8-7-2018

PRIVACY LEAKAGE THROUGH SENSORY DATA ON SMART DEVICES

Yi Liang

PRIVACY LEAKAGE THROUGH SENSORY DATA ON SMART DEVICES

by

YI LIANG

Under the Direction of Zhipeng Cai, Ph.D. and Yingshu Li Ph.D.

ABSTRACT

Mobile devices are becoming more and more indispensable in people's daily life. They bring variety of conveniences. However, many privacy issues also arise along with the ubiquitous usage of smart devices. Nowadays, people rely on smart devices for business and work, thus much sensitive information is released. Although smart device manufactures spend much effort to provide system level strategies for privacy preservation, lots of studies have shown that these strategies are far from perfect. In this dissertation, many privacy risks are explored. Smart devices are becoming more and more powerful as more and more sensors are embedded into smart devices. In this thesis, the relationship between sensory data and a user's location information is analyzed first. A novel inference model and a corresponding algorithm are proposed to infer a user's location information solely based on sensory data. The proposed approach is validated towards real-world sensory data. Another privacy issue investigated in this thesis is the inference of user behaviors based on sensory data. From extensive experiment results, it is observed that there is a strong correlation between sensory data and the tap position on a smart device's screen. A sensory data collection app is developed to collect sensory data from more than 100 volunteers. A conventional neural network

based method is proposed to infer a user's input on a smart phone. The proposed inference model and algorithm are compared with several previous methods through extensive experiments. The results show that our method has much better accuracy. Furthermore, based on this inference model, several possible ways to steal private information are illustrated.

INDEX WORDS: Privacy preserving, Localization, Smart devices, Sensors, Data mining, Deep learning

PRIVACY LEAKAGE THROUGH SENSORY DATA ON SMART DEVICES

by

Yi Liang

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2018

PRIVACY LEAKAGE THROUGH SENSORY DATA ON SMART DEVICES

by

YI LIANG

Committee Chair: Zhipeng Cai

Committee: Yingshu Li

Wei Li

Donghyun Kim

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

August 2018

DEDICATION

This dissertation is dedicated to all my family members, supervisors, and friends. Specially, this dissertation is dedicated to my beloved wife Shanshan Yuan, for her unselfish support and love during my PhD study.

ACKNOWLEDGEMENTS

It would be impossible for me to finish the dissertation without the guidance of my committee members, help from my group, and support from my family and friends. I would like to show my great gratitude to all of them.

First and most important, I would like to give my deepest thanks to my advisors, Dr. Zhipeng Cai and Dr. Yingshu Li, for their generous guidance, encouragement, assistance, caring, patience, and support. Dr. Cai and Dr. Li always gave me the greatest support during my graduate study. They patiently inspired me and financially supported my research.

I would like to thank my committee members Dr. David Kim and Dr. Wei Li. They gave me constructive suggestions for my PhD proposal and dissertation. They also kindly provided me with valuable comments for my research.

I would like to thank other professors and staff members in our department, especially Ms. Tammie Dudley. Their warm-hearted help makes my life much more easier.

I would like to thank my group members. Special thanks go to Xu Zheng, Dongjin Miao, Yan Huang, Meng Han, Mingyuan Yan, Zhuojun Duan, Zaobo He, Ji Li, Ke Yan, Gesu Li, *etc.*, who provided me with help and happiness.

Last but not least, I would like to thank my family members for their continuing support, trust, and love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Smart devices and mobile apps are rolling out at swift speeds over the last decade, turning these devices into convenient and general-purpose computing platforms. However, smart devices also bring risks of privacy leakage. Sensory data collected from smart devices are important resources to nourish mobile services, and they are regarded as innocuous information which can be obtained without user permission and awareness. In this thesis, two privacy issues caused by sensory data collected from smart devices are investigated. The first one is location privacy leakage. The second one is deep learning based inference of private information.

In the first part of this dissertation, it is shown that only by using the data collected from the embedded sensors in smart devices instead of GPS data, a user's location information can be inferred with high accuracy. Three issues are addressed which are route identification, user localization in a specific route, and user localization in a bounded area. A Dynamic Time Warping based technique and a Hidden Markov Model are developed to solve the localization problem. Real experiments are performed to evaluate the proposed methods.

In the second part of this dissertation, it is shown that the seemingly innocuous sensory data could cause serious privacy threat. First, it is demonstrated that users' tap positions on the screens of smart devices can be identified based on sensory data by employing some deep learning techniques. Second, it is shown that tap stream profiles for each type of apps can be derived so that a user's app usage habit can be accurately inferred. The sensory data and app usage information of 102 volunteers were collected. The experiment results demonstrate that the inference accuracy of tap position can be at least 90% through utilizing a convolutional neural network based method. Furthermore, based on the inferred tap position information, it is shown that users' app usage habits and passwords may be inferred with high accuracy.

Chapter 2

RELATED WORK

Sensory data based privacy attacks should not very unfamiliar to us since this has been studied for many years in wireless sensor network [1, 2, 3, 4, 5, 6, 7, 8, 9] Although lots of existing studies focus on the privacy preserving issues of smart devices [10]. It was hard to find existing works exploring the possibility of privacy leakage of utilizing sensory data solely. In this section, the research progress and related literature are summarized.

2.1 Location Privacy Leakage through Sensory Data

There are many previous works trying to acquire user's privacy information by analyzing user‘s data collected from mobile devices. We roughly classify them into four categories based on the type of privacy information.

The first category focuses on mobile devices user's identity privacy preserving based [11, 12, 13]. Sanorita et al. [14] conducted extensive experiments to show that the accelerometer readings are identical for each user so that they can be used to infer user identity. While in [15], in order to identify an individual device, the speaker of a smart phone is used to construct the fingerprint of a user. Different from [15], the work in [16] proposed and implemented two approaches, one based on analyzing the frequency response of the speakerphone-microphone system and the other is based on studying device-specific accelerometer calibration errors to construct a fingerprint to de-anonymize mobile devices. No matter what embedded sensors they make use of to extract unique fingerprints of users, the works fully prove that user privacy is being threatened by smart phones.

The second category aims at getting users' location information [17, 18, 19, 20, 21]. Without making use of GPS information, Han et al. [22] proposed an approach to locate users only based on accelerometer readings. Their method firstly tries to reconstruct mo-

tion trajectory given the acceleration measurements collected from a user's phone. Then it matches the constructed trajectory with the map information to infer the user's location. Their work is similar to ours. However, their method is mainly based on probability and statistic models which need tremendous background information. Thus, it may have limited ability to infer location. Similar to [22], Lonut et al. [23] try to make use of a smart phone's accelerometer and electronic compass to get the moving speed and the direction so that they can construct a directional trail which can be matched with the local area map. In this way, they can infer a user's location based on the best matched path segment. But they need to use GPS information to get the initial location which cannot be satisfied in many situations. Two works related to open permission sensors have been proposed by Michalevsky et al. In [24], they argue that a smart phone's location greatly affects the power consumed by the phone's cellular radio which is the most power-intensive part. Thus, they can use a mobile device's aggregated power consumption profile to learn the location information based on the cellular radio map. But the power consumption in smart phones can be affected greatly and many factors such as playing game affect localization accuracy heavily. Moustafa et al. [25] designed a step counting method based on a lightweight finite state machine to estimate the walking distance so that they can track pedestrians. Their method is too simple to deal with complicated scenarios. The most popular methods to get user's location indoor without using GPS component is to utilize the WiFi signal. John, et.al [26] designed and implemented a system called LOCADIO to infer the motion and location of a user., This kind of works cannot work without WiFi device (outdoor). The work in [27] tries to explore the possibility of developing an electronic escort service by inferring the walking trail of a user. This work is not trying to get user's privacy information secretly. It requires users to share their location information with others which is not preferred by most users. Martin et al. [28] argue that logical location, which means location fingerprint characterized by surrounding sound, color, light, etc. can be captured by the embedded sensors in smart phones. They try to utilize location logical fingerprint matching to localize users indoor. It is obvious that their is infeasible outdoor. Different from all these works, our work is the

first one that combines two kinds of sensor readings to infer a user's location information outdoor without using GPS information.

All other privacy issues were considered in category three. These works open an interesting way to make use of in-built sensors to poach privacy information. Such as in [29], the authors proposed a method to steal the acoustic signals by using gyroscope in a smart phone. The work in [30] studies the feasibility of getting a user's tap inputs through motion sensors embedded in cell phones. Accelerometer is used at [31] to infer if the user is taking a metro.In this paper, they first extract the feature of the accelerometer sensor data, then utilize supervised learning based classifier to infer the interval of riding a metro. While [32] focuses on inferring a user's private information in Android system leveraging the system bugs.

Some other miscellany works were grouped into category four [33, 34, 35, 36, 37, 38, 39]. Attackers not only want to infer privacy information, but also try to do it efficiently [40]. In order to save energy, Yadav et al. [41] proposed their low cost GSM-based localization method based on Cell Broadcast Messages and war-driving. To tackle the problem that the Maximum Likelihood estimator for received signal strength (RSS) based localization is nonconvex. Robin, et al. [42] proposed an Semidefinite Programming (SDP) relaxation technique to solve this problem. Further, even some works has been proposed to improve the service quality instead of getting privacy information from user. Actually, using the integrated sensor in phone to monitor the road condition and traffic problem has been proposed by Prashanth[43], however, their work focus on the detecting rough road condition and traffic jam. In [44], the author argue that the slight localization error may cause inconvenient result, so they proposed that use accelerometer signatures to mark user's location to place mobile phone in a right context. However, the accelerometer signatures were just used as a side channel to give a more meaningful localization information for user when using GPS.

As we can see, most of the aforementioned related works either have strong assumptions about their application scenarios or hava limited inference ability.

2.2 Deep Learning Based Inference of Private Information Using Embedded Sensors in Smart Devices

Privacy preseving is always a big challenge in traditional social media [45, 46, 47, 48, 49, 50, 51] and also becoming a problem in mobile social media world as long with the development of smart devices [52, 53, 54]. By proposing a principled machine learning approach which only leverages accelerometer measurements, the work in [55] shows that accelerometer is a powerful side channel to infer user input on smartphones. Their experiment results show that 80% of their predictions are concentrated within an error of 0 or 1 key distance. This means it is easy to capture tap positions merely with measurements of accelerometer. Another similar work is presented in [56]. It is also based on the observation that tap on different locations of screens leads to distinguishable vibrations and motions of smartphones. The work in [56] makes use of this observation to infer users' sensitive input. The proposed method can correctly infer 70% of the input keys, which also validates the feasibility of using accelerometer to capture keystroke features. The work in [57] focuses on inferring tap positions. Same as [56], this work leverages the embedded sensors in smartphones to perform inference, while their method makes use of some machine learning techniques. Their work only focuses on the inference of tap positions, while our work associates tap positions with app inference. Instead of obtaining users' exact tap positions, the work in [58] investigates how to infer hand gestures such as which hand is holding a smartphone, which finger is touching the screen, *etc.*

Aviv *et. al.* [59] investigated how to utilize accelerometer to infer 4-digit PIN and android password pattern (gesture/swiping). Compared with gyroscope, their results show that accelerometer performs better in inferring user input. From their tremendous experiments, we can definitely believe that sensors in smartphones really threaten user privacy. By studying the correlation between tap position and gesture change in a tap action, the work in [60] shows the feasibility to infer user input with orientation sensory data. In the real scenario, a trojan app is installed along with a host app. The trojan app trains its intersection pattern when the host app is running. When a user is running other apps, the trojan app

stealthily monitor the sensory data to detect tap events. With the learned patterns, user input can then be inferred. The difficulties in these works mainly lie in tap event detection and the mapping between sensory data and tap positions.

Touch screens provide us with various interaction methods. In [61], the authors explored a new technique to detect side tap, which can provide more functions of touch screens. Their method is based on the built-in sensors and tap position inference. More generally, the work in [62] develops a framework to collect sensory data from smartphones to infer user privacy including users' emotion.

Besides the sensors embedded in smartphones, sensors in wearable devices also threaten user privacy. In [63], the authors explored the harmfulness brought by the sensors in smart watches. Sensory data are used to infer keystrokes when users wear smart watches and type the keyboards.

In the recent years, the deep learning methods have been widely employed in many areas. The work in [64] takes advantage of CNN to recognize human activities. The work in [65] also develops a CNN based feature learning system to address the human activity recognition problem based on raw time series data.

Even through many existing works proposed different methods to protect user from many aspects including social media [66, 67, 68, 69], networks [70, 71, 72, 73, 74, 75, 76, 77, **?**] and sensory data related privacy issues

Chapter 3

LOCATION PRIVACY LEAKAGE THROUGH SENSORY DATA

3.1 Introduction

While people are enjoying the many benefits brought by mobile devices, people have to take the risk of losing privacy by leaking location information[78, 79, 80, 81]. Users' location information is excessively collected by third party apps, on which are heavily relied by users.

Such apps provide users with convenience, while threatening users' privacy. Location information is so sensitive that malicious adversaries can make use of location information to attack users or even threat the public. Therefore, location privacy has attracted tremendous attentions from researchers who are struggling to protect location privacy without degrading service qualities of third party Apps.

The most common way for apps to obtain location information is to get access to the GPS [82, 83] module in a mobile device. Thus, some methods proposed aiming at controlling the access to the GPS module to protect location privacy. In reality, third party apps need users to authorize the access to the GPS module so that users may control the tradeoff between service quality and privacy preservation. Such a strategy seems to provide satisfiable location privacy preserving. However, it has been pointed out in many works that without accessing GPS data, apps can still infer private information, such as input on touch-screen [30] and motion status [31], through the data collected by general embedded sensors in mobile devices [24, 22, 25, 41, 26, 14]. Unfortunately, few works try to utilize built-in sensors like accelerometer, magnetometer, gyroscope, *etc.* to do localization due to the limitation of those sensors. These sensors are very sensitive and their readings may contains lots of noises due to stochastic events such as tiny vibrations of mobile devices. Thus it is extremely challenging to infer location information merely based on noisy sensory data. However, combined with reasonable background knowledge, readings from these sensors can

be utilized to infer a user's location information. Such sensor readings are considered non-sensitive and can be obtained without user permission, which causes a big threat to location privacy[84].

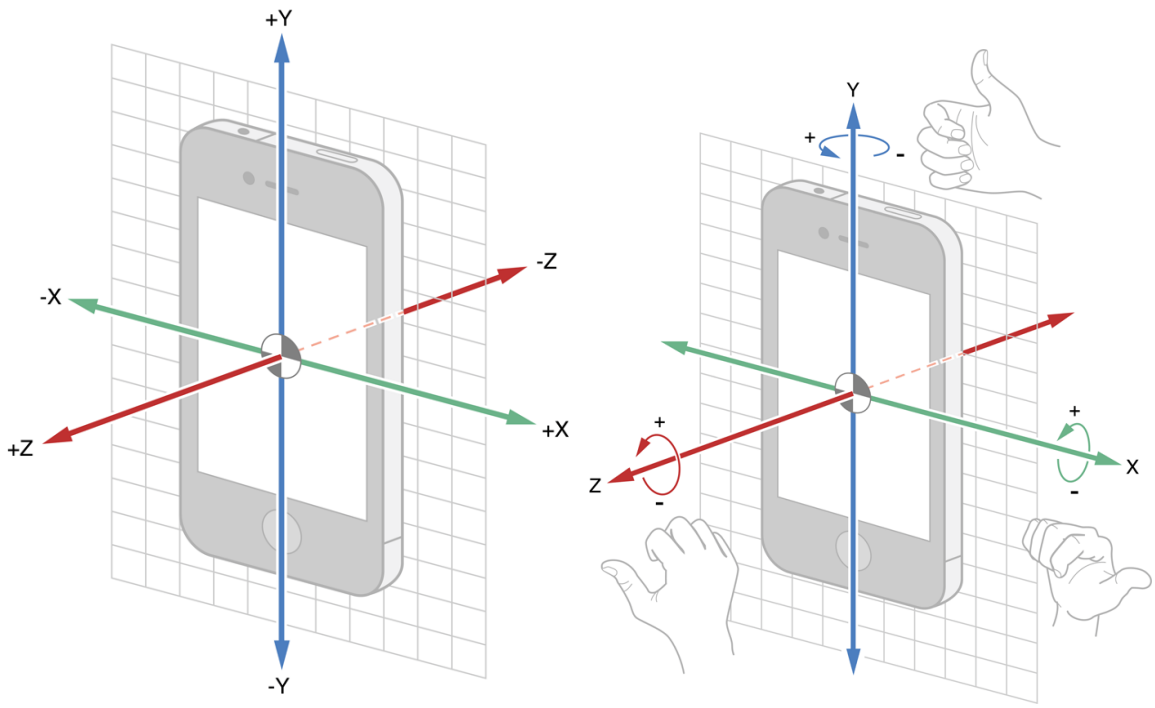
In this paper, a novel method is proposed to infer a user's location, which only utilizes the sensory data collected from the accelerometer and gyroscope in a mobile device. Such data can even be collected easily without users' awareness [85]. Our work is inspired by the fact that sensor readings are highly related to the route a user is taking, which can reveal the user's location. Besides, most people generally have relatively stable life patterns in their daily lives. We then take driving pattern as a case study in this work. We believe driving pattern is unique for each person based on our observation and experience. We take advantage of this feature to infer users' location information through unique fingerprints collected from people's daily lives. Regarding driving pattern, we have the following observations. It is very common that a person takes the same route to go to work/school or go home at specific time every weekday. Also, a person may be jammed on the same road segments everyday. There are only several reasonable routes that people would like to choose to drive to a specific destination. The time it takes everyday to drive to a particular location along the same route is almost the same. If adversaries can obtain the sensory data profiles for a set of known routes in advance, they can track a mobile device on those routes by secretly gathering sensory data from that device and matching it with the pre-recorded profiles. Based on these observations, we address the following three issues in this paper based on the sensory data collected from mobile devices.

1. Given a set of possible routes that a user would like to drive along everyday, how to decide which route the user is driving along?

2. Given the selected route of a user, how to infer the user's location in real time?

3. In a bounded area, how to trace a user?

These three problems are the location privacy issue in three different aspects, and the difficulty increasing along with the order. To our best of knowledge, this is the first work to



(a) Accelerometer (b) Gyroscope

Figure 3.1. Sensors in a smart phone.

