Behavior-Based Probabilistic User Identification Through Passive Sensing

Submitted in partial fulfillment of the requirements for

the degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Xiao Wang

B.S., Information Engineering, Shanghai Jiao Tong University
M.S., Electrical and Computer Engineering, Carnegie Mellon University

Carnegie Mellon University
Pittsburgh, PA

July, 2017

Abstract

A notion of identity is of vital importance to each individual and the society. Its significance is manifest from two perspectives. On one hand, many systems rely on identity to provide proper services and fulfill their functionality. On the other hand, loss or leakage of identity can cause disastrous consequences. Identity is hence the sought-after to both benign and malicious entities. The representation, acquisition, validation and application of identity have been evolving consistently with the advances in technology. The last decade has witnessed the rapid adoption of mobile electronic devices and the emergence of Internet-of-Things hardware. Their onboard solid-state sensors enable ubiquitous sensing of user, environment and the interaction between them, accumulating ample sensory data that can be leveraged for user identification. In this dissertation, we study behavior-based user identification through passive sensing and its application in new real-world scenarios. We apply statistical modeling methods to the sensory measurements and extract sufficient entropy to establish user identity. A corresponding behavior-based identification framework is specified with necessary components and steps. We contrast the behavior-based approach to existing mechanisms and highlight its advantages. To demonstrate the practical implications of the proposed approach, we propose two application scenarios in the mobile and IoT realms, and conduct experiments. The mobile application investigates user recognition across mobile devices. We leverage the way users interact with mobile apps to track them even when they switch between multiple mobile devices. The IoT application, following the emergence of IoT-equipped buildings, studies room-level indoor localization. We model residents' mobility patterns through occupancy measurements and further learn their locations. Our experimental results demonstrate the effectiveness of identifying users through behavioral patterns under distinct application scenarios.

Acknowledgements

This journey is challenging yet enjoyable. There are an army of people how make my PhD life at CMU truly memorable. First of all, I would like to thank my advisor Professor Patrick Tague for his excellent mentorship and support throughout my PhD career. I benefit from his academic knowledge about problem formulation, paper writing and presentation, as well as his life attitude in stress handling, time management and emotional regulation. Also, thanks for all the group lunches, dinners and barbecues where many research ideas were born. I also want to thank my committee members who devote valuable time to help with the thesis. Thank Professor Ole Mengshoel for his detailed comments and feedback. Moreover, thanks for serving on my qualification committee. It was also an great experience collaborating with him on a paper. Thank Professor Osman Yagan also for his time and help on my thesis. My experience as a teaching assistant in his class was really wonderful. Thank Dr Sandeep Bhatkar for all the suggestions and discussions. His sharp questions improve the thesis. I also enjoyed working with him during my internship and all the table tennis at that time.

I also want to thank the fellow PhD students. Thank Jun Han and Yuan Tian for the company at Room 1033 during all the days and nights of research. Thanks to Yu Seung Kim and Bruce DeBruhl for all the warmhearted help. They are like big brothers. Thank Arjun Athreya for the good old times at CMU. Thank Eric Chen, my first roommate, for all the funny chats. Thank Le Nguyen for all the inspiring discussions. Besides the officemates, I also want to thank Zheng Sun for all the help from the very beginning, all the insightful discussions and all the tasty meals we had together that enriched my life at CMU. Thanks to Lu Zheng for the inspiring chats and wild ideas. Thank Hengtze Cheng for the help during my qual exam and for sharing the experience as a senior student. Also, it was fun playing tennis with him and Zheng. Thank Jiang Zhu for being an excellent team partner in our course project, and all the advice as a veteran in this field. Thank Dongzhen Piao for all the wild discussions. Thanks to Ming Zeng for all discussions, meals and fun activities. Thank Tong Yu for the collaborations on papers. Thank my Pittsburgh friends Chen Chen, Miao Yu, Zongwei Zong, Yanlin Li, Jun Zhao, Youzhi Bao and Jiacheng Hong for all the fun during my study there. Many thanks to all my CMU friends David Huang, Frank Mokaya, David Cohen, Shijia Pan, Brian Ricks, William Chan, Annirudha Basak, Pang Wu, Abhinav Jauhri, Yong Zhuang, Yingrui Zhang, and more.

In addition to the colleagues and friends at CMU, I would also like to thank my undergraduate research advisor Professor Xinbing Wang at Shanghai Jiao Tong University for his advice and all the help with my graduate school application. I also want to thank my undergraduate friends Yun Wang, Yuanzhong Xu, Xiangyang Liu, Hao Li, Rong Du, Zhuo Li, Minjie Zang, Zhongliang Zheng, Qiuyu Peng, Huan Tang, Jia Zhao, Xu Ha, and more for all the fun times. The work in this thesis is supported by the National Science Foundation under awards CNS-1149582.

Finally and most importantly, I want to thank my parents for their love and support all along. Without them I would never have the courage, capability and faith to pursue my dream. I never felt lost in life because of the love I always feel from them. This thesis is dedicated to beloved parents.

Contents

Chapter 1

Introduction

Human are social animals which interact extensively with other people. In social activities, identity plays a vital role. Validation of user identity guarantees that entities involved in an activity are the intended ones. The value of identity is two-fold from the benign and adversarial perspectives, respectively. On one hand, it is the foundation for many systems to operate and function properly. Banks rely on identity to process transactions. Internet content companies rely on it to provide services. Identity offers the capability to differentiate users and respond accordingly. On the other hand, identity is sought-after for malicious entities. This is a natural outcome of its aforementioned importance. Identity information is traded in underground markets and the estimated damage is tremendous. Loss or leakage of identity can lead to disastrous consequences. Losing identity may cause severe damage to property. Any leakage also rises deep concerns about privacy.

The importance of identity does not exist apart from specific real-world scenarios. The representation, acquisition, validation and application of identity have been consistently evolving with advances in technology and consequently the emergence of new challenges. For example, traditionally, it requires manual verification of ID cards. Then, dedicated hardware replaced human labor in many circumstances such as RFID reader for badge check-in. In what follows, we discuss the evolution of identification in terms of the representation, acquisition, validation and application of identity.

The *representation* of user identity generally falls into three categories including *something you know*, *something you have* and *something you are*. For instance, passwords, date of birth and social security number are the information

you know which can represent your identity, while keys, driver's license and other documents are the materials you have to demonstrate your identity. Something you are is more intrinsic to human. It typically refers to biometrics which are further categorized as physiological such as fingerprints and iris scans, or behavioral such as gait and keystroke.

The *acquisition* of identity may require user involvement, meaning the principal to be identified needs to explicitly present the identity to another principal. A simple example is someone presenting her passport to a TSA officer. Advances in technology is enabling more means to collect identity information. Depending on different representations of identity, it can be acquired by card readers, fingerprint scanners and other hardware. These new techniques significantly facilitate identity acquisition and expand the set of information that can be used as for identification.

Identity *validation* examines the obtained identity against established records. Essentially, validation is based on matching, which can be as simple as verifying a social security number. Some representations of identity require advanced techniques for validation such as fingerprint and other biometric information. Dedicated and complicated algorithms are needed to perform the procedure. Moreover, validation results in this circumstance are typically probabilistic due to the nature of these representations and the introduced algorithms.

The *applications* of identity are specific to real-world scenarios, and can be of benign or malicious intent. Technology innovation creates ample application scenarios by breaking down borders and connecting people. Information is now increasingly communicated and disseminated. To understand, leverage and regulate the information, identity plays a key role. This thesis will report our study in two brand new application scenarios that represent some new development in related fields and demonstrate how our approach can be effectively applied to real practice.

As aforementioned, the representation of identity falls into three categories. Each has its advantages and disadvantages. Something you know is the most common source of identity. It is straightforward and easy to use. However, something you know can become something you forget. It is not a rare case for users to forget their passwords since they have to memorize plenty of passwords for different usage. Something you have overcomes the drawback of forgetting. However, you have to always bring it with you any time you need your identity. Moreover, it brings inherent risk of losing it or being stolen by others. Something you are is something intrinsic to yourself and difficult to lose. We can think of fingerprint for an example. Also, it removes the hassle to keep the identity with you. To obtain and validate physiological biometrics, however, it typically requires dedicated hardware and complicated algorithms.

Something you are includes physiological and behavioral biometrics, which have distinct properties and are ap-

plicable to different scenarios. Physiological biometrics such as fingerprint and iris are explicit identities that provide strong guarantee for accurate identification. Something you know or have also offers this guarantee. This advantage, on another hand, is a major source of concern for privacy.

Behavioral biometric distinguishes itself from them in the sense that it provides favorable trade-off between identification accuracy and user privacy, and is hence applicable to many applications of benign purposes such as authentication and personalization. From the perspective of attackers, behavioral information provides some unique opportunities of identifying users without their attention. For explicit identities such as passwords, keys and fingerprint, it is difficult for attackers to obtain them since it typically requires the user to present the identity, while behaviors are intrinsic to users and often exposed to observation. Furthermore, people are less cautious about behavioral biometrics than explicit identities, allowing a better chance for adversaries to obtain behavioral information for malicious purposes.

Previously, it is rare to employ behavioral biometrics as identity representation due to difficulty in the acquisition procedure. The situation, however, has been changing with the rising of new application scenarios and new technology. The last decade has witnessed the rapid adoption of mobile electronic consumer devices and the emergence of Internet-of-Things devices which are equipped with a rich set of sensors. This capability enables ubiquitous sensing of user, environment and their interaction, and accumulates ample behavioral data that can be leveraged for user identification. The dissertation will report our efforts in studying behavior-based probabilistic user identification and its applications in new real-world scenarios. We will investigate both the benign and adversarial aspects of the proposed behavior-based approach, which will be manifested in the application scenarios.

To demonstrate the practical implications of the behavior-based identification, we study two application scenarios in the mobile and IoT realm, respectively. The first application concentrates on recognizing users across mobile devices. The importance of this problem arises from the fact that more and more user are becoming multi-device owners. They switch between multiple mobile devices such as smartphone and tablet on a daily basis. For apps installed across these devices, it is desirable to know they are actually used by the same user. This knowledge can greatly facilitate the enhancement of advertising, recommendation and user experience. This notion of identity is, however, missing for many users and many apps for various reasons. First, a number of apps do not enforce user login as it might prevent users from using the app in the first place. Second, a device might be shared by multiple users. There is a chance that the actual user is not the user who signed in. Under this circumstance, the absence of explicit

identifiers creates opportunities for the behavior-based approach. We will show how we may leverage the way a user interacts with apps to reveal the trace of that user. We will present a system named XRec that can provide this notion of identity to other services such as the recommender systems that will benefit from the information to trace users' past activities.

The second application scenario demonstrates the potential of the behavior-based approach from an adversarial perspective. It aims to reveal residents' indoor location through their mobility patterns. In IoT-equipped buildings, many sensors are placed to sense the environment and its users for the purpose of automated activation and smart building management. The information, on another hand, offers opportunities for revealing some private details about building residents. This dissertation shows how we can model users' mobility patterns through occupancy data and further learn their indoor locations.

This dissertation is organized as follows. In Chapter 2, we present the overall design for behavior-based identification, describing necessary components and steps to achieve the goal. Next, we introduces real-world problems that can benefit from the proposed framework and demonstrate implications of our approach. In Chapter 3, we present the benign application of cross-device user recognition. In Chapter 4, we present the malicious application of room-level indoor localization in smart buildings. We conclude in Chapter 5 and outline future research directions on this topic.

Chapter 2

Behavior-Based User Identification

In this chapter, we begin with a discussion about our behavior-based user identification approach. We propose a framework including indispensable components and steps as well as practical considerations with regards to design and implementation.

2.1 Background

Identification deals with the problem of determining who is who. It aims to provide mechanisms to describe principals involved in certain activities. Typically, identification is meaningful in interactions involving two or more parties. The identity information manifests its importance when it is communicated to other parties. Following the validation of the information, proper activities or services are performed.

Identification happens frequently in daily life. For instance, flight passengers present photo IDs to TSA officials to prove their identity. Bank customers use username and password as identity to log into online banking systems. In the past decades, the significant advances in communication, networking, internet and related services have greatly facilitated the dissemination of information, and transform various aspects of identification. Especially, with high penetration of solid-state sensors into electronic products, it has become convenient to collect and store informative behavior data about ordinary users. The data provides entropy in distinguishing the users. On another hand, the last decade has also observed enormous progress in machine learning techniques. New methods and tools enable effective learning of valuable information from ample data. The combination of the development in both areas presents novel

opportunities for behavior-based user identification.

The behavior data enrich the representation of identity. Traditional representation is some deterministic information only owned by a user, such as date of birth, password and driver's license. In contrast, behavioral representation of identity is typically probabilistic. The information is usually imprecise, requiring more sophisticated modeling methods to represent the information. It needs proper processing of the raw sensory data. One common way is to extract statistical features from the data. This compression makes the information robust to noise and more effective for distinguishing one principal from another.

The acquisition of behavioral identity is enabled by various types of sensors. These sensors are deployed to either devices directly used by users such as smartphone and tablet, or devices in a user's living environment such as thermostats and motion detectors. The sensor measurements take a snapshot of user behaviors with respect to certain life routines such as using an app on smartphone or walking in an office. Collecting the raw data is the first step to obtain user identity. Following the data collection, data processing and statistical modeling are necessary to establish identity from the raw data, because behaviors are inherently probabilistic and the way sensors record measurements typically introduces noise. The same argument also applies to the validation process.

The validation of behavioral identity may require sophisticated methods as the information is typically non-deterministic. Under this circumstance, simple matching does not work. Naturally, the validation methods match with the acquisition process that establishes the identity. In this sense, they can correspond to the training and test routines of a statistical model, respectively. The validation output is therefore probabilistic in nature due to the methods employed. This property presents constraints on the application of behavior-based identification.

Behavioral identity expands the horizon of the application of identification. Many application opportunities are otherwise impossible. Due to privacy concerns, explicit identities are seldom used in many online services. Moreover, exact identification with full accuracy is unnecessary for some applications. In other words, probabilistic identification is already highly desirable. In these scenarios, behavior-based identification sees its unique role. Under many circumstances, to fulfill the functionality of a service, data collection about user behaviors is inevitable, guaranteeing the availability of behavior sensory data for further analysis.

2.2 Overall Design

User identification involves collecting user-specific identifiable information and validating it against established knowledge. There are several facets of identification to be systematically investigated in our design, including purpose, practice, performance, principle and particulars. The purpose refers to the goals of performing identification, which can be benign or malicious. Benign purposes include authentication, personalization and user targeting, while adversarial intent is usually related to user privacy. The thesis will study our approach in both benign and malicious applications. The practice refers to aforementioned three representations of identity: something you know, somthing you have, and something you are. We propose to use behavioral biometrics that are desirable for many application scenarios. Performance of identification is typically measured by accuracy. Additionally, we want to evaluate the complexity and overhead of the proposed approach. In the design and implementation of identification mechanisms, there are several key principles to follow with regard to security, usability and privacy. Our case studies will manifest the principles and give solid examples. Last, identification does not exist apart from specific application scenarios. The particulars refer to application requirements and constraints. These five facets of identification will be manifest in two application scenarios to be present. Though identification is specific to application, we begin with a generic framework and general discussion about design and implementation of each component within the framework.

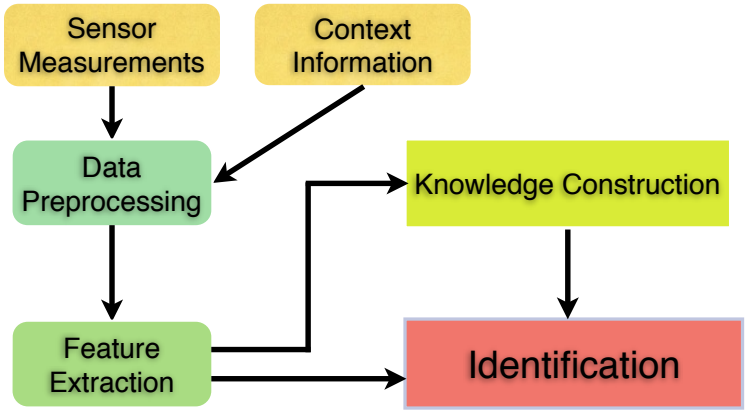


Figure 2.1: We show the framework for behavior-based user identification through passive sensing.

Figure 2.1 illustrates our framework for behavior-based user identification through passive sensing. We discriminate users through usage behaviors. This framework contains indispensable components including behavior measurement collection, data preparation and algorithmic inference. In what follows, we describe these components and

necessary steps, and discuss general principles and practical considerations for design and implementation of the system.

To model user behaviors, the flow begins with collecting behavior measurements and auxiliary context information. Our work concentrates on the area of mobile and IoT application scenarios where a rich set of sensors are employed to sense the environment and people within, creating ample measurements about user behaviors. To achieve satisfactory performance, we need to collect *discriminative* measurements that can help well describe and model usage patterns. It is important that the measurements convey sufficient information and entropy to distinguish between different users.

In real deployment, we have to consider *availability* of desired measurements, *cost* of implementation and runtime *overhead*. In what follows, we discuss the three aspects in behavior-based identification.

Availability limits our access to certain sensor data. On some mobile devices certain sensors may not exist. In this case, we are unable to collect desired sensor measurements. Moreover, sensor accessibility may require user acknowledgement. It is preferred to use sensor permissions that are inherently granted to the app for its normal functionality. If the app requires additional sensor permissions for the purpose of user recognition, privacy-cautious users may decline it. For example, many apps provide location-based services which inherently require GPS data to enable the services. Besides the normal usage of the location data, the apps can also leverage the data for user identification. On another hand, some other apps do not require GPS information to fulfill their functionality. Requiring unnecessary sensor permissions raise privacy ref flag for users.

Cost of implementation is another factor to consider. Collecting certain measurements can be demanding and thus less desirable. We are inclined to use measurements that are inherently used by mobile or IoT systems for normal functionality, or additional measurements that can be easily collected.

Last, we need to bear in mind runtime overhead of measurement collection. It is supposed to be non-invasive. We prefer it not to interfere with daily use of mobile or IoT systems.

In the phase of data preparation, raw measurements are processed into formatted data that is subsequently fed into machine learning algorithms. The preparation consists of preprocessing and feature extraction.

In raw measurements, there might be missing values or corrupted data. Based on domain knowledge and data inspection, we can clean our dataset to facilitate further steps. To give an example, some mobile devices either do not have certain sensors, and simulate fake values or return constant values in response to queries for the sensory

data. To prevent the values from disturbing the inference algorithm used in our approach, we may want to explicitly assign values to indicate the case. Some IoT devices sacrifice accuracy for lower cost. Measurements are thus prone to errors. It is important to employ algorithmic methods to suppress errors such as missing values, calibration offset and hardware noise.

Moreover, simple data formatting and calibration can be performed. For instance, data from devices of different manufacturers can introduce inherent differences. With contextual knowledge about how different devices measure and format data, we perform calibration to make data from different sources consistent.

After preprocessing, we want to extract useful features from numeric data to model user behaviors. To gain sufficient entropy for distinguishing different users, it is desirable to use a combination of ample features that describe a variety of aspects of behaviors, hoping each feature can contribute some entropy for identification.

For a biometric to be an identifiable behavioral characteristic, it is required to have *universality*, meaning every user has this characteristic. It also has to have *distinctiveness* so that users are distinguishable in terms of the characteristic. Moreover, it needs to bear *permanence*: the feature is stable over a period of time. In specific applications, we need to bear in mind these properties to engineer valuable features.

These numeric and categorical features can be used as identity of users. The framework first leverage the features and other context information to establish knowledge about user identity which will subsequently be compared to new data for identification. To fulfill this procedure, we need to make use of adequate algorithms for modeling and learning.

The proposed framework can have promising application in the mobile and IoT realm because of prominent sensing capability in relative applications. We investigate two typical applications of behavior-based user identification through passive sensing. Both are brand new scenarios that represent novel research problems. They can demonstrate practical implications of the proposed approach and, additionally, cast insight into related study in mobile and IoT realm. Through the two scenarios, we seek to solidify our understandings of the approach in general and examine specific challenges and corresponding solutions in these scenarios.

2.3 Application Scenarios

Behavior-based user identification can potentially have many applications in real-world scenarios. The applications can be benign or malicious. Benign scenarios concentrates in the area of authentication, authorization and security protection, while malicious ones typically involves privacy violation such as identity theft.

2.3.1 Benign Application

For benign application scenarios, the behavior-based approach has several advantages. First, it requires no user effort and no dedicated hardware. Users need not to explicitly identify themselves to the systems. Passive sensing enables identity acquisition without user involvement. Second, behavior-based approach preserves user privacy. There is no binding to explicit identifiers. User behavioral patterns are specific to unique context. Lastly, behavior patterns are reliable. It is difficult to mimic behaviors. Faking behaviors goes against normal activities and usage.

2.3.2 Adversarial Application

For malicious applications, the behavior-based approach may have some appealing advantages. First, it brings unique opportunities for identification. Explicit identities are usually hard to obtain. Behaviors are constantly exhibited and exposed to observation. Second, it enables stealthy identification without user's attention. Users are less cautious about behavioral biometrics. Many sensing applications are themselves benign and can hardly raise privacy red flag, preventing cautions from users.

Chapter 3

User Recognition Across Mobile Devices

3.1 Application Scenario

The last decade has witnessed the emergence and rapid spread of mobile electronic devices such as smartphones and tablets. These new technologies have continually worked their way into various dimensions of our individual lives. As the diversity and quantity of networked consumer products continue to grow, more and more customers are becoming multi-device users [28]. According to market research, 31% of US adults own both smartphones and tablets [5], UK households now own on average three different types of Internet-enabled devices [68], and 54% of tablet users share their device with other users [39]. From such studies, we see that (1) users switch between or simultaneously use more than one device and (2) devices are shared among multiple users.

