Security and Usability in Mobile and IoT Systems

by

Tao Li

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved January 2020 by the
Graduate Supervisory Committee:

Yanchao Zhang, Chair
Junshan Zhang
Guoliang Xue
Deliang Fan

ARIZONA STATE UNIVERSITY

August 2020

ABSTRACT

Mobile and Internet-of-Things (IoT) systems have been widely used in many aspects of human's life. These systems are storing and operating on more and more sensitive data of users. Attackers may want to obtain the data to peek at users' privacy or pollute the data to cause system malfunction. In addition, these systems are not user-friendly for some people such as children, senior citizens, and visually impaired users. Therefore, it is of cardinal significance to improve both security and usability of mobile and IoT systems. This report consists of four parts: one automatic locking system for mobile devices, one systematic study of security issues in crowdsourced indoor positioning systems, one usable indoor navigation system, and practical attacks on home alarm IoT systems.

Chapter 1 overviews the challenges and existing solutions in these areas. Chapater 2 introduces a novel system *ilock* which can automatically and immediately lock the mobile devices to prevent data theft. Chapter 3 proposes attacks and countermeasures for crowdsourced indoor positioning systems. Chapter 4 presents a context-aware indoor navigation system which is more user-friendly for visual impaired people. Chapter 5 investigates some novel attacks on commercial home alarm systems. Chapter 6 concludes the report and discuss the future work.

To My Family.

ACKNOWLEDGMENTS

I owe my gratitude to several people who have advised, supported, or inspired me during the course of the work.

First, I want to truly thank my advisor Dr. Yanchao Zhang, who through his immense patience and forbearance has shown me an exciting world of research. You have been a constant source and guide of knowledge for the past five years. Without your encouragement and help, I would not have completed this work. Thank you.

Second, I would like to express my sincere appreciation and thanks to Dr. Junshan Zhang, Dr. Guoliang Xue, and Dr. Deliang Fan for serving on my supervisory committee. They give me many constructive and brilliant comments and suggestions in many different ways over these years..

I have received the help and support from a great number of people, including, but not limited to, my colleagues Dr. Rui Zhang, Dr. Jinxue Zhang, Dr. Jingchao Sun, Dr. Yimin Chen, Dianqi Han, Ang Li, Yan Zhang, Lili Zhang, Jiawei Li, Dr. Junwei Zhang, and Dr. Xin Yao, who I have closely worked with over the years.

My research work and the writing of this dissertation could not have been completed without the enormous support of my family. I thank my family members for being a constant source of support and encouragement. My deepest gratitude also goes to my friends who encourage and believe in me over the time.

I also gratefully acknowledge the financial support I received from the National Science Foundation through grant CNS-1933069, CNS-1824355, CNS-1619251, CNS-1514381, CNS-1421999, and CNS-1320906.

TABLE OF CONTENTS

CHAPTER Page

CHAPTER Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

Figure Page

Chapter 1

INTRODUCTION

1.1 Mobile Authentication

The human society is in a wireless and mobile era. According to the Cisco Virtual Networking Index [1], 497 million mobile devices (mainly tablets, smartphones, and laptops) were added in 2014, and the number of global mobile devices in 2014 reached 7.4 billion and will reach 11.5 billion by 2019 at a CAGR of 9%. People are using mobile devices in every aspect of life, including work, education, voice/video communications, Internet browsing, web transactions, online banking, reading, multimedia playing, etc.

Mobile device losses/thefts are skyrocketing and posing severe threats to data security. According to a 2012 Kensington study [2], one laptop is stolen every 53 seconds; 70 million smartphones are lost each year, with only 7% recovered; and 4.3% of company-issued smartphones are lost/stolen every year. The true cost of a lost/stolen mobile device goes far beyond the device cost due to the lost productivity, the loss of intellectual property, data breaches, and legal fees.

The most common defense against device losses/thefts is to set a password on the mobile device. Unfortunately, the 2015 Kaspersky Lab survey [3] shows that 31% of smartphones and 41% of tablets are not password-protected. In addition, the time window for a password-protected device going from the unlocked mode to the locked mode may be long enough for a capable attacker to access all the sensitive information on the lost/stolen device. For example, the auto-lock options on iPad 2 include 2 min, 5 min, 10 min, 15 min, and NