcal{P}$), recall ($\mathcal{R}$), accuracy ($\mathcal{A}$), and F1 score to understand the overall quality of our user activity inferring experiments. The precision is calculated as $\mathcal{P} = \frac{\mathcal{TP}}{\mathcal{TP}+\mathcal{FP}}$, while the recall is $\mathcal{R} = \frac{\mathcal{TP}}{\mathcal{TP}+\mathcal{FN}}$. The accuracy can be calculated as $\mathcal{A} = \frac{\mathcal{TP}+\mathcal{TN}}{\mathcal{TP}+\mathcal{TN}+\mathcal{FP}+\mathcal{FN}}$. The F1 score is the harmonic mean of precision and recall and can be calculated as $2 \times \frac{\mathcal{P}\times\mathcal{R}}{\mathcal{P}+\mathcal{R}}$.

C. Experimental Results

Using performance evaluation metrics introduced in Section V-B, we present the performance of our proposed user activity inference system in three phases. In the first phase, we present the experimental results of running the $0_{<}$approximate matching algorithm, i.e., requiring the *exact* matching of device event sequences for all user activities. In the second phase, we present the result of running the $1_{<}$approximate matching algorithm, allowing at most one missing device event due to packet delays and losses or device malfunctions. In the third and final phase, we add the optimization rules to the $1_{<}$approximate matching algorithm for substantially improving the performance of user activity inference.

1) Phase I: $0_{<}$approximate matching. Our experiments of running $0_{<}$approximate matching algorithm for inferring thousands of user activities in the real-world smart home environment have shown that the overall value of accuracy, precision, and recall are 0.96, 0.99, and 0.97, respectively, as illustrated in columns 6 to 8 of the last row in Table III. We can observe that most of the user activities can be detected correctly. However, the user activity inference algorithms occasionally fail to detect user activities $\#2, \#4, \#6, \#8, \#10, \#11, \#12$, and $\#14$, leading to a small number of false negatives, as shown on the fifth column of Table III.

Our in-depth investigation has discovered that nearly all of these false negatives are caused by the "missing" Ring Doorbell's motion detection event from the device event sequences of these user activities. The underlying root cause of these "missing" events is the multiple-second long delay of reporting these motion events by the battery-powered Ring

TABLE III
EXPERIMENTAL RESULTS OF OUR PROPOSED USER ACTIVITY INFERENCE ALGORITHMS IN THE REAL-WORLD SMART HOME ENVIRONMENT

User Act.	#	Phase I						Phase II						Phase III					
		$\mathcal{TP}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{TP}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{TP}$	$\mathcal{FP}$	$\mathcal{FN}$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$
1	112	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00
2	115	105	0	10	0.91	1.00	0.91	115	0	0	1.00	1.00	1.00	115	0	0	1.00	1.00	1.00
3	112	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00
4	115	106	0	9	0.92	1.00	0.92	115	0	0	1.00	1.00	1.00	115	0	0	1.00	1.00	1.00
5	113	113	0	0	1.00	1.00	1.00	113	0	0	1.00	1.00	1.00	113	0	0	1.00	1.00	1.00
6	115	97	0	18	0.84	1.00	0.84	115	0	0	1.00	1.00	1.00	115	0	0	1.00	1.00	1.00
7	112	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00
8	112	110	0	2	0.98	1.00	0.98	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00
9	121	121	0	0	1.00	1.00	1.00	121	78	0	0.61	0.61	1.00	121	0	0	1.00	1.00	1.00
10	112	105	0	7	0.94	1.00	0.94	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00
11	350	347	0	3	0.99	1.00	0.99	350	0	0	1.00	1.00	1.00	350	0	0	1.00	1.00	1.00
12	294	269	0	25	0.91	1.00	0.91	294	0	0	1.00	1.00	1.00	294	0	0	1.00	1.00	1.00
13	116	116	0	0	1.00	1.00	1.00	116	0	0	1.00	1.00	1.00	116	0	0	1.00	1.00	1.00
14	113	98	0	15	0.87	1.00	0.87	113	0	0	1.00	1.00	1.00	113	0	0	1.00	1.00	1.00
15	138	138	0	0	1.00	1.00	1.00	138	33	0	0.81	0.81	1.00	138	0	0	1.00	1.00	1.00
16	132	132	0	0	1.00	1.00	1.00	132	33	0	0.81	0.81	1.00	132	0	0	1.00	1.00	1.00
17	112	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00
18	224	224	16	0	0.93	0.93	1.00	224	16	0	0.93	0.93	1.00	224	16	0	0.93	0.93	1.00
19	112	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00	112	0	0	1.00	1.00	1.00
20	113	113	0	0	1.00	1.00	1.00	113	0	0	1.00	1.00	1.00	113	0	0	1.00	1.00	1.00
21	116	116	0	0	1.00	1.00	1.00	116	0	0	1.00	1.00	1.00	116	0	0	1.00	1.00	1.00
all	2959	2870	16	89	0.96	0.99	0.97	2959	158	0	0.95	0.95	1.00	2959	16	0	0.99	0.99	1.00

Doorbell [4], leading to the *out-of-order* device events for these user activities. In other words, these motion events are simply delayed, actually not missing. However, as the $0_{\leq}$approximate matching algorithm requires the exact matching of device event sequences for all user activities, these cases with out-of-order device events are reported as false negatives.