The complexity of multi-device, multi-user interaction presents significant challenges to service providers that provide value-added services by collecting and analyzing user behavioral data, namely because there are no effective techniques to properly label the data with the active user of the device. If the service provider had a mechanism by which *to differentiate between multiple users on the same device* and *to match the app user across different devices*, however, this *user recognition* capability would provide significant improvements to advertising services, recommendation systems, and general user experience.

In the multi-device, multi-user scenario, we define a *session* as a single visit to the app by one user on one device. In the presence of device sharing, the app might observe sessions of multiple users on a shared device. By user recog-

nition, an app vendor or service provider aims to attribute sessions on different devices to the same user. In essence, the app back-end system reveals underlying linkage of sessions on devices and their users. The system can partition the set of sessions of the app into groups, where each group includes the set of sessions involving one user during a particular time period of interest. Depending on whether the partitioning is conducted on a single device, the problem is further categorized as: (1) multi-user differentiation on the same device, and (2) user recognition across multiple devices. For the first problem, classification of multiple users sharing the same device can be approached using identification mechanisms proposed for the single-device scenario [10, 36, 16, 92]. Hence, to avoid reinventing the wheel, we concentrate our efforts on the second problem of cross-device recognition, assuming users of a single device are already segmented and can be treated individually. An example of the resulting cross-device user recognition concept is illustrated in Figure 3.1.

As a motivating example of cross-device recognition, consider a scenario in which a user browses the Yelp app on each of their mobile devices. For the sake of convenience and privacy, the user prefers not to log in to Yelp, since this is not forced by the app. In this case, Yelp cannot recognize the browsing activity as belonging to the same user across the multiple devices, and it misses an opportunity to gain valuable insight into the user's preferences and tendencies. At the same time, the user misses out on the potential added value that Yelp could provide in terms of personalized recommendations. In this case the user's choice not to log in negatively affects both parties' utility. However, Yelp may still be interested in finding ways to provide notions of personalization that do not impede on users' personal privacy, providing a win-win solution.

Based on the complexity of the multi-device landscape and varied user privacy and usability preferences, our primary goal in this work is to enable approximate user-session partitions in a way that provides added value to app vendors and service providers while also considering privacy and usability concerns of device users. Toward this goal, we begin by outlining the current state of the art in recognizing users in web and mobile systems.

Traditional approaches for user recognition rely on either user authentication or explicit tracking. On a single device, websites and apps may employ cookies to track a user's online activity [2, 89]. However, locally stored cookies cannot span devices to enable recognition in the cross-device scenario [67]. Native system identifiers such as Android's Advertising-ID and iOS's IDFA are limited to usage by different apps on a single device instead of cross-device tracking [72]. In addition, a system-level solution cannot traverse different mobile ecosystems (Android,

iOS and so forth). Moreover, such solutions, like accessing user accounts saved locally on each device, bring severe privacy concerns of revealing the explicit identity to a third party. Even worse for the app, relying on device accounts may lead the provider to erroneously label the owner of the device as the user of the application, failing to realize that devices are shared by multiple users. Similarly, privacy-conscious users may opt to use different accounts on different devices, for example to keep their personal and corporate data separate to comply with policies. Hence, these current approaches are insufficient, and in many cases, the app will fail to recognize users across multiple devices.

To address the current limitation to cross-device attribution of user activities, we propose a behavior-based user recognition framework named *XRec* for the multi-device scenario. Our approach leverages behavioral data, namely where, when and how a user interacts with the app. XRec relies on several behavioral characteristics of users, which can collectively serve as an effective yet potentially anonymous way to link user sessions across devices. The behavioral characteristics include location (GPS readings, IP addresses), app-specific statistics, and touchscreen gesture patterns. XRec approaches this problem in two rounds of partitioning. First, XRec creates a course-grained partition of devices into small groups using location information. The intuition lays in the fact that devices of one user are likely to consistently collocate with each other. After this coarse-grained grouping, a user's devices are likely to be mixed with devices of family, friends and colleagues. Subsequently, within each group, XRec performs pairing based on behavior patterns such as touchscreen gestures and browsing behaviors, then refines the set of pairings into a proper partition.

The behavior-based approach has two advantages. First, *it requires no user effort beyond normal app usage*. Users can use apps as usual without explicitly identifying themselves. Second, *it is privacy-preserving as behaviors are bound to neither real identities nor explicit identifiers*. Also, the behavioral patterns are specific to certain context, and can be used for identification only in particular applications. Hence, even leakage of the information does not expose user identity.

Our cross-device recognition approach distinguishes itself from previous behavior-based approaches for identification and authentication. Previous approaches have been proposed for identification and authentication on individual devices in which the mobile operating system aims to thwart unauthorized device access. The differences between these problems are significant. First, an operating system agent typically performs user authentication on a single device, while the *app back-end system* performs cross-device recognition in our approach. The two entities have their

unique advantages and constraints in terms of behavior-related measurement collection. For example, the app back-end enjoys the advantage of obtaining app-specific usage patterns. Second, the user authentication is formalized as a binary classification problem with the owner's behaviors as positive instances. In cross-device recognition, however, we are trying to group devices that belong to the same user. The user's real identity is transparent to the app back-end and also of no interest. Hence, the cross-device recognition becomes a set partitioning problem, detailed in Chapter 3.3, that requires some algorithmic innovation to solve. Third, the authentication problem requires high accuracy since it is critical for security and privacy, while our approach to cross-device recognition is intended to add value to non-sensitive services such as advertising and user experience, so high accuracy is desired but not needed.

In our design and implementation of XRec, there are components of measurement collection, feature extraction and algorithmic inference. To present our approach in a more clear and solid fashion, we implement XRec using an Android app named HackerNews Reader [75], which is discussed in more detail in Chapter 3.3.2. Since it is the app back-end that attempts to recognize users, the provider can leverage exclusive usage information it collects through the app.

Summary of our work: In summary, our overall goal in this work is to *develop, test, and evaluate a proof-of-concept system for anonymously recognizing the same user across app usage sessions on multiple mobile devices*. To the best of our knowledge, this is the first attempt for multi-device user recognition without relying on explicit user authentication. Toward this goal, we provide the following contributions.

- We design the XRec protocol for anonymous, behavior-based user recognition across mobile devices based on application usage patterns and sensor measurements.
- We propose a classification-plus-refinement approach to identify user partitions within an observed set of sessions.
- We deploy XRec with an experimental Android app to collect and evaluate real usage data in a proof-of-concept experiment.

Roadmap. The chapter is organized as follows. We first contrast our study with existing work. Next in Chapter 3.3, we describe background information about problem formulation and the experimental app. We then present in Chapter 3.4 our framework for cross-devices user recognition with details regarding technical assumptions, data collection, feature extraction and algorithm design. Chapter 4.7 and 3.6 present our experiment setup and results. In

Chapter 3.7, we conclude and discuss future directions.

3.2 Related Work

Incorporation of sensors into commodity mobile devices enables passive sensing for a variety of purposes, among which enormous research efforts have been devoted into device-based user identification. A device leverages sensory data to model users and distinguishes its owner from other users or thieves.

Researchers propose to use biometrics for user identification. Biometric approaches have been studies for many years, but only consider *user authentication on single device* rather than multiple devices. Biometrics are categorized as physiological such as fingerprint and iris, and behavioral such as gait, keystroke, touch and swipe [61]. Behavioral biometrics have the advantage of requiring no dedicated hardware [73]. Feng et al. [34] introduced a touchscreen-based approach that authenticates users through finger gestures. Vu [83] proposed an identification approach by exploiting capacitive touchscreens. Meng et al. [63, 62] focused on touch dynamics and experimented with touch biometrics. Frank et al. [36] and Xu et al. [87] investigated continuous and passive authentication based on how a user touches a screen. Jain et al. [47] developed an authentication approach by analyzing a user's behavioral traits modeled by acceleration data, curvature of swipes and finger area. Later, Li et al. [53] designed a mechanism to re-authenticate current users based on finger movements. Draffin et al. [27] modeled users' micro-behavior during their interactions with soft keyboard, including key press position, drift from finger down to finger up and touch pressure. Sae-Bae et al. [79] showed that multi-touch gestures are applicable to user authentication. De et al. [23] collected touchscreen input data and used dynamic time warping for identification. Dey et al. [25] leveraged accelerometer data to extract frequency features to obtain unique fingerprints. Zhu et al. [92] proposed a continuous authentication system that uses various sensors to profile users. Our work investigates user recognition across multiple devices that differs from authentication on a single device. *The problem introduces another dimension (device), and inherently requires multi-class user classification on multiple devices.*

Cross-device patterns received academic attention in the online search domain. Studies of usage patterns (topic of interests, reading behavior, etc.) have examined user behavior in the search engine. Wang et al. [84] examined cross-device task continuation from PC to smartphone for a particular sets of search tasks. They used topics and transition time to predict whether a search task is the continuation from PC to mobile. Montañez et al. [64] studied

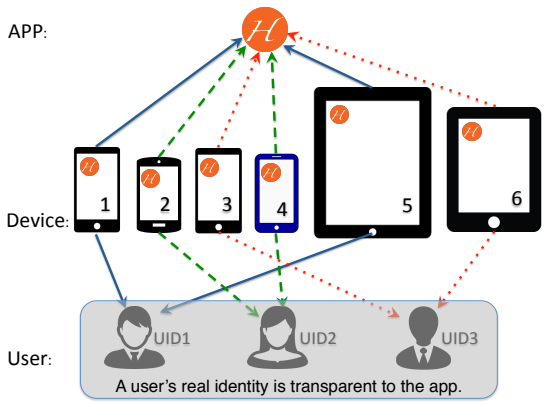


Figure 3.1: We illustrate the scenario of user recognition across devices: associating devices that belong to a same user. Solid lines indicate synced devices whose association is already known to the app while dashed lines present unsynced devices to be associated using XRec. This example assumes no sharing of any device by multiple users. In the presence of sharing, the recognition is performed at the granularity of sessions instead of devcies.

search behaviors and device transition features to predict cross-device search transitions. Karlson et al. [49] analyzed the usage log of desktops and mobile phones to understand patterns how people transition between devices. Our work investigates app usage behaviors across mobile devices. We believe there are still many research questions regarding how users and information transition between multiple devices.

3.3 Problem Definition

We formally define our problem through our example scenario, and highlight its distinction in contrast with the problem of user identification on a single device. We also introduce the experimental app for this study to assist with understanding of the problem and our approach.

3.3.1 Problem Statement

In our problem, an app's back-end aims to recognize the same users across multiple devices. The example scenario is illustrated in Figure 3.1. Each user uses the app on two devices as indicated by the lines. We approach the problem from the perspective that users are transparent to the app back-end. The back-end can only observe those devices and user interactions with the app. By recognizing users across devices, the app does not attribute a device to any user in reality. Instead, it essentially links devices together based on its inference whether they belong to the same user. In this sense, the output of our recognition algorithm is a *partition* of the set of devices, where a partition of a set is a

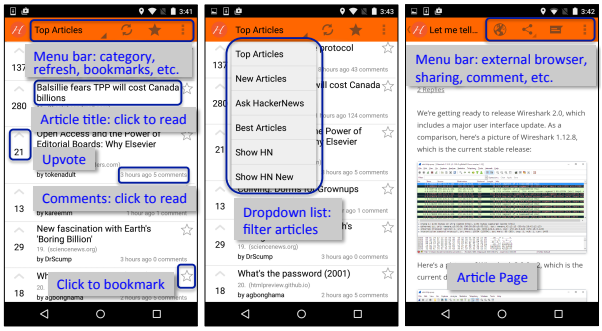


Figure 3.2: Our experimental app is a news reader with diverse functionality including article filtering and bookmarking. It thus enables ample interaction with users, creating opportunities of obtaining effective behavioral patterns. The app also demonstrates key design components of mobile apps such as the list and menu layout, which helps generalize our ideas to many other apps.

division of the set's elements into non-empty subsets, each corresponding to a unique user. In the example scenario, the correct partition is $\{\{1, 5\}, \{2, 4\}, \{3, 6\}\}$.

For all devices, we further categorize them as *synced* and *unsynced*. A device is regarded as synced if it is linked to at least one of the other devices through "out-of-band" information. The information is usually user logins as some users may log in to the app on those devices. Otherwise, in the extreme case that no user opt to log in to the app, app developers may recruit people to use the app on multiple devices to manually create synced devices. In practice, those synced devices provide training samples for our algorithms. In Figure 3.1, device 1 and 5 are synced possibly because UID1 logs in on both devices while other devices are unsynced since users opt not to log in. If UID1 only logs in on device 1, then both 1 and 5 are unsynced. We have to notice that synced devices might be further linked to other devices. In reality, it is possible that a user logs in only on some devices or uses different accounts.

3.3.2 The Experimental App

Some previously published works study continuous authentication on single devices through touchscreen analytics. To collect measurements from users, they employ their own experimental app with particularly designed user interfaces. For example, in *touchalytics* [36], experiment subjects are instructed to use a simple image comparison app. They have to move the screen content through some touchscreen gestures in order to navigate between images. In another work by Xu et al. [87], the authors design an experimental app that instructs participants to perform pre-defined operations such as typing in a sentence using the soft-keyboard and writing down a character on the screen.

In our work, we instead *make use of a real-world app to avoid constraints in controlled settings*. Moreover, the

touchscreen gestures consequently incorporate contextual information of the design and content of the app. That means we can analyze the gestures with respect to the app context instead of purely analyzing isolated gestures. For instance, a user's click locations may exhibit certain patterns due to the layout of the user interface.

To this end, we create a custom-instrumented version of the open-source HackerNews app [75] which is a popular mobile client for browsing YC Hacker news [65]. This app manifests typical design principles of mobile apps on the market. For instance, it employs the list-plus-menu-bar layout that is especially suitable for content display and page transition on small-screen devices. As shown in Figure 3.2, the app lists news articles for users to browse. Moreover, it enables functionality such as article filtering by category, upvoting/downvoting, sharing with friends and bookmarking. With all the diverse functionality, the app enables ample interaction with users, creating opportunities to identify useful patterns among users for the recognition purpose. Our version of the app measures context and behavior and reports back to our server.

This example can facilitate our explanation of the problem context and the intuition behind our proposed algorithms. Although our exposition is specific to this app, the framework and methodology are not limited to this app; instead the app serves as a proof-of-concept implementation using common design principles of mobile apps. Additionally, we will leverage app-independent features that are not associated with the design of an app. We admittedly notice that one single app cannot be representative of all apps and possible interactions with users. However, with the aforementioned reasons, we believe our approach to be generalizable to a broad class of apps. Again, XRec provides cross-device user recognition as added-value, and the inability to support the requirements simply means the app cannot enjoy the benefit of cross-device recognition brought by our approach and all device users would be treated independently.

3.4 Design and Implementation

In this chapter, we present our design and implementation of XRec with the experimental app. We give examples how user behaviors offer identifiable information and help with cross-device recognition. As shown in Figure 3.3, XRec contains indispensable components of measurement collection, data preparation and algorithmic inference.

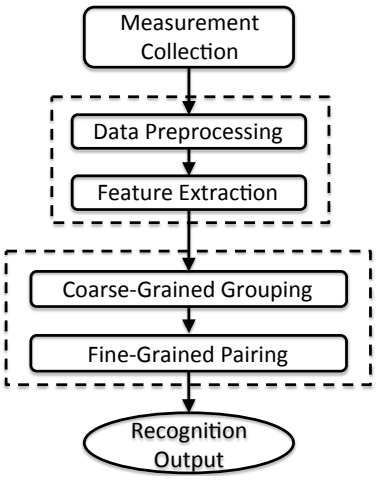


Figure 3.3: This flowchart presents necessary components and steps of XRec for cross-device user recognition.

3.4.1 App Instrumentation and Measurement Collection

We instrument the HackerNews app [75] and set up a server to collect usage-related measurements. Data are first stored locally on the device before being uploaded to the server. To reduce runtime overhead, the instrumented code only runs when the app is open, never in background. Moreover, data are uploaded through WiFi networks to avoid cellular data.

The instrumented code collects timestamped measurements about user actions. To model a user's usage pattern, the app records the user's interactions with the app including clicks, swipes and other actions that lead to status change of the app. Ideally, the sequence of measurements will serve as a behavioral fingerprint that is sufficiently different from that of other users. Since measurement collection relies on what features we attempt to obtain, particulars will be presented in the next subchapter with discussions about potential features.

To obtain ground truth information for evaluating our approach experimentally, we require users to register or provide login credentials the first time the app runs on a new device. In this way, we are aware of which user uses the app on which devices and are able to obtain labeled data for our experiment.

In real deployment, app developers have to consider *availability* of desired measurements, *cost* of implementation, runtime *overhead* and *privacy* concerns [87, 91, 9].

Logs on server and device are easy to obtain. Some sensors may not exist on certain devices. Moreover, sensor accessibility may require user acknowledgement. It is preferred to use sensor permissions that are inherently granted

to the app for its normal functionalities. If the app requires additional sensor permissions for the purpose of user recognition, privacy-cautious users may decline it. For example, Yelp inherently needs GPS data to fulfill its services. The GPS info can be leveraged for user recognition. However, if the app further requests for irrelevant permissions such as contacts, it may raise a red flag. On another hand, user acknowledgement is exempted for some permissions. For example, in Android, many permissions are designated as `protection_normal` such access to coarse-grained location, network state and WiFi state. If an app requests a normal permission in its manifest, the system automatically grants it at install time while users are not prompted to manually grant the permission, and users cannot revoke it.

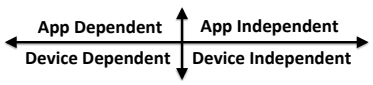
Effort of developers is another factor to consider. Collecting certain measurements can be demanding and less desirable. Preferably, developers are inclined to measurements that are inherently used by the app for its normal functionalities, and measurements that can be easily collected.

Developers need to bear in mind runtime overhead of measurement collection. It is supposed to be non-invasive. We prefer it not to interfere with daily use of the app. Compared to systems that attempt to provide continuous authentication, user recognition is not a frequent service. Measurement collection can thus be disabled most of time. In this sense, any negative effects it may impose to app performance are significantly contained.

Developers may take user privacy into consideration. Collection of any explicit user identifiers is privacy-invasive. The amount and types of data to be measured should not exceed the minimum need for linking multiple devices to a same user instead of identifying any users in the real world.

3.4.2 Feature Preparation

To obtain sufficient entropy for distinguishing different users, we consider a variety of features that cover various dimensions of a user's usage patterns. We hope the features can collectively model user behaviors and facilitate cross-device recognition. Generally, the features fall into different categories as shown below determined by whether the features are stable across apps or devices.



The multi-device scenario presents new challenges since some users might have different behaviors on different devices in terms of certain features. *This requires us to include features from different aspects of user behaviors, each of which contributes some entropy for identification. In the worst case, however, we cannot rule out the possibility that*

some users might act differently on different devices measured by all these features. Then for these users, it is difficult for the app to gain the added-value offered by our approach, which is not favorable but not critical, either. To reduce the possibility of this case, it is important to incorporate device-independent features that maintain permanence across devices, or exhibit relatively constant difference between different devices.

We also categorize features based on their app-dependence. App-dependent features are created with respect to the context of the app. App-independent features do not rely on specific apps, and thus can be directly applied to other apps. Nevertheless, ideas in creating app-dependent features also applies to a broader class of apps.

The (in)dependence across devices or apps can be measured by some statistic metrics such as the Kullback-Leibler divergence. The KL divergence measures how one probability distribution is different from another probability distribution. For numeric features, we can obtain the their empirical distributions on different devices, and compare the distributions to have a quantitative impression of the behavioral difference across devices.

In the following, we discuss several categories of features, and illustrate data from real users using our experimental app to facilitate understandings.

Session Characteristics

We leverage session-level discrepancy among different users. Each session represents a user's activities during a single visit to the app. The activities captured by our instrumented code are all time stamped. We are able to divide the sequence of activities into sessions. For each session, we compute the following statistics.

(*stat-1*). Number of articles read per session

(*stat-2*). Time spent on each article

(*stat-3*). Number of sessions during each hour of a day

In Figure 3.4, we show above session characteristics of three randomly chosen users on two types of devices: Nexus 4 phone and Nexus 7 tablet. Within subfigures, (a-1) and (a-2) plot the CDF of *stat-1*. As it shows, one user tends to read only a few articles per session while another consumes significantly more. They maintain their habit across two devices except for one user who exhibits less consistency. In subfigures (b-1) and (b-2), *stat-2* manifests itself differently for three users: one takes long reading while the other two spend less time. The trend also migrates across devices. Subfigures (c-1) and (c-2) depict preference over 24 hours in a day for using the app on each device.