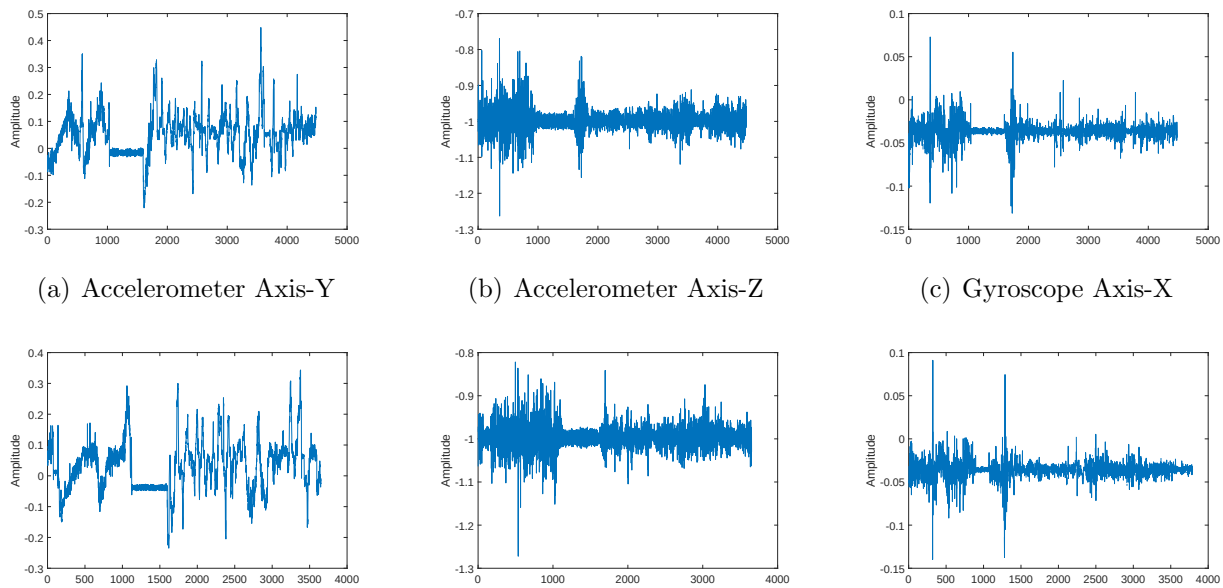
make use of sensory data collected from embedded sensors in mobile devices to infer location information without considering GPS data. Section 2 will discuss how we collect the data, followed by our attack model in section 3. Section 4,5,6 will give the detail of how we addressed these three questions separately. Section 7 will show our experiment result.

3.2 Sensory Data Collected by Mobile Devices

Some real data was collected to validate and depict our observations. We adopt smart phones with accelerometer and gyroscope as our mobile devices. Almost every smart phone has at least such two kinds of sensors which have three axes x, y and z. Each axis represents a dimension of a smart phone as shown in Fig.3.1.

We collected the sensory data of several routes for 10 days continously and all the sensory data of the same route show similarity. Fig.3.2 shows an example data set for one route, which was collected for 2 different days. Fig.3.2(a), Fig.3.2(b) and Fig.3.2(c) are for

one day. Fig.3.2(d), Fig.3.2(e) and Fig.3.2(f) are for the other day. As we can see, the data patterns for these two different days demonstrate high similarity. The X-axis of each subfigure in Fig.3.2 represents the total time to collect the data. These two days have different total data collection durations because the driving speeds in these two days are different. Then we can tell that even with different driving speeds in different days, as long as it is for a same route, similar data patterns always present. This is to say, each route has its unique data pattern. Such a fact assures that we can definitely infer location information through sensory data collected from the sensors embedded in mobile devices without accessing GPS data. Our extensive experiment results in Section 3.7 also validate that.



(a) Accelerometer Axis-Y (b) Accelerometer Axis-Z (c) Gyroscope Axis-X

(d) Accelerometer Axis-Y (e) Accelerometer Axis-Z (f) Gyroscope Axis-X

Figure 3.2. Sensory data showing similarity for a same route

To figure out which kind of sensory data can characterize driving pattern for a specific route is a fundamental issue. We take accelerometer and gyroscope, which are two common sensing units in a mobile device, as two representative kinds of sensors in this work. Accelerometer can measure linear acceleration and gyroscope can track angular velocity of three axes of a smart phone as shown in Fig.3.1. Actions such as speeding up, breaking, turning left/right are the most common driving actions. All such actions can be precisely

captured by linear acceleration and angular velocity which can be conveniently measured by accelerometer and gyroscope respectively. We conducted extensive experiments to prove our hypothesis. In our experiments, each smart phone was placed on the dashboard of a car with screen facing up and the positive Y-axis of accelerometer towards the driving direction. Note that a smart phone is not necessary to be placed in this way in real applications. Our purpose is to simplify the experiments. As depicted in Fig.3.2, the crests in Fig.3.2(a) and Fig.3.2(d) represent breaks, while the troughs represents accelerations. The crests in Fig.3.2(c) and Fig.3.2(f) represent left turns, and the troughs represent right turns. We can see that breaks and accelerations can be captured by the Y-axis readings of the accelerometer, steering can be captured by the X-axis readings of the accelerometer and the X-axis and Y-axis readings of the gyroscope, and road conditions (bump, downhill, slope *etc.*) can be captured by the Z-axis readings of the accelerometer.

Fig.3.3 shows the Z-axis readings of the gyroscope for some sharp/slow turns made at the same intersection. We can see that the only difference between sharp turns and slow turns is the shape of the corresponding peaks which are greatly different from that of the line representing no turning. This example indicates that even if people have different driving habits, the resultant data for a same action present the same pattern and the only difference lies in the actual values, which means that driving behaviors may affect sensory readings but have no impact on data patterns. Therefore, we can infer one's location information given pre-collected data for targeted routes.

Different sensory readings have different usefulness. Initially, we used Y-axis readings of accelerometer to infer one's accurate location because Y-axis readings of accelerometer can capture the break actions. Unfortunately, we find this method is not quite effective for localization in practice, as one may break arbitrarily anywhere in a road segment, making it hard to precisely locate a user. This is because even if the road conditions are the same every day, the real time traffic conditions affecting one's driving speed may be quite different. Sometimes, Y-axis readings of accelerometer may even hinder us from locating users. However, we find that a user may have a same break frequency or speed up frequency on some

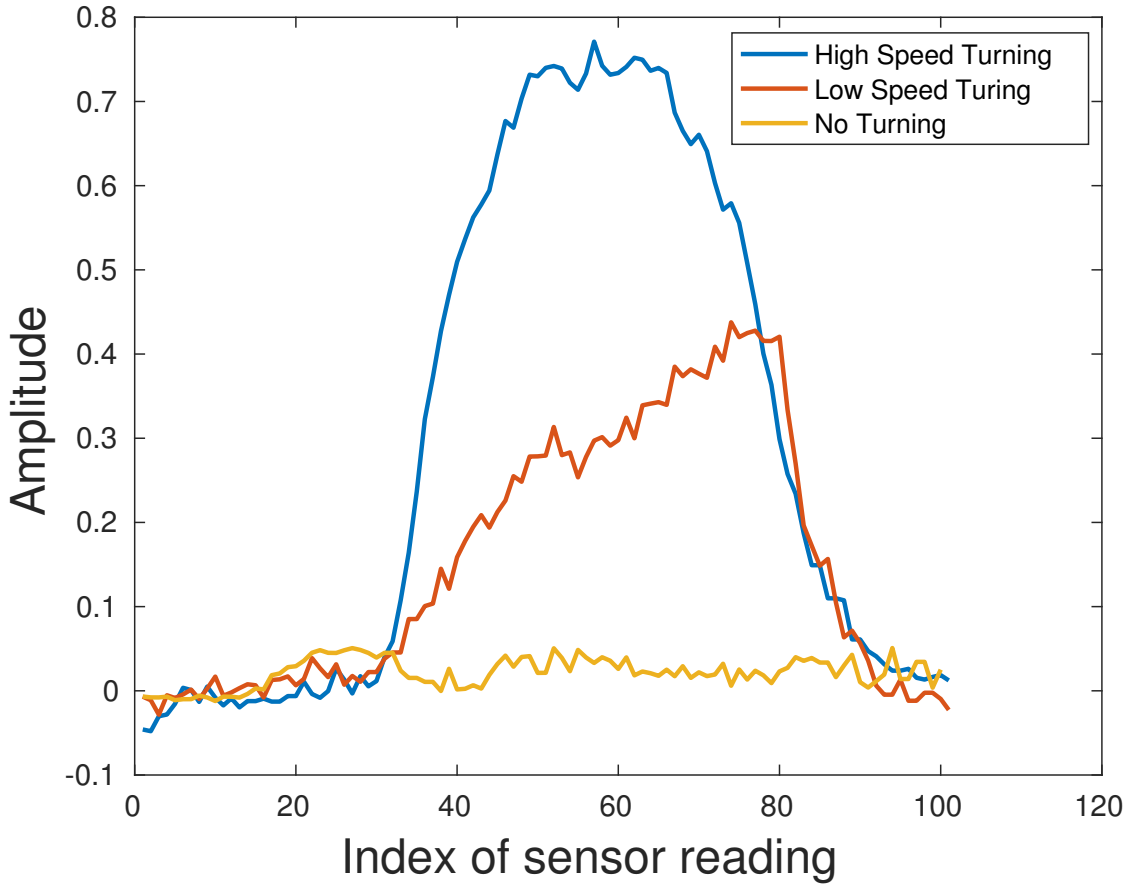


Figure 3.3. Influence of driving habit.

particular road segment which can help with route identification. For example, one may break very frequently on a particular road segment resulting in many crests in the collected data, and such a pattern is useful for identifying this road segment. Therefore, we make use of Y-axis readings of accelerometer for route identification, not for location inference. We also find that road conditions are generally stable, as the locations of downhill, uphill, and intersections one need to make turns are the same for each route.

Thanks to the aforementioned observations, we are able to locate mobile devices based on the sensory data which can be easily collected. From the sensory data, we can also extract unique fingerprints to identify road segments with high accuracy.

3.3 Attack Model

We have no special requirements regarding the attack model and our attack model is very reasonable compared with the ones in the previous works. Basically, there are just two roles in our attack model, attackers and users. Attackers are the adversarial app providers, users is the one who installed these adversarial apps. An attacker tries to obtain a user's location information secretely, assuming the attacker has successfully attracted the user to install malicious Apps on the mobile device. Then, the attacker can easily collect the user's sensory data because many sensors like accelerometer and gyroscope can be accessed by malicious Apps without user permissions. The only requirement is that the malicious Apps can upload the sensory data to the attacker's backend server through Internet so that the attacker can analyze the sensory data for location inference. All of these actions can be carried out without user's awareness. As mentioned before, sensory data collected by mobile devices may threaten privacy. Even worse, most users and many manufacturers have not even realized such a threat.

3.4 Route Identification

In this section, we explain how to identify a route, which is the first step towards location inference. Suppose users drive to work every weekday morning and the number of possible destinations for each user is limited. Moreover, for each destination, there are only a few reasonable routes that a user would like to take. All in all, the set of all the possible routes for each user is limited. From personal perspective and experience, we think this assumption is reasonable. If we can infer which route a user is taking, then to infer all the possible destinations for each user becomes possible which threatens user privacy.

Each route has relatively unique road conditions involving intersections, stop signs, traffic lights, *etc.* Then the resultant sensory data from a user can characterize each route. For example, a user, who goes to work every weekday, may be jammed on the same road segments and stop at the same places for traffic lights and stop signs. Then the corresponding

sensory data are unique and stable. Without loss of generality and for simplicity, we assume a known finite set of routes for each user. Each route in the set has a corresponding sensory data pattern as shown in Section II. We can collect the sensory data profile for each route beforehand, then we can compare a user's sensory data with the available profiles to identify routes. However, the following challenges present. For a specific route, the data collected on multiple days may be different because of real time driving speed and traffic conditions. Many unpredictable events may occur, which also results in data difference. Furthermore, the collected data may have noises caused by shaking of cars, slight movement of smart phones, *etc.* All these factors degrade the quality of the collected sensory data and make route identification even more challenging. Actually, route identification is to match sensory data patterns. If two sets of sensory data present a same pattern, we strongly believe they represent the same route. For a same route, since there are so many factors causing data difference, we cannot expect two sets of sensory data representing the same route to present exactly the same pattern. In order to address theses challenges, we first need to define similarity between two sets of sensory data. The data collection durations for different routes vary greatly. Then simple measurement Euclidean distance is obviously ineffective to measure similarity because Euclidean distance can only be used for phase aligned sequence. In order to accommodate noises and various data collection durations, it is better to consider the shape of a sequence of sensor readings for distinguishment.

3.4.1 Dynamic time warping

Dynamic Time Warping (DTW) is a powerful tool to measure a distance-like quantity between two time series which may vary in speed and duration [86]. The obtained distance-like quantity reflects the similarity between these two non-linearly aligned time series. This is exactly what we need for sensory data matching, since real-time traffic is unpredictable resulting in various data collection durations. Therefore, we employ DTW to find out along which route a user is driving given a set of possible routes that the user would like to drive along.

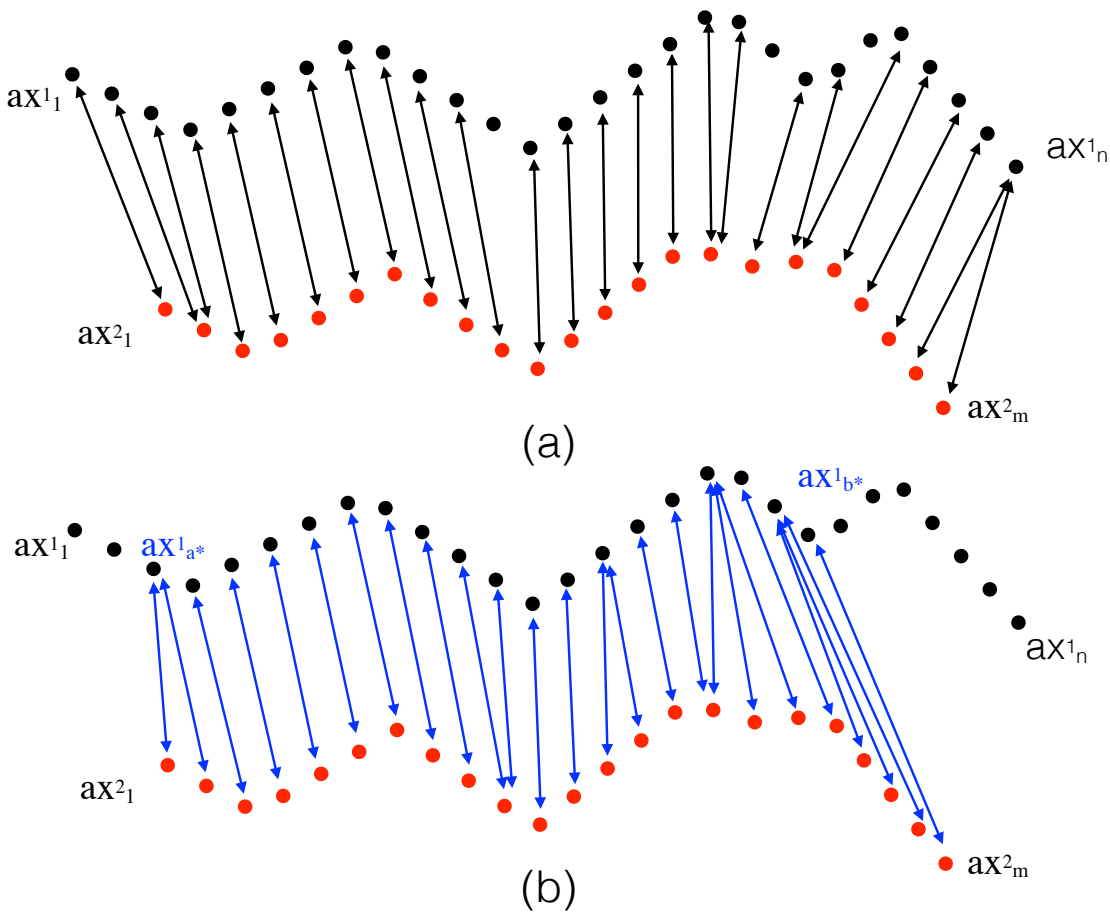


Figure 3.4. Two examples for DTW-based sequence alignment.

Let ax, ay, az and gx, gy, gz be the X, Y, Z axis values of accelerometer and gyroscope respectively. Assume $ax^1 = (ax^1_1, ax^1_2, ax^1_3, ..., ax^1_n)$ and $ax^2 = (ax^2_1, ax^2_2, ax^2_3, ..., ax^2_m)$ are an accelerometer's X-axis readings for two different days, where n and m are their collection durations respectively. If n and m are different, the Euclidian distance is not proper for measuring the similarity between these two sequences. Our primary task is to compare two sensor reading sequences collected on different days for a same route even if they have different collection durations. Then we define similarity based on a time warping path. First, we use an $n \times m$ matrix M to represent the point-to-point distance between two sensor reading sequences ax^1 and ax^2. Fig.3.4(a) shows two sensor reading sequences with similar data collection durations. Fig.3.4(b) shows two sensor reading sequences with much

different data collection durations. Entry M_{ij} in M indicates the way we align ax_i^1 and ax_j^2. Then we can derive a time warping path $P = (p_1, p_2, p_3, ..., p_K)$ to represent the alignment and matching relationship between ax^1 and ax^2, where $p_k = (i, j)$ $(1 \leq k \leq K)$ indicates the alignment and matching between ax_i^1 and ax_j^2 with $min(m, n) \leq K \leq m + n - 1$. The different data collection durations of different sensor reading sequences are resulted by the different driving speeds in different days. Since DTW can align multiple sensor readings in one sequence to a particular sensor reading of another sequence, we are able to successfully align two sequences with different data collection durations. Based on the obtained time warping path $P = (p_1, p_2, p_3, ..., p_K)$, we define the distance between two sensor reading sequences ax^1 and ax^2 as follows

$$Dist(ax^1, ax^2) = \sum_{k=1}^{K} d_k, \tag{3.1}$$

where d_k denotes the distance between ax_i^1 and ax_j^2.