In addition to these false negatives, there are also 16 cases of false positive for user activity #18. Our follow-up analysis reveals that the real user activities indeed have happened as inferred, but are not recorded in our controlled experiments. This observation suggests that our user activity inference algorithms could potentially monitor real-time IoT network traffic and alert homeowners on unexpected or anomalous user activities for home safety and other applications.

2) Phase II: $1_{\leq}$approximate matching. Running the $0_{\leq}$approximate matching algorithm for inferring thousands of user activities achieves high accuracy, but returns a non-trivial of false negatives. Therefore, in phase II we explore the increase of the value of k in the matching algorithm for accommodating the missing and out-of-order device events when searching and matching the signatures of user activities. To understand and quantify the performance tradeoff, specifically false positives and false negatives with varying k, we run the $k_{\leq}$approximate matching with the value of k increasing from 0 to 5. As illustrated in Fig. 4, the increase of k leads to lower false negatives but higher false positives. In addition, increasing k from 1 to $2, 3, 4, 5$ achieves marginal improvements in false negatives, but incurs significant penalties in false positives. Therefore, we choose $1_{\leq}$approximate matching algorithms for the remaining experiment evaluations.

By setting k as 1, the approximate matching algorithm allows at most one missing or out-of-order device event when matching the signatures of user activities. Such flexibility effectively addresses the challenge of the long delay of the

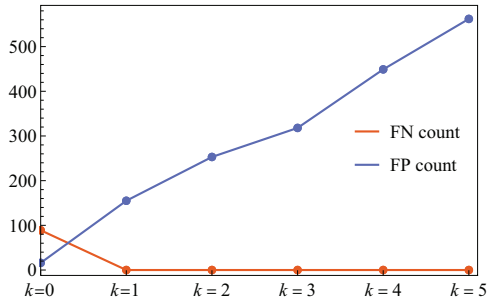

Fig. 4. Tradeoff analysis of false positive vs. false negatives with varying k

Ring Doorbell's motion detection event, and significantly improves the performance of our user activity inference system in the smart home environment. As shown in the column 11 of Table III, the false negative measures of user activities $\#2, \#4, \#6, \#8, \#10, \#11, \#12$, and $\#14$ drop to 0.

However, in column 10 of the same table, we also observe the increase of false positive measures from 0 to 78, 33, 33 for user activities $\#9, \#15$, and $\#16$, respectively. Our follow-up analysis discovers that the signatures of all of these three user activities ($\#9, \#15$, and $\#16$) have a length of 2 device events. The short length of 2 device events in the signatures explains the high false positives, since $1_{\leq}$approximate matching algorithm will report the matching success of one of these activities whenever finding one single device event in their signatures. Given the above observations, we continue to explore optimization rules in the approximate matching algorithm to reduce high false positives while maintaining low false negatives during the matching process.

3) Phase III: $1_{\leq}$approximate matching with optimization. Inspired by the findings of high false positives for the short device event sequences, we add a simple yet effective optimization rule for the approximate matching algorithm. Specifically, when the length of the device event sequence for a user activity is equal to or less than a certain threshold, referred

to as θ, we only run the $0_{\leq}$approximate matching algorithm to infer such activities. The intuition of such optimization rule lies in the observation that when the length of the device event sequence in the user activity signature is short, the probability of finding false positives is very high. Based on our empirical results, we set θ as 3 in our experiments.

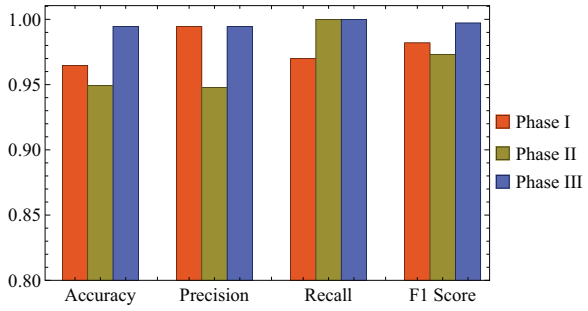

Fig. 5. Overall accuracy, precision, recall, and F1 scores of Phase I, Phase II, and Phase III, respectively

As shown in columns 16 and 17 in Table III, the false positives of user activities $\#9, \#15$, and $\#16$ drop to 0 thanks to the simple yet effective optimization rule. The only remaining false positives correspond to user activity $\#18$, which are caused by real but uncontrolled user activities that were not recorded in our experiments. Fig. 5 compares the overall accuracy, precision, recall, and F1 scores of running our proposed user activity inference algorithms over three phases: $0_{\leq}$approximate matching, $1_{\leq}$approximate matching, $1_{\leq}$approximate matching with optimization, respectively. Clearly, $1_{\leq}$approximate matching with the simple and intuitive optimization rule achieves the best performance.