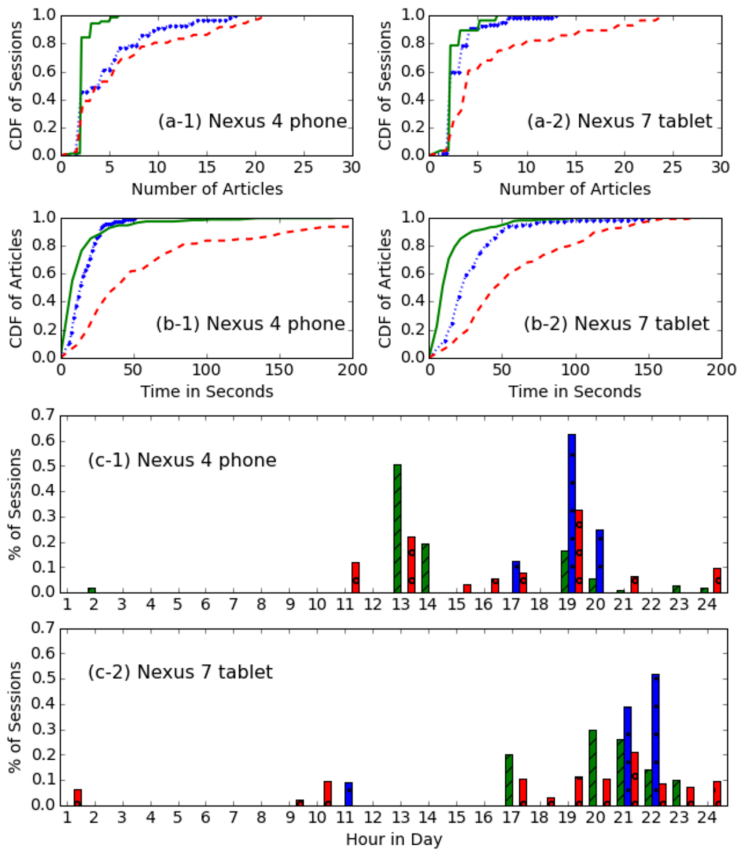


Figure 3.4: We illustrate three session characteristics of three users on two types of devices. Each pair of colored lines (bars) correspond to one characteristic of one user's behaviors on a phone vs. a tablet. They present high-level patterns regarding user behaviors across devices.

Phones gain popularity at noon and evening while tablets are more preferred at night. All session statistics give entropy to distinguish different users on multiple devices.

Clickstream Analysis

We zoom in to investigate clicks within each session. Each click typically triggers certain functionality of the app, and enables navigation within the app. We can leverage differences in click ordering to model usage behaviors. The click sequence helps characterize how a user transitions from one activity to another. Formally, we use a Markov Chain model to analyze click transitions of a user. The Markov transition matrix can be directly used as features for classification.

In this model, each state is an app state, and edges represent transitions between app states triggered by user actions such as touchscreen clicks. Our instrumented code listens on `onClick` events, and records detailed information about

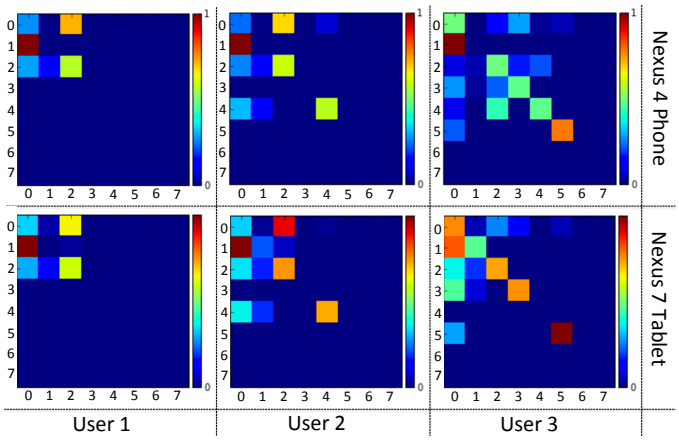


Figure 3.5: To give a straightforward comparison of clickstreams, we visualize transition matrices as heat maps. The numbers 0-7 represent locations within the app such as top articles, new articles and bookmarks.

each click including timestamp and what activities are triggered.

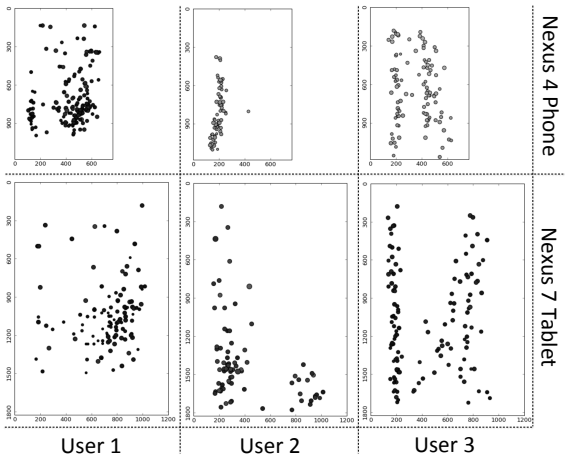
For better comparison, we visualize transition matrices as heat maps in Figure 3.5. Inspecting all subfigures, column-wise they show three users have different transition patterns. Row-wise, the same user maintains similar behavior on both devices. This feature is device-independent for many users.

Clicks and Swipes

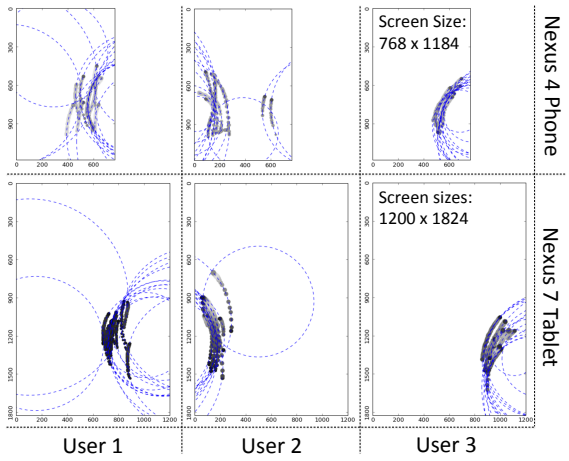
Swipes and clicks, as illustrated in Figure 3.6, are important gestures to interact with apps. In our app, a user has to swipe on the screen to scroll the list of articles, and click on article title to read it. How a user swipes and clicks may bear unique biometric information about individuals. It is desirable to include the information for user recognition.

To support normal usage of a mobile device, when we touch the screen of a device, dedicated hardware automatically generates a set of data and reports them to the operating system as raw events. The system or the app further processes the raw data to understand user gestures and responds with corresponding functionality. Taking Android as an example, the raw event records measurements of *position, size, pressure* and *timestamp* of a screen touch. Position is measured in terms of pixels along two dimensions of the device. It is fine-grained. Size and pressure information are less accurate. Especially for pressure, some devices do not have dedicated sensor for this measurement. They thus report constant value or simulate the value from size measurement. We need to preprocess the data based on domain knowledge and observations to calibrate inherent differences across devices introduced by hardware.

In our app, there are icons for users to click on. Since their positions are fixed, their corresponding click measure-



(a) A tap has position on screen, size and pressure. Each point in above scatter plots represents a tap. Its size visualizes the size of the corresponding tap. The pressure is visualized as darkness. For each user on each device, we randomly and uniformly sample 100 clicks to illustrate their click preference.



(b) A swipe is a series of taps. In the figure, for each user on each device, we randomly and uniformly sample 15 swipes for illustration. We also fit a circle to each swipe to show its radius and direction.

Figure 3.6: We depict users' taps on screen when clicking on article titles, and swipes when scrolling the article list.

ments have negligible variance on position, and are thus less favorable as features. For articles, users scroll the list to find articles of interest, and click on the titles. In this sense, click positions have more variance and entropy for different users. As shown in Figure 3.6(a), users give different click positions. User 1's clicks are more spread on the screen. User 2 might be left-handed due to the imbalance to the left side of the screen. User 3 is equal between left and right.

Users swipe on the screen to browse articles. A swipe is a sequence of taps. Each tap has its position, size, pressure and timestamp. We also fit a circle to each swipe to show its radius and direction. Figure 3.6(b) shows that swipes can be an even stronger indication of handedness; user 2 is most likely left-handed while users 1 and 3 are opposite. Despite that, user 3's swipes have smaller radius and are more concentrated than user 1, which adds further biometric features.

Click and swipe features are relatively sensitive to hardware differences that introduce inherent dissimilarity of measurements across devices. Instead of leaving it fully to our learning model to automatically compensate for the difference, we calibrate some difference beforehand using heuristics and domain knowledge. To give an example, for the pressure and size data, sensors on different devices have distinct sensitivity and measurement reference. With the tap data of the same users on different devices, we can learn the relative difference across devices and normalize the

ID	Feature Description
fc_1, fc_2	The mean of click position in vertical and horizontal directions.
fc_3	The mean of click pressure.
fc_4	The mean of click size.
fc_5, fc_6	The variance of click position in vertical and horizontal directions.
fc_7, fc_8	Ratio of vertical clicks or horizontal clicks to all clicks.
$fc_9, \cdots, fc_{28}$	The index (frequency) of the 10 highest FFT value of click data.
$fc_{29}, fc_{33}, fc_{37}$	The max, min and median values of click position in vertical direction.
$fc_{30}, fc_{34}, fc_{38}$	The max, min and median values of click position in horizontal direction.
$fc_{31}, fc_{35}, fc_{39}$	The max, min and median values of click pressure.
$fc_{32}, fc_{36}, fc_{40}$	The max, min and median values of click size.
fs_1, fs_2	The mean of swipe trace in vertical and horizontal directions.
fs_3	The mean of pressure in swipes.
fs_4	The mean of touch area of taps in swipes.
fs_5, fs_6	The variance of swipe trace in vertical and horizontal directions.
fs_7, fs_8	Ratio of vertical swipe or horizontal swipe actions among all swipe actions.
$fs_9, \cdots, fs_{28}$	The index (frequency) of the 10 highest FFT value of swipe data.
$fs_{29}, fs_{33}, fs_{37}$	The max, min and median of swipe trace in vertical direction
$fs_{30}, fs_{34}, fs_{38}$	The max, min and median of swipe trace in horizontal direction
$fs_{31}, fs_{35}, fs_{39}$	The max, min and median of swipe pressure
$fs_{32}, fs_{36}, fs_{40}$	The max, min and median of swipe touch area
fs_{41}	The angle of moving during swiping
$fs_{42}, fs_{46}, fs_{50}, fs_{54}$	The mean, max, min and median of velocity during swiping in vertical direction
$fs_{43}, fs_{47}, fs_{51}, fs_{55}$	The mean, max, min and median of velocity during swiping in horizontal direction
$fs_{44}, fs_{48}, fs_{52}, fs_{56}$	The mean, max, min and median speeds of swipe pressure change
$fs_{45}, fs_{49}, fs_{53}, fs_{57}$	The mean, max, min and median speeds of swipe touch area change
fs_{58}	The acceleration of moving during swiping in vertical direction
fs_{59}	The acceleration of moving during swiping in horizontal direction
fs_{60}	The mean of swipe radius
fs_{61}	The mean of swipe direction at the start point
fs_{62}	The mean of swipe direction at the stop point
fs_{63}	The mean of swipe latency

Table 3.1: We summarize our features engineered from user clicks and swipes. The features include common statistics and frequency domain components. They capture characteristics of user interacting with our app.

data in terms of mean and variance. For the position of clicks, we create relative values in addition to the deterministic values to facilitate generation of features that focus on relative positions.

To generate click and swipe features, we compute statistics in time and frequency domains. Table 3.1 summarizes the features.

Feature Summary

We extract both numerical and categorical features from aforementioned domains and more, giving a total of 173 features. To give a summary, we compute time domain features such as mean, deviation and frequency domain features including FFT values. Moreover, we include features unique to our study. For example, we divide the screen

into six areas and compute percentage of clicks falling into each area. We employ the transition matrix of clickstreams to take into account app-specific behavioral patterns. We also add a feature to indicate if two devices belong to the same type or model so that our algorithm can compensate for inherent differences across devices.

3.4.3 Recognition Procedure

The multi-device case is formalized as a set partition problem. The recognition procedure comprises 1) *coarse-grained grouping*, and 2) *fine-grained pairing*.

Coarse-Grained Grouping By Overlaps

An app may have millions of installations. We do not directly apply supervised algorithms to all devices at that scale. A wiser way is to first break them down into small groups using rule-based heuristic methods. A common criteria for coarse-grained grouping is *location proximity*. The underlying intuition lays in the fact that devices of the same users are likely to colocate with each other during certain time periods (it is theoretically possible for some users that their devices never appear together, in which case our approach cannot provide added-value for those devices). The colocation can be geographically or in terms of network address. Therefore, overlaps of two devices at physical locations or network segments enable the grouping which can significantly reduce the search space for devices potentially owned by the same users.

We admittedly have no theoretical justification on location-based grouping (which itself can be a serious study similar to the famous 6 degrees of separation [85]), neither experimental study on any large scale dataset from industry as they have legal concerns about publishing relevant results. We have only attempted to survey related literature to obtain understandings about this step. We have investigated papers about location-based user identification [14, 13, 45]. Locations can serve as quasi-identifier of users. With sufficient location information, we are able to uniquely identify a user. Mobile devices colocate with their users. In fact, a main method to measure user locations is through their devices. In this sense, we are able to identify devices with locations. On another hand, we survey papers on locations, geo-clustering and social relations [7, 22]. The finding is locations and social relations interact with each other. Social relations imply location proximity. Hence, devices of the same users or socially close users bear similar proximity, which becomes the foundation for location-based grouping of devices.

An app can obtain location information of devices through GPS, WiFi SSID and IP addresses. The heuristic criteria to pair devices together is that they are frequently within physical proximity, connect to the same WiFi Access Points, or appear on the same IP segments for certain amount of time.

Fine-Grained Pairing By Usage Patterns

The coarse-grained grouping is straightforward yet effective in reducing the search space. After grouping, a user's device is likely to be mixed with devices of family, colleagues and friends since the social relations typically imply physical proximity of people and their devices as well. To further identify devices of the same owners, we leverage distinctions in usage patterns of different users. Since it is formulated as a set partitioning problem, the k-means algorithm might qualify as a candidate. However, it is not suitable for this problem due to two main reasons: 1) k-means requires a moderate sample size and does not perform well with sparse data [77] while our group size is well under 100 and the graph is sparse, and 2) k-means underperforms with high-dimensional data [93] while our features are of dimension 173. To tackle the challenges, we propose a fine-grained pairing method of two steps: *pairwise classification* and *refinement*.

After feature extraction, each device d_i has its feature vector fv_i. For any pair of devices $\langle d_i, d_j \rangle (i \neq j)$, we construct a feature vector fv_{ij} which measures the similarity between behaviors on the two devices. The pairwise feature vector fv_{ij} is defined as the element-wise squared vector difference $||fv_i - fv_j||^2$, which is widely used as a distance and similarity metric. The resulting feature vector is also of dimension 173. Additionally, we add an extra field to indicate whether the two devices are of the same type in terms of phone or tablet so that learning algorithms can take into account inherent difference between devices. With this pairwise setup, the first step is to determine the $(0, 1)$-label for each pair $\langle d_i, d_j \rangle$ to indicate whether they belong to the same user.

XRec employs random forest to classify device pairs. Random forest is an ensemble learning method that constructs many decision trees and outputs the mean prediction of individual trees. It is able to address nonlinear features and inherently performs feature selection. Random forest achieves good robustness over single classification approaches [19, 26]. Moreover, we use regression to yield a mean prediction between 0 and 1, which can serve as a confidence score. Then we can introduce a threshold to generate $(0, 1)$-labels.

The second step after pairwise classification is refinement. Think of each device d_i as a vertex in a graph $\mathcal{G}$, there

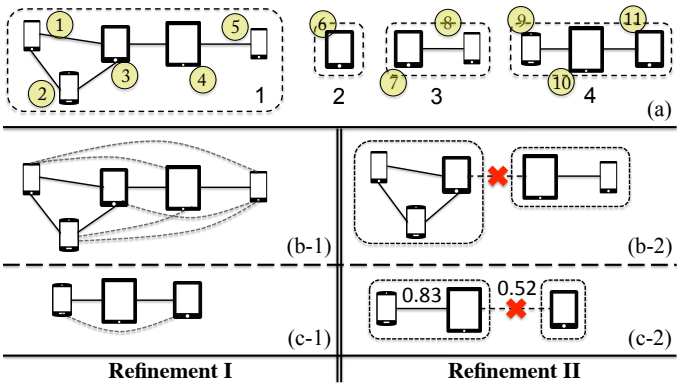


Figure 3.7: We illustrate examples of refinement I and II after pairwise classification, which constructs fully-connected subsets by adding or removing certain edges.

exists an edge between two devices $\langle d_i, d_j \rangle$ if the predicted label for the pair is 1. We can populate all potential edges based on classification results. The graph of devices is thus partitioned into connected subgraphs. A graph is connected if there is a path from any vertex to any other vertex in this graph. As shown in Figure 3.7, (a) illustrates a sample output of pairwise classification. Each subset is expected to by *fully connected* which does not hold for subset 1 and 4. The fully connectivity requirement comes from the fact that all devices in a subset are supposed to belong to one user. Conflict arises when `dev_1` and `dev_2`, `dev_2` and `dev_3` belong to the same user while `dev_1` and `dev_3` do not. Our refinement algorithm updates edges so that all subsets fulfill the fully connectivity requirement. We consider two approaches: 1) *refinement I* – adding edges to connect nodes in a partially connected subset, and 2) *refinement II* – removing edges to partition a subset into several fully connected subsets. Subfigures (b-1) and (c-1) illustrate two examples of the refinement I which is straightforward. The refinement II keeps fully connected subsets and remove extra edges. Subfigure (b-2) presents a simple example: removing one edge partitions the set into two fully connected subsets. A tricky scenario is depicted in (c-2) where we can remove either edge to succeed. In certain cases, we want to delete the edge with lower confidence score.

Refinement II alternatively iterates through two steps: 1) finding the maximum clique in the current graph, and 2) removing the maximum clique and all associated edges from the graph. In the presence of multiple maximum cliques, we choose one such that the associated edges to be removed have lowest confidence scores. To find the maximum clique, we use existing maximum clique algorithms [17]. Readers may refer to Algorithm 1 for details.

Refinement II works with the binary classification to achieve satisfying recognition performance. In the binary classification phase, we can tune parameters to intentionally increase recall (namely, predict more edges in the graph).

Algorithm 1: Refinement Algorithm II — Reduction

Input: Adjacency matrix $\mathcal{A}_{n\times n}$, confidence score $\mathcal{S}_{n\times n}$
Output: New adjacency matrix $\mathcal{A}'_{n\times n}$
Procedure REFINEMENT
$\mathcal{A}'_{n\times n} = \mathcal{A}_{n\times n}$;
while $\mathcal{A}_{n\times n} \neq$ ***[0]*** **do**
    $cliques$ = FindMaximumClique($\mathcal{A}_{n\times n}$);
    if $cliques.size() < 2$ **then**
        break;
    end
    $clique = \emptyset$;
    $edgesToRemove = \emptyset$;
    $minScore = +\infty$;
    for c ***in*** $cliques$ **do**
        $victims$ = FindEdgesToRemove($\mathcal{A}_{n\times n}, c$);
        $scores$ = GetScores($\mathcal{S}_{n\times n}, victims$);
        if $scores < minScore$ **then**
            $edgesToRemove = victims$;
            $minScore = scores$;
            $clique = c$;
        end
    end
    RemoveEdges($\mathcal{A}'_{n\times n}, edgesToRemove$);
    RemoveClique($\mathcal{A}_{n\times n}, clique$);
end
return $\mathcal{A}'_{n\times n}$;

Then, in the refinement stage, the algorithm removes edges in accordance with the topological constraint that each clique should be fully connected. The topological constraint provides extra information in addition to behavioral patterns on different devices. Most of the removed edges are false positives. Later in Section **??**, we present an illustrative example and experimental results to show the effect of refinement.

3.5 Experiment Setup

In this chapter, we present our experiment setup. The experiment concentrates on fine-grained pairing using real usage data, obtained under IRB approval.

3.5.1 Data Collection

In this cross-device recognition study, each subject in our study uses the experimental app on four types of devices: Nexus S, Nexus 4, Nexus 7-2012 and Nexus 7-2013. For each type, a subject is required to use the app on two devices of that type. In total, we obtain usage data of each subject on eight devices. Though it is rare for a real user to own

eight devices, with this setup we do not actually mean for them to have eight devices. We can opt to use part of the data to form different scenarios. Four pairs of different devices allow us to study usage behaviors of each subject on 1) identical devices, 2) same type but different models (Nexus S phone *vs.* Nexus 4 phone), and 3) different types of devices (Nexus 4 phone *vs.* Nexus 7 tablet). Also, with eight sets of usage data per subject, we have the flexibility of evaluating the problem for cases of various numbers of devices per user from one to eight. Moreover, with this setup we can test the performance of our approach in extreme case and also obtain insights of user behaviors on different types of devices.

In addition to usage behaviors of each subject across eight devices, we also need to study the variation of behaviors on the same devices by different subjects. To this end, we recruit twenty subjects from different demographic groups to participate in the data collection, giving essentially 160 sets of usage data in terms of the $\langle user, device \rangle$ pair. With the data, we are able to study inter-user, intra-user, inter-device and intra-device behavioral similarity or difference. Moreover, per our discussion in Chapter 3.4.3, the coarse-grained grouping can significantly narrow down the cross-device recognition to within relatively small groups. From the 160 sets of usage data, we can effectively construct such groups for studying fine-grained pairing within the groups. That said, we admittedly note that it is favorable to include more devices and participants into this user study. However, our effort is limited by the constraints of time and hardware. We do not have dozens of mobile devices for simultaneous data collection from many participants, and collecting data on eight devices for even one user is already time-demanding. In our work, we hope to deliver proof-of-concept results on this new cross-device recognition problem, which we believe can be fulfilled with the 160 sets of usage data.

After we distribute devices to a subject for data collection, we do not give specific instructions to use the experimental app other than a basic introduction of the app. They use the app in accordance to their own preference and habit. In this fashion, we hope the subjects exhibit their own patterns in the usage that are not biased by any particular recipe. For the twenty subjects, we have collect usage data of effectively two to ten weeks, including hundreds to thousands of clicks and swipes.