We collected the sensory data along the 6 dimensions for each route in a set of routes for several days. These data are used as our training data. Then we computed the distance between the test data and our training data. For each dimension, we derive a similarity score between the test data and training data. The final similarity score for each route is the sum of these 6 similarity scores. The route with the smallest similarity score is the identified one that matches a route in the training data set. As sensory data have a lot of noises, we need to smooth the data before computing similarity scores. Furthermore, we use two classic methods to optimize the DTW algorithm whose time complexity is $O(nm)$, where n and m are the data collection durations for two sensor reading sequences. The first method is based on the fact that although the durations vary, their difference is limited. Suppose the maximum time duration difference for a same route is MD. Then we can reduce the searching space in our algorithm. Assuming that the sampling rate is r, each alignment and matching in $P = (p_1, p_2, p_3, ...p_K)$ does not exceed $r * MD * 60$. That is, for every $p_k = (i, j)$ $(1 \leq k \leq K)$

$$\max |i - j| \leq r * MD * 60. \tag{3.2}$$

For all of our testing routes, the maximum difference for a same route is 4 minutes. We limit the searching space within a bounded area to increase the searching speed. The second method is called the multi-scale DTW. Because we just identify the route with the smallest similarity distance as the result, exact similarity distance is not necessary. Then we can re-sample the sensory data sequences to reduce the dimension of matrix M. This method also substantially speeds up the searching speed.

3.5 Location Inference on a Particular Route

This section discusses how to locate a user on a particular route given real time sensory data. We assume an attacker knows which route along which a user is driving. In this case, the attacker can collect the data for a small road segment from malicious Apps installed in the user's mobile device. To locate a user, sub-sequence matching needs to be performed between real time sensory data and the data for the entire route. Here, real time sensory data is the test data, and the data for the entire route is the training data.

DTW can also be used for sub-sequence matching with minor modifications. For instance, for the ax dimension, a route's sensory data is $ax^1 = (ax^1_1, ax^1_2, ax^1_3, ..., ax^1_n)$ and the query segment's sensory data is $ax^2 = (ax^2_1, ax^2_2, ax^2_3, ...ax^2_m)$, where $n \gg m$. It is different from route identification in which the start and end points of ax^1 are aligned with the start and end points of ax^2 as shown in Fig.**??**(a). In sub-sequence matching, the start and end points of ax^1 can be aligned to any points in ax^2. That is, in route identification, $p_1 = (1, 1)$ and $p_K = (n, m)$. While for sub-sequence matching, this requirement is not necessary.

Our goal is to find sub-sequence $ax^1(a^* : b^*) = (ax^1_{a^*}, ax^1_{a^*+1}, ax^1_{a^*+2}, ..., ax^1_{b^*})$ with $1 \leq a^* \leq b^* \leq n$ minimizing the time warping distance to ax^2 over all possible sub-sequences of ax^1. In other words,

$$(a^*, b^*) = \underset{(1 \leq a \leq b \leq n)}{\arg\min} (Dist(ax^1(a : b), ax^2)). \tag{3.3}$$

The algorithm is pretty simple which can be found in [86]. Let $P^* = (p_1^*, p_2^*, p_3^*, ..., p_K^*)$ be the identified warping path, we have $p_1^* = (a^*, 1)$ and $p_K^* = (b^*, m)$ as shown in Fig.3.4(b). Then we can infer b^* is the current location of the user.

Roughly speaking, to infer a user's location, we employ the modified DTW algorithm to find the most likely sub-sequence along a route, and consider the end point of the sub-sequence as the inferred location of the user. The most challenging issue is that there may exist many similar sub-sequences along a very long route, *e.g.*, a user is driving along a highway with constant velocity. In this case, we need to take account in other information such as time and traffic conditions. The simplest method to deal with this issue is to consider the time difference. We assume the start time of a training sequence for the given route and the start time of the test sub-sequence are both known. According to all the training sequences for a given route, we can reduce the searching space to a specific range to reduce inference error. Even though we cannot completely eliminate such errors, in practice, the dynamically changing road and traffic conditions, sudden events, and climate reasons can all help with characterizing sequences. Then the number of the similar sub-sequences along a route is not large. In our experiments, such a issue does not present.

3.6 Location Inference in a Bounded Area

There are some works for tracing a user in a bounded area based on private location information of users without user awareness [24, 22]. However, some assumptions in these works may not be practical. For example, users may choose to detour due to traffic jam or emergencies. In this case, since there are no training data for the new route, it may be impossible to identify the route. Another challenge is that there are so many possible routes and it is impossible to collect the training data for all the possible routes. Furthermore, it is infeasible to compute the similarity distance between the query sensory data and the entire database data.

In order to develop a more general method for location inference, we employ a Hidden Markov Model (HMM). As shown in Fig.3.5, we split all the routes into small segments

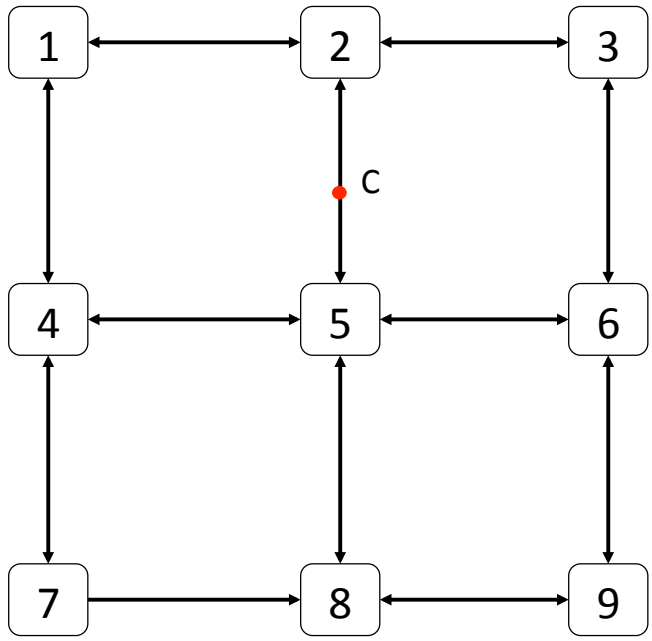


Figure 3.5. Road Map.

based on intersections. Each rounded rectangle represents an intersection and each arrow represents a road segment. Bidirectional arrows represent two-way roads, while directional arrows represent one-way roads. Let I denote the set of intersections and S denote the set of road segments. A road segment is denoted by $s = (i, j)$, indicating s is between intersection i and intersection j. We consider a road segment s as a state, and the transition probability of s is determined by s's outgoing degree. For example, from segment s_1, a user can go to segments s_2 and s_3. Then the transition probability from s_1 to s_2 or s_3 is 0.5. We may also define transition probability based on real time traffic. For example, 30% of cars go from s_1 to s_2, then the transition probability from s_1 to s_2 is 0.3. This method requires real time traffic information at each intersection and usually it is impractical. Thus, we make use of outgoing degree to define transition probability.

To calculate a user's probability of arriving at a particular location, we have the following strategy. As shown in Fig.3.5, when a user is driving along road segment $(2, 5)$, if the user passes intersection 5, the state changes from 2 to 5. Otherwise, suppose the user stops at

a particular location C on segment $(2, 5)$, and C is the location to be inferred. Then the probability of arriving at C is determined by the similarity distance between the observed segment sensory data for $(2, C)$ and the training sensory data for $(2, 5)$. The similarity distance can be computed using the DTW algorithm presented in the previous section. With this probability, we are able to identify the final location of the user on segment $(2, 5)$.

In summary, to trace a user in a bounded area, we first need to collect the sensory data for all the road segments within the area, which is possible since the number of the road segments in the bounded area is limited. Obviously, the route traversed by a user is a concatenation of a subset of all the road segments. Once we collect the sensory data from a user's mobile device, we can infer the most possible route traversed by the user. Then the user can be located in a bounded area. Following are the details of this location inference method in a bounded area.

3.6.1 The Modified Viterbi Algorithm

The viterbi algorithm [87] is a dynamic programming algorithm to find the most likely sequence of hidden states which generates the input sequence of observed events. In our work, each road segment is regarded as a hidden state, and the sensory data from a mobile device are regarded as the input sequence of observed events. Then given the sensory data of some road segments from a user's mobile device in terms of a sequence, the viterbi algorithm can help us identify the most likely route traversed by this user. However, the viterbi algorithm deals with discrete events, while our collected sensory data are continuous. Besides, the data collection duration for a road segment may not be fixed, and it is hard to find the intersections that divide the sensory data into road segments. Thus, we modify the viterbi algorithm so that it can be used to locate a user in a given area.

For each road segment, we have the corresponding training data. Let RD^i be the training data for road segment s_i. Let T^i_{min} and T^i_{max} be the shortest and longest time to traverse s_i respectively. T^i_{min} and T^i_{max} can be obtained from the training data of s_i. Let l_{ob} be the data collection duration of query route OB. The collected sensory data for OB,

which is a sensor reading sequence, correspond to the entire route traversed by the user and this route consists of road segments. The purpose is to identify the route traversed by the user through matching the sequence of OB with the training data. Our basic idea is to break OB into different road segments so that we can employ the modified viterbi algorithm to locate a user.

To break OB into road segments, at each stage, we need to cut OB utilizing the DTW algorithm. For example, at the first stage, we start from the first point of OB. Then for all the possible next road segments s_i where $i \in 1, 2, ..., m$, we compute the similarity distance between s_i and a sub-sequence of OB. Let this sub-sequence be $\hat{s}_i$. As we know, $\hat{s}_i$ starts from the first point of OB. Since we do not know the user's exact travel time of s_i, the end point of $\hat{s}_i$ could be reached in time T^i_{min} to T^i_{max} as mentioned above. We need to compute the similarity distance between all such possible $\hat{s}_i$ and s_i, and choose the smallest similarity distance as the similarity distance for s_i. The end point of $\hat{s}_i$ is the point that derives the smallest similarity distance for s_i. The end point of $\hat{s}_i$ is regarded as the start point at the next stage. Then the above process is repeated. In this way, we can break OB into road segments. Algorithm 1 is the pseudocode for the modified viterbi algorithm which breaks OB into road segments. The input of Algorithm 1 includes the initial probabilities π_i $(1 \leq i \leq n)$ for a user to start from road segment s_i, transition probability matrix T_{Prob} among road segments with size $K * K$, observed sensory data OB, and the training data RD^i for road segments s_i $(1 \leq i \leq n)$. Let MPS^t_i be the probability of road segment s_i being determined for stage t. After running Algorithm 1, we derive vector MPS_i for road segment s_i for all the stages. By tracing back from the final stage, we can obtain the most possible trajectory for OB.

Algorithm 2 is to determine the exact end point of a particular road segment $\hat{s}_i$ given the start point of $\hat{s}_i$. Let $Dist_i$ be the similarity distance between a $\hat{s}_i$ and the training data, and end_point_i be the end point of $\hat{s}_i$. mps_{i-1} is the end point of the previous stage and the step size τ is 1 second. Actually, we consider vector $Dist$ as the emission probability that generates the observation at each stage.

Algorithm 1: Modified Viterbi Algorithm

Require: π, T_{Prob}, OB, RD^i for each road segment s_i
Ensure: The most likely trajectory MPS for OB

```
for all stage t do
    if (t == 1) then
        [Dist, end_point_1] = FRS(1)
        MPS_i^t = max(π_i * Dist)
        Let the end_point be the start_point at the next stage
    else
        for all segment s_i do
            MPS_i^t = max(MPS_i^{t-1}) * T_Prob
        end for
        [Dist, end_point_t] = FRS(end_point_{t-1})
        MPS_i^t = MPS_i^t * Dist
    end if
end for
```

A user may finally stop at an intermediate point on a road segment. That is, the collected sensory data OB is a concatenation of several complete segments and a partial segment. However, Algorithm 1 can only derive a rounded sequence representing a set of complete road segments. Let us call the last complete road segment derived from Algorithm 1 as the final complete road segment, after which the next possible segment that the user would like to go must be an adjacent road segment of it. Then we can employ the method of

Algorithm 2: Find Road Segment (FRS) for Stage i

Require: End point mps_{i-1},
Ensure: Similarity distances for all possible road segments and corresponding end points mps_i

```
for all RD^i do
    if T_min^i + τ of RD^i is smaller than l_ob − mps_{i-1} then
        Dist_i = min_{τ∈[0, T_max^i − T_min^i]} Dist(ob(mps_{i-1} : T_min^i + τ), RD^i)
    else
        Dist_i = Infinity
    end if
    return Dist and mps_{i-1} + τ + 1
end for
```

location inference on a particular route introduced in the previous section to find the most likely sub-sequences for $OB(mps_{final} : l_{ob})$ on all the adjacent road segments. Among all these sub-sequences, we consider the one with the smallest similarity distance as the road segment that the user finally selects. The end point of this sub-sequence is the final location of the user.

3.7 Experiment Results

3.7.1 Data collection

All the data employed in our experiments are real data. For route identification, we drove around Atlanta, USA and Wuhan, China to collect data. The sensor data for 48 unique routes were collected, with 32 routes in Atlanta and 16 routes in Wuhan. The lengths of the routes vary from about 1 kilometer to 3 kilometers. All the data were collected by iphone 5, iphone 5s, iphone 6 plus, and iphone 6s. For a specific route, we collected its data in at least consecutive 5 days. Thus, we have at least 5 sensor data profiles for each route. For localization in a particular route, we collected the data for a very long route. For localization in a bounded area, we collected the data of all the road segments in a limited area located at the Decatur county in Atlanta, USA.

3.7.2 Assumptions

It is obvious that the way a smart phone is placed in a car greatly affects the collected sensor data. In our work, we assume that the query data follow the same dimensions of the collected training data. Even though different users may place their smart phones in different ways, the similarity between the sensor data for different days of the same route does not change. In our experiments, the only requirement is that a volunteer places the smart phone the same way everyday. Since it is easy to detect the position of a smart phone, we believe it is possible to project the sensor data into a uniform position coordinate system and this is out of the scope of this paper.

3.7.3 Route Identification

For route identification, 16 volunteers participated to collect sensor data along their daily routes. We do not have any strict requirements about the start time. We find it does not have much impact on the experiment results. Totally, we have 48 routes, some of which overlap with each other. For any pair of routes, the overlap rate is from 0 to 70%. However, we can still distinguish them efficiently. We also find that the longer the route, the easier to distinguish it from other routes, because the longer the route, the more unique features it has. For each route, the volunteers are required to collect sensor data for at least a week so that we can evaluate the impact of the size of the training data.

For the 48 routes, we have a testing set and a training set. The size of the testing set and the training set are both 48. Each route has only one profile. We run our algorithm for each route in the testing set. If our algorithm can identify the route in the training set, we consider it as successful. Our success rate is 100% for identifying a route. It indicates our method is effective in identifying a route even if the routes may overlap with each other, as our method makes use of the data collected from 6 dimensions which vividly depict the unique features of a route. Our method outperforms the work in [24] that employs power footprint collected from the base station. As the number of the reference profiles increases, we obtain better results even if we try to identify more unique routes.

3.7.4 Localization in a specific route

For localization in a specific route, we randomly select a long route which is about 20 kilometers, and collect 10 sensor data profiles for it. We choose one of the profiles as the training data. We randomly select another profile as the testing data. That means we only have one profile in each experiment. We split the testing data into several road segments as if they are collected from a user's smart phone in real time. First, we want to know that whether our method could distinguish road segments. We are also interested in the impact of the number of profiles. The results are shown in Table 3.1. When there are 10 road segments, the success rate is 84.2%. When there are 5 road segments, the success rate

increases to 90.3%. The main reason is that if there are only 5 road segments, each road segment is longer and more unique so that it is easier to distinguish them. If we increase the number of the profiles to 5, the success rate increases to 100% even if there are 10 road segments.

Table 3.1. Road segment Identification

# of road segments	# of profiles	Ave success rate
10	1	84.2 %
5	1	90.3%
10	5	100 %

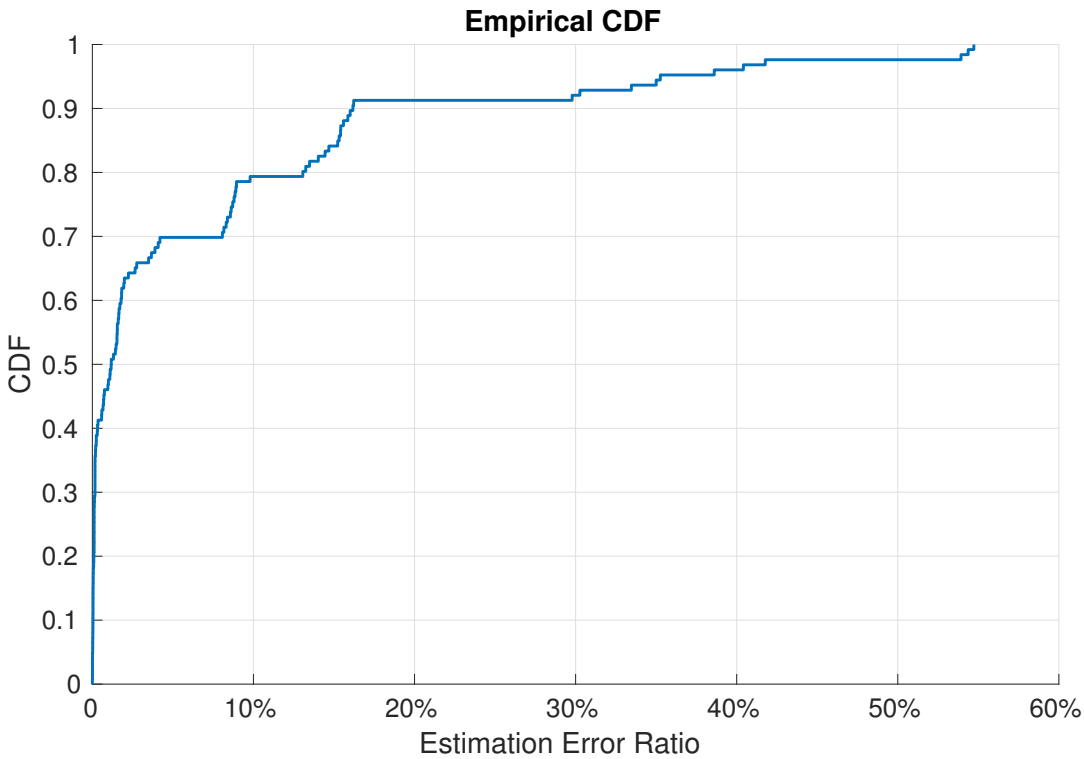


Figure 3.6. Estimation error cumulative distribution.