In summary, our two-month long experiments have shown that our proposed user activity inference system is able to effectively and accurately detect and infer user activities from IoT network traffic in smart homes. By applying the approximation matching algorithm with the simple optimization rule, we achieve the overall values of accuracy, precision, and recall as 0.99, 0.99, and 1.00, respectively.

VI. Related Work

The wide deployment and growth of IoT devices in smart homes in the last two decades have attracted significant research interests in studying network traffic of IoT devices [5, 19, 21, 26, 28, 29, 32, 36, 37]. These research studies collect TCP/IP packets or network flows of IoT devices, extract a rich set of features from these traffic data, and provide critical insights on IoT communications patterns and behavioral dynamics. The deep understanding of IoT traffic fingerprints benefits a variety of applications such as anomaly detection, security monitoring, IoT device classification, and IoT device event detection.

Given the prevalent threats and attacks targeting IoT devices [2, 3, 13, 18, 22, 23, 27, 30, 40], detecting IoT device events has become a crucial task for IoT security [35]. For example, a recent research [18] has presented new event-eliminating and event-spoofing attacks on commercial wireless home alarm systems. Several parallel research efforts have developed different approaches for detecting IoT devices events [6, 24, 31, 33, 38]. For example, the studies in [6, 24] first extract and learn various traffic features for IoT device events, and subsequently develop machine learning classifiers for distinguishing IoT device events. HoMonit [38] designs a deterministic finite automaton (DFA) model to detect the specific events for many Samsung SmartThings devices based on the the link-layer traffic data. By studying the network layer and transport layer network traffic data, PingPong [31] and IoTAthena [33] develop deterministic algorithms for detecting IoT device events using the signatures generated from TCP/IP packets.

A few recent studies have explored the events and statuses of IoT devices for recognizing user activities and behaviors [1, 12, 25]. For example, the researchers of [25] deploy numerous sensors in a three-bedroom apartment and develop a hidden Markov model (HMM) based system to recognize and track user activities. Peek-a-Boo [1] is able to launch privacy attacks in smart homes by passively sniffing the encrypted network traffic over the air and utilizing machine learning approaches to recognize six different user activities. Similarly, the study in [12] proposes Home Automation Watcher (HAWatcher), a semantics-aware anomaly detection system for appified smart homes, for discovering the correlation according to semantic information in smart homes, and exploits these correlations for modeling a smart home's normal behaviors. Different from these prior work, IoTMosaic first detects IoT device events from smart home network traffic and then recognizes and profiles user activities to map them to their triggered IoT device event sequences. Subsequently, IoTMosaic develops an effective approximate matching algorithm for inferring user activities in smart homes, which could provide homeowners critical insights on what is happening in their homes instantly and automatically.

VII. Conclusions and Future Work

This paper proposes IoTMosaic for inferring user activities from IoT network traffic. Based on the IoT device events detected from smart home network traffic, IoTMosaic generates the signatures of diverse user activities consisting of IoT device event sequences. Given the observation of missing and out-of-order device events due to device malfunctions and varying network latencies, we design the approximate matching algorithms to capture the exact or approximate matches of user activities' signatures in the device event sequence. In addition, we devise a heuristic trimming strategy to resolve the conflicts in multiple matches of user activities due to overlapping device events in their signatures. Our two-month experimental results with thousands of user activities in a real-world smart home show that our proposed algorithms can infer different user activities with accuracy, precision, and recall of 0.99, 0.99, and 1.00, respectively. Our future work is centered on deploying IoTMosaic in other smart environments and exploring the applications of the system such as home safety and anomaly detection.

References

[1] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, and 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. 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] C. Barry, "Ring Video Doorbells Rescued By This Great New Feature," 2020, https://www.forbes.com/sites/barrycollins/2020/11/29/ring-video-doorbells-rescued-by-this-great-new-feature/.

[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] V. Bhosale, L.-D. Carli, and I. Ray, "Detection of Anomalous User Activity for Home IoT Devices," in *Proc. of IoTBDS*, 2021.

[7] S. Birnbach, S. Eberz, and I. Martinovic, "Peeves: Physical Event Verification in Smart Homes," in *Proc. of ACM CCS*, 2019.