Last, due to limited resources, the device models we use are only a small portion of all models on market. The four device types in our experiment consist of phones and tablets of different models, sizes and hardware. Although they are not completely representative of all devices, they present differences we hope to have and deliver meaningful

results.

3.5.2 Experimental Data Generation

Without access to large-scale data that a commercial app provider would collect, we have attempted to approximate various scenarios. We collected usage data of real users using our app on real devices. However, we are not able to attract a large number of users to study the coarse-grained grouping step. Therefore, we simulate probable grouping outputs using the 160 sets of usage data. There are three primary parameters describing a group: 1) total number of devices of the group, 2) number of devices per user, which determines the number of underlying users given the total number of devices, and 3) diversity of devices which defines the number of different types of devices. With respect to the set partitioning problem, the first parameter specifies the size of the set to be partitioned. The second parameter tells the number of clusters. The third parameter describes diversity which affects the similarity and difference between behaviors on devices within a group.

Our data allows us the flexibility to simulate various groups of different parameter settings. With 160 sets of $\langle user, device \rangle$ data, we can simulate a group of up to 160 devices with its correct partition being twenty subsets each of eight devices, which might be rare but can test some properties about our approach such as scalability. The size will be smaller considering the withheld training data in our supervised approach.

We will conduct several experiments in the next chapter. Without otherwise stated, we use 80 devices from ten users as training data, and the remaining for testing. Additionally, the results are averaged over ten different sets of training-test split. Some other details about the setup will be presented along with results in the next chapter. The training phase essentially models similarities or differences of behaviors on 80 devices, which are the *distances* between behaviors represented as feature vectors instead of isolated behaviors on individual devices. In training, we include all eight devices of ten users to obtain sufficient positive and negative instances of device pairs. A positive instance refers to the case that two devices belong to the same user, while a negative instance denotes the opposite. Having two devices of 100 users can also help achieve the goal of having sufficient positive and negative instances. Unfortunately, we are not able to perform a study at that scale. Though it might be difficult to find real users with eight devices, the setting does not undermine practical implications of this study. Having eight device from each user gives no advantages to modeling. It is more about having enough positive instances with less users under our experimental

settings. Additionally, we will evaluate XRec against different training settings where we have less training instances.

3.6 Performance Evaluation

In this chapter, we evaluate our approach in terms of its recognition performance. We aim to provide insights by investigating the following questions:

- What is the overall performance of XRec?
- How does the group size affect recognition results?
- How does the recognition performance change with the number of devices per user on average?
- How well our approach can perform in presence of various mixture of multiple different types of devices?
- What is the generalization capability in terms of users, i.e. can XRec generalize from a few users to many others?
- What is the generalization capability in terms of devices, i.e. can XRec generalize from a few devices to other devices?

3.6.1 Performance Metrics

Recall from previous discussion, the user recognition problem can be formalized as *set partition*. To evaluate XRec, we need to compute the distance between the estimated partition $\hat{\rho}$ and the correct partition ρ. Denoeud et al. [24] compared several distance metrics between set partitions such as the Rand index, the Jaccard index, the corrected Rand index, the Wallace index and the normalized Lerman index. We employ two widely used metrics for partitioning evaluation: the Jaccard and Rand index.

To compute the *Jaccard index* between $\hat{\rho}$ and ρ, we need to evaluate on all pairs of elements in the set. For any pair $\langle x, y \rangle$, they can be joined or separated in each partition: joined means they are labeled as 1 while separated defines the opposite. We denote as r the number of pairs simultaneously joined in both partition, s the number of pairs simultaneously separated, u the number of pairs joined in $\hat{\rho}$ and separated in ρ, and v the number of pairs separated in $\hat{\rho}$ and joined in ρ. Then the Jaccard index of the two partition is defined as:

$$\mathbb{J}(\rho, \hat{\rho}) = \frac{r}{r + u + v}.$$

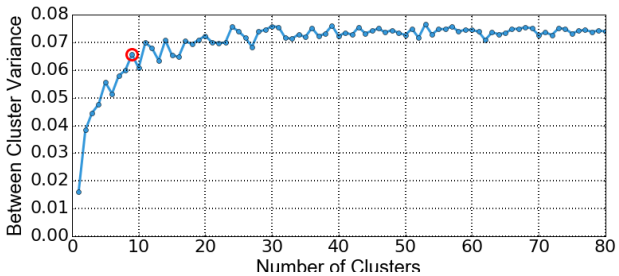


Figure 3.8: We illustrate the elbow method used in k-means algorithm when the number of clusters is not known. It examines the percentage of variance explained, and identifies the 'elbow' point indicated by the red circle.

The Jaccard index omits s since its inventor believes partition is typically interpreted as joining elements. In our evaluation, we may also want to reward correct separations. The *Rand index* gives equal emphasis on simultaneously joined or separated pairs. We also assess our approach using this index to offer another angle of perspective. Rand index is defined as:

$$\mathbb{R}(\rho, \hat{\rho}) = \frac{r+s}{r+s+u+v}.$$

The evaluation metrics on group partitioning is in terms of pairs within the group. From the perspective of pairwise binary classification, the aforementioned r, s, u, v are respectively true positive (tp), true negative (tn), false positive (fp) and false negative (fn). In this sense, the Rand index is essentially the *accuracy*. To help understand the effect of our refinement algorithm, we also use metrics defined as follows:

$$precision = \frac{r}{r+u} = \frac{tp}{tp+fp}, \quad recall = \frac{r}{r+v} = \frac{tp}{tp+fn}.$$

3.6.2 Recognition Performance At A Glance

First, we evaluate on all test data of 80 devices from ten users. Even though a group of 80 devices might be rare, we hope the result can give an overall impression on the performance, and also demonstrate the scalability of the algorithm. We also compare the performance with several baseline approaches. Moreover, we present analytical results to highlight the effect of the refinement algorithm.

Baseline Approaches

Our approach is compared to three baseline approaches including: 1) a naive baseline method (`Naive`); 2) k-means algorithm (`Kmeans`); and 3) a heuristic method (`Heuristic`).

The `Naive` baseline randomly guesses (0,1)-labels for all pairs during the binary classification phase and employs our refinement algorithm I and II. The `Kmeans` approach incorporates the elbow method [82], which is used to determine the number of clusters in k-means. As shown in Figure 3.8, the method plots the percentage of variance explained by the clusters against the number of clusters. The 'elbow' point is indicated by the red circle. The `Heuristic` method trains a distance threshold on labeled data that maximizes the Jaccard value. Then the distance is used to link any pair within the threshold during the binary classification. Refinement methods I and II are applied subsequently.

Both the `Naive` and `Heuristic` methods require refinement since they first make a binary prediction of each pair. The `Kmeans` approach, on the other hand, directly partitions the device set and requires no refinement.

Performance Comparison

As shown in Figure 3.9(a), XRec with refinement II achieves the highest performance by Jaccard index, Rand index and precision. For 80 devices and thus $\binom{80}{2} = 3160$ pairs, XRec achieves 70% Jaccard value, 98% Rand value and precision, and 70% recall. XRec with refinement I obtains high recall value, but loses precision. Refinement I causes low performance for `Naïve`, `Heuristic` and XRec since it blindly increases recall by aggressively adding positive instances, many of which are unfortunately false. On another hand, refinement II leverages a topology constraint and makes appropriate adjustments to the binary classification results. We will present experimental results shortly to further reveal its mechanism.

As also shown in the figure, XRec performs significantly better than all baseline results. Overall, the Rand value is satisfying since there are many more negative instances than positive ones and it is relatively easy to correctly predict the negatives. However, for the baseline approaches that employ refinement I, the Rand values decrease significantly since refinement I predicts many negative instances as positive. The recall values for some baseline approaches are satisfying. The high recall is due to the fact that these methods make sufficient amount of positive predictions so that the true positives are predicted. All baseline methods underperform in terms of Jaccard and precision values.

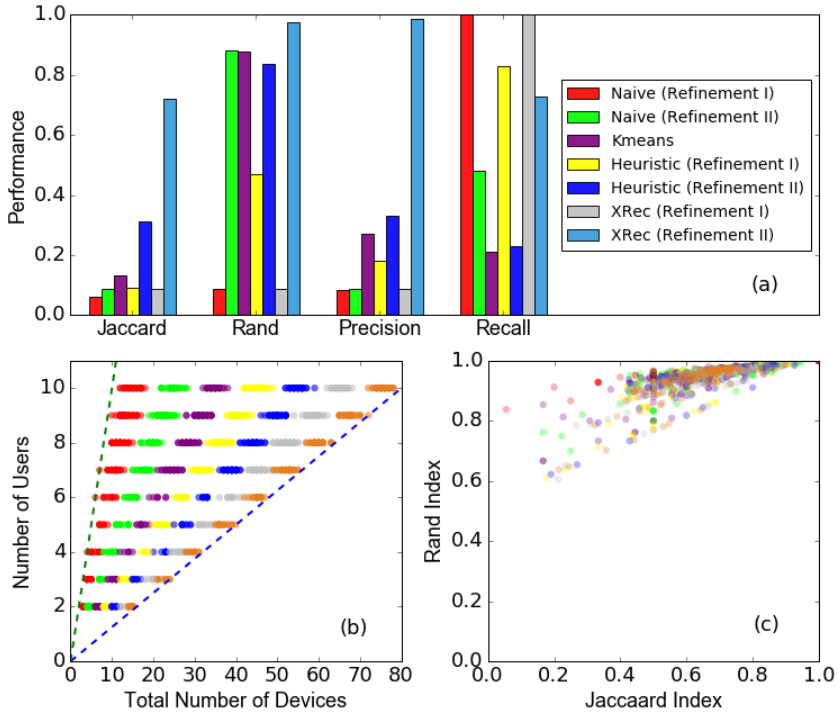


Figure 3.9: We contrast XRec with baseline approach. Refinement I and II are also compared. We further illustrate the performance for potentially different outputs from coarse-grained grouping.

The `Naïve` method makes random guesses and gives a lower bound for the performance. Its precision and Jaccard values are low regardless of the refinement methods employed. `Kmeans` slightly outperforms the `Naïve` method. The `Kmeans` method, as a clustering method in nature, makes use of topological characteristics of the group of devices. However, without knowledge of the underlying number of clusters, the uncertainty compromises its performance. Additionally, the high dimensionality of the feature space makes clustering even more difficult. The `Heuristic` with refinement II achieves about 30% Jaccard value, outperforming other baseline approaches with a Jaccard value below 15%. This method learns a distance threshold to maximize the Jaccard value. The learning process takes into account behavioral similarity between devices and topological characteristics of groups. The excessive amount of information incorporated in the learning process creates sparsity. When the learned model is applied to test data, the performance suffers from the lack of generalizability because the test data might exhibit dissimilar behavioral and topological patterns.

We also want to evaluate XRec on test data of diverse formations representing various group scenarios. To this end, we randomly sample from the whole test data to construct a variety of different groups. A group is defined by the three parameters as discussed before: 1) total number of devices, 2) number of underlying users, and 3) number of different

types of devices. To construct different types of groups, we first sample two to ten users. For each user, we sample one to eight devices. In total, we obtain 2000 sets of test data. Figure 3.9(b) shows two of the three parameters for the 2000 groups. The two dotted lines define possible area for the two parameters as each user has one to eight devices. For each group, we depict its Jaccard and Rand indexes obtained in Figure 3.9(c). As shown in Figure 3.9(c), points concentrate on upper-right corner. The variance of performance comes from not only the difference in group formation but also specific users included in certain groups. If some users in a group have very similar behavioral patterns, the performance will be undermined since our approach relies on behavioral distinctions. In this case, the app simply loses the potential added-value brought by XRec on these users if they happen to be in the same group. On average for all randomly sampled 2000 groups, XRec (refinement II) achieves 91% $\mathbb{R}$-value and 61% $\mathbb{J}$-value. Since XRec relies on behavioral patterns of users and the topological constraint, its performance hinges on various factors. For some specific group scenarios, the performance is far below the average level. It might be caused by various reasons such as: 1) some users in the same group exhibit similar behaviors, 2) the group is small and each misclassification thus incurs high performance loss. To further improve the recognition performance, we may want to introduce new features that capture other aspects of user behaviors. For instance, we can possibly use accelerometer data to reveal user's holding posture. This potential improvement comes with the price of extra overhead since accelerometer has a relatively high sampling rate. Moreover, it requires extra permission to access the data.

The added value brought by XRec can enhance advertising, recommendation and user experience. The working mechanism is similar to cookie-based online tracking. Web cookies track users across websites while XRec tracks users across mobile devices. The primary goal for both is to link the current user to his previous activities. In this sense, XRec works in parallel with the advertising and recommendation systems. XRec makes predictions about the linkage among users on different devices. Then the content viewed by the user on other devices also indicate his interest and can be leveraged for optimization on the current device. The XRec outcome is fed to recommendation algorithms so that they can make use of the history content viewed by the user to recommend new content the user might be interested in. The same procedure applies to advertising. The past trace exposes user interest and there is higher chance that related advertisements being clicked. XRec also offers opportunity for enhancing user experience with the app. For example, a news article left open on another device can be displayed on the current device in case the user may want to continue reading that article. Though the experience optimization process is app specific and

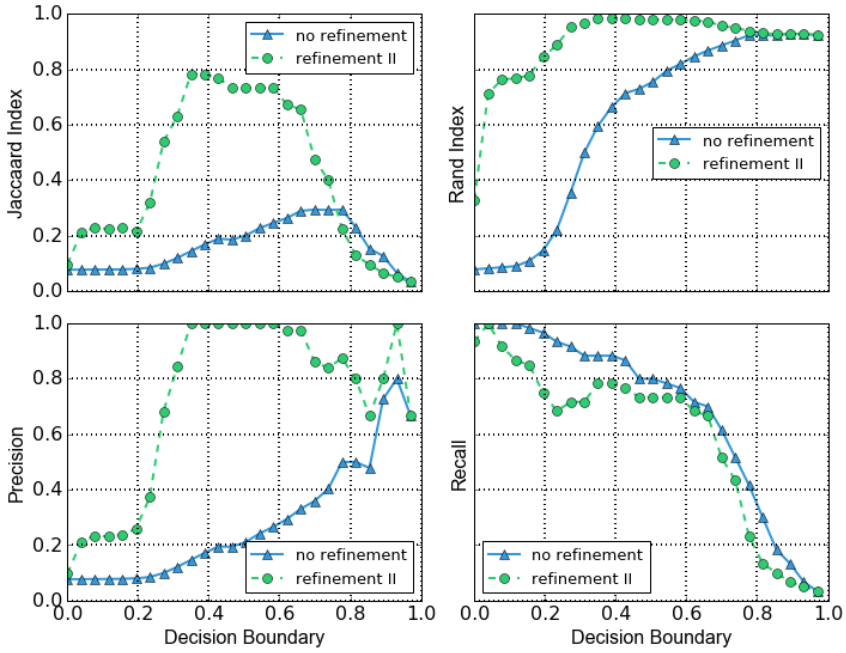


Figure 3.10: We illustrate analytical results under different decision boundaries to demonstrate how refinement boost recognition performance. The refinement after binary classification significantly enhance precision without sacrificing much recall.

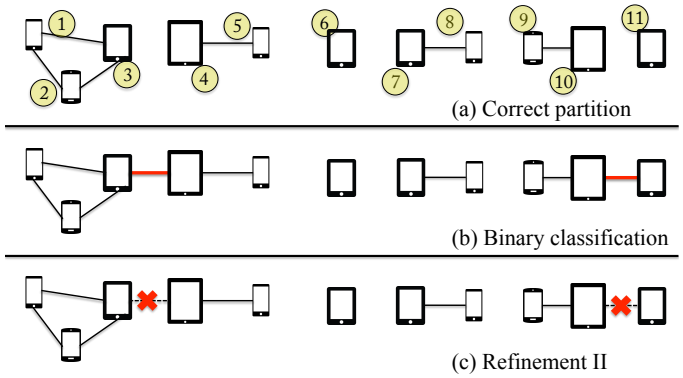


Figure 3.11: We present an illustrative example to show how refinement can improve recognition performance. The binary classification excessively predicts associated pairs, which creates false positives (indicated as red lines) compared to the correct partition. The subsequent refinement procedure selectively removes edges, most of which are false positives.

requires careful design and implementation, the information relating to user identity is generally useful. XRec aims to serve downstream services that favor a notion of user identity to better fulfill their goals.

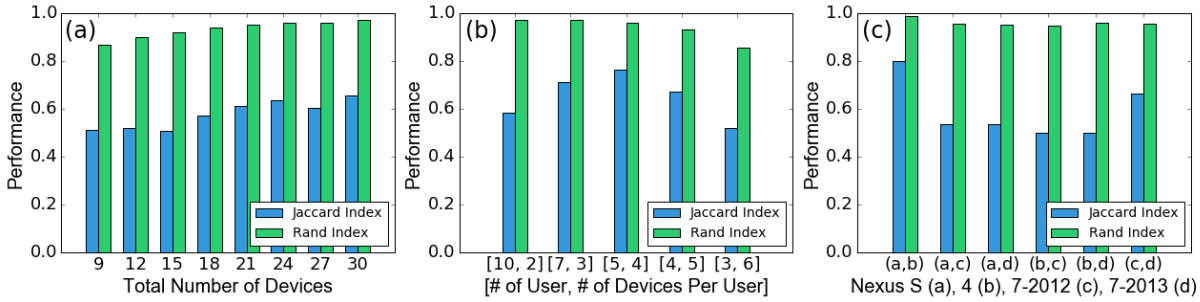


Figure 3.12: We illustrate the performance against changes of three controlling factors, respectively.

Effect of Refinement

We also reveal the working mechanism of the proposed classification-plus-refinement algorithm. The pairwise classification outputs confidence scores from 0 to 1. A boundary is chosen to further produce (0,1)-labels indicating whether a pair of devices belong to the same user. Figure 3.10 shows the effect of the decision boundary on the performance of refinement II measured in four metrics. We use 80 devices as training data and the remaining 80 devices as test data. In terms of all four metrics except recall, we can observe that performance with refinement II is consistently better than the performance without refinement. Refinement II only slightly reduces the recall value by about 5%. By wisely removing positives as shown in Figure 3.7, refinement II boosts precision by 80% without sacrificing much recall. *Recall* evaluates if all true positives get predicted, while *precision* assesses if the predicted positives are true. The balance between recall and precision can be achieved by adjusting the decision boundary in binary classification. As shown in the figure, a suitable decision boundary value from 0.4 to 0.6 gives satisfactory results in our experiment.

Figure 3.11 presents an illustrative example how the refinement based on the topological constraint improves performance. Sub-figure (a) shows the correct partitioning of eleven devices. In the binary classification, we intentionally adjust the decision boundary to predict more positive instances, maintaining high recall value. However, aggressive prediction generates some false positives indicated as red lines in sub-figure (b). Subsequently, the refinement phase removes a substantial amount of false positives by enforcing the topological constraint (namely the fully-connection clique constraint) in the graph.

3.6.3 Three Controlling Parameters

As mentioned above, a group is described with three parameters. We study the impact of different group scenarios on recognition performance. We employ XRec with refinement II, and use the same model trained in the above experiment.

Comparison of Different Group Sizes

Group size is the total number of devices in a group obtained from coarse-grained grouping. Depending on the strictness of the grouping method, physical proximity of devices, population density and other factors, the group size may vary. To generate test data for this problem, we fix the number of devices per user at three: a Nexus 4, a Nexus 7-2012 and a Nexus 7-2013, while the number of users changes from three to ten (the maximum number of users available in test data), giving a range of group size from 9 to 30. We generate nine sets of data per setting except for the number of users being ten, and compute the average performance value.

Figure 3.12(a) plots the results. For most settings, XRec can achieve about 55% $\mathbb{J}$-value and 90% $\mathbb{R}$-index. With the increase of devices, the performance is on a slight upward trend. An explanation is more devices introduce more positive instances which is captured by the Jaccard index. Moreover, the refinement can effectively reduce false positives. The combined effect brings the values slightly higher. Intuitively, a smaller group size can make the partitioning task less challenging. However, it also depends on the dissimilarity of behaviors from different users. If users within a small group happen to share common behavioral patterns. XRec is likely to treat them as the same user. Moreover, when the group size is small, each misclassification can incur higher performance loss.

Number of Partitions

The number of partitions of a group is essentially the number of underlying users. To prepare test data, the group size is fixed at 20 devices. The number of partitions from three to ten. For each number from three to ten, we randomly generate several corresponding partitions. For instance, three implies partitions such as $(8, 8, 4)$, and $(8, 7, 5)$. Moreover, the three users are randomly chosen from all ten users. For each setting, we generate ten sets of test data and obtain average performance.

As shown in Figure 3.12(b), XRec achieves above 50% Jaccard value and 80% Rand value. The case of 5 users

each with 4 devices gives best Jaccard value. Going either side decreases the value. Similarly, the reason might be the number of positive instances in each combination. The true positives can be well-presented by the Jaccard index. Moreover, the refinement in XRec relies on the topological constraint that every clique is fully connected. With 5 users and 4 devices each, the clique size and the number of cliques are medium. A graph with many small clusters is sparse. It is easier for XRec to output small cliques since they require much less mutually connected pairs. For example, an edge alone can generate a two-node clique. The performance is penalized if a small clique is actually incorrect. On another hand, a graph with a few large clusters is dense. In this type of graphs, to output a large clique, XRec needs to correctly predict all pairs within the clique. Missing edges will break the large clique into smaller ones, and incur performance loss. From this perspective, a medium number of cliques of medium size can potentially benefit the partitioning.

Different Combinations of Devices

Our experiment uses four device types. We are interested in possible variance between recognizing users across two phones and across phone-tablet. To prepare test data, we fix group size at twenty: ten users each of two device types.