By using our algorithm, it is easy to know which road segment a user is traveling. However, we still want to locate a user more accurately. In our experiments, the minimum length of a subsequence is 1 minute, and the step size is 5 seconds. One of the 15 days' sensor

data is chosen as the testing data, and we randomly select another one as the training data. The total length of the route is 19 kilometers. The x-axis of Fig.3.6 shows the estimation error ratio with respect to the total length. It can be seen that almost 40% of the estimations are error free, and almost 80% of the estimations have an error less than 2 kilometers. Even though sometimes there are big errors, it can be avoided if we take time into consideration. Since we already know which route a user is traversing, based on the time information, we can narrow down the search space to avoid a big error.

For the impact of the size of training set, we fix the length of a subsequence as 4 minutes. We compute the location for each training route in the training set for the query subsequence, then the averaged location is regarded as the final estimation. As we increase the size of the training set, we can obtain a more accurate result. As shown in Fig.3.7, when we use more route profiles to localize a user, the average estimation error is reduced.

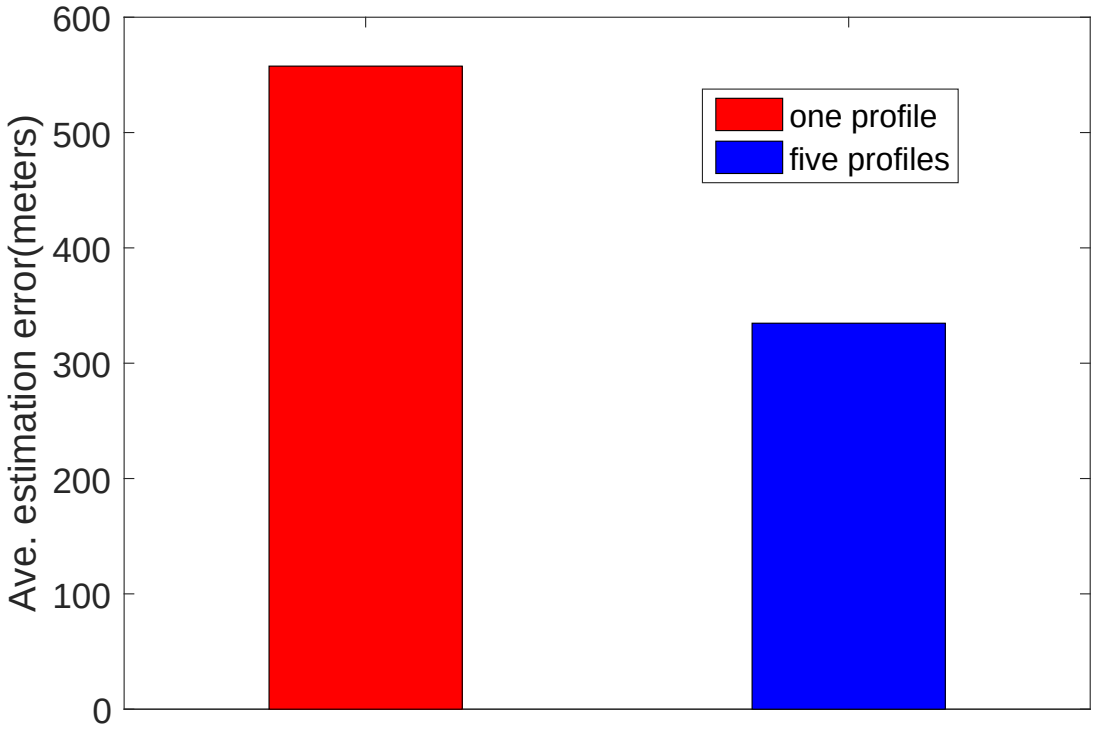


Figure 3.7. Localization accuracy with more training profiles.

3.7.5 Localization in a bounded area

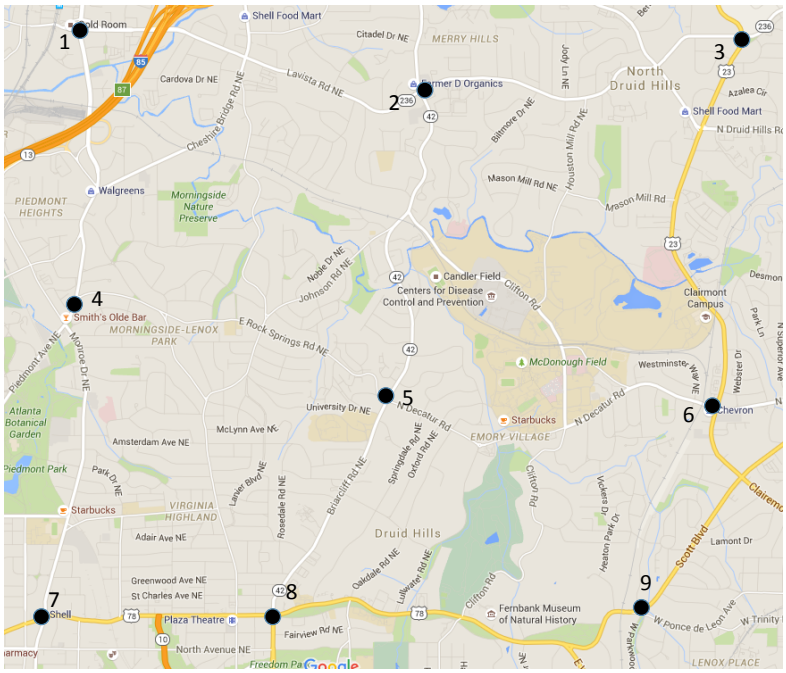


Figure 3.8. A bounded area.

For localization in a bounded area, we collected data from an area shown in Fig.3.8. This area locates at the center of Atlanta, USA. We select 9 intersections and 12 segments determined by these intersections. That means, in the HMM model, we have 12 states. The average length of the road segments is about 3 kilometers. We assume the probability of a road segment to be the starting segment of a user is 1/12. There are many methods to determine transition probability. In our experiments, we adopt the simplest one. The transition probability for road segments is evenly distributed over all the possible transitions. For each road segment, we collected at least one profile. One of them is chosen as the testing data, and the rest are considered as the training data. The probability of some sensor data to be related to a specific road segment can be calculated by the DTW algorithm. Actually, this is the observation probability. Now, we have the initial probability, transition probability, observation probability, then by using our method we can infer the route and location of a user.

First, we want to make sure that our method can successfully infer the route traversed by a user. For simplicity, we only consider one direction in the map which is from top left to bottom right. All the possible routes have been tested as listed in Table 3.2. We tested all the possible routes from intersection 1 to intersection 9. For a full route which means a car stops immediately after passing intersection 9, we want to know whether we can infer the route correctly. The results are shown in Table 3.2.

Table 3.2. Route Inference

Routes	# of profiles	Ave. Success Rate
1-2-3-5-9	3	95%
1-2-4-5-9	3	98%
1-2-4-8-9	3	100%
1-3-4-5-9	3	100%
1-3-6-8-9	3	100%

Basically, we can infer all the routes successfully, then we can know the final location of a user is at intersection 9. However, it is quite possible that a user may stop at any point in the area. The ultimate goal of our work is to infer a user's location. Thus, we also test some routes ending at any point in the area. Totally, we tested 200 sub routes of the full routes in the previous group of experiments. These sub routes are randomly taken from the full routes. The total length of each route is up to 11 kilometers. The idea is to infer the part of the route consisting of several complete road segments. We can get the most possible intersection that a user should be. As we know, there are many possible associated road segments for each intersection. By using the method introduced in Section 3.5, we can compute the similarity distance between the testing partial road segment and all the possible roads. The one with the minimum similarity distance is the inferred location of a user. The results are shown in Fig.3.9. It can be seen that almost 65% of the estimations are error free, and almost 86% of the estimations have an error less than 0.5 kilometer.

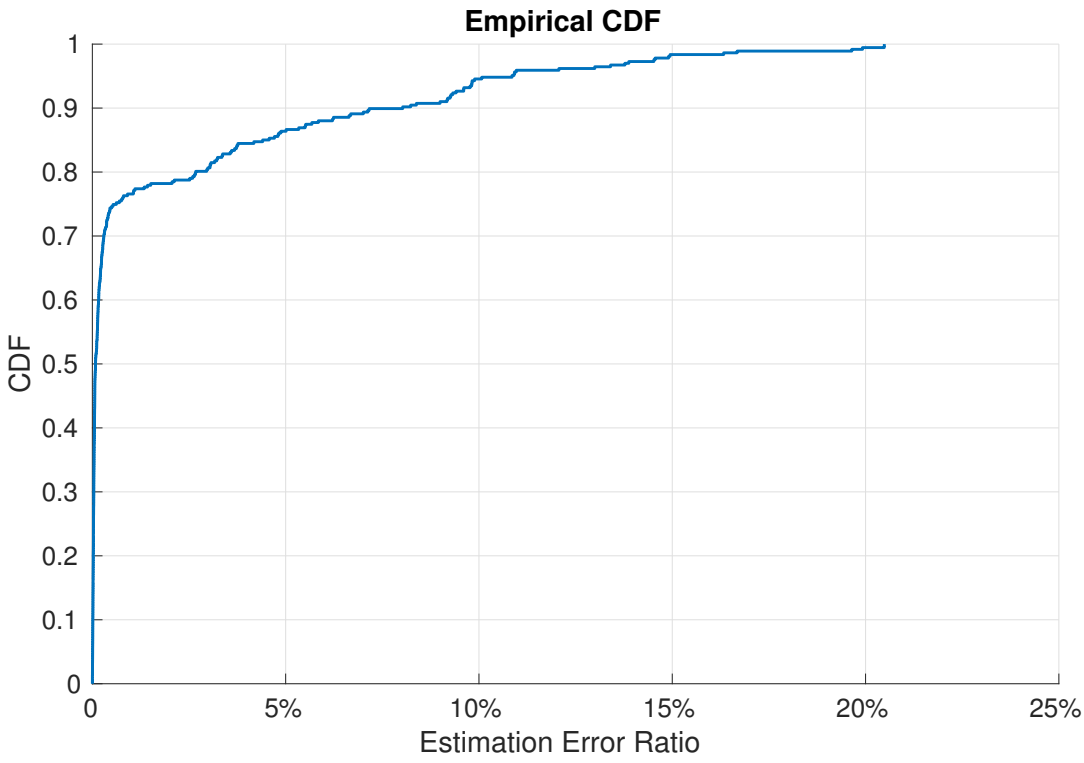


Figure 3.9. Estimation error for localization in a bounded area

Chapter 4

DEEP LEARNING BASED INFERENCE OF PRIVATE INFORMATION USING EMBEDDED SENSORS IN SMART DEVICES

4.1 Introduction

The usage of smart mobile devices for personal and business purposes grows increasingly in popularity over the last decade [88]. Smart mobile devices provide great convenience for people's daily lives. Unfortunately, mobile attacks have simultaneously exploded and become more sophisticated, especially when more and more users rely on mobile devices to manage their financial and personal data. Many services in smart devices are reliant on the sensory data collected from the sensors embedded in smart devices. Those sensory data nourish the mobile app design to provide incredibly convenient services to people, and the collected data are widely regarded as innocuous information. In most mobile platforms, such as IOS and android systems in smart phones, the sensory data readings are considered non-sensitive and can be easily collected without user permissions. Recent studies indicate that freely accessible built-in sensors can be easily utilized by adversaries to launch inference attacks [55, 56, 60, 62, 58, 89]. Even some system level defense mechanisms are proven to be ineffective. Some recent works mentioned that built-in sensors can be utilized to recognize human activities [90, 91] and infer screen based input [61, 60] based on the assumption that different human activities or gestures can create unique sensory data "patterns". The existing approaches extract patterns from raw sensor data directly and predict a user's pattern based on the same user's data, which is usually infeasible in practice. In this paper, we propose a novel approach to identify user tap position even if we do not have any historical sensory data of this particular user. Furthermore, we make use of the identified tap patterns for each type of apps to further discover users' app usage habits and daily life patterns, which are definitely private information to most users.

Intuitively, different types of apps incur different tap patterns due to the usage nature of each app. For instance, users type frequently and fast in chatting apps such as snapchat and wechat, while users scroll down/up on screens and type occasionally when reading news. Based on those observations, we first investigate how to infer tap positions based on sensory data, the inferred tap positions can then help with deriving tap sequences, according to which we can record the traces of tap events of users when they use smart devices. A tap sequence consists of a series of positions tapped by a user on a screen. Tap sequences can help with distinguishing different apps and predicting apps being used or used before, which is referred as a usage habit of a user. Note that usage habit information is private which can be taken advantage by adversaries to infer more private information such as age, gender, *etc.*

In order to achieve the aforementioned goals, we propose several models to encode tap sequences and the intervals between taps in a tap sequence. Specifically, n-gram is used to measure the similarity distance between two tap sequences and a few machine learning models are applied to recognize the app being used by a user. We validate our work towards the sensory data collected from 102 volunteers. The experimental results demonstrate that our proposed deep learning based method can predict tap sequences and infer app usage habits with high accuracy. The key contributions of our work are summarized as follows:

1. To the best of our knowledge, this is the first work to demonstrate that tapstream pattern can distinguish apps accurately, which reveals the fact that seemingly innocuous sensory data from smart devices can seriously threaten user privacy.

2. Several methods and models are proposed and evaluated in this paper. We first employ some traditional classification methods and deep neural network to recognize tap position sequences. Our experiment results demonstrate that deep learning, such as deep conventional neural network, is very effective for identifying tap position sequence. Furthermore, we propose a robust model to identify app usage of a user.

3. All the experimental data are collected from real traces. We develop a new app on the Android system and run it on 102 volunteers' smartphones. We "steal" data from

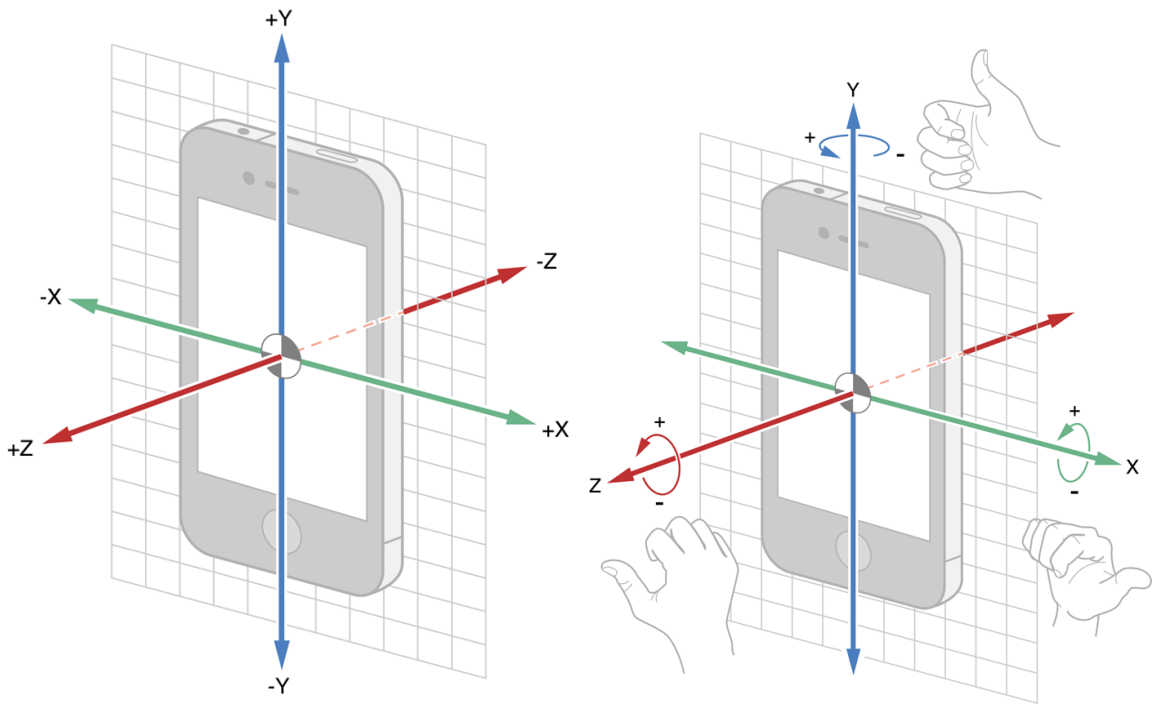
their smartphones in a nontrivial way when they use their smartphones and all the experiment results validate our hypothesis that sensory data can be easily utilized to carry out privacy attacks.

4.2 Background Knowledge

In this section, we introduce the characteristics of a smartphone and the reasons why the proposed approach could work. Then, we briefly introduce our attack model and the experiment data.

4.2.1 Sensors in smart devices

Our work is inspired by the previous works which try to infer the tap gesture on smartphones. With the emergence of smart devices with touch screens, users rely more and more on smart devices to deal with daily business, even for extremely private and sensitive business involving personal and financial data. In this paper, we use smartphone as a case study. Nowadays, off-the-shelf smartphones are equipped with sensors which can provide various interaction functionalities. The most common sensors include accelerometer, gyroscope, and rotation vector in Android based smartphones, as shown in Fig.4.1. An accelerometer sensor is used for measuring the linear accelerations for three axes x, y, and z, a gyroscope sensor can measure the angular velocity of the three axes, and a rotation vector sensor measures pitch, roll, and azimuth angle. Each kind of sensor can cover three dimensions. We denote the 9 dimensions of sensory data as A_x, A_y, A_z, G_x, G_y, G_z, R_x, R_y, and R_z respectively. R_x represents the rotation angle along the x-axis, R_y represents the rotation angle along the y-axis, and R_z represents the rotation angle along the z-axis. Tap actions performed on the screen of a smartphone can be easily captured by these sensors. The data sampling rate could be as high as 100Hz, while our data sampling rate is at most 20 Hz. Therefore, our method is applicable to almost all kinds of smartphones.