[8] H. Chi, Q. Zeng, X. Du, and L. Luo, "PFirewall: Semantics-Aware Customizable Data Flow Control for Home Automation Systems," in *Proc. of NDSS*, 2021.

[9] 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.

[10] J. Fan, Y. He, B. Tang, Q. Li, and R. Sandhu, "Ruledger: Ensuring Execution Integrity in Trigger-Action IoT Platforms," in *Proc. of IEEE INFOCOM*, 2021.

[11] E. Fernandes, J. Jung, and A. Prakash, "Security Analysis of Emerging Smart Home Applications," in *Proc. of IEEE S&P*, 2016.

[12] C. Fu, Q. Zeng, and X. Du, "HAWatcher: Semantics-Aware Anomaly Detection for Appified Smart Homes," in *Proc. of USENIX Security*, 2021.

[13] S. Herwig, K. Harvey, G. Hughey, R. Roberts, and D. Levin, "Measurement and Analysis of Hajime, a Peer-to-peer IoT Botnet," in *Proc. of NDSS*, 2019.

[14] 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.

[15] Y. Jia, L. Xing, Y. Mao, D. Zhao, X. Wang, S. Zhao, and Y. Zhang, "Burglars' IoT Paradise: Understanding and Mitigating Security Risks of General Messaging Protocols on IoT Clouds," in *Proc. of IEEE S&P*, 2020.

[16] J. Kleinberg and E. Tardos, *Algorithm Design*. Pearson/Addison-Wesley, 2006.

[17] 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.

[18] T. Li, D. Han, J. Li, A. Li, Y. Zhang, R. Zhang, and Y. Zhang, "Your Home is Insecure: Practical Attacks on Wireless Home Alarm Systems," in *Proc. of IEEE INFOCOM*, 2021.

[19] X. Ma, J. Qu, J. Li, J. C. Lui, Z. Li, and X. Guan, "Pinpointing Hidden IoT Devices via Spatial-temporal Traffic Fingerprinting," in *Proc. of IEEE INFOCOM*, 2020.

[20] S. Manandhar, K. Moran, K. Kafle, R. Tang, D. Poshyvanyk, and A. Nadkarni, "Towards a Natural Perspective of Smart Homes for Practical Security and Safety Analyses," in *Proc. of IEEE S&P*, 2020.

[21] 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.

[22] P. Morgner, S. Mattejat, Z. Benenson, C. Müller, and F. Armknecht, "Insecure to the Touch: Attacking ZigBee 3.0 via Touchlink Commissioning," in *Proc. of ACM WiSec*, 2017.

[23] A. Mosenia and N.-K. Jha, "A Comprehensive Study of Security of Internet-of-Things," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 4, pp. 586 – 602, 2016.

[24] T. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves, and A.-R. Sadeghi, "HomeSnitch: Behavior Transparency and Control for Smart Home IoT Devicess," in *Proc. of ACM WiSec*, 2019.

[25] P. Rashidi, D. J. Cook, L. B. Holder, and M. Schmitter-Edgecombe, "Discovering Activities to Recognize and Rrack in a Smart Environment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 4, pp. 527–539, 2011.

[26] 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 *Proc. of ACM IMC*, 2019.

[27] 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.

[28] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "IoT Devices Recognition Through Network Traffic Analysis," in *Proc. of IEEE International Conference on Big Data*, 2018.

[29] 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*, vol. 18, no. 8, pp. 1745–1759, 2018.

[30] V. Sivaraman, D. Chan, D. Earl, and R. Boreli, "Smart-Phones Attacking Smart-Homes," in *Proc. of ACM WiSec*, 2016.

[31] R. Trimananda, J. Varmarken, A. Markopoulou, and B. Demsky, "PingPong: Packet-Level Signatures for Smart Home Device Events," in *Proc. of NDSS*, 2019.

[32] 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*, vol. 8, no. 1, pp. 89–101, 2020.

[33] Y. Wan, K. Xu, F. Wang, and G. Xue, "IoTAthena: Unveiling IoT Device Activities from Network Traffic," *IEEE Transactions on Wireless Communications*, vol. 21, no. 1, pp. 651–664, 2022.

[34] 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.

[35] Q. Wang, W. Hassan, A. Bates, and C. Gunter, "Fear and Logging in the Internet of Things," in *Proc. of NDSS*, 2018.

[36] 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.

[37] 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.

[38] 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.

[39] W. Zhou, Y. Jia, Y. Yao, L. Zhu, L. Guan, Y. Mao, P. Liu, and Y. Zhang, "Discovering and Understanding the Security Hazards in the Interactions between IoT Devices, Mobile Apps, and Clouds on Smart Home Platforms," in *Proc. of USENIX Security*, 2019.

[40] T. Zillner, "ZigBee Exploited: The Good, the Bad and the Ugly," in *Proc. of the DeepSec Conferences In Depth Security*, 2017.