Figure 3.12(c) shows the results for different combinations. Identifying users across Nexus S and 4 reaches 80% Jaccard value, while it is about 50% for the phone-tablet combinations, and 65% between the two tablets. The recognition across the same type of devices outperforms that across phone-tablet. User behaviors on the same type of devices tend to be similar. The screen size and hardware on different devices might cause subtle changes on some aspects of the behaviors. In this sense, XRec loses some discriminating information from those device-dependent features. The calibration during data preprocessing can help with the situation to some extent. Our recognition is probabilistic in nature. From this perspective, the recognition made for the same type of devices has higher chance to be correct. This observation can be taken into consideration by downstream services when they make use of the XRec outputs.

3.6.4 Different Settings for Training Data

In this chapter, we change our training setting (80 devices from ten users) to investigate how XRec generalizes its knowledge on users and devices. We only use a subset of the original training data. The test data consists of the remaining 80 devices.

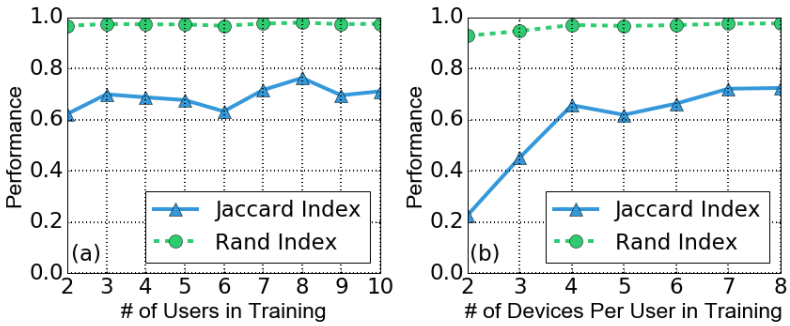


Figure 3.13: We illustrate the performance against changes of two training data parameters, respectively.

Cross-User Generalization

For training, we choose two to ten users each with eight devices. We plot our result in Figure 3.13(a). XRec achieves above 60% $\mathbb{J}$-value and the trend is stable. Even with two users each of eight devices, XRec can capture behavioral differences among users across devices. It reduces the demand for extensive training samples. By the design of XRec, the input feature vector to the binary classification algorithm is the *difference* between behavioral patterns on two devices. The algorithm aims to capture the relative difference instead of absolute patterns, which lowers the importance of learning the patterns of any specific users. In this sense, a few users on multiple devices can offer sufficient instances for XRec to model behavioral differences across devices. This is a favorable advantage even though it is not difficult to recruit more users.

Cross-Device Generalization

We vary the number of devices per user from two to eight with ten users for training and 80 other devices for test. Figure 3.13(b) shows that the $\mathbb{J}$-value initially increases with the number of devices per user and then becomes stable. With two or three devices, XRec might fail to capture behavioral differences of users between enough devices of different types. As we use more devices per user for training, XRec observes more devices and can better model the differences. In this experiment, the test dataset contains all eight devices. It is desirable that XRec can observe instances from all devices in training. If the test data and the training data come from different distributions, the trained model is likely to underperform for the test data. Hence, if we do not have all devices in the training data, we hope the devices at least are different in type so that the algorithm can incorporate this difference into the model. In real practice, XRec favors to include more types of devices into its training dataset so that the optimal performance can be

achieved.

3.7 Discussion

We present the design, implementation and evaluation of XRec for cross-device user recognition. Distinct from existing research work that investigates user identification or authentication by the operating system on an individual device, XRec deals with tracking users across multiple mobile devices by an app provider's back-end. Moreover, previous work typically concentrates on binary identification of owner or non-owner of a single device while our problem inherently demands multi-class recognition of multiple users across multiple devices. To tackle challenges present in this problem, we propose to use coarse-grained grouping based on location proximity plus fine-grained pairing based on usage behaviors. Algorithmically, XRec leverages pairwise classification plus refinement to solve the particular partitioning problem required for fine-grained pairing in this context. We evaluate XRec on 160 sets of usage data on eight devices of four types. For different group scenarios, XRec achieves an average Rand value of 91% and Jaccard value of 61%. These results offer confidence that user behaviors provide entropy for recognition even across device types and form factors. XRec offers the opportunity to optimize many downstream services across devices such as recommendation, advertising and user experience enhancement. XRec provides a notion of identity to these services so that they can track down users' activities across multiple devices. The history activities on other devices also indicate a user's interest and preference. XRec enables the mining of data at the user level instead of at the device level, enriching information available to other services for better performance.

Chapter 4

Room-Level Indoor Localization Through Occupancy Measurements

4.1 Application Scenario

With the advance in sensing technologies and the decrease in cost, power and size of solid-state devices, it is now possible to support ubiquitous sensing and actuation at scale with increasing temporal resolution [78]. During consistent sensing, ample data is collected, stored, and communicated which brings opportunities for a variety of data-driven services and applications [88, 18, 66, 86]. Building management, among all kinds of application scenarios, benefits extensively from such a sensing infrastructure and inspires a variety of useful services regarding heating, cooling, illumination, safety and many other purposes. Residing on top of the sensing layer, service providers, either in-building or cloud-based, can be aware of building status and conditions through sensor measurements, and react properly and automatically to environmental variations, essentially creating a *smart building* [30, 76, 35, 50].

Enabling this level of intelligence, however, inevitably exposes rich information about the building environment as well as its occupants because many types of sensor measurements are either subtly related to user behaviors or even user-centric in nature. Hence, the very process of generating, storing and transporting sensor data for building management presents inherent risk to the *privacy* of building users. Understanding these risks, quantifying the associated tradeoffs between privacy and utility, and designing smart building systems that respect users' privacy concerns while

still meeting operational goals are all open problems.

To highlight the value of building sensory data and the possibility to exploit it for inferring private information, we consider as a motivating example the *occupancy* data, namely the number of users in each room. Given its value in building management, occupancy data has become one of the most sought-after pieces of analytical output of the sensing infrastructure, and it has been inspiring extensive research [3, 38, 33, 80, 31] as well as a number of commercial products. Many passive sensing or indoor tracking platforms can create real-time occupancy data with high accuracy [57, 60, 44].

Among a variety of applications, occupancy is highly valuable especially in heating, ventilation, and air conditioning (HVAC) systems [32, 69, 11, 8]. There has been a considerable effort in industry and academia toward optimizing the HVAC system to enable intelligent controls in response to occupancy variations. While enjoying the benefits brought by occupancy data, people fail to realize potential risks in the data. Occupancy detection is not designed for surveillance purposes. It has been believed that occupancy data is privacy-preserving because it reports only the number of users and reveals identity of nobody [41]. This carefree attitude is risky. In this chapter, we show that even aggregated measurements that appear privacy-preserving or privacy-enhancing are still subject to inference attacks using side information that can reveal some of the same sensitive information as privacy-invasive techniques such as video surveillance. With access to occupancy data over a period of time, a malicious or curious individual can possibly infer occupants' indoor locations since dynamics of occupancy leak information about mobility patterns, given the intuition that changes in room occupancy correlate to user transitions between rooms.

To understand this location privacy threat, we first study the adversarial aspect of the problem setting by developing inference techniques and algorithms to reconstruct user location traces from the time-series occupancy measurements being generated. Specifically, we build a stochastic framework to infer location traces and evaluate the inference performance. The framework is based on the *factorial hidden Markov model* [37]. The model captures the dependence of occupancy measurements on user locations and the Markovian property of user location transitions between rooms over time. Prior work has studied privacy issues in other types of sensor data, including smart meter readings [58, 74]. Unlike these heuristic or information theoretic approaches, we perform an *empirical and quantitative* study. We design the location inference attack and evaluate it quantitatively with real data. Our experimental results can therefore explain how privacy breach possibilities can present realistic and achievable risks, and how we can quantify the risks.

Our quantitative treatment of the privacy problem also allows us to consider design of privacy-enhancing sensing systems, instead of only considering the adversarial angle. The performance of the HVAC system relies on the granularity of the occupancy data; with a high sensing rate, the infrastructure can continually sense the environment and enable prompt response to occupancy variations. In other words, the system enjoys more data utility. However, more fine-grained occupancy data, on the other hand, leads to more information leakage to the adversaries, i.e. reduced location privacy. This conflict essentially presents a tradeoff between *data utility and user privacy*. Therefore, knowing the variations of utility and privacy with respect to the system configurations, we can configure the sensing system to act in a privacy-conscious manner and make informed and automated decisions to achieve the optimal tradeoff, for example by minimizing privacy risk while satisfying a minimum requirement on sensing quality.

In this chapter, leveraging previous research outcomes on location privacy and mature machine learning techniques, we present a stochastic framework for inferring locations of building users from the occupancy data. We achieve the following objectives.

- We demonstrate potential location privacy leakage from the occupancy data through a formal approach that allows us to reconstruct location traces of occupants.
- We show how our framework enables quantitative analysis of the inference performance and location privacy of users with respect to several impacting factors and design parameters of the sensing system.
- We illustrate how our results can potentially cast light on the design of privacy-aware sensing and actuation.

Our investigation of location privacy in smart buildings begins with a summary of related work. We then present our location inference framework, followed by a detailed description of the enabling machine learning techniques. We then provide detailed exposition of our proposed location inference attacks and experimental study. Finally, we show how our results can be used for adaptive privacy control by the sensing system.

4.2 Related Work

Previous research has addressed the issue of explicitly tracking users or computing building occupancy as an intermediate statistic for more abstract building information collection. As examples, Manzoor et al. [57] proposed the use of RFID readers in doorways and passages to localize users. Hnat et al. developed the Doorjamb system [44] to achieve unobtrusive room-level tracking using doorway sensors. In our work, we show that it is able to track users

using occupancy data instead of explicitly tracking with dedicated systems, which can be exploited by adversaries who have access to the occupancy data.

Occupancy measurements are useful to HVAC and many other systems. Occupancy detection has received much attention in recent years. To obtain occupancy data, previous work proposed the use of infrared sensors, acoustic sensors, CO_2 sensors, motion sensors, cameras and other types of sensors coupled with estimation theory, vision algorithms and machine learning techniques [3, 33, 80, 8]. Another straight-forward yet effective approach is to aggregate location data from indoor tracking system to generate occupancy outputs.

While offering rich utility for building management, the dedicated systems described above together with other general-purpose sensing systems open the breach of user privacy. McDaniel et al. [58] studied privacy issues in smart grid and revealed how attackers, heuristically, can infer user activities/behaviors through smart meter readings. They discussed the problem qualitatively and raised the concern about privacy risks in smart environments. Our work investigates specifically the location privacy issue via occupancy data and, in contrast, provides analytical results in addition to high level intuitions. Gruteser et al. [41] propose privacy-aware techniques to collect occupancy and location data without compromising the privacy of users; unfortunately, these techniques do not protect against our location inference approach. User privacy preservation in pervasive sensor-rich environments has been studied recently by Pallapa et al. [70, 71] and by Hengartner [43], where the authors provided context-aware privacy preservation techniques that attempt to minimize privacy risks. Other studies of privacy in sensor-rich environments primarily study either location privacy or user/data anonymization [1, 4, 20, 21, 46, 42, 55, 59, 15, 90, 54]. Anonymization of the occupancy data, however, offers no protection against attackers who have direct access to the sensing database. Additionally, anonymization is not desired as management operations are usually room-specific and require room identity in nature.

There are many related studies on location privacy in particular. Gruteser and Grunwald [40] surveyed privacy issues related to location-based services, and introduced a quadtree-based algorithm to guarantee k-anonymity. Krumm proposed several location privacy protection schemes [52]. To evaluate the schemes, researchers proposed two popular metrics: k-anonymity and entropy-based criteria [12]. Shokri et al. showed that the two metrics are not adequate. They developed a probabilistic framework to quantify location privacy and further to analyze different protection strategies [81]. Their approach reconstructed outdoor location traces from anonymized and obfuscated traces. In our

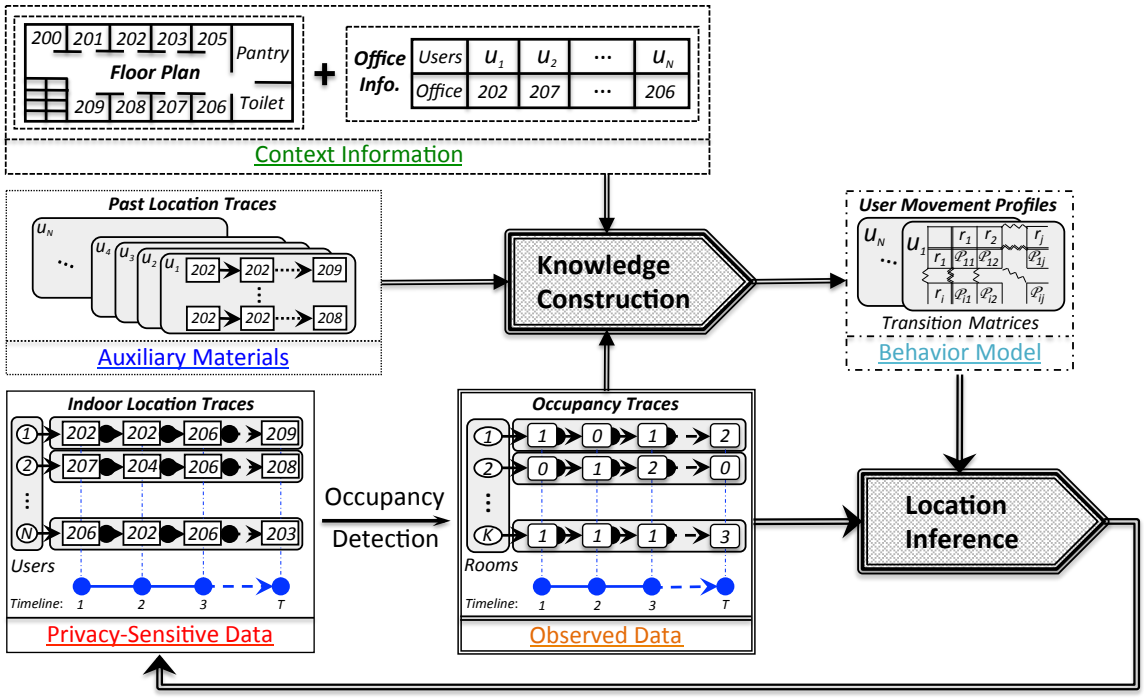


Figure 4.1: We illustrate the framework for indoor location inference from occupancy data. The privacy-sensitive location traces of building users correlate to the occupancy traces of rooms. The occupancy traces together with past location traces and available context information are fed to the knowledge construction module to create movement profiles for all users. The profile specifies a user's pattern with respect to transitioning between rooms. Using the profiles, the location inference module reconstructs location traces from the occupancy data during a certain period of time.

indoor scenario, we leverage the occupancy data instead of sanitized user locations.

A common goal for system design is to achieve satisfactory utility without sacrificing user privacy. Rajagopalan et al. [74] studied the tradeoff between utility and privacy in smart grid. To quantify this tradeoff, they applied information theoretic methods. We take a quantitative treatment as well, yet through a realizable approach. In contrast, the information theoretic bound might not be achievable by feasible algorithms. Eney at al. [29] proposed a theoretic scheme for balancing utility and privacy in smart sensor applications, in which the raw data is transformed. In our work, however, the adversaries have access to the raw occupancy data.

4.3 Location Inference Framework

In this chapter, we present our framework for inferring indoor locations from time-series occupancy data. The framework specifies essential components under our problem settings. As illustrated in Figure 4.1, the framework comprises

the use of observable occupancy data in conjunction with available context information and past user location traces to construct the user movement profiles that constitute the knowledge of adversaries. The knowledge is fed to the inference module to enable reconstructing underlying user location traces from occupancy. The overall building management system includes a sensing infrastructure, a data repository and many data consumers (service providers). The malicious or curious individual can be the database administrator or anyone who has direct access to the sensor data, and thus all raw time-series occupancy data for all rooms.

In the following chapters, we formally define these components and describe the relations among them. We also propose metrics to evaluate our inference results. For the ease of presentation, we denote random variables using calligraphic letters, realizations of random variables using lower case letters, and sets from which the random variables take values using blackboard bold letters. For instance, random variable $\mathcal{X}$ takes value x from set $\mathbb{X}$.

4.3.1 Occupancy Trace

The adversary aims to infer user locations from room occupancy. Occupancy of a room is defined as the number of users in that room. Formally, the building area of interest consists of K rooms/regions. A special location is `away` which means users are not present in any of the rooms of interest. Defining the `away` location allows the adversary to partition a huge building area into several subareas (floors, sections, departments). The `away` state aids in defining user's transitions between subareas. As so, the adversary can perform a divide-and-conquer approach in which location inference is conducted in each subarea separately. We will revisit this point later on. Readers can refer to the floor plan in the Augsburg benchmark as an example [48]. The set of locations is denoted as $\mathbb{R} = \{r_1, r_2, \cdots, r_K\}$. The occupancy for room r_k at time t is a random variable denoted as $\mathcal{O}_t^{(k)}$ $(k = 1, 2, \cdots, K)$. The occupancy variable takes a value from the set $\mathbb{O}_t^{(k)} = \{0, 1, \cdots, N\}$, where N is the total number of users in the building. We denote the occupancy for the building at time t as $\mathcal{O}_t = (\mathcal{O}_t^{(1)}, \mathcal{O}_t^{(2)}, \cdots, \mathcal{O}_t^{(K)})$. It takes values from a subset of $\mathbb{O}_t = \mathbb{O}_t^{(1)} \times \mathbb{O}_t^{(2)} \times \cdots \times \mathbb{O}_t^{(K)}$ subject to the constraint $\sum_{k=1}^{K} \mathcal{O}_t^{(k)} = N$.

The occupancy trace for each room is a timestamped sequence of occupancy measurements. Further, an occupancy trace for the tracking area is a time series of occupancy vectors for all rooms. The corresponding set of timestamps is an ordered set denoted as $\mathbb{T}_o = \{t_1, t_2, \cdots, t_T\}$. Those time instants are when there are occupancy changes. Since the detection system generates occupancy measurements periodically based on a sensing interval, timestamps in $\mathbb{T}_o$

are aligned to fixed sensing instants. Therefore, delay is expected in the timestamps compared to the instants when occupancy variations actually happen. We will emphasize this difference again when defining the location trace. The timestamp sequence also gives the length of the occupancy trace, i.e. T. For brevity, we denote the time sequence using only indices so that $\mathbb{T}_o = \{1, 2, \cdots, T\}$. The occupancy trace for the building area is thus a random process $\mathcal{O} = \{\mathcal{O}_t : t \in \mathbb{T}_o\}$.

4.3.2 Location Trace

We consider a total of N building users. The user set is denoted as $\mathbb{U} = \{u_1, u_2, \cdots, u_N\}$. The set $\mathbb{R}$ defines possible locations for all users. The location for user u_n at time t is a random variable denoted as $\mathcal{L}_t^{(n)}$ $(n = 1, 2, \cdots, N)$, which takes values from location set $\mathbb{L}^{(n)} = \mathbb{R}$.

A location trace for a user is defined as a time series of locations visited by that user. For the purpose of evaluation, we have to clarify two different types of location traces: *real location trace* and *estimated location trace*. A main difference between them is their corresponding timestamps. First, the location trace for user u_n, in general, is a random process $\mathcal{L}^{(n)} = \{\mathcal{L}_t^{(n)} : t \in \mathbb{T}\}$. For the real location trace, the time instants are when there are actually location transitions for u_n, and the corresponding timestamp set is denoted as $\mathbb{T}_r$. The time sequence for the estimated location trace constructed by the adversary is the same set $\mathbb{T}_o$ for the corresponding occupancy trace because the location trace is inferred from the occupancy trace. Usually, $\mathbb{T}_o \neq \mathbb{T}_r$ holds since occupancy detection instants are discrete and periodic, and lagging behind when actual location transitions happen.

In addition, we define $\mathcal{L}_t = (\mathcal{L}_t^{(1)}, \cdots \mathcal{L}_t^{(N)})$ as the collection of the location variables at time t. It takes values from set $\mathbb{L} = \mathbb{L}^{(1)} \times \cdots \mathbb{L}^{(N)}$. The location trace for all users therefore is defined as $\mathcal{L} = \{\mathcal{L}_t : t \in \mathbb{T}\}$.

4.3.3 Context/Auxiliary Information

Many people claim that occupancy is a privacy-preserving metric mainly because it reveals no identity information. However, occupancy combined with context information discloses private details about indoor locations of building users. A common piece of context information is office directory. This mapping between users and office rooms offers us the opportunity to link an occupancy reading of a room with the presence of a specific user since the one in the office is more likely to be the office owner. The office information can be leveraged to associate estimated location

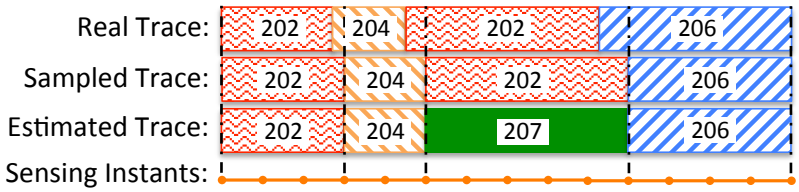


Figure 4.2: To distinguish between transition-based accuracy and time-averaged accuracy, we show the difference between the real location trace and the estimated location trace.

traces to specific users. The role of office directory will be manifest as we describe the intuition of our approach in the next chapter.

There might also be some past location traces available with which we can establish our prior knowledge about a user's mobility pattern. We will specify in later chapter how the prior knowledge is established and incorporated.