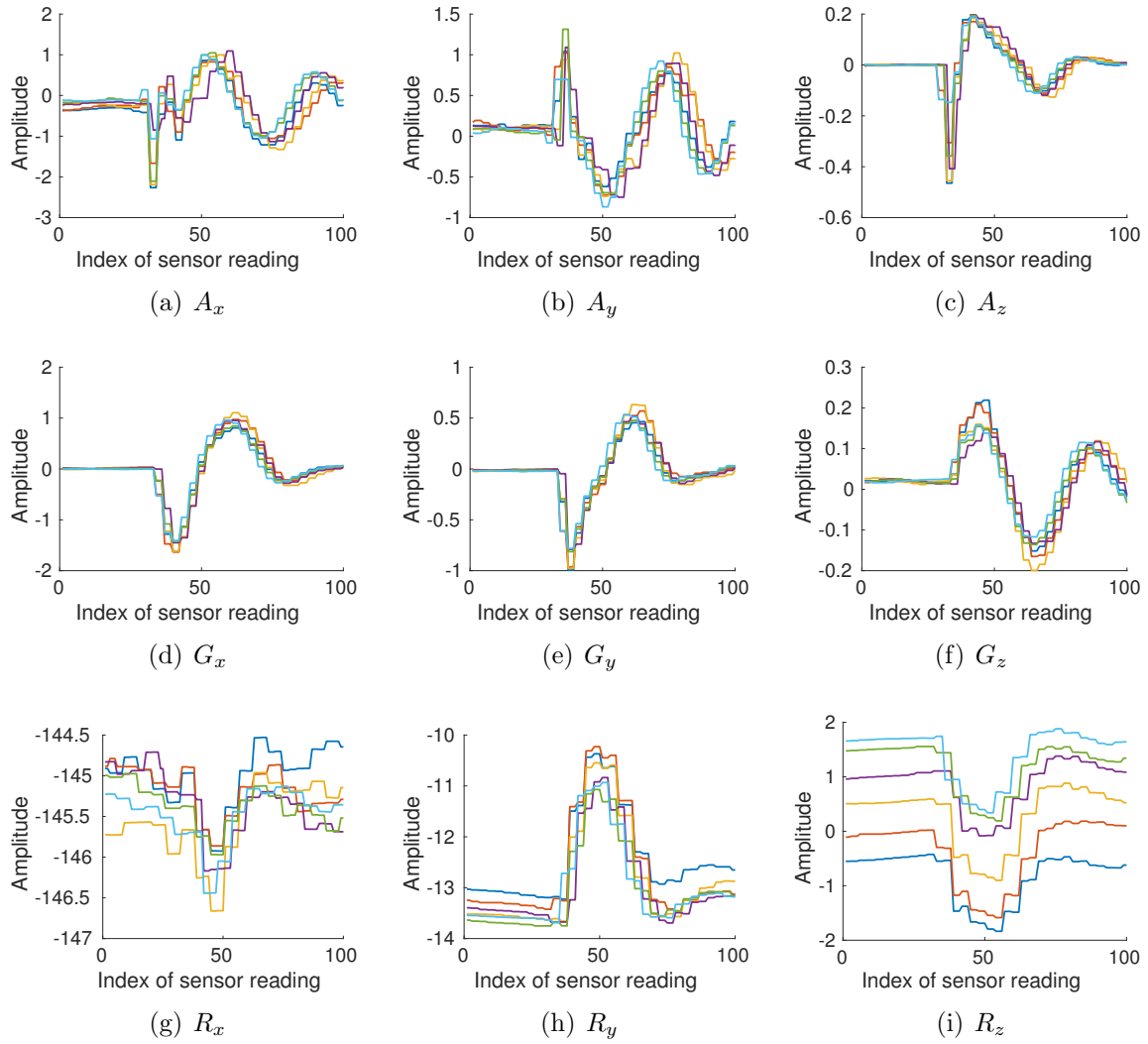
(a) Accelerometer (b) Gyroscope

Figure 4.1. Sensors in a smartphone.

4.2.2 Correlation between sensory data and tap position

It is well known that malicious apps can steal sensory data secretly because these sensors can be accessed without user permissions [92], resulting in serious privacy issues. In this paper, we first show that what we type through the soft keyboard, which is the most common input method in smartphones, can be inferred through sensory data. Note that, very sensitive information such as passwords, PINs, social security numbers, and credit card numbers are generally input through soft keyboard. Then, we demonstrate that current running apps can also be identified through mining sensory data. The information regarding running apps is also sensitive. Unfortunately, most users are not even aware of this fact.

It is not surprising that side channel information can be utilized by attackers [59, 55, 60, 57]. Lots of previous related works have shown that the changing angle and vibration of a touch screen on a smartphone are highly correlated to tap positions. In Fig.4.2, the lines with different colors represent multiple taps on the same position. Each sub-figure



(a) A_x (b) A_y (c) A_z

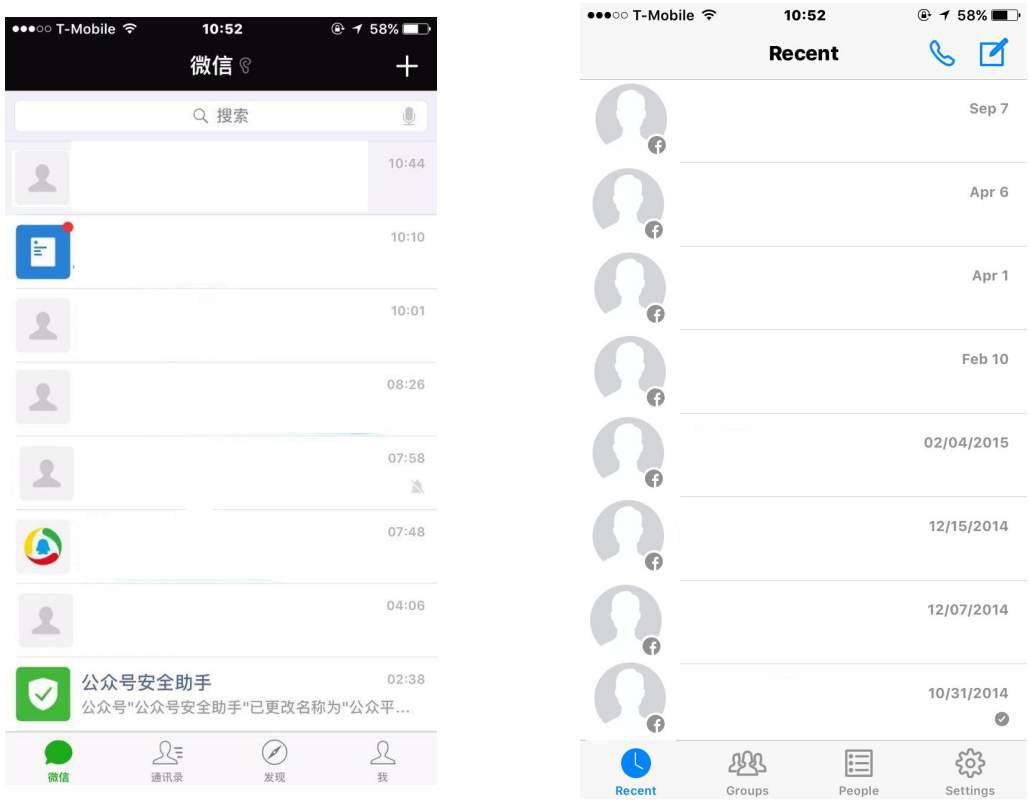
(d) G_x (e) G_y (f) G_z

(g) R_x (h) R_y (i) R_z

Figure 4.2. Similarity of sensory data for a same tap position.

depicts a specific dimension of a particular kind of sensor. Similarity presents in each of the 9 dimensions of sensory data. Then we derive the observation that a tap position has unique sensory data patterns, and different tap positions have different sensory data patterns (also refer to Fig.4.6 and Fig.4.7). Based on this observation, we can try to infer a user's tap position. Our experiment results show that we are able to infer the keys typed in through a number-only soft keyboard with 99% accuracy.

Furthermore, we observe that same kind of apps share similar user interface layouts, *e.g.*, chatting apps such as *WeChat* and *Messenger* as shown in Fig.4.3. Users carry out similar



(a) Wechat (b) Messenger

Figure 4.3. Similar layout of chatting apps.

actions for a same kind of apps. For example, when a user uses *WeChat* or *Messenger*, the general procedure is to start the app, select a friend, and type words, then go back to the friend list. In news browsing apps, such as *BBC*, *CNN*, and *New York Times*, users scroll down/up on screens, select an interested news and read it, then go back to the news selection menu. We believe that the similarity of the same kind of apps is unique and can be used to distinguish different kinds of apps. Our experiment results also validate this fact. Moreover, we believe that what apps a user uses is relevant to some private properties such as gender and age. In this paper, we try to explore the feasibility of inferring app usage habits based on sensory data collected from smart devices. In other words, based on the unique behavior pattern for each type of apps, we try to infer what apps a user is using. Then, we can further infer some private information of a user according to the user's app usage habits [62].

4.2.3 Attack model

In our attack model, we assume users have been tricked to install our malicious app on their smartphones so that we can collect their sensory data [63]. The most common way is to develop an app similar to a popular paid app and make it free in the Android app store. Lots of careless users will be tricked. Once a user launches the malicious app, the app starts to collect sensory data secretly. Then the malicious app can send the sensory data back to our back-end to train our inference models and launch inference attacks.

Our system consists of several components including tap detection, keystroke recognition, tap position inference, tap sequence pattern recognition, and app inference, as shown in Fig.4.4. Initially, a tap event is captured by the tap detection component when a user taps on the screen. Then lots of features can be extracted from the sensory data and easily associated with tap positions during our training process. If a user switches to another app, we can detect a tap event and record sensory data. The sensory data will be compared with our training data to infer tap positions.

To infer app usage of a user, we can record all the sensory data when the user uses an app. Based on our tap position inference results, we can derive a tap sequence representing a unique pattern for each kind of apps. As long as we collect enough training data and tune our inference model well enough, this inference model can be used to infer an app.

All of this is just an outline of our idea, there is lots of challenges on our way to our final result, we are going to show you step by step in following sections.

4.2.4 Sensory Data Collection

To collect sensory data from smartphones secretly, we design and implement a trojan app, named *Informer*, on the Android platform which has two parts, *sensor reading service* and *host app*.

Sensor reading service is responsible for gathering sensory data from smartphones. *Host app* is a luringly installed malicious app such as tools, media, and games. We can "steal" sensory data from users' smartphones without users' notice because the sensors in smart-

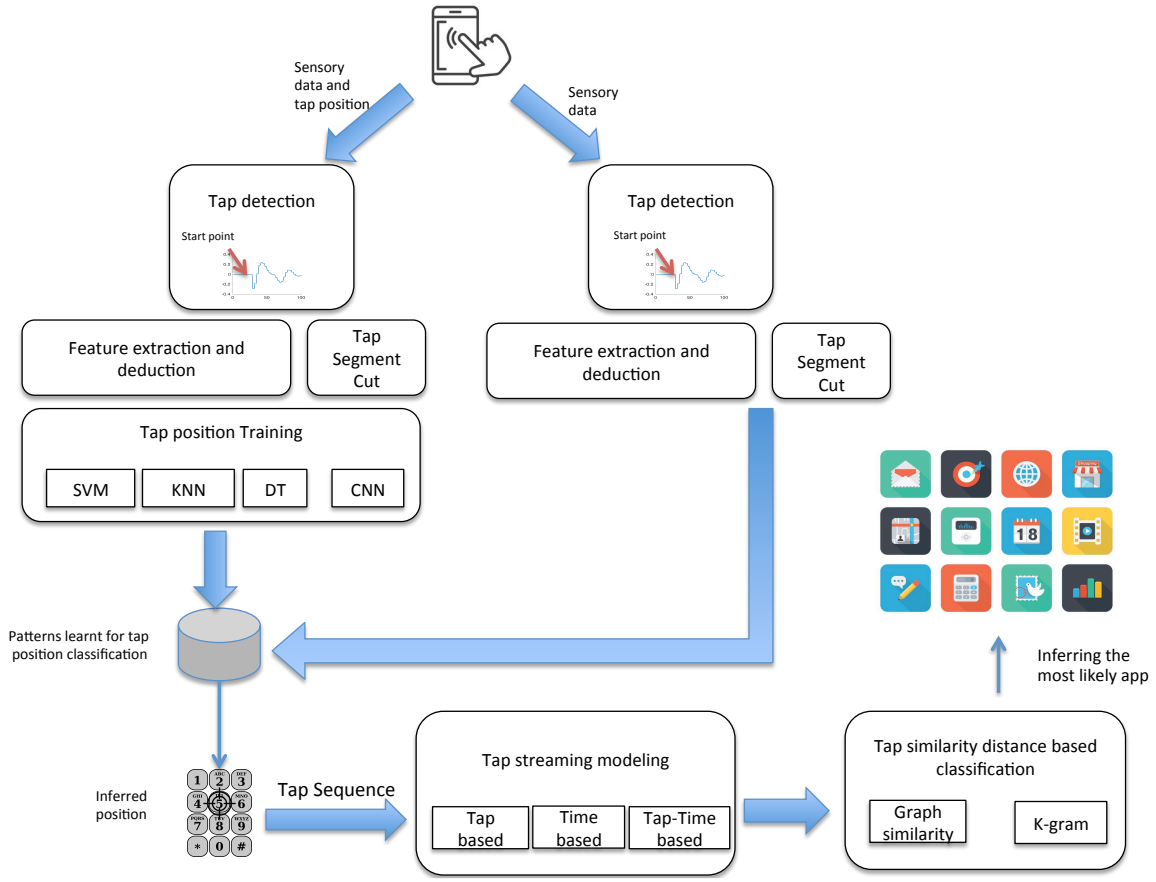


Figure 4.4. System overview.

phones can be accessed without user permissions [92]. There are lots of strict system level restrictions about who can receive a tap event with corresponding coordinates. In the Android system, only the current foreground view and activity on the touch screen can receive a tap event and the coordinates. Thus, it is impossible for third parties to retrieve these information. *Informer* has two stages: training data collection and sensory data recording. In the training data collection stage, users interact with the *host app* so that the tap positions and sensory data can both be collected. Then we can extract the features of the sensory data for different positions. In this way, we can easily associate sensory data with tap positions to form an inference database. In the sensory data recording stage, if a user is not interacting with the *host app*, *Informer* cannot capture tap events and positions. However, we can still

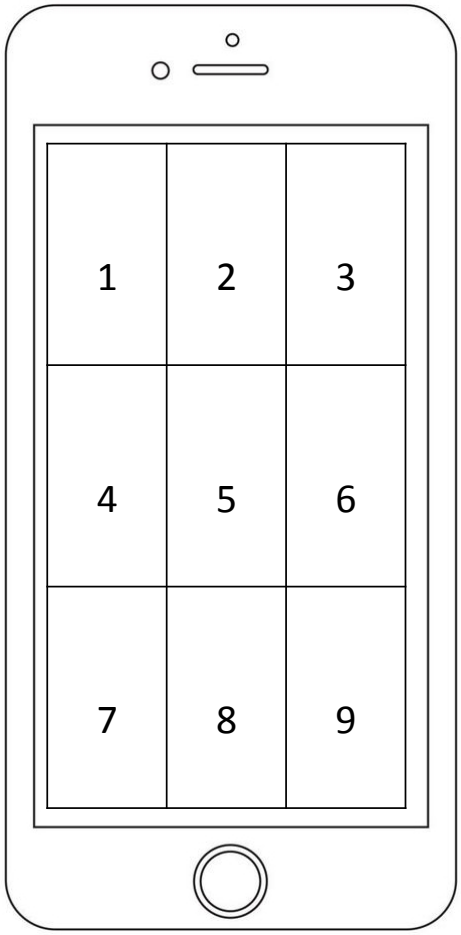


Figure 4.5. Layout of app during training status in our experiment

infer tap positions. Because the sensor reading service keeps recording sensory data which can be used for inferring tap positions. In order to collect sensory data for all the possible positions on a screen, we design the layout of the host app carefully so that a user may have to tap all the possible positions on a screen when interacting with *Informer*. In our experiment, the layout of the app is show as Fig 4.5

4.3 Tap Recognition

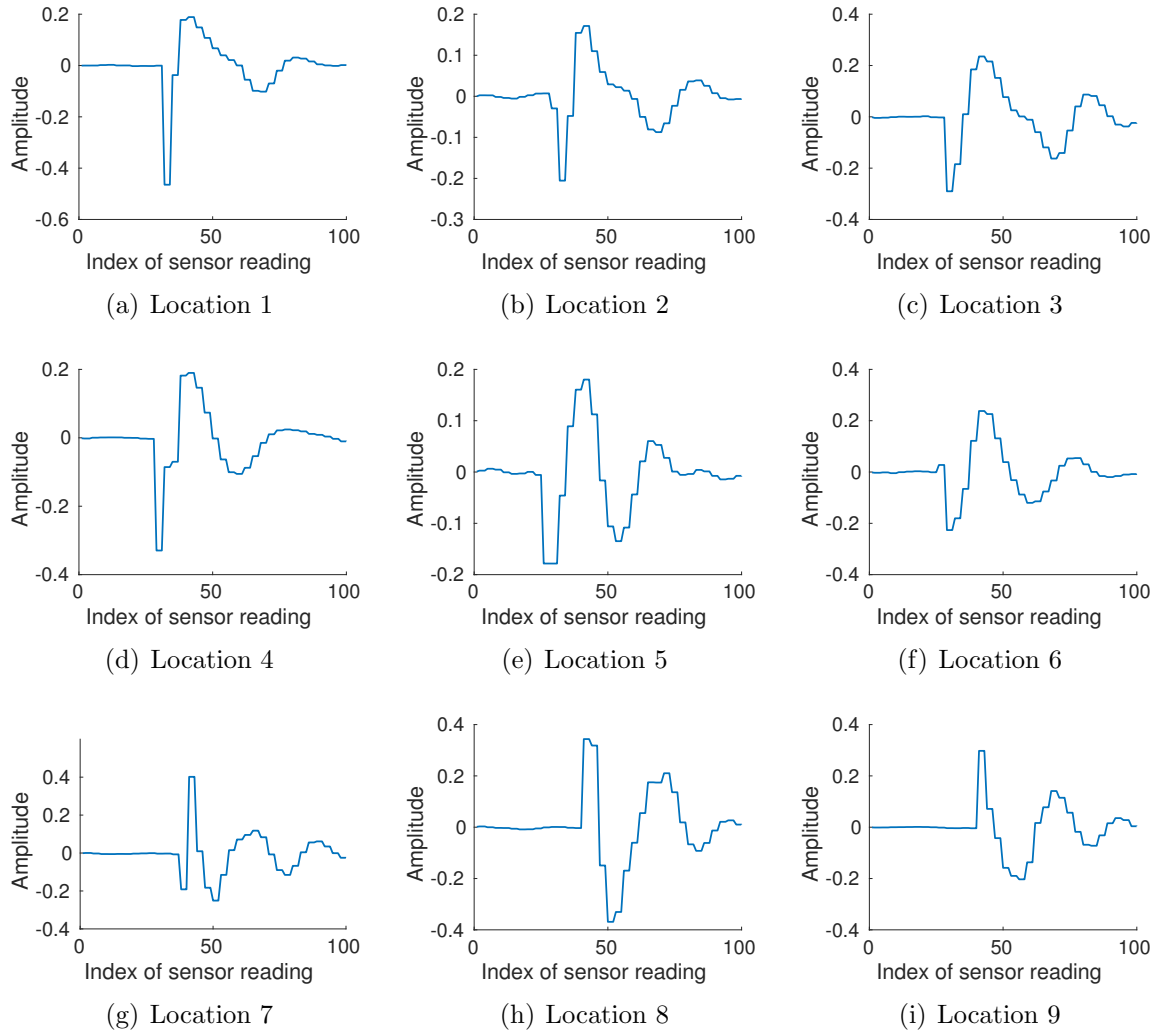
We now introduce our two different ways to distinguish different tap positions.