4.3.4 User Movement Profiles

A user's movement profile describes the pattern of transitions between different rooms in the building. Essentially, this profile is represented by a Markov transition matrix as shown in Figure 4.1 with P_{ij} specifying the probability of moving from room r_i to r_j when there is a transition. In our approach, the location trace is modelled as a Markov process.

4.3.5 Objectives and Evaluation Metrics

The objective specifies what private information the adversary aims to extract from occupancy data. Potential objectives include: *localization* in which the adversary attempts to find out locations of users at certain time instants; *meeting disclosure* in which the adversary is interested in finding out who met whom at a time instant in a given room; or *full tracking* in which the adversary aims to reconstruct the most probable joint location trace for all users [81].

For the localization and meeting disclosure attacks, to evaluate the inference results we can compare the estimated locations with the actual locations of users. In this chapter, we focus more on the full tracking attack. To assess the outcome of the attack, we compare the estimated location trace with the real location trace. Noteworthily, we can evaluate the trace for each individual user or the joint trace for all users.

The real location for user u_n at time t is denoted as $l_t^{(n)}$ and the full trace denoted as $l^{(n)}$. We denote as $\hat{l}_t^{(n)}$ the estimated location for user u_n at time t, and the full trace as $\hat{l}^{(n)}$.

As illustrated in Figure 4.2, by comparing the estimated trace $\hat{l}^{(n)}$ with the real trace $l^{(n)}$ over the time span and finding the percentage of overlap, we can compute the *time-averaged accuracy* `ta_acc` of $\hat{l}^{(n)}$:

$$\texttt{ta\_acc}(\hat{l}^{(n)}, l^{(n)}) = \frac{1}{t_T - t_1} \int_{t_1}^{t_T} \mathbb{1}(\hat{l}_\tau^{(n)}, l_\tau^{(n)})\, d\tau,$$

where $\mathbb{1}(x, y)$ is the binary indicator function that equals to 1 when $x = y$ and 0 otherwise.

On the other hand, we may only be interested in the correctness of the estimation of location transitions. In this case, we compare the estimated trace $\hat{l}^{(n)}$ with the sampled location trace. As shown in Figure 4.2, to obtain the sampled trace, we align location transitions in the real trace to sensing instants. We denote as $l_t^{(n)}[t_s]$ the sampled location for user u_n at time t with sensing interval t_s, and the full sampled trace as $l^{(n)}[t_s]$. Then we can define the *transition-based accuracy* `tb_acc` of $\hat{l}^{(n)}$ compared to $l^{(n)}[t_s]$ as:

$$\texttt{tb\_acc}(\hat{l}^{(n)}, l^{(n)}[t_s]) = \frac{1}{T} \sum_{t \in \mathbb{T}_o} \mathbb{1}(\hat{l}_t^{(n)}, l_t^{(n)}[t_s]).$$

Since the location transitions are of interest, we compare the estimated trace with the sampled trace only at instants specified by $\mathbb{T}_o$ as defined in the location trace subsection.

The above definitions of accuracy evaluate the estimated location trace for each individual user. If combining the locations for all users at each time into a N-sized tuple, we can evaluate the joint location trace for all users. The accuracy in this case can be calculated in rather similar way.

4.4 From Occupancy to Location

In this chapter, we start with the intuition of applying the factorial hidden Markov model to inferring indoor locations from occupancy. Then we present the model and related technical details in the context of this location inference problem.

4.4.1 Overview of the Approach

Reconstructing location traces from occupancy data is non-trivial because: 1) occupancy measurements are aggregates of locations, revealing no identities of users in certain rooms, and 2) location traces of multiple users interleave with each other, incurring ambiguity when translating occupancy changes to location transitions of different users. For example, occupancy measurements report that two people are present in room 100, and at next instant there is one person in room 101 and the other in room 102. Each time along with an occupancy change, there is a 'location reshuffle' of users. However, different location transition combinations can lead to the same occupancy change, so occupancy is lossy.

To address the two challenges, we leverage *context information* as well as *user mobility properties*. Specifically in our approach, the context information is the directory that provides office information about users. Even through an occupancy measurement itself is sanitized, directory information can act as a side-channel for inferring underlying identities of occupants. An office can thus serve as a user's identifier. The one sitting in an office is most likely to be the office owner. The context information can be captured by the mobility model as we will explain later.

As previously mentioned, an occupancy change may lead to multiple location transition combinations. The adversary needs to find the correct combination. Without explicitly tracking users, deterministic approaches prove infeasible. Fortunately, the location reshuffle is not uniformly random over all combinations, and users' mobility patterns can be leveraged to determine the preferences over them. Hence, the mobility pattern enables attributing tangled transitions to specific users. Each user's location transitions can be modelled as a first-order Markov process, which proves feasible in describing user mobility patterns [56, 51]. In addition, this first-order assumption makes our analysis easier and the inference algorithm simpler while still yielding satisfactory inference results as we will demonstrate in Chapter 4.7. Another concern regarding the Markov transition model may arise if a user exhibits different patterns during different time periods within a day. This drawback can be mitigated by introducing a heterogeneous Markov model to capture one's movement patterns during different periods in a day.

The context information can be incorporated into the Markov model since a user's location transitions are usually centred on his office. As we can observe from a real dataset, over 50% of transitions are either from or to the office. Hence, the Markov transition matrices for different users are guaranteed to be distinct if users have different offices. The discrepancy enables differentiating various users and attributing occupancy changes to location transitions. If

multiple users share an office, their transition matrices might or might not be different enough to distinguish any of them. The feasibility hinges on the discrepancy between transition matrices, which is not necessarily present in all real scenarios. The degree of discrepancy is affected by working environment, personal habits and many other factors. Hence, a definitive conclusion would require a comprehensive study of human mobility behavior in various indoor environments, which is beyond the scope of this dissertation. Instead, we focus on the problem of modelling the relationship between the users' mobility patterns and the occupancy data to demonstrate the ability to infer user activities. Given user locations, we can obtain occupancy through deterministic aggregation. The reverse process, however, is probabilistic and complicated, and its success relies on the information loss in the location-to-occupancy process and the inherent entropy in user location transitions over time

To give an example, if we observed two users in one room and then one of them left the room and entered another room, we are unable to find which one of the two made this transition by this occupancy change. However, if the one who left entered an office, the user can be identified with high probability based on the ownership of the office. This explains how context information helps in removing ambiguity. The context information can be incorporated into the Markov transition matrix such that a user has higher probability of returning his office from other rooms than other users going to his office.

Returning to the discussion about the intuition behind our approach, the occupancy traces are generated by the occupancy detection system. Occupancy at a time only relies on user locations at that instant. The characteristics of the detection system determines the relation between user locations and room occupancy. The relation can be deterministic when detection is accurate or probabilistic if there is random noise. The detection system characteristics can be investigated beforehand.

Taking into account the Markovian property of location traces and the relation between occupancy and location, we can hence describe the inference framework using the hidden Markov model (HMM) [6]. In a HMM, the current state of the process cannot be observed directly. Instead, some outputs from the current state can be observed. The probability distribution of the outputs depends only on the current state. Specifically in our problem, we apply the factorial hidden Markov model (FHMM) instead of the HMM. The FHMM offers us several benefits as we will discuss soon.

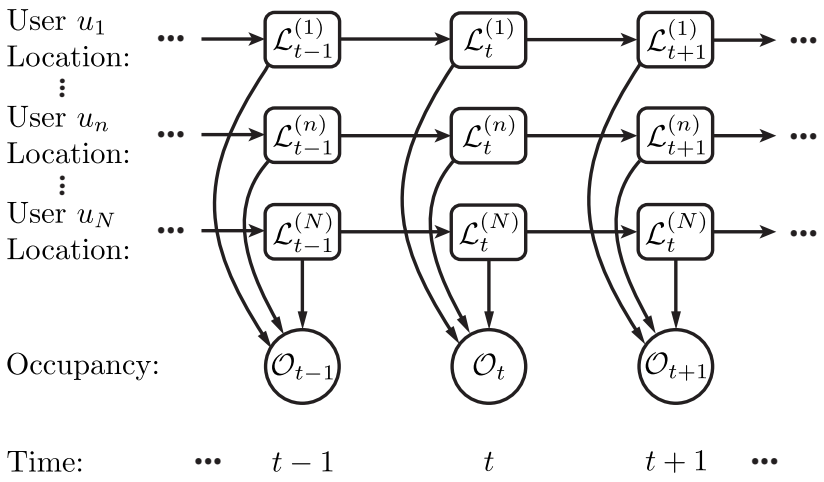


Figure 4.3: We illustrate the FHMM model used for location inference. The model consists of multiple latent state chains each of which represents the location trace of one user. The observation sequence shows occupancy changes for all locations.

4.4.2 Factorial Hidden Markov Model

In the HMM, locations of all users are combined into one latent variable. All user locations are allowed to interact arbitrarily. The advantage is the model can incorporate possible interactions of location transitions among users. However, the drawbacks render HMM infeasible. The size of the state space for the single latent variable is K^N, and leads to intolerable computational complexity. Additionally, it requires more learning data for the purpose of establishing the Markov transition matrix. Hence we resort to FHMM which decouples location transitions of different users. The location traces for different users are assumed to be independent from each other. They are linked together by the occupancy trace. The state space is thereafter reduced to KN. The computational efficiency can be significantly improved. Even though the independence assumption might not hold in some scenarios, it makes the algorithm trackable without undermining the inference accuracy.

As illustrated in Figure 4.3, the sequence of observations is the *occupancy trace*. The FHMM consists of multiple latent state sequences, each of which represents the *location trace* of a user. Each location trace follows its own Markovian transition nature as users exhibit different mobility patterns. At each time instant, the occupancy measurement depends on locations of all users so we can see arrows to the observation from corresponding latent states in Figure 4.3. Two fundamental problems for FHMM are *learning* and *inference*. We will explain them later in the context of our scenario.

4.4.3 Model Parameters

The model is governed by a set of parameters: initial state probabilities, transition probabilities and emission probabilities. In the context of our scenario, initial state probabilities specify the chance of each user starting from certain locations. Since every user is initially located outside the building, with probability 1 the initial latent state is `away`. Initial state probabilities are invariant.

The transition probabilities specify the mobility pattern of a user, which is denoted as a $K \times K$ transition matrix. We define as $\mathbf{A}^{(n)} = [a_{ij}^{(n)}]$ $(i, j = 1, 2, \cdots, K)$ the transition matrix for user u_n, where $a_{ij}^{(n)} = P(\mathcal{L}_{t+1}^{(n)} = r_j | \mathcal{L}_t^{(n)} = r_i)$ for $t = 1, 2, \cdots, T-1$ as the transition model is homogeneous such that transition probabilities do not change over time.

The emission probabilities characterize the conditional distribution of occupancy measurements given user locations, defined as $P(\mathcal{O}_t | \mathcal{L}_t)$. The conditional distribution hinges on specific occupancy detection systems. We consider the worst case of privacy leakage in which occupancy is accurately detected. Given precise occupancy information, location inference can achieve better performance. This gives us an upper bound on the potential privacy risk. The conditional distribution hence becomes degenerate and gives the correct occupancy.

The learning problem in HMM is to establish model parameters. In our model, we need to learn the transition matrix for all users. We can denote the model parameters as $\lambda = (\mathbf{A}^{(1)}, \cdots, \mathbf{A}^{(N)})$. Input to the learning module includes the occupancy trace, context information and possibly some past location trace samples.

4.5 Knowledge Construction

In order to infer locations from occupancy, we first need to establish the knowledge about user mobility patterns. Knowledges construction, in the context of the FHMM, translates into learning the model parameters, namely the transition matrices for all users. We present two methods for parameter estimation.

4.5.1 Learning Transition Matrices

The model parameter can be established using the maximum likelihood estimation (MLE). Formally, we need to find the parameter $\lambda^\star$ that maximizes the probability of the observation (occupancy) sequence, i.e.: $\lambda^\star = \text{argmax}_\lambda \, P(\mathcal{O}|\lambda)$.

Solving the optimization problem involves the expectation maximization (EM) algorithm, which is known as the Baum-Welch algorithm in the HMM [6]. The EM algorithm iterates between two steps commonly referred to as E and M. In the E step, we compute the posterior distribution over latent states using current parameters, and obtain the expected log-likelihood of observations as a function of parameters. In the M step, we maximize the expectation and update the parameters. The EM algorithm for MLE is defined as:

$$Q\left(\lambda^{(i+1)}|\lambda^{(i)}\right) = E_{\mathcal{L}|\mathcal{O},\lambda^{(i)}}\left[\log P(\mathcal{L},\mathcal{O}|\lambda^{(i+1)})\right],$$

where $\lambda^{(i)}, \lambda^{(i+1)}$ are the parameters at the i^{th} and $(i+1)^{th}$ iteration. The Q function takes conditional expectation over all possible location traces and is maximized with respect to $\lambda^{(i+1)}$. The E step requires computing posterior probabilities $P(\mathcal{L}|\mathcal{O},\lambda^{(i)})$, which incurs intensive computation for exact inference. Hence, Gibbs sampling is applied to generate location traces and approximate the posterior distributions [37].

Each iteration of EM samples all TN latent location variables. After adequate iterations, the samples can approximate the posterior probabilities. Then in the M step, we can update the transition matrix using the obtained posterior probabilities

$$a_{ij}^{(n)} = \frac{\sum_{t=1}^{T-1} P(\mathcal{L}_t^{(n)} = r_i, \mathcal{L}_{t+1}^{(n)} = r_j|\mathcal{O},\lambda)}{\sum_{t=1}^{T-1} P(\mathcal{L}_t^{(n)} = r_i|\mathcal{O},\lambda)}.$$

The algorithm iterates back and forth between the two step until convergence of the log-likelihood function. We thereafter obtain the estimates of the user movement profiles.

4.5.2 Prior Knowledge

The approach described above leverages no prior knowledge that we may gain from auxiliary information. Past location traces, if available, provide us the prior knowledge about users' mobility patterns. For the transition matrix $\mathbf{A}^{(n)} = [a_{ij}^{(n)}]$ $(i,j = 1,2,\cdots,K)$, we assume that the rows of $\mathbf{A}$ are independent and their densities follow Dirichlet distributions. By tuning the parameters of a certain Dirichlet distribution, we are able to generate various density functions for a row vector in $\mathbf{A}^{(n)}$. A transition matrix is therefore governed by K Dirichlet distributions. In such a way, we can create various movement profiles for each user. Considering the independence between rows of a

transition matrix and between the matrices of different users, the density of the model parameters becomes

$$P(\lambda) = C \cdot \prod_{n=1}^{N} \left(\prod_{j=1}^{K} \left(\prod_{i=1}^{K} \left(a_{ij}^{(n)} \right)^{\eta_{ij}^{(n)}-1} \right) \right),$$

where C is the normalizing factor, $[\eta_{ij}^{(n)}]$ $(i = 1, \cdots, K)$ is the set of parameters of the Dirichlet distribution for the j^{th} $(j = 1, \cdots, K)$ row of the transition matrix $\mathbf{A}^{(n)}$.

Given past location traces of a user n, we can obtain the set of parameters for the Dirichlet distributions. The parameter $\eta_{ij}^{(n)}$ equals to the number of location transitions of user n from r_i to r_j. The parameters represent the prior knowledge of movement patterns, and is taken into account when estimating the location transition probabilities. The method is called maximum a posterior (MAP) estimation, defined as $\lambda^{\star} = \text{argmax}_{\lambda}\ P(\lambda|\mathcal{O}) = \text{argmax}_{\lambda}\ P(\mathcal{O}|\lambda)P(\lambda)$. Similarly, we need to apply the EM algorithm for MAP which introduces the additional term $P(\lambda^{(i+1)}$ compared to the version for MLE, yielding

$$Q'\left(\lambda^{(i+1)}|\lambda^{(i)}\right) = Q(\lambda^{(i+1)}|\lambda^{(i)}) + \log P(\lambda^{(i+1)}),$$

where we can observe an additional term of the logarithm density of model parameters compared to the Q function for MLE. The posterior probabilities are also approximated using Gibbs sampling. The resulting update for the transition matrix in the M step becomes

$$a_{ij}^{(n)} = \frac{\sum_{t=1}^{T-1} P(\mathcal{L}_t^{(n)} = r_i, \mathcal{L}_{t+1}^{(n)} = r_j|\mathcal{O}, \lambda) + \eta_{ij}^{(n)} - 1}{\sum_{t=1}^{T-1} P(\mathcal{L}_t^{(n)} = r_i|\mathcal{O}, \lambda) + \sum_{j=1}^{K}(\eta_{ij}^{(n)} - 1)}.$$

Each update incorporates the transitions seen in past location traces and hence forces the estimation to lean to prior knowledge in contrast to the MLE estimation.

4.6 Location Inference Attacks

Leveraging the transition matrices of all users we obtained, we are able to mount several types of location inference attacks. Recall from the objectives, the adversary aim to infer locations at certain time instants or reconstruct the whole trace.

4.6.1 Localization and Meeting Disclosure Attacks

In localization attacks, the attacker aims to find the location of a user at a specific time instant. Formally, we need to compute $P(\mathcal{L}_t^{(n)} = r_i|\mathcal{O}, \lambda)$, which specifies the distribution of the location of user n at time t. The probability can be computed using the modified forward-backward algorithm as proposed by Ghahramani et al. [37]. The forward algorithm calculates the probability of locations for all users at a certain time and the occupancy sequence up to that time, defined as $\alpha_t = P(\mathcal{L}_t^{(1)}, \cdots, \mathcal{L}_t^{(N)}, \mathcal{O}_1, \cdots, \mathcal{O}_t|\lambda)$. The backward algorithm computes the probability of future occupancy trace conditioned on current locations of all users, defined as $\beta_t = P(\mathcal{O}_{t+1}, \cdots, \mathcal{O}_T|\mathcal{L}_t^{(1)}, \cdots, \mathcal{L}_t^{(N)}, \lambda)$.

Given the forward and backward variables at time t, we can compute the posterior probabilities at that time. Summing over locations of other users, we can obtain the marginal probability of the location for user n:

$$P(\mathcal{L}_t^{(n)}|\mathcal{O}, \lambda) = \sum_{\mathcal{L}_t^{(m)}: m \neq n} P(\mathcal{L}_t^{(1)}, \cdots, \mathcal{L}_t^{(N)}|\mathcal{O}, \lambda).$$

In a meeting disclosure attack, the attacker aims to determine whether a pair of users u_n and u_m met in room r_i at time t. Formally, the attacker needs to compute $P(\mathcal{L}_t^{(n)} = r_i, \mathcal{L}_t^{(m)} = r_i|\mathcal{O}, \lambda)$, which can be computed by marginalizing all other variables:

$$P(\mathcal{L}_t^{(n)}, \mathcal{L}_t^{(m)}|\mathcal{O}, \lambda) = \sum_{\mathcal{L}_t^{(l)}: l \neq n, m} P(\mathcal{L}_t^{(1)}, \cdots, \mathcal{L}_t^{(N)}|\mathcal{O}, \lambda).$$

Depending on different objectives, we can extend the meeting disclosure attacks to consider more than two users at multiple instants.

4.6.2 Location Tracking Attack

In location tracking attack, the objective is to reconstruct the most probable location trace given the occupancy trace. Formally, we need to find $\hat{l} = \text{argmax}_l \; P(\mathcal{L} = l|\mathcal{O}, \lambda)$. The approach to this problem is well known as the Viterbi algorithm in HMM. In our context, a straightforward approach is to treat the locations of all users as a single latent variable and directly apply the Viterbi algorithm. The approach, however, is infeasible as the state space for the location variable is K^N.

We modify the Viterbi algorithm to reconstruct the most likely location trace in FHMM. Considering that the

location traces are independent, we can sequentially advance the location trace for each user to next time instant. The state space is reduced to KN and the computational overhead is significantly reduced.

For the ease of presentation, we denote as $\{\mathcal{L}_.\}_p^q$ the location sequence $\mathcal{L}_p, \cdots, \mathcal{L}_q$. Similarly, $\{\mathcal{O}_.\}_p^q = \mathcal{O}_p, \cdots, \mathcal{O}_q$ and $\{\mathcal{L}_t^{(\cdot)}\}_m^n = \mathcal{L}_t^{(m)}, \cdots, \mathcal{L}_t^{(n)}$. We can then define the recursion variables

$$\begin{aligned}
\phi_t &= \max_{\{\mathcal{L}_.\}_1^{t-1}} P(\{\mathcal{L}_.\}_1^{t-1}, \{\mathcal{L}_t^{(\cdot)}\}_1^N, \{\mathcal{O}_.\}_1^t) \\
\phi_t^{(0)} &= \max_{\{\mathcal{L}_.\}_1^{t-1}} P(\{\mathcal{L}_.\}_1^{t-1}, \{\mathcal{L}_t^{(\cdot)}\}_1^N, \{\mathcal{O}_.\}_1^{t-1}) \\
\phi_t^{(1)} &= \max_{\{\mathcal{L}_.\}_1^{t-2}, \{\mathcal{L}_{t-1}^{(\cdot)}\}_2^N} P(\{\mathcal{L}_.\}_1^{t-2}, \{\mathcal{L}_{t-1}^{(\cdot)}\}_2^N, \\
&\qquad\qquad \mathcal{L}_{t-1}^{(1)}, \{\mathcal{L}_t^{(\cdot)}\}_2^N, \{\mathcal{O}_.\}_1^{t-1}) \\
&\vdots \\
\phi_t^{(N)} &= \max_{\{\mathcal{L}_.\}_1^{t-2}} P(\{\mathcal{L}_.\}_1^{t-2}, \{\mathcal{L}_{t-1}^{(\cdot)}\}_1^N, \{\mathcal{O}_.\}_1^{t-1}).
\end{aligned}$$

Here, ϕ_t is a function of $\mathcal{L}_t$, which gives the maximum probability of location traces ending with $\mathcal{L}_t$. Different from the original Viterbi algorithm, the modified algorithm has a sequence of intermediate variables which update the location of each user to the next time instant. Intermediate variable $\phi_t^{(n)}$ is a function of a combination of locations at time t and locations at time $t-1$, or precisely $\left\{\{\mathcal{L}_{t-1}^{(\cdot)}\}_1^n, \{\mathcal{L}_t^{(\cdot)}\}_{n+1}^N\right\}$. We can therefore obtain recursion relations as below

$$\phi_t = \phi_t^{(0)} P(\mathcal{O}_t|\mathcal{L}_t) \tag{4.1}$$

$$\phi_t^{(n-1)} = \max_{\mathcal{L}_{t-1}^{(n)}} \phi_t^{(n)} P(\mathcal{L}_t^{(n)}|\mathcal{L}_t^{(n)}) \tag{4.2}$$

$$\phi_t^{(N)} = \phi_{t-1}(\mathcal{L}_{t-1}). \tag{4.3}$$

For this factorial model, we update the location for each user independently and sequentially as specified by (4.2).