4.3.1 Traditional Method

Many approaches have been proposed for tap detection and recognition [55, 56, 59, 60]. However, our experiment results show that the previous methods have some limitations. Initially, we believe that sensory data should show very different patterns for different tap positions in each dimension. It turns out that this is not always true. For instance, no matter where you tap on the screen, there must be a downward power impact on the smartphone. So the sensory data pattern is very similar for the A_z axis regardless of tap positions as shown in Fig.4.6. It is then impossible to distinguish tap positions simply based on the extracted features from an individual dimension. Thus, the methods in [59, 55, 60] may not be effective in some situations.

We also find the correlation among axes is unique for each tap position and is very stable as shown in Fig.4.7 which depicts the angle relation between roll and pitch for different tap positions. Note that, the lines with different colors in Fig.4.7 represent different tap actions. It is shown in each subfigure that different tap actions at a same location result in highly similar correlations among different types of sensory data. Such coorrelations are definitely meaningful features that can help with identifying a tap action at a particular location. Therefore, we consider not only the features for each type of sensor data, but also the correlations among different types of sensory data.

Tap event detection In the training data collection stage, our data collection app naturally receives tap events. In the sensory data recording stage, we can only derive tap events through sensory data. Thus, in this stage, the main challenge is to detect tap events, for which we only take accelerometer into consideration. In our experiments, we find that no matter where you tap on the screen, there is a great impact on the accelerometer along axis A_z. It is intuitive because all tap actions have a downward power on the screen. Hence, we mainly utilize the sensory data of A_z to detect tap actions. We first normalize raw sensory data, then set a threshold λ for the square sum $SquareSum = (A_z^i)^2$. If the square sum exceeds λ, there is a peak candidate at time i.



(a) Location 1 (b) Location 2 (c) Location 3

(d) Location 4 (e) Location 5 (f) Location 6

(g) Location 7 (h) Location 8 (i) Location 9

Figure 4.6. Accelerometer Z-axis data for different tap positions.

We may obtain lots of peak candidates and we need to filter out noises. There must be an interval between two sequential tap actions and the peak width should fall into a constant range. So we set another four thresholds for the peaks in A_z, which are the minimum peak interval length, minimum peak height, minimum peak width, and maximum peak width. Then, all tap actions can be captured from the sensory data.

Feature Extraction We can obtain an array of peak indices from the tap detection module. Note that these peak indices are not the real peak indices for all the axes, because even for a same tap, the sensory data peaks may be various for different axes. In other

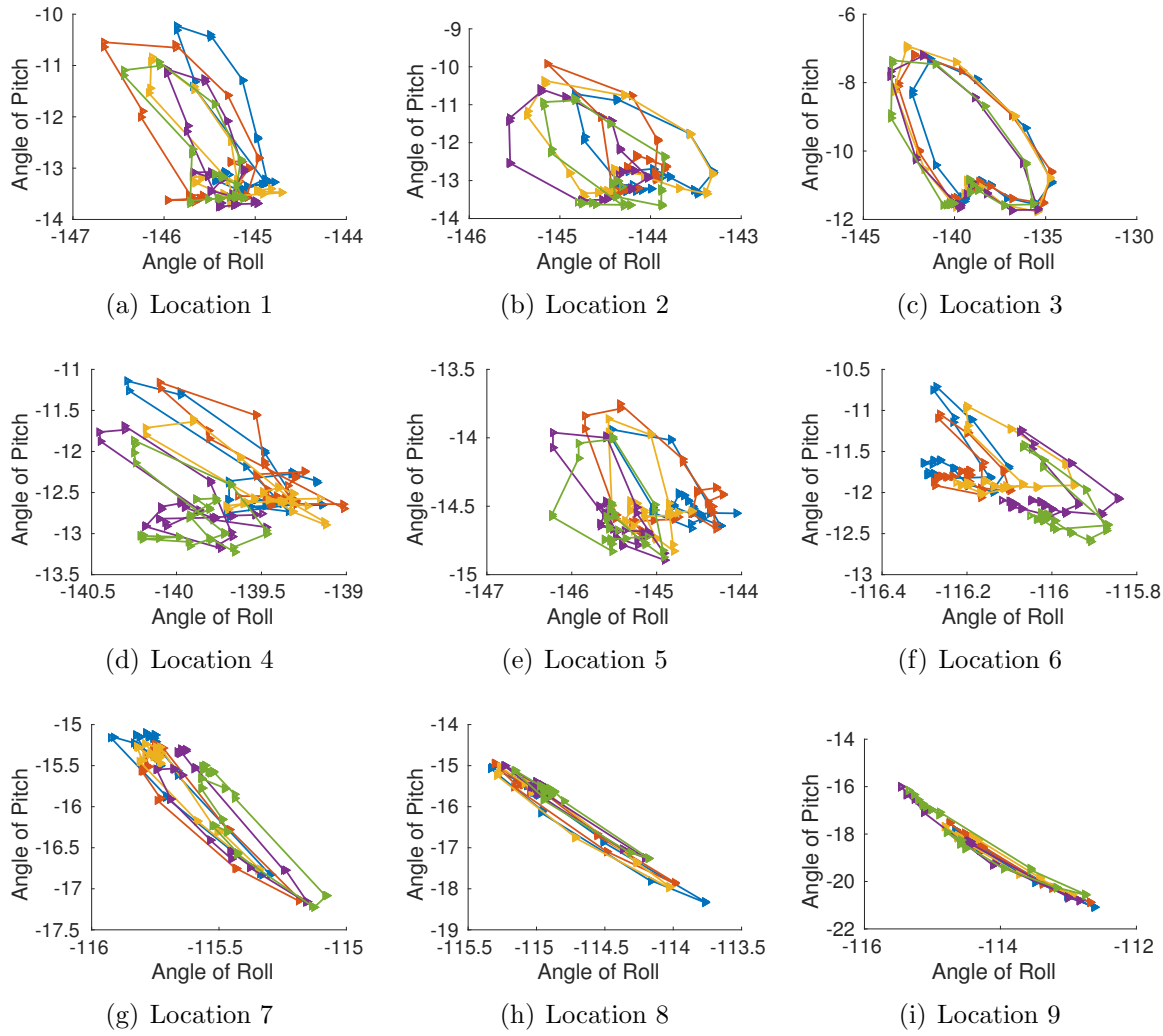
words, the sensory data reach peaks at different time for different axes. In our experiments, we use these peaks as an approximate index and cut off the small sections before and after these peaks indices. Let us call these sections as tap event windows which are processed respectively. It means we extract features for each axis respectively so that we can combine all the features of the 9 dimensions of sensory data. The extracted features include the min, max, average, number of peaks and crests, index difference between min and max. We extract these features for each axis respectively. The extracted features are listed in Table 4.1.

Table 4.1. Extracted Features

Feature	Description	Sensor
Min	The min element and the index	A, G
Max	The max element and the index	A, G
InDiff	The index difference between min and max	A, G
AVE_Deltas	the average sample by sample change	A, G
Num_peaks	the number of peaks	A, G
Num_crest	the number of crests	A, G
Sqsum	the square sum of pitch and roll angle	Ang
Sqmax	the maximum value of the square sum	Ang
Ag	the angle difference of sample by sample	Ang

The third column indicates the sensors from where the features are extracted. For instance, in the first row feature *Min* is acquired from accelerometer and gyroscope. In our experiments, the volunteers hold the smartphones in their left hands and use right hands to tap. In this scenario, there is only tiny influence on the azimuth angle. So, for features $Sqsum$, $Sqmax$ and Ag, we only extract them from R_x and R_y.

As aforementioned, we not only extract features for each axis respectively, but also take the correlations among axes into consideration. In our experiments, we find that different axes in each sensor show strong correlations for each tap position. Take the correlation between the angles of roll and pitch as an example as shown in Fig.4.7. It is obvious that the correlation between roll and pitch is very strong. Actually, such correlations are found among different axes in each sensor. There are 3 correlated axis pairs. We extract correlation



(a) Location 1 (b) Location 2 (c) Location 3

(d) Location 4 (e) Location 5 (f) Location 6

(g) Location 7 (h) Location 8 (i) Location 9

Figure 4.7. Correlation between angle of roll and pitch.

features for each pair of axes. We extract totally 136 features for each tap action. It is very time-consuming and unnecessary to leverage all these 136 features to discriminate different tap positions, because we do not know what features contribute more to the characteristics of a tap action's sensory data for different positions

If we can recognize the features which have the strongest correlations with a tap position, we can not only reduce noise but also improve inference speed. We utilize Principle Component Analysis (PCA) to filter features.

As discussed above, dimension reduction is essential for tap inference model training and tap inference. Suppose matrix $M_{N\times 136}$ represents the sensory data, where N is the size

of the training data sample and 136 is the feature space. First, we can derive the covariance matrix. Then we calculate the eigenvalues and the corresponding eigenvectors of the matrix. By sorting the eigenvalues in ascending order, we select the largest m eigenvalues and then use the corresponding eigenvectors as column vectors to construct an eigenvector matrix. So the dimension of the eigenvector matrix is $136 \times m$. Then we can project the original data into this new space using this matrix:

$$M_r(N \times m) = M_{N\times 136} \times EigenVectorMatrix_{136\times m} \tag{4.1}$$

$M_r(N \times m)$ is the new representation in the new data space. We can see that the feature dimensions have been reduced to m. To decide the value of m is tricky. From our experiments, $m = 10$ incurs the highest accuracy ratio.

Tap classification After we derive $EigenVectorMatrix_{136\times m}$, it means we have a new coordinate system for our data. Our keystroke recognition is based on this new coordinate system. To infer a tap position, we adopt three methods which are k-nearest neighbor (KNN), decision tree, and SVM to perform classification. In KNN, for each tap action, we calculate the standard Euclidean distance between this tap and all the taps in the training data set under the new coordinate system. We check the majority ones among the closest 5 taps, then label this tap with the majority tap position.

4.3.2 Conventional Neural Networks

Traditional tap recognition methods have some nonnegligible drawbacks, *e.g.*, feature extraction sacrifices data information. We propose to employ Conventional Neural Networks (CNN) to accomplish tap recognition.

CNN is one of the most popular deep learning methods and has attracted much attentions [93] [94]. Particularly, CNN has become a powerful tool in many areas, especially in image recognition and natural language processing [95, 96, 97]. CNN is a feed-forward artificial neural network, which consists of conventional layers, pooling layers and fully connected

layers. Conventional layer and pooling layer can be viewed as a whole and stacked together so that we can create a CNN model as complex as possible. CNN requires that the input data contain some kinds of 'spatial' correlations, such as image data and digital signal data [64].

As aforementioned, our collected sensory data from smartphones show similar patterns if users tap on the same location on smartphone screens. The 'shape' of the sensory data curve shown in Fig.4.6 can be considered as signal data which has spatial correlation. Moreover, inspired by the multi-channel image processing, our sensory data can be naturally treated as multi-channel signals because there are 9 axes of sensory data for each tap action. Different from image processing where the input is a 2-dimensional array, our input is just a 1-dimensional vector. It does not become more challenging to adopt CNN for our data since we just need to adjust the kernel shape accordingly.

There are many attractive advantages to make use of CNN to address the classification problem. One of the most powerful strengths of CNN is that we do not have to extract the features of tap sensory data manually. Inappropriate feature extraction leads to catastrophe consequence for classification. Even an experienced data analyst can hardly guarantee the effectiveness of feature extraction. Furthermore, extracting as many features as possible [56, 60] reduces efficiency. While in CNN, all the important features are extracted automatically during the convention process and the weights are updated during the back-propagation process. The 'spatial' correlations are also recorded through parameter sharing. These superior strengths make CNN very suitable for solving our classification problem. This conclusion is also validated by our experiment results, which show CNN outperforms the traditional methods significantly.

4.4 Application of tap recognition

In this section, we introduce two possible applications of tap inference.

4.4.1 App Usage Inference

A tap position can be inferred accurately as discussed in Section 3. So we can infer app usage based on the obtained tap sequences. Intuitively, apps with similar functions should have similar operation patterns. For example, in social media apps, we can chat with friends and browse the contents shared by friends. In news apps, we keep scanning news until we find something attractive, then we click the news link to read it carefully.

In this paper, we explore the feasibility of inferring app usage. A tap sequence refers to a series of tap actions. Tap sequences can be utilized to infer usage of an app. First, we explain how to model tap sequences and measure the similarity of tap sequences.

Tap Sequence Modeling In our experiments, we divide a smart device's screen into 9 zones similar to numeric only keyboard on smartphone, defined as $TL = \{l_1, l_2, l_3 ... l_9\}$. We record both the timestamp and tap position for each tap event. Let t_i be the timestamp for tap action T_i.

The following three models are considered for tap sequences.

i. Position based model. The most straightforward way is to only take tap positions into consideration. A tap sequence is recorded as a series of tap positions sorted by timestamps, *e.g.*, $\{l_3, l_4, \cdots l_i, \cdots, l_8\}$ where $l_i \in TL$.

ii. Time based model. Our observation indicates that the interval between a pair of consecutive tap actions is an important factor to infer app usage. We model a tap sequence as a time interval sequence $\{t_2 - t_1, t_3 - t_2, ...t_n - t_{n-1}\}$, where $t_i - t_{i-1}$ is the time interval between tap action T_i and tap action T_{i-1}. In order to utilize the n-gram algorithm, we categorize intervals into 5 groups $A = [0, 500ms)$, $B = [500ms, 1000ms)$, $C = [1000ms, 1500ms)$, $D = [1500ms, 2000ms)$, and $E = [2000ms, 2500ms)$. A time based sequence is then represented by a sequence of time intervals, such as $\{A, B, D\}$.

iii. Hybrid model. We take both tap positions and time intervals into consideration. We model a tap sequence as a list of tap positions and time intervals. For example, $\{l_3, A, l_1, B, l_4, A, l_5, C \cdots l_9, D\}$.

We group the tap sequences of the same type of apps together to form a 'profile' and utilize it to determine to which type of apps the upcoming tap sequence belongs.

Tap Sequence Similarity Now we discuss the metrics for measuring the similarity between two tap sequences. We mainly focus on n-gram similarity and average element-wise matrix similarity [98].

n-gram similarity. The ratio of the common subsequences of two tap sequences over the total number of subsequences can be utilized to measure their similarity [98]. Let S_n be a set of subsequences of length n appearing in one tap sequence. Thus, for tap sequence $Q = \{q_1, q_2, \cdots, q_N\}$, $S_k(Q) = \{subseq | subseq = \{q_i, q_{i+1}, \cdots, q_{i+n-1}\}, i \in [1, N+1-n]\}$. The similarity of two sequences Q_1 and Q_2 is defined as

$$Distance(Q_1, Q_2) = 1 - \frac{|S_n(Q_1) \cap S_n(Q_2)|}{|S_n(Q_1) \cup S_n(Q_2)|}. \tag{4.2}$$

n-gram follows the assumption that two similar apps should result in more common subsequences. The value of n is the number of the necessary tap actions to complete an operation and n varies for different apps.

Average element-wise matrix similarity. We construct a transition matrix based on a tap sequence, where each node represents a tap position. There are two ways to define the weight of an edge from node l_i to node l_j. The first way is to use the number of transitions from tap positions l_i to l_j over the total number of transitions. The other one is to use the average transition time from tap positions l_i to l_j. The similarity between two tap sequences is defined as the average element-wise distance of their transition matrices. For each type of apps, we collect a large number of tap sequences as training data. The similarity between the tap sequences in the training data set and a coming tap sequence is utilized to do the classification.

4.4.2 Password inference

Password is the most sensitive information. A system level protection strategy may be provided to protect it, *e.g.*, only the currently active activity can receive the tap events [92]. However, the released sensory data can still threaten password privacy.

We adopt the CNN method for password inference where the CNN outputs a probability vector for each single tap action. Each entry in the vector represents the probability of this position being tapped. We choose the tap position sequence with the highest joint probability as our inferred password. As users are usually allowed to try 3 times when type in the password, our method outputs the top 3 inferred passwords with the highest joint probabilities. If the real password is included in the top three ones, we consider that our inference is correct.

4.5 Experiment Results

According to our attack model, our system has two working status, training status and attacking status. For an Android app, there are many functional components which are mainly activities and services [92]. Both in training status and attacking status, our service components keep running in the background without user notice. Adversaries can inject services in apps to secretly collect sensory data. An activity component is responsible for user interactions such as a tap event on the screen. When a user is running an attacker's app, all the tap positions and sensory data are collected and sent to the backend as training data.

In the attacking status, due to the system level protection mechanisms, we cannot receive tap events anymore if a user is running other apps. However, service components may keep running so that we can still collect sensory data. In this way, we can then make inference based on the collected sensory data as aforementioned.

We implement our adversary app in an Android system API level 23 and test it with many kinds of smartphone including Samsung Galaxy S7 edge and Samsung Note

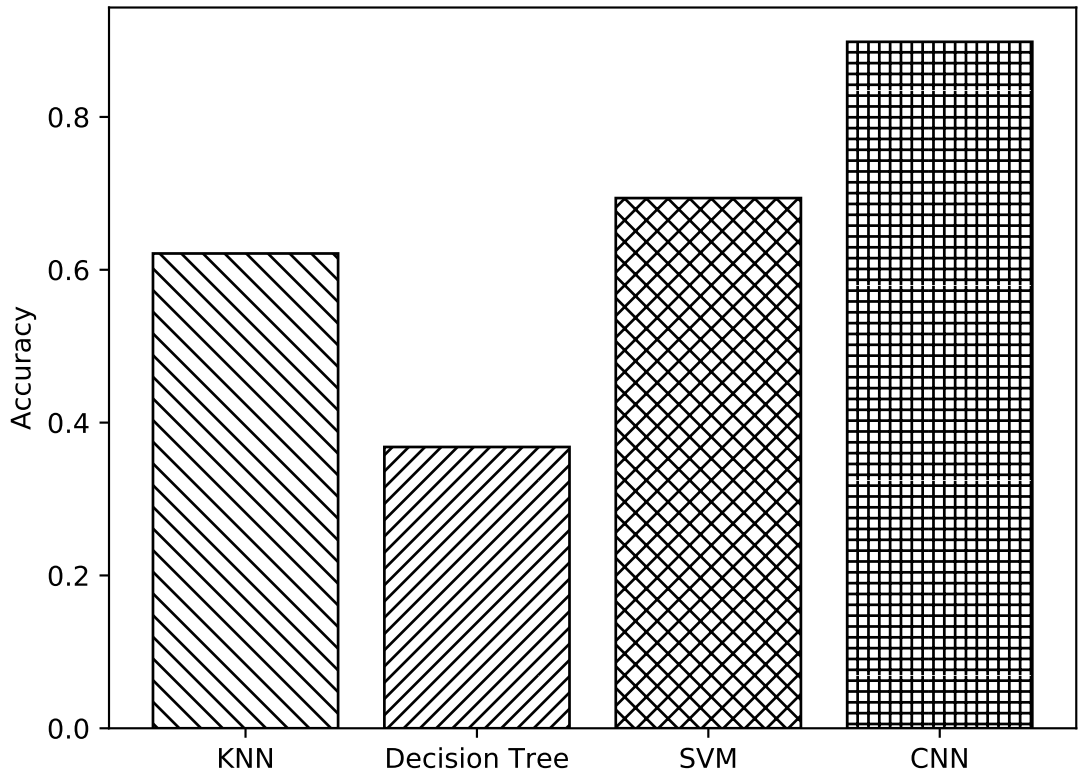


Figure 4.8. Average inference accuracy.

5, HUAWEI meta 9, HUAWEI Honor 8,LG G5, HTC M8 *e.g.* Our experiments only require volunteers to hold smartphones with left hands and use point fingers to tap on screens.

4.5.1 Tap inference

We have 102 volunteers. A smartphone screen is divided into 9 zones. During the training data collection stage, each volunteer taps in each zone on the screen for at least 100 times. During the data recording stage, the volunteers use smartphones as usual.

The average inference accuracy is shown in Fig.4.8 We can see, SVM and CNN achieve better results in the first part of our work.

Fig.4.9 shows the Cumulative Distribution Function (CDF) of our results. In our CNN

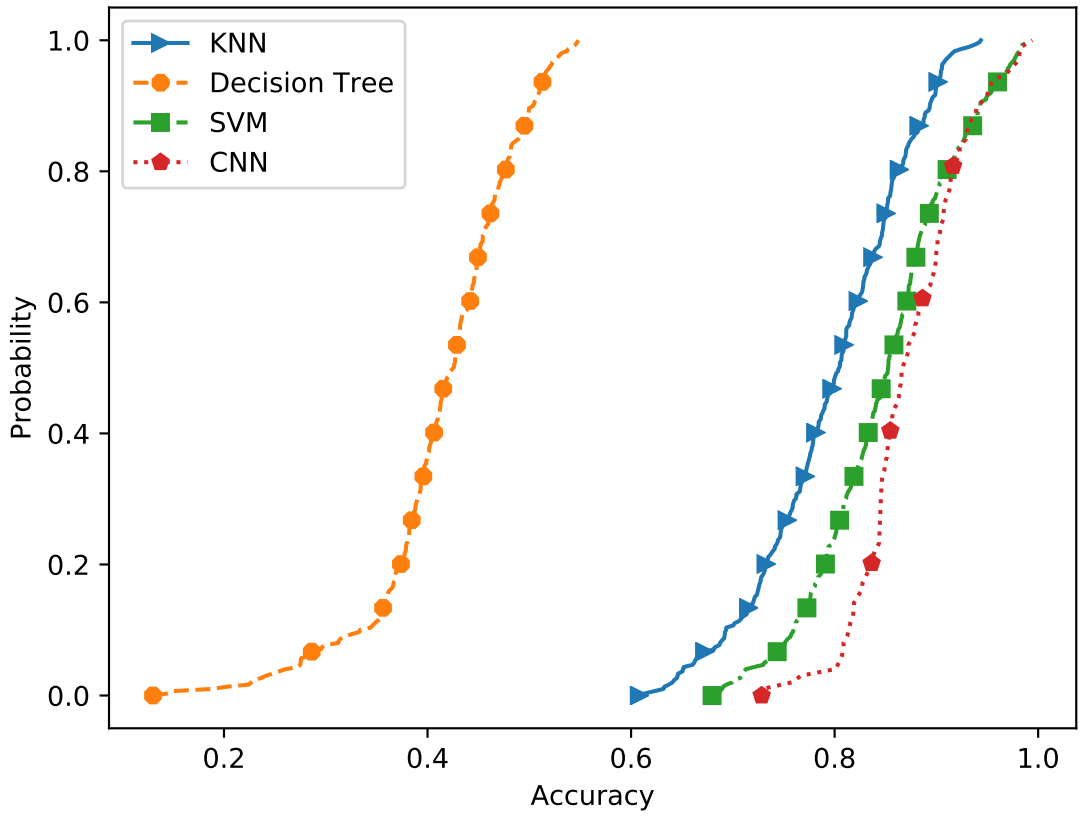


Figure 4.9. Cumulative distribution function of accuracy.

method, more than 95% percent inference accuracy is better than 80%, which is very impressive.

The traditional methods mainly focus on feature extraction, which might sacrifice accuracy. As shown in Fig.4.10, for SVM and KNN, with only 10 features, we can have an accuracy of 80%. As the number of features increases, the accuracy is not substantially improved. This validates that feature extraction based methods may not be effective in practice.

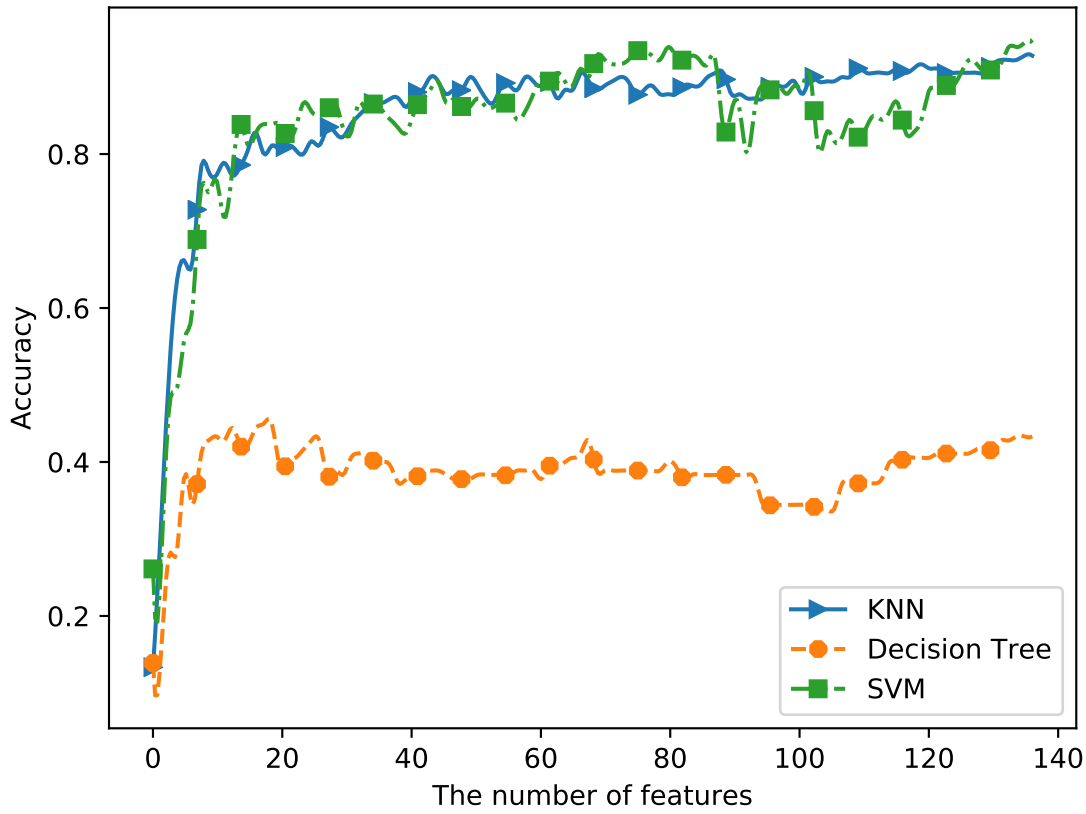


Figure 4.10. Impact of features.

4.5.2 App inference

For app inference, we classify the apps into 7 categories based on the functions as shown in table: 4.2

We employ three tap sequence models which are *tap-sequence*, *time-based*, and *tap-time*. In the position based (Tap) and time based (Interval) models, we use n-gram distance to measure app similarity distance. In the hybrid model, we adopt three similarity distance methods: n-gram (Hybrid), graph-count (GC), and graph-interval (GI). The results are shown in Fig.4.11. Our tap-time model with n-gram (Hybrid) achieves the best results.

Table 4.2. Apps tested in our experiment catogories

Categories	Apps
Game	Temple Run, Paris Metro
Shopping	TaoBao, Amazon
Brower	QQ browser, UC browser
Social Media	Weibo, Facebook
Instant Chat	WeChat, Messenger
Music Player	QQ Music, WangYi Yun Music

4.5.3 Password inference

In the second part of our work, we apply the CNN model for app password inference. Although it is also related to tap positions, password inference is harder because it is considered as failed even with one single inference error. In our experiments, we consider passwords with different lengths including 4 digit passwords, 6 digit passwords, and 8 digit passwords. If our top 3 candidates contain the correct password, we consider it as correct. The accuracy results are shown in Table 4.3.

Table 4.3. Password Inference Accuracy

	4-Digit	6-Digit	8-Digit
Accuracy	94.3 %	92.0 %	89.9%

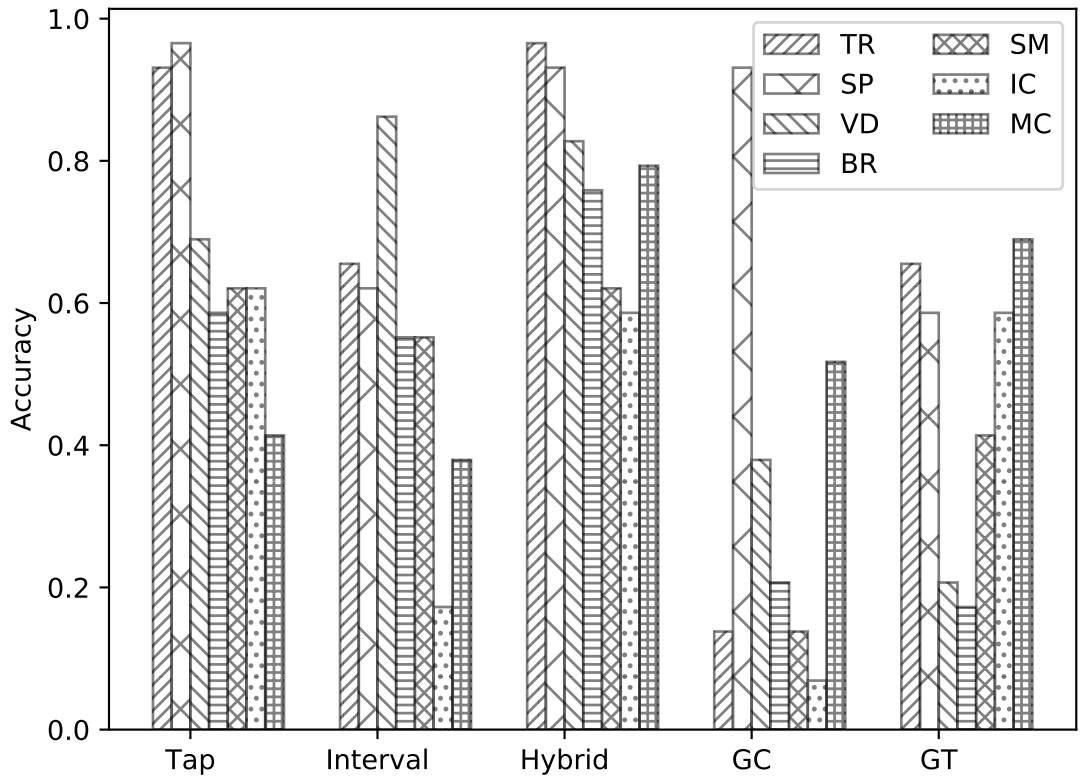


Figure 4.11. App inference accuracy.

Chapter 5

FUTURE RESEARCH DIRECTIONS

5.1 Potential interesting problems

5.1.1 How to determine if the user start driving?

So far, we have been talking about privacy leakage based on sensory data in smart device, even though we have proved that our approaches can infer privacy info solely based on sensory data, there still several interesting problem that need to be addressed to improve the performance of our work. In our sensory data based location inference, we assume that we already know the start point of driving, but there still one interesting problem: how to determine if the user is start driving. Since we only interested in some sensory data when user is driving, it is very crucial to detect the start and end point of driving. It is not always that case that user get up on time and start driving the same time.

There are many existing works focus on recognize the human activities based on the sensory data including determeine if a user is taking a marta. While few of works study on determine the start or end point to diferent activities. So how to determine is the user start driving is an interesting problem.

5.1.2 How to reduce the usage habit impacts on sensory data?

In both parts of our experiment, in order to reduce the noisy impact on our sensory data, we require volunteers to held the smart device tight and in the same position both for testing and training data, however, in real world, users may hold their smart device arbitrarily. In order to make our attack more general and effective, on possible solution is to project the sensory data into one same space without caring about how gesture of smart devices. The insight is behind that no matter how user hold the devices, his tap action is respect to the smart devices. Besides, sinc we can read the gyoscope data which will enable

us to know the relative angle between the smart phone and horizontal, it is not impossible to project the sensory data to one consistant coordinate space. If this work can be done, it will greatly facilite our previous works.

5.1.3 How to determine if the user is inputting password

In the password inference part, we just simply assuming that the user is inputting password, so the training data and testing data is all "human made", which means we only type number pad to input password. However, in reality, we only can detect user is typing on the screen, but do not know what the user is type, even we do not know if the user is typing or it is just the noisy. What we interested is the password of user, since we can infer inputs using sensory data, we can infer the password of user if we know the user is inputting password. There is papers analyzes the router network traffic to know what service the user is using, then based on that to determine whether the user is typing some sensitive information.

5.2 Sensory data fusion to improve infer accuracy

5.2.1 Different data model fusion

Another interesting potential work is data fusion to input accuracy.

For example, when use is typing, not only we can record the accelerometer data , but also the sounds of typing. If we can combine these two different data source, it will improve our inference accuracy.

The data fusion is not limited to same data model fusion, we can also us different data model to perform fusion.

5.2.2 Data conflict resolving

Besides, when two data source is conflict with each other, how to resolve the conflicts.

For instance, when one dimension sensory data indicates that user is tapping on the

location 1, the other dimension sensory data indicates that the user is tapping on the location 2.

Currently, the most common way is to average the sensory data before performing inference. This method is not so accurate, we may can apply data fusino techinics to solve this problem.

Chapter 6

CONCLUSION

User privacy is being threatened by the sensors embedded in mobile devices, as these sensors may release data without users' awareness. In this paper, we show that a user's location information can be inferred by utilizing the sensory data collected from embedded sensors in users' mobile devices. We make use of the sensory data to construct fingerprints of routes and dynamic time warping is employed to perform route inference. We address three issues including route identification, localization in a specific route, and localization in a bounded area. Real experiments were performed to evaluate our work. The extensive experiment results show that we can effectively identify routes and localize a user in a real time manner unconsciously.

Though smartphones are indispensable in people's modern life, most people do not realize that smartphones also threaten our privacy. People usually ignore the fact that sensory data can be secretly collected from the sensors embedded in smartphones without user permissions. In this paper, we present the feasibility of inferring users' app usage habits solely based on sensory data. More specific, we propose three improved traditional methods and one deep neural network method to recognize users' tap positions by analyzing the secretly collected sensory data. The extensive experiment results show that our proposed method achieve high accuracy and are very effective for tap classification, app inference and password inference.

REFERENCES

[1] L. Zhang, X. Wang, J. Lu, P. Li, and Z. Cai, "An efficient privacy preserving data aggregation approach for mobile sensing," *Security and Communication Networks*, vol. 9, no. 16, pp. 3844–3853, 2016.

[2] Q. Chen, H. Gao, S. Cheng, X. Fang, Z. Cai, and J. Li, "Centralized and distributed delay-bounded scheduling algorithms for multicast in duty-cycled wireless sensor networks," *IEEE/ACM Transactions on Networking*, vol. PP, no. 99, pp. 1–14, 2017.

[3] Y. Wang, Z. Cai, G. Yin, Y. Gao, X. Tong, and Q. Han, "A game theory-based trust measurement model for social networks," *Computational Social Networks*, vol. 3, no. 1, p. 2, 2016.

[4] Y. Huang, Z. Cai, and A. G. Bourgeois, "Location privacy protection with accurate service," *Journal of Network and Computer Applications*, vol. 103, p. 146156, 2018.

[5] S. Cheng, Z. Cai, J. Li, and H. Gao, "Extracting kernel dataset from big sensory data in wireless sensor networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 4, pp. 813–827, April 2017.