To construct the most probable location trace, two kinds of back-tracking procedures are performed: local and global. The local backtracking finds the most probable joint locations for all users at a time instant, namely $\mathcal{L}_t$. The global backtracking finds the most probable location trace over time, namely $\{\mathcal{L}_.\}_1^T$. The update from ϕ_{t-1} to ϕ_t includes N steps specified by (4.2). Each step we store the previous location of user n that leads to the maximum

probability

$$\Phi_t^{(n-1)}(\mathcal{L}_t^{(n)}) = \underset{\mathcal{L}_{t-1}^{(n)}}{\text{argmax}}\ \phi_t^{(n)} P(\mathcal{L}_t^{(n)}|\mathcal{L}_{t-1}^{(n)}).$$

Then by local backtracking, we can progressively obtain the optimal location for each individual user and eventually construct the most probable joint locations for all users at time t. The procedure is defined as

$$\mathcal{L}_{t-1}^{\star(1)} = \underset{\mathcal{L}_{t-1}^{(1)}}{\text{argmax}}\ \phi_t^{(1)}, \quad \mathcal{L}_{t-1}^{\star(n)} = \Phi_{t-1}^{(n-1)}(\mathcal{L}_{t-1}^{\star(n-1)})$$

for $n = 1, \cdots, N$. At each time t, we store the previous joint locations of all users that result in the maximum probability

$$\Phi_t(\mathcal{L}_t) = \text{argmax}_{\mathcal{L}_{t-1}}\ \phi(\mathcal{L}_t) P(\mathcal{L}_t|\mathcal{L}_{t-1}).$$

Then applying global backtracking which is similar to the backtracking in Viterbi algorithm, we start from the last time instant and reconstruct the location trace backward.

4.7 Experimental Study

In this chapter, we evaluate our inference approach. The metric used to quantify the inference performance is *accuracy*. The purpose of assessing the inference method is to demonstrate the possibility of inferring locations from occupancy data in certain environments and to reveal the underlying privacy risk. Moreover, we hope our results can provide insights into design and implementation of privacy-preserving sensing systems. Toward our goal of privacy-preserving sensing system design, we study two impacting factors: the sensing interval and the number of users.

4.7.1 Dataset

To setup our experiment, we use synthetic data as well as real-world data from the the *Augsburg Indoor Location Tracking Benchmark* [48]. The dataset includes location traces for 4 users in an office building with 14 rooms. The benchmark dataset contains location data over a period of 2 to 9 weeks. Each trace records location transitions of

Table 4.1: We show the average number of transitions each user made per day, and the average percentage of transitions from or to one's office.

User	avg num of trans per day	avg percent of trans from/to office per day
1	26.2	52.5%
2	37.7	70.7%
3	35.2	66.8%
4	26.6	65.7%

one user during a day. All traces are timestamped. The Augsburg dataset is the only suitable indoor room-level location dataset we found publicly available. Though relatively small in scale, the location traces in the dataset are representative of the transition patterns of many people in certain working environments.

As previously discussed, in many scenarios the office room assumes an important role in a user's transitions. In Table 4.1, we show two statistics about the Augsburg dataset. Noteworthily, of all transitions per day, 52% to 70% are either originate from or destined to one's office, and office can represent one's identity. Each user's transitions are heavily linked to his office, resulting in discrepancy in their transition matrices.

To investigate the impact of the population, we create synthetic data that simulates location traces for 20 users based on the Augsburg dataset. To simulate the location traces, we apply the random way-point mobility model. A user randomly chooses the next location according to his movement profile, i.e. the Markov transition matrix. The stay time at one location is normally distributed. To precisely simulate the transitions, we also take into account the walking speed and the distance between rooms.

All the parameters for the mobility model are created based on the empirical study of the Augsburg dataset. Tweaking the existing movement profiles, we create transition matrices for another 16 users. The stay time in a room varies for different users and rooms. Typically, the distribution for one's office has larger mean values and larger variance, while the distribution for other rooms has smaller mean and variance.

4.7.2 Methodology

First, we experiment with the benchmark data. We study the impact of the sensing interval. We aim to infer location traces of the 4 users under different sensing intervals. Though the dataset provides location traces over a period of 2 to 9 weeks, there were 10 days when all 4 users were present in the building. The location traces in the 10 days

exhibit differences in the number of transitions, the total length and the stay time at rooms. We assume the detection system can accurately detect occupancy at each room. Hence, given the location traces of all users, we can obtain the occupancy traces for all rooms by a simple counting. We apply different intervals and get occupancy traces of different granularities. For each sensing interval, the resulting occupancy trace is used to perform 10 fold cross-validation. Specifically, to perform the cross-validation we partition the location traces and the corresponding occupancy trace into 10 complementary subsets. One round of validation involves performing training on 9 subsets (called the training set), and validating the model on the remaining subset (called the testing set). To reduce variability, cross-validation is conducted using different training sets, and the validation results are averaged over 10 rounds.

Second, we study the impact of population using the synthetic data. The sensing interval is fixed to 1 second to show the optimal inference performance. We create scenarios of different numbers of users from 1 to 20. Each time, we can obtain the occupancy traces from all location traces. Similarly as the experiment with the Augsburg data, we perform 10 fold cross-validation over the synthetic data of 10 days. As the interval is 1 second, the `tb_acc` and the `ta_acc` becomes equivalent. We compare the estimated location traces with real location traces and evaluate the accuracy.

We conduct our experiment based on available sample traces. To also show the statistical significance, we evaluate our estimation of the mean value of accuracy. We assume that the accuracy in each iteration of the cross-validation is a normally distributed random variable. Denote as X_i $(i = 1, \cdots, 10)$ the accuracy result obtained in the i^{th} run of the cross-validation, and $X_i \sim N(\mu, \sigma^2)$. The sample mean $\overline{X}$ follows normal distribution. The sample variance S^2 follows chi-square distribution. Then we can obtain that $U = \frac{\overline{X}-\mu}{S/\sqrt{10}} \sim t(9)$, where the random variable U follows t-distribution with the degree of freedom 9. Therefore, we are able to obtain that $P(|\overline{X}-\mu| \leq 0.03) = P(|U| \leq \frac{0.03}{S/\sqrt{10}})$, which essentially computes the probability that the estimated mean is within a small bound around the true mean value. This probability describes the confidence in estimations of the accuracy on average.

4.7.3 Accuracy and Sensing Interval

Figure 4.4 illustrates the results about the impact of the sensing interval, and the difference between transition-based accuracy and time-averaged accuracy. Figure 4.4(a) shows the performance using metric `tb_acc` while Figure 4.4(b) shows the result for `ta_acc`. In both figures, the sensing interval starts from 1 second and increases to 128 seconds.

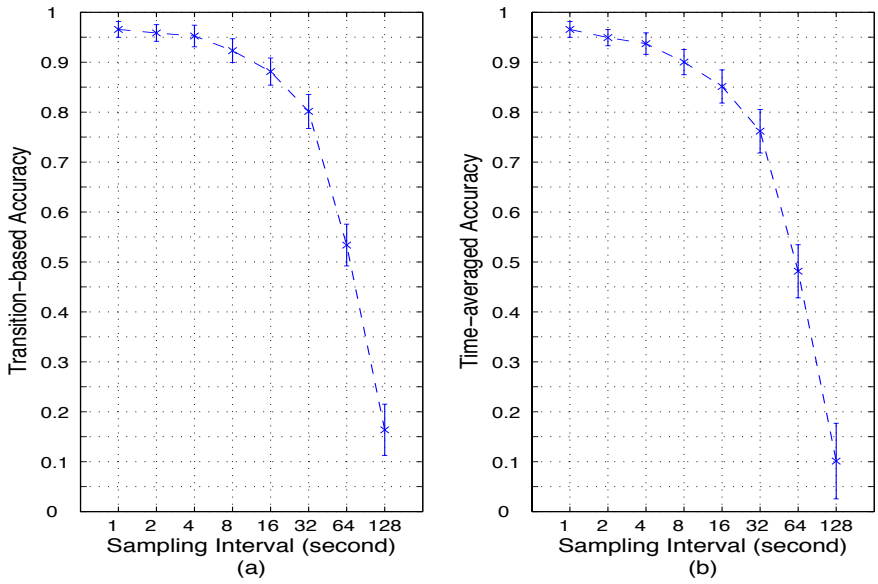


Figure 4.4: We show the results of the 10 fold cross-validation. The number of users is 4. The errorbar plot in (a) and (b) illustrates the mean and standard deviation of transition-based accuracy and time-averaged accuracy, respectively, under sensing intervals from 1 to 128 seconds.

For each sensing interval, the errorbar plot depicts mean and standard deviation of the 10 fold cross-validation.

In Figure 4.4(a), when the sensing interval is 1 second, `tb_acc` can achieve 0.96 on average. As the sensing interval increases to 128 seconds, the mean gradually decreases to below 0.2. The standard deviation slightly increases.

In Figure 4.4(b), the errorbar plot for `ta_acc` exhibits similar trend as the plot for `tb_acc`. When the sensing interval is 1 second, `ta_acc` becomes `tb_acc`. It reaches about 0.96. As the sensing interval increases to 128 seconds, `ta_acc` slides to about 0.1 on average. With a larger sensing interval, the occupancy detection system might miss certain occupancy changes and render the inference system fail to reconstruct certain location transitions that led to those changes. In addition, a larger sensing interval results in imprecision in timestamps. Comparing figure (b) with figure (a), we can find that `tb_acc` outperforms `ta_acc`by a margin that expands as the sensing interval increases.

Figure 4.6(a) shows the t-analysis result for the transition-based accuracy. We evaluate the probability that the sample mean of the transition-based accuracy is within ± 0.03 of the real mean value. The probability maintains 1 when the sensing interval is below 16 seconds, and declines to about 0.98 as the sensing interval increases to 128 seconds. This result agrees with the Figure 4.6(a) as the variance increases with the sensing interval. The estimation therefore becomes less accurate. Figure 4.6(b) shows the t-analysis result for the real accuracy. The curve exhibits similar trend. It is, by contrast, smooth as the variance in Figure 4.6(b) does not increase significantly with the sensing

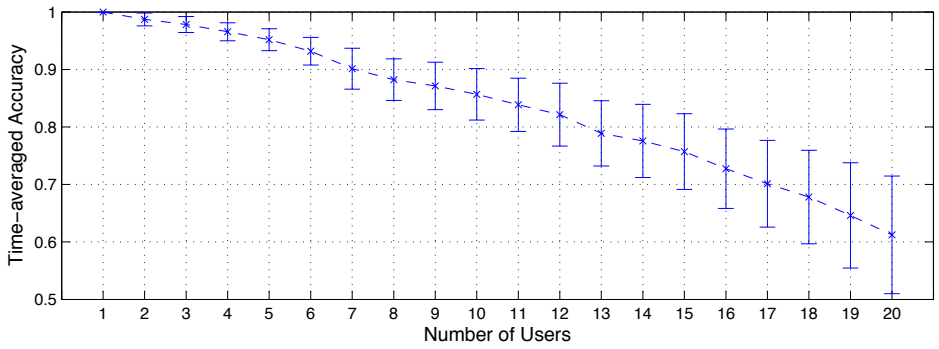


Figure 4.5: We show the 10 fold cross-validation results in the presence of different numbers of users from 1 to 20. The errorbar plot illustrates the mean and standard deviation of inference accuracy. The sensing interval is fixed at 1 second. Transition-based accuracy and time-averaged accuracy become interchangeable in this case.

interval.

4.7.4 Accuracy and Number of Users

Figure 4.5 illustrates the results that we have obtained about the impact of the number of users. The sensing interval is chosen to be 1 second. The number of users increases from 1 to 20. In each case, the errorbar plot depicts the mean value and the standard deviation of the inference accuracy. Since the transition-based accuracy and the time-averaged accuracy becomes interchangeable when the sensing interval is 1 second, we do not distinguish between these two metrics. When there is only one user, occupancy is equivalent to location, which leads to fully correct estimation of location trace of that user. As the number of users increases, the location traces of different users interleave with each other, which leads to uncertainty in reconstructing the location traces and inaccuracy in estimation. The inference accuracy on average declines to about 0.61 when there are 20 users. The variance is gradually increasing with the number of users, which also supports the fact that more users incur more uncertainty. Figure 4.6(c) shows the t-analysis result. The confidence of estimation is close to 1 with less than 8 users. It slides in case of more users since the variance climbs.

4.7.5 Average Accuracy and Joint Accuracy

As explained when defining the metrics, the metrics used average the accuracy of all location traces. We call it *average accuracy*. Figure 4.7 compares the average accuracy with *joint accuracy* that denotes the accuracy of the joint location trace for all users. In the joint location trace, locations of all users at a time are recognized as a tuple, and hence a

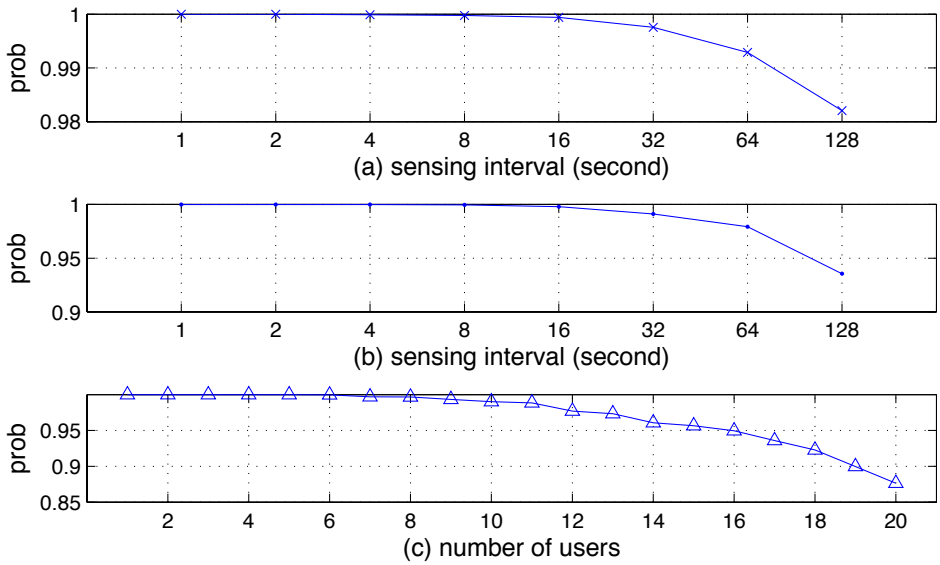


Figure 4.6: We show through t-analysis the probability that the sample mean of accuracy is within ± 0.03 of the actual mean value. Figure (a) shows the result about the transition-based accuracy in Figure 4.4(a). Figure (b) shows the result about the time-averaged accuracy shown in Figure 4.4(b). Figure (c) shows the result about the accuracy illustrated in Figure 4.5.

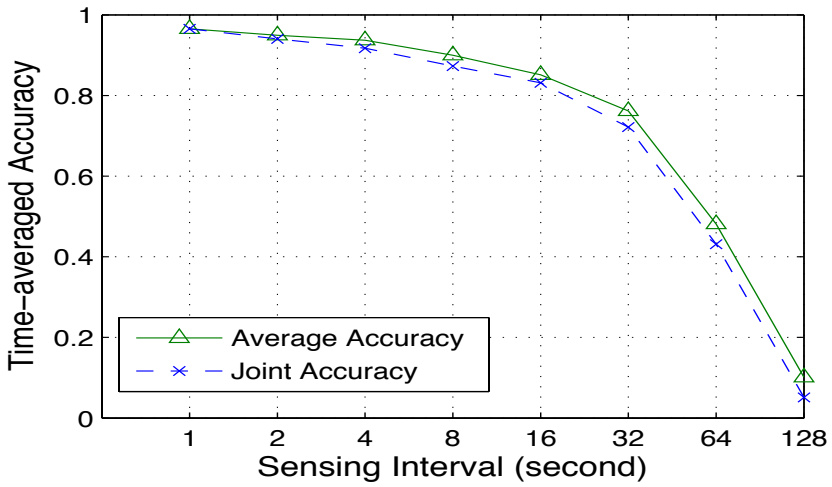


Figure 4.7: The two curves depict the mean value of joint accuracy and average accuracy, respectively, as the sensing interval increases from 1 to 128 seconds.

mistake for even one user renders the whole tuple incorrect. The joint accuracy provides a lower bound on the accuracy for individual users. From the figure, we can observe a non-negligible difference between the joint accuracy and the average accuracy. When the sensing rate is 1 second, the average accuracy outperforms by a small margin of about 0.01. The difference expands to 0.05 as the sensing interval increases.

4.8 Sensing System Design

The sensing system can leverage the results of our empirical study to achieve adaptive control of privacy. The accuracy metric we use measures the loss of location privacy. Accuracy is affected by several factors, among which the sensing interval is a configurable system parameter while the number of users is a context factor that the system cannot change. We discuss how the system takes into account multiple factors and performs adaptive control.

4.8.1 Design Metrics

The sensing system feeds sensor measurements to a variety of service providers residing above it to enable automatic building management. A central goal of the sensing system is thus to provide adequate data to data consumers so that they can perform their functionalities properly. In other words, the sensing system needs to provide sufficient *data utility*.

There are several system parameters affecting data utility: sensor deployment density, sensing interval, and measurement accuracy. The parameters can be configured to achieve different operation goals. Other than the configuration parameters, the utility is also impacted by the context including the population in the building, the behavior patterns of the users and the topology of the building. We incorporate impacting factors and the context information with a single utility function $\texttt{Util}_{\mathcal{C}}(\mathcal{P})$ in which $\mathcal{P}$ denotes relevant parameters that the system can configure, and $\mathcal{C}$ represents the contextual settings that the system cannot really change yet might be aware of through sensing, such as the population in the building.

The deployment density represents the spatial granularity of the sensing system. In general, with more sensors the system provides more utility because more area is under coverage. The sensing interval represents the temporal granularity of the system. More frequent sensing enables service providers to have a finer temporal granularity in their view of the building environment, and allows the building management system to respond to environmental variations

properly and promptly. The measurement accuracy represents the inherent quality of sensor readings, i.e. how close the measurement is to the real physical properties such as temperature, humidity, illumination and so forth. The utility function $\texttt{Util}_{\mathcal{C}}(\mathcal{P})$ is an increasing function of the parameters described above.

On the other hand, system parameters also affect *user privacy*. In order to evaluate the sensing system with respect to protecting private details of building occupants, we need a privacy function $\texttt{Priv}_{\mathcal{C}}(\mathcal{P})$ which characterizes to what extent the system preserves the specific privacy of interest. The context information $\mathcal{C}$ also affects our assessment of privacy. For example, when there are more users in the building, the resulting higher entropy leads to more privacy for the users. Considering the system parameters mentioned above, higher deployment density leads to less privacy for users. Smaller sensing interval also allows attackers to infer user activities with higher accuracy and confidence. Likewise, increasing measure accuracy offers more opportunities to perform effective inference. In addition to these parameters, in our indoor location inference scenario $\texttt{Priv}_{\mathcal{C}}(\mathcal{P})$ also depends on the number of users and their mobility patterns. The number of users represents inherent complexity of the tracking problem. In the case of only one user, location inference becomes trivial as occupancy amounts to the location of that user. However, when more users are present, it becomes generally more difficult to reconstruct location from occupancy since location traces of different users interleave with each other. The privacy function is a decreasing function of the parameters, which essentially presents a tradeoff with the utility function.

4.8.2 Adaptive Control

The goal of the sensing system is to achieve privacy-aware and context-aware adaptive control. In what follows, we discuss how the system can be configured under various contextual settings to operate in accordance to the utility-privacy tradeoff and specific requirements for performance.

Formally, given the utility function $\texttt{Util}_{\mathcal{C}}(\mathcal{P})$ and the privacy function $\texttt{Priv}_{\mathcal{C}}(\mathcal{P})$, we can formulate an optimization problem by combining two functions into a single objective function according to adjustable design goal

$$\mathcal{P}^{\star} = \text{argmax}_{\mathcal{P}} \left(u\texttt{Util}_{\mathcal{C}}(\mathcal{P}) + v\texttt{Priv}_{\mathcal{C}}(\mathcal{P}) \right), \tag{4.4}$$

where $u, v \geq 0$ are weighting constants specifying the system's emphasis over utility and privacy.

We study the indoor location privacy with respect to sensing interval and number of users. In the presence of

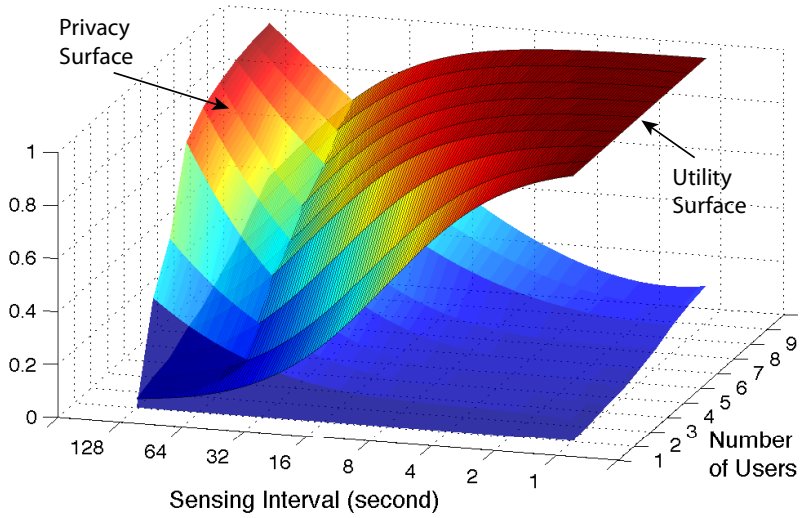


Figure 4.8: We depict the surface of privacy which has higher values with larger sensing intervals, and the surface of utility which has value 1 when the sensing interval is 1. The surfaces also evolve with the other dimension: the number of users. The system can adjust the sensing interval to achieve desired tradeoff between utility and privacy.

different population in the building, by controlling the sensing interval, the system can achieve different operation goals with respect to utility and privacy.

4.8.3 An Example

As a proof-of-concept example, we discuss how the occupancy detection system can operate under the adaptive control framework. In the last section, we present our results about inference accuracy. As explained, inference accuracy and user location privacy are two sides of the same coin. We can provide an empirical location privacy function computed using our results $\texttt{Priv}_{\mathcal{C}}(\mathcal{P}) = 1 - \texttt{tb\_acc}$. The parameter $\mathcal{P}$ is the sensing interval in this example, and the context $\mathcal{C}$ is the number of users.

Due to the absence of a formal study on utility under different sensing intervals and population, we construct the utility function according to several intuitions. For the purpose of illustrating the utility-privacy tradeoff and adaptive control, we simplify some details, and focus on high-level ideas. Though this carefree attitude might be controversial, we hope it can serve as the first dialogue on this topic. In spite of various interpretations of utility, we concentrate on energy efficiency. The intuitions about the utility function include but are not limited to the following.

1. Utility is normalized to 0 to 1.
2. Smaller sensing interval leads to more utility. There is a limiting property about the utility the system can

achieve.

3. The utility versus sensing interval relation changes with different numbers of users. With more users, the system inherently enjoys more information and more data utility.

We adopt the generalized logistic function to characterize data utility. In Figure 4.8, we plot the privacy function and utility function with respect to different contexts (number of users) and system parameters (sensing interval). Given an optimization goal defined in (4.4), the system can can continually adjust its sensing interval to achieve the desired tradeoff as the number of users changes.

4.9 Discussion

We have demonstrated that time-series occupancy data collected for environmental control in smart buildings leaks privacy-sensitive information about the building users. Toward quantitative privacy assessment, we propose a stochastic framework using the factorial hidden Markov model that captures the characteristics of user mobility and occupancy measurement. In our experiment with the real and synthetic data, we can achieve high inference accuracy with fine-grained occupancy data. More importantly, we show quantitatively the decrease of inference accuracy with respect to two factors in the sensing system: sensing interval and number of users. Based on our framework, the system can pursue fine privacy control by adjusting the sensing interval with respect to the number of users.

While our approach demonstrates the privacy leakage in smart buildings through data analytics, our results only scratch the surface of the potential privacy issues in heavily-sensed environments. The wealth of data collected in such environments goes far beyond occupancy data, suggesting that additional privacy risks exist and must be incorporated into the design of privacy-respecting management systems. Another valuable direction of further study is to conduct extensive experimentation and data collection at scale to validate our belief that the proposed techniques can scale to fit practical scenarios.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this dissertation, we have studied the behavior-based user identification and its application to real-world practice. The wide deployment of solid-state sensors and the advances in machine learning techniques offer unique opportunities for establishing identity from behavior related sensory data. We highlight advantages of the behavior-based approach compared to traditional mechanisms from both benign and adversarial perspectives, and propose a framework for identification through behavioral patterns. To demonstrate the practical implications of the approach, we identify two novel application scenarios in the mobile and IoT realms.

The first application studies cross-device user recognition. We show the feasibility of tracking a multi-device user based on the similarity of user behaviors while interacting with apps on different devices. The outcome offers opportunities for the optimization of recommendation, advertising and experience enhancement across multiple devices. Experimental results demonstrate that behavior-based identification can achieve satisfying performance under practical settings and provide added-value to mobile apps.

The second application embraces the emergence of IoT-based services, and investigates potential privacy breaches through building sensory data. We are able to leverage occupancy measurements to model the mobility patterns of building residents, and reveal their room-level locations. The study shows that some seemingly privacy-preserving sensory data might contain sensitive details.

The two case studies present benign and malicious applications of the proposed approach. They exemplify necessary procedures in modeling the sensory data. Our results show how the behavior-based user identification can be applied to real practice.

5.2 Future Work

Our results show that it is feasible to leverage user behaviors as identifiable information. Though the methodologies and algorithms explored in this research as well as the implementation details are specific to the proposed application scenarios, they can be reused or extended to facilitate new research in new applications. We specify some promising future research directions and areas for improvement.

5.2.1 New Application Scenarios

The rise of cyber-physical systems greatly enhance the penetration of sensor-equipped devices into daily life. We are expected to embrace new hardware, new software and new application scenarios. Following this trend, it is inevitable that more information relating to user activities is exposed to ubiquitous sensing. It is tempting to come up with new applications of the behavior-based identification approaches under different practical contexts.

5.2.2 Collaborative Sensing Across Sensors

Our work on cross-device user recognition investigates identification through sensory readings from different sources. Though the data come from different devices, they are still measured by the same type of sensors. An interesting problem arises when two data sources are different types of sensors observing the same behaviors. The study can be useful when there are no same set of sensors. Obviously, the sensory readings from different sensors exhibit differences even for the same behaviors. The degree of the difference depends on sensor types. For example, the response of an accelerometer can be similar to that of a geophone. However, the data from a camera can be totally different. It is important to model the similarity and the linkage between different data. A potential approach is to train a classification model to determine if two pieces of data correspond to the behaviors of the same user.

5.2.3 Long-Term Characteristics of Behavior

Some behaviors studied in this work are short-term behaviors. The duration can be as short as a few seconds. The behaviors are subject to changes because of variations in the environment or due to other changes. It is important to study how those short-term behaviors might evolve in the long run. The results can provide insights about the necessary freshness of models. Also, the behaviors might exhibit seasonal variations. The new dimension of time introduces a new degree of freedom and requires proper modeling to tackle the changes in that dimension.

5.2.4 New Representations and Algorithms

Deep learning has received enormous attention and achieved tremendous success in many practical application areas. The way deep learning extract high level features from raw sensory data might be helpful in improving the behavior-based identification. In our current approach, we handcraft features using domain knowledge. We can also simply evaluate the effectiveness of specific features using standard methods. With deep learning, we can feed raw sensory data to the neural networks for automatic feature extraction. It learns an adequate representation of the raw data that contains discriminative information for distinguishing different users.

Bibliography

[1] A. Abbasi, A. Khonsari, and M. S. Talebi. Source location anonymity for sensor networks. In *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, pages 1–5. IEEE, 2009.

[2] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *CCS*, pages 674–689, 2014.

[3] Y. Agarwal, B. Balaji, R. Gupta, J. Lyles, M. Wei, and T. Weng. Occupancy-driven energy management for smart building automation. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, pages 1–6. ACM, 2010.

[4] B. Alomair, A. Clark, J. Cuellar, and R. Poovendran. Toward a statistical framework for source anonymity in sensor networks. *Mobile Computing, IEEE Transactions on*, 12(2):248–260, 2013.

[5] T. R. Amy Mitchell and L. Christian. Mobile devices and news consumption: Some good signs for journalism.

[6] Y. Anzai. *Pattern Recognition & Machine Learning*. Elsevier, 2012.

[7] L. Backstrom, E. Sun, and C. Marlow. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th international conference on World wide web*, pages 61–70. ACM, 2010.

[8] B. Balaji, J. Xu, A. Nwokafor, R. Gupta, and Y. Agarwal. Sentinel: occupancy based hvac actuation using existing wifi infrastructure within commercial buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 17. ACM, 2013.

[9] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 280–293, 2009.

[10] S. P. Banerjee and D. L. Woodard. Biometric authentication and identification using keystroke dynamics: A survey. *Journal of Pattern Recognition Research*, 7(1):116–139, 2012.

[11] A. Beltran, V. L. Erickson, and A. E. Cerpa. Thermosense: Occupancy thermal based sensing for hvac control. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings*, pages 1–8. ACM, 2013.

[12] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive computing*, (1):46–55, 2003.

[13] A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. 2004.

[14] C. Bettini, X. S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In *Secure data management*, pages 185–199. Springer, 2005.

[15] C. Bettini, X. S. Wang, and S. Jajodia. Protecting privacy against location-based personal identification. In *Secure data management*, pages 185–199. Springer, 2005.

[16] C. Bo, L. Zhang, X.-Y. Li, Q. Huang, and Y. Wang. Silentsense: silent user identification via touch and movement behavioral biometrics. In *Proceedings of the 19th annual international conference on Mobile computing & networking*, pages 187–190. ACM, 2013.

[17] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[18] D. Bourgeois, C. Reinhart, and I. Macdonald. Adding advanced behavioural models in whole building energy simulation: a study on the total energy impact of manual and automated lighting control. *Energy and buildings*, 38(7):814–823, 2006.

[19] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[20] S. Chakraborty, K. R. Raghavan, M. P. Johnson, and M. B. Srivastava. A framework for context-aware privacy of sensor data on mobile systems. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, page 11. ACM, 2013.

[21] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 211–224. ACM, 2008.

[22] J. Cranshaw, E. Toch, J. Hong, A. Kittur, and N. Sadeh. Bridging the gap between physical location and online social networks. In *Proceedings of the 12th ACM international conference on Ubiquitous computing*, pages 119–128. ACM, 2010.

[23] A. De Luca, A. Hang, F. Brudy, C. Lindner, and H. Hussmann. Touch me once and i know it's you!: implicit authentication based on touch screen patterns. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 987–996. ACM, 2012.

[24] L. Denœud and A. Guénoche. Comparison of distance indices between partitions. In *Data Science and Classification*, pages 21–28. Springer, 2006.

[25] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi. Accelprint: Imperfections of accelerometers make smartphones trackable. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014.

[26] T. G. Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[27] B. Draffin, J. Zhu, and J. Zhang. Keysens: passive user authentication through micro-behavior modeling of soft keyboard interaction. In *Mobile Computing, Applications, and Services*, pages 184–201. Springer, 2014.

[28] Econsultancy. `https://econsultancy.com/blog/64464-more-than-40-of-online-adults-are-multi\`
`\-device-users-stats/`.

[29] M. Enev, J. Jung, L. Bo, X. Ren, and T. Kohno. Sensorsift: balancing sensor data privacy and utility in automated face understanding. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 149–158. ACM, 2012.

[30] Enlight. `http://enlightedinc.com/solutions/products/`.

[31] V. L. Erickson, S. Achleitner, and A. E. Cerpa. Poem: Power-efficient occupancy-based energy management system. In *Information Processing in Sensor Networks (IPSN), 2013 ACM/IEEE International Conference on*, pages 203–216. IEEE, 2013.

[32] V. L. Erickson, M. Á. Carreira-Perpiñán, and A. E. Cerpa. Observe: Occupancy-based system for efficient reduction of hvac energy. In *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on*, pages 258–269. IEEE, 2011.

[33] V. L. Erickson and A. E. Cerpa. Occupancy based demand response hvac control strategy. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, pages 7–12. ACM, 2010.

[34] T. Feng, Z. Liu, K.-A. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. K. Nguyen. Continuous mobile authentication using touchscreen gestures. In *Homeland Security (HST), 2012 IEEE Conference on Technologies for*, pages 451–456. IEEE, 2012.

[35] K. Framling, I. Oliver, J. Honkola, and J. Nyman. Smart spaces for ubiquitously smart buildings. In *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2009. UBICOMM'09. Third International Conference on*, pages 295–300. IEEE, 2009.

[36] M. Frank, R. Biedert, E.-D. Ma, I. Martinovic, and D. Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *Information Forensics and Security, IEEE Transactions on*, 8(1):136–148, 2013.

[37] Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine learning*, 29(2-3):245–273, 1997.

[38] S. K. Ghai, L. V. Thanayankizil, D. P. Seetharam, and D. Chakraborty. Occupancy detection in commercial buildings using opportunistic context sources. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pages 463–466. IEEE, 2012.

[39] GlobalWebIndex. 54% of tablet users share their device with others, 2014. https://www.globalwebindex.net/blog/tablet-sharing.

[40] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.

[41] M. Gruteser, G. Schelle, A. Jain, R. Han, and D. Grunwald. Privacy-aware location sensor networks. In *HotOS*, volume 3, pages 163–168, 2003.

[42] W. He, X. Liu, H. V. Nguyen, K. Nahrstedt, and T. Abdelzaher. Pda: privacy-preserving data aggregation for information collection. *ACM Transactions on Sensor Networks (TOSN)*, 8(1):6, 2011.

[43] U. Hengartner and P. Steenkiste. Avoiding privacy violations caused by context-sensitive services. *Pervasive and mobile computing*, 2(4):427–452, 2006.

[44] T. W. Hnat, E. Griffiths, R. Dawson, and K. Whitehouse. Doorjamb: unobtrusive room-level tracking of people in homes using doorway sensors. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 309–322. ACM, 2012.

[45] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 194–205. IEEE, 2005.

[46] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 194–205. IEEE, 2005.

[47] A. Jain and V. Kanhangad. Exploring orientation and accelerometer sensor data for personal authentication in smartphones using touchscreen gestures. *Pattern Recognition Letters*, 2015.

[48] p. Jan Petzold, year=2004. Augsburg indoor location tracking benchmarks.

[49] A. K. Karlson, B. R. Meyers, A. Jacobs, P. Johns, and S. K. Kane. Working overtime: Patterns of smartphone and pc usage in the day of an information worker. In *Pervasive computing*, pages 398–405. Springer, 2009.

[50] J. Kleissl and Y. Agarwal. Cyber-physical energy systems: focus on smart buildings. In *Proceedings of the 47th Design Automation Conference*, pages 749–754. ACM, 2010.

[51] J. Kołodziej, S. U. Khan, L. Wang, N. Min-Allah, S. A. Madani, N. Ghani, and H. Li. An application of markov jump process model for activity-based indoor mobility prediction in wireless networks. In *Frontiers of Information Technology (FIT), 2011*, pages 51–56. IEEE, 2011.

[52] J. Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[53] L. Li, X. Zhao, and G. Xue. Unobservable re-authentication for smartphones. In *NDSS*, 2013.

[54] N. Li, N. Zhang, S. K. Das, and B. Thuraisingham. Privacy preservation in wireless sensor networks: A state-of-the-art survey. *Ad Hoc Networks*, 7(8):1501–1514, 2009.

[55] Y. Li and J. Ren. Preserving source-location privacy in wireless sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON'09. 6th Annual IEEE Communications Society Conference on*, pages 1–9. IEEE, 2009.

[56] D. Madigan, E. Einahrawy, R. P. Martin, W.-H. Ju, P. Krishnan, and A. Krishnakumar. Bayesian indoor positioning systems. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 2, pages 1217–1227. IEEE, 2005.

[57] F. Manzoor, Z. Cong, P. Stack, and K. Menzel. Tracking occupants and inventory items in buildings using rfid technology. In *the 18th International Conference on the Application of Computer Science and Mathematics in Architecture and Civil Engineering, Weimar, Germany*, 2009.

[58] P. McDaniel and S. McLaughlin. Security and privacy challenges in the smart grid. *IEEE Security & Privacy*, (3):75–77, 2009.

[59] K. Mehta, D. Liu, and M. Wright. Location privacy in sensor networks against a global eavesdropper. In *Network Protocols, 2007. ICNP 2007. IEEE International Conference on*, pages 314–323. IEEE, 2007.

[60] R. Melfi, B. Rosenblum, B. Nordman, and K. Christensen. Measuring building occupancy using existing network infrastructure. In *Green Computing Conference and Workshops (IGCC), 2011 International*, pages 1–8. IEEE, 2011.

[61] W. Meng, D. Wong, S. Furnell, and J. Zhou. Surveying the development of biometric user authentication on mobile phones. 2015.

[62] Y. Meng, D. S. Wong, et al. Design of touch dynamics based user authentication with an adaptive mechanism on mobile phones. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 1680–1687. ACM, 2014.

[63] Y. Meng, D. S. Wong, R. Schlegel, et al. Touch gestures based biometric authentication scheme for touchscreen mobile phones. In *Information Security and Cryptology*, pages 331–350. Springer, 2013.

[64] G. D. Montañez, R. W. White, and X. Huang. Cross-device search. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1669–1678. ACM, 2014.

[65] Y. H. News. `https://news.ycombinator.com/`.

[66] T. A. Nguyen and M. Aiello. Energy intelligent buildings based on user activity: A survey. *Energy and buildings*, 56:244–257, 2013.

[67] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Security and privacy (SP), 2013 IEEE symposium on*, pages 541–555. IEEE, 2013.

[68] OFCOM. `http://media.ofcom.org.uk/news/2012/uk-is-now-texting-more-than-talking/`.

[69] F. Oldewurtel, D. Sturzenegger, and M. Morari. Importance of occupancy information for building climate control. *Applied Energy*, 101:521–532, 2013.

[70] G. Pallapa, M. D. Francesco, and S. K. Das. Adaptive and context-aware privacy preservation schemes exploiting user interactions in pervasive environments. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–6. IEEE, 2012.

[71] G. Pallapa, N. Roy, and S. K. Das. A scheme for quantizing privacy in context-aware ubiquitous computing. In *Intelligent Environments, 2008 IET 4th International Conference on*, pages 1–8. IET, 2008.

[72] P. Pearce, A. P. Felt, G. Nunez, and D. Wagner. Addroid: Privilege separation for applications and advertisers in android. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, pages 71–72. ACM, 2012.

[73] A. P. Pons and P. Polak. Understanding user perspectives on biometric technology. *Communications of the ACM*, 51(9):115–118, 2008.

[74] S. R. Rajagopalan, L. Sankar, S. Mohajer, and H. V. Poor. Smart meter privacy: A utility-privacy framework. In *Smart Grid Communications (SmartGridComm), 2011 IEEE International Conference on*, pages 190–195. IEEE, 2011.

[75] H. Reader. `https://play.google.com/store/apps/details?id=com.xw.hackernews&hl=en`.

[76] Redwood. `http://redwoodsys.com/solutions`.

[77] C. Romesburg. *Cluster analysis for researchers*. 2004.

[78] A. Rowe, M. E. Berges, G. Bhatia, E. Goldman, R. Rajkumar, J. H. Garrett Jr, J. M. Moura, and L. Soibelman. Sensor andrew: Large-scale campus-wide sensing and actuation. *IBM Journal of Research and Development*, 55(1.2):6–1, 2011.

[79] N. Sae-Bae, N. Memon, K. Isbister, and K. Ahmed. Multitouch gesture-based authentication. *Information Forensics and Security, IEEE Transactions on*, 9(4):568–582, 2014.

[80] J. Scott, A. Bernheim Brush, J. Krumm, B. Meyers, M. Hazas, S. Hodges, and N. Villar. Preheat: controlling home heating using occupancy prediction. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 281–290. ACM, 2011.

[81] R. Shokri, G. Theodorakopoulos, J.-Y. Le Boudec, and J.-P. Hubaux. Quantifying location privacy. In *Security and privacy (sp), 2011 ieee symposium on*, pages 247–262. IEEE, 2011.

[82] R. L. Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953.

[83] T. Vu, A. Ashok, A. Baid, M. Gruteser, R. Howard, J. Lindqvist, P. Spasojevic, and J. Walling. Demo: user identification and authentication with capacitive touch communication. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pages 483–484. ACM, 2012.

[84] Y. Wang, X. Huang, and R. W. White. Characterizing and supporting cross-device search tasks. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 707–716. ACM, 2013.

[85] D. J. Watts and S. H. Strogatz. Collective dynamics of ?small-world?networks. *nature*, 393(6684):440–442, 1998.

[86] A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, L. Fang, Z. He, S. Lin, and J. Stankovic. Alarm-net: Wireless sensor networks for assisted-living and residential monitoring. *University of Virginia Computer Science Department Technical Report*, 2, 2006.

[87] H. Xu, Y. Zhou, and M. R. Lyu. Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones. In *Symposium On Usable Privacy and Security, SOUPS*, volume 14, pages 187–198, 2014.

[88] N. Xu. A survey of sensor network applications. *IEEE Communications Magazine*, 40(8):102–114, 2002.

[89] Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *IMC*, pages 329–344, 2011.

[90] T. Xu and Y. Cai. Feeling-based location privacy protection for location-based services. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 348–357. ACM, 2009.

[91] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer. Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach. In *2012 16th International Symposium on Wearable Computers*, pages 17–24, June 2012.

[92] J. Zhu, P. Wu, X. Wang, and J. Zhang. Sensec: Mobile security through passive sensing. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 1128–1133. IEEE, 2013.

[93] A. Zimek, E. Schubert, and H. Kriegel. A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining*, 5(5):363–387, 2012.