[6] K. Zhang, Q. Han, Z. Cai, and G. Yin, "Rippas: a ring-based privacy-preserving aggregation scheme in wireless sensor networks," *Sensors*, vol. 17, no. 2, p. 300, 2017.

[7] M. Han, J. Li, Z. Cai, and Q. Han, "Privacy reserved influence maximization in gps-enabled cyber-physical and online social networks," in *Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom), 2016 IEEE International Conferences on.* IEEE, 2016, pp. 284–292.

[8] X. Wang, Z. He, X. Zhao, C. Lin, Y. Pan, and Z. Cai, "Reaction-diffusion modeling of

malware propagation in mobile wireless sensor networks," *Science China Information Sciences*, vol. 56, no. 9, pp. 1–18, 2013.

[9] L. Guo, C. Ai, X. Wang, Z. Cai, and Y. Li, "Real time clustering of sensory data in wireless sensor networks." in *The 28th IEEE International Performance Computing and Communications Conference (IPCCC 2009)*, 2009, pp. 33–40.

[10] L. Zhang, Z. Cai, and X. Wang, "Fakemask: a novel privacy preserving approach for smartphones," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 335–348, 2016.

[11] X. Zheng, Z. Cai, and Y. Li, "Data linkage in smart iot systems: A consideration from privacy perspective," *IEEE Communications Magazine*, 2018.

[12] J. Wang, Z. Cai, Y. Li, D. Yang, J. Li, and H. Gao, "Protecting query privacy with differentially private k-anonymity in location-based services," *Personal and Ubiquitous Computing*, pp. 1–17, 2018.

[13] Z. Cai, C. Wang, and A. Bourgeois, "Preface: Special issue on computing and combinatorics conference and wireless algorithms, systems, and applications conference," *Journal of Combinatorial Optimization*, vol. 32, no. 4, pp. 983–984, Nov 2016.

[14] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, "Accelprint: Imperfections of accelerometers make smartphones trackable." in *NDSS*. Citeseer, 2014.

[15] A. Das and N. Borisov, "Poster: Fingerprinting smartphones through speaker," in *Poster at the IEEE Security and Privacy Symposium*. Citeseer, 2014.

[16] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, "Mobile device identification via sensor fingerprinting," *arXiv preprint arXiv:1408.1416*, 2014.

[17] J. Li, Z. Cai, J. Wang, M. Han, and Y. Li, "Truthful incentive mechanisms for geographical position conflicting mobile crowdsensing systems," *IEEE Transactions on Computational Social Systems*, 2018.

[18] Y. Liang, Z. Cai, Q. Han, and Y. Li, "Location privacy leakage through sensory data," *Security and Communication Networks*, vol. 2017, 2017.

[19] Y. Wang, Z. Cai, X. Tong, Y. Gao, and G. Yin, "Truthful incentive mechanism with location privacy-preserving for mobile crowdsourcing systems," *Computer Networks*, vol. 135, pp. 32–43, 2018.

[20] J. Wang, Z. Cai, C. Ai, D. Yang, H. Gao, , and X. Cheng, "Differentially private k-anonymity: achieving query privacy in location-based services," in *The International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI 2016)*, 2016.

[21] X. Zheng, Z. Cai, J. Li, and H. Gao, "Location-privacy-aware review publication mechanism for local business service systems," in *The 36th Annual IEEE International Conference on Computer Communications (INFOCOM 2017)*, May 2017, pp. 1–9.

[22] J. Han, E. Owusu, L. T. Nguyen, A. Perrig, and J. Zhang, "Accomplice: Location inference using accelerometers on smartphones," in *Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on*. IEEE, 2012, pp. 1–9.

[23] I. Constandache, R. R. Choudhury, and I. Rhee, "Towards mobile phone localization without war-driving," in *Infocom, 2010 proceedings ieee*. IEEE, 2010, pp. 1–9.

[24] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, "Powerspy: Location tracking using mobile device power analysis," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 785–800.

[25] M. Alzantot and M. Youssef, "Uptime: Ubiquitous pedestrian tracking using mobile phones," in *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*. IEEE, 2012, pp. 3204–3209.

[26] J. Krumm and E. Horvitz, "Locadio: Inferring motion and location from wi-fi signal strengths." in *Mobiquitous*, 2004, pp. 4–13.

[27] I. Constandache, X. Bao, M. Azizyan, and R. R. Choudhury, "Did you see bob?: human localization using mobile phones," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking.* ACM, 2010, pp. 149–160.

[28] M. Azizyan, I. Constandache, and R. Roy Choudhury, "Surroundsense: mobile phone localization via ambience fingerprinting," in *Proceedings of the 15th annual international conference on Mobile computing and networking.* ACM, 2009, pp. 261–272.

[29] Y. Michalevsky, D. Boneh, and G. Nakibly, "Gyrophone: Recognizing speech from gyroscope signals," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 1053–1067.

[30] Z. Xu, K. Bai, and S. Zhu, "Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks.* ACM, 2012, pp. 113–124.

[31] J. Hua, Z. Shen, and S. Zhong, "We can track you if you take the metro: Tracking metro riders using accelerometers on smartphones," *arXiv preprint arXiv:1505.05958*, 2015.

[32] X. Zhou, S. Demetriou, D. He, M. Naveed, X. Pan, X. Wang, C. A. Gunter, and K. Nahrstedt, "Identity, location, disease and more: Inferring your secrets from android public resources," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security.* ACM, 2013, pp. 1017–1028.

[33] Z. Cai, Z.-Z. Chen, G. Lin, and L. Wang, "An improved approximation algorithm for the capacitated multicast tree routing problem," *The 2nd Annual International Conference on Combinatorial Optimization and Applications (COCOA2008)*, vol. 5165, pp. 286–295, 2008.

[34] K. Zhang, Q. Han, Z. Cai, G. Yin, and J. Lin, "Doami: A distributed on-line algorithm to minimize interference for routing in wireless sensor networks," *Theoretical Computer Science*, 2016.

[35] Z. Duan, W. Li, and Z. Cai, "Distributed auctions for task assignment and scheduling in mobile crowdsensing systems," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 635–644.

[36] L. Guo, Y. Li, and Z. Cai, "Minimum-latency aggregation scheduling in wireless sensor network," *Journal of Combinatorial Optimization*, vol. 31, no. 1, pp. 279–310, 2016.

[37] Z. Cai, C. Wang, and A. Bourgeois, "Preface: Special issue on computing and combinatorics conference and wireless algorithms, systems, and applications conference," 2016.

[38] Z. Cai, R. Goebel, and G. Lin, "Size-constrained tree partitioning: approximating the multicast k-tree routing problem," *Theoretical Computer Science*, vol. 412, no. 3, pp. 240–245, 2011.

[39] ——, "Size-constrained tree partitioning: A story on approximation algorithm design for the multicast k-tree routing problem." in *The 3nd Annual International Conference on Combinatorial Optimization and Applications (COCOA2009)*. Springer, 2009, pp. 363–374.

[40] L. Zhang, X. Wang, J. Lu, P. Li, and Z. Cai, "An efficient privacy preserving data aggregation approach for mobile sensing," *Security and Communication Networks*, vol. 9, no. 16, pp. 3844–3853, 2016.

[41] K. Yadav, V. Naik, P. Singh, and A. Singh, "Alternative localization approach for mobile phones without gps," in *Middleware'10 Posters and Demos Track*. ACM, 2010, p. 1.

[42] W. R. Ouyang, A. K.-S. Wong, C.-T. A. Lea, and V. Y. Zhang, "Received signal strength-based wireless localization via semidefinite programming." in *Globecom*, 2009, pp. 1–6.

[43] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and

traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 323–336.

[44] A. Ofstad, E. Nicholas, R. Szcodronski, and R. R. Choudhury, "Aampl: Accelerometer augmented mobile phone localization," in *Proceedings of the First ACM International Workshop on Mobile Entity Localization and Tracking in GPS-less Environments*, ser. MELT '08. New York, NY, USA: ACM, 2008, pp. 13–18. [Online]. Available: http://doi.acm.org/10.1145/1410012.1410016

[45] X. Zheng, Z. Cai, J. Yu, C. Wang, and Y. Li, "Follow but no track: Privacy preserved profile publishing in cyber-physical social systems," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1868–1878, 2017.

[46] X. Zheng, G. Luo, and Z. Cai, "A fair mechanism for private data publication in online social networks," *IEEE Transactions on Network Science and Engineering*, 2018.

[47] Z. Cai, Z. He, X. Guan, and Y. Li, "Collective data-sanitization for preventing sensitive information inference attacks in social networks," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[48] T. Shi, S. Cheng, Z. Cai, Y. Li, and J. Li, "Retrieving the maximal time-bounded positive influence set from social networks," *Personal and Ubiquitous Computing*, vol. 20, no. 5, pp. 717–730, 2016.

[49] J. Li, Z. Cai, M. Yan, and Y. Li, "Using crowdsourced data in location-based social networks to explore influence maximization," in *The 35th Annual IEEE International Conference on Computer Communications (INFOCOM 2016)*, April 2016, pp. 1–9.

[50] Z. He, Z. Cai, Q. Han, W. Tong, L. Sun, and Y. Li, "An energy efficient privacy-preserving content sharing scheme in mobile social networks," *Personal and Ubiquitous Computing*, vol. 20, no. 5, pp. 833–846, 2016.

[51] Z. He, Z. Cai, and J. Yu, "Latent-data privacy preserving with customized data utility for social network data," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 1, pp. 665–673, 2018.

[52] Y. Wang, Z. Cai, G. Yin, Y. Gao, X. Tong, and G. Wu, "An incentive mechanism with privacy protection in mobile crowdsourcing systems," *Computer Networks*, vol. 102, pp. 157–171, 2016.

[53] Z. Duan, M. Yan, Z. Cai, X. Wang, M. Han, and Y. Li, "Truthful incentive mechanisms for social cost minimization in mobile crowdsourcing systems," *Sensors*, vol. 16, no. 4, p. 481, 2016.

[54] Y. Wang, G. Yin, Z. Cai, Y. Dong, and H. Dong, "A trust-based probabilistic recommendation model for social networks," *Journal of Network and Computer Applications*, vol. 55, pp. 59–67, 2015.

[55] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, "Accessory: password inference using accelerometers on smartphones," in *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*. ACM, 2012, p. 9.

[56] L. Cai and H. Chen, "Touchlogger: Inferring keystrokes on touch screen from smartphone motion." *HotSec*, vol. 11, pp. 9–9, 2011.

[57] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "Tapprints: your finger taps have fingerprints," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 323–336.

[58] M. Goel, J. Wobbrock, and S. Patel, "Gripsense: using built-in sensors to detect hand posture and pressure on commodity mobile phones," in *Proceedings of the 25th annual ACM symposium on User interface software and technology*. ACM, 2012, pp. 545–554.

[59] A. J. Aviv, B. Sapp, M. Blaze, and J. M. Smith, "Practicality of accelerometer side

channels on smartphones," in *Proceedings of the 28th Annual Computer Security Applications Conference.* ACM, 2012, pp. 41–50.

[60] Z. Xu, K. Bai, and S. Zhu, "Taplogger: Inferring user inputs on smartphone touchscreens using on-board motion sensors," in *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks.* ACM, 2012, pp. 113–124.

[61] W. McGrath and Y. Li, "Detecting tapping motion on the side of mobile devices by probabilistically combining hand postures," in *Proceedings of the 27th annual ACM symposium on User interface software and technology.* ACM, 2014, pp. 215–219.

[62] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas, "Emotionsense: a mobile phones based adaptive platform for experimental social psychology research," in *Proceedings of the 12th ACM international conference on Ubiquitous computing.* ACM, 2010, pp. 281–290.

[63] X. Liu, Z. Zhou, W. Diao, Z. Li, and K. Zhang, "When good becomes evil: Keystroke inference with smartwatch," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security.* ACM, 2015, pp. 1273–1285.

[64] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, J. Zhu, P. Wu, and J. Zhang, "Convolutional neural networks for human activity recognition using mobile sensors," in *Mobile Computing, Applications and Services (MobiCASE), 2014 6th International Conference on.* IEEE, 2014, pp. 197–205.

[65] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition." in *IJCAI*, 2015, pp. 3995–4001.

[66] Y. Wu, X. Zhang, Y. Bian, Z. Cai, X. Lian, X. Liao, and F. Zhao, "Second-order random walk-based proximity measures in graph analysis: formulations and algorithms," *The VLDB Journal*, vol. 27, no. 1, pp. 127–152, 2018.

[67] J. Chen, C. Wang, H. Lin, W. Wang, Z. Cai, and J. Wang, "Learning the structures of online asynchronous conversations," in *The 22rd International Conference on Database Systems for Advanced Applications (DASFAA 2017)*. Springer, 2017, pp. 19–34.

[68] Z. He, Z. Cai, J. Yu, X. Wang, Y. Sun, and Y. Li, "Cost-efficient strategies for restraining rumor spreading in mobile social networks," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 3, pp. 2789–2800, March 2017.

[69] M. Han, M. Yan, Z. Cai, Y. Li, X. Cai, and J. Yu, "Influence maximization by probing partial communities in dynamic online social networks," *Transactions on Emerging Telecommunications Technologies*, vol. 28, no. 4, 2017.

[70] Z. Cai, G. Lin, and G. Xue, "Improved approximation algorithms for the capacitated multicast routing problem," in *In Proceedings of the 11th International Computing and Combinatorics Conference (COCOON 2005)*, vol. 8881. Springer, 2005, pp. 136–145.

[71] Q. Chen, H. Gao, S. Cheng, J. Li, and Z. Cai, "Distributed non-structure based data aggregation for duty-cycle wireless sensor networks," in *The 36th Annual IEEE International Conference on Computer Communications (INFOCOM 2017)*, May 2017, pp. 1–9.

[72] Z. He, Z. Cai, Y. Sun, Y. Li, and X. Cheng, "Customized privacy preserving for inherent data and latent data," *Personal and Ubiquitous Computing*, vol. 21, no. 1, pp. 43–54, 2017.

[73] T. Shi, S. Cheng, Z. Cai, and J. Li, "Adaptive connected dominating set discovering algorithm in energy-harvest sensor networks," in *The 35th Annual IEEE International Conference on Computer Communications (INFOCOM 2016)*, April 2016, pp. 1–9.

[74] X. Liu, Z. Cai, D. Miao, and J. Li, "On the complexity of extracting subtree with keeping distinguishability," in *The 10th Annual International Conference on Combinatorial Optimization and Applications (COCOA 2016)*. Springer, 2016, pp. 230–240.

[75] M. Yan, M. Han, C. Ai, Z. Cai, and Y. Li, "Data aggregation scheduling in probabilistic wireless networks with cognitive radio capability (globecom 2016)," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[76] J. Lu, Z. Cai, X. Wang, L. Zhang, P. Li, and Z. He, "Primary and secondary social activity aware routing for cognitive radio networks." in *The International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI 2016)*, 2016.

[77] T. Zhu, S. Cheng, Z. Cai, and J. Li, "Critical data points retrieving method for big sensory data in wireless sensor networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, p. 18, 2016.

[78] J. H. Ziegeldorf, O. G. Morchon, and K. Wehrle, "Privacy in the internet of things: threats and challenges," *Security and Communication Networks*, vol. 7, no. 12, pp. 2728–2742, 2014.

[79] K. Zhao, H. Jin, D. Zou, W. Dai, and Y. Xiang, "A privacy-preserving location tracking system for smartphones based on cloud storage," *Security and Communication Networks*, vol. 8, no. 3, pp. 446–458, 2015.

[80] Q. Yang, A. Lim, X. Ruan, X. Qin, and D. Kim, "Location-preserved contention-based routing in vehicular ad hoc networks," *Security and Communication Networks*, 2014.

[81] L. L. J. L. Meng Han, Qilong Han and Y. Li, "Maximizing influence in sensed heterogenous social network with privacy preservation," *International Journal of Sensor Networks (IJSNet)*, 2017.

[82] Y. Zhao, "Mobile phone location determination and its impact on intelligent transportation systems," *IEEE Transactions on intelligent transportation systems*, vol. 1, no. 1, pp. 55–64, 2000.

[83] M. Han, J. Li, Z. Cai, and Q. Han, "Privacy reserved influence maximization in gps-enabled cyber-physical and online social networks," in *Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom), 2016 IEEE International Conferences on.* IEEE, 2016, pp. 284–292.

[84] N. Eagle, A. S. Pentland, and D. Lazer, "Inferring friendship network structure by using mobile phone data," *Proceedings of the national academy of sciences*, vol. 106, no. 36, pp. 15 274–15 278, 2009.

[85] https://developer.android.com/guide/topics/sensors/index.html, [Online; accessed Dec-2016].

[86] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[87] G. D. Forney Jr, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[88] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin, "Diversity in smartphone usage," in *Proceedings of the 8th international conference on Mobile systems, applications, and services.* ACM, 2010, pp. 179–194.

[89] L. Pei, R. Guinness, R. Chen, J. Liu, H. Kuusniemi, Y. Chen, L. Chen, and J. Kaistinen, "Human behavior cognition using smartphone sensors," *Sensors*, vol. 13, no. 2, pp. 1402–1424, 2013.

[90] X. Su, H. Tong, and P. Ji, "Activity recognition with smartphone sensors," *Tsinghua Science and Technology*, vol. 19, no. 3, pp. 235–249, 2014.

[91] M. Fahim, I. Fatima, S. Lee, and Y.-K. Lee, "Daily life activity tracking application for smart homes using android smartphone," in *Advanced Communication Technology (ICACT), 2012 14th International Conference on.* IEEE, 2012, pp. 241–245.

[92] Google. (2017) Android developer. [Online]. Available: https://developer.android.com/

[93] C. M. Bishop, *Neural networks for pattern recognition.* Oxford university press, 1995.

[94] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[95] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[96] L. Xu, J. S. Ren, C. Liu, and J. Jia, "Deep convolutional neural network for image deconvolution," in *Advances in Neural Information Processing Systems*, 2014, pp. 1790–1798.

[97] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification." in *AAAI*, vol. 333, 2015, pp. 2267–2273.

[98] G. Wang, T. Konolige, C. Wilson, X. Wang, H. Zheng, and B. Y. Zhao, "You are how you click: Clickstream analysis for sybil detection." in *Usenix Security*, vol. 14, 2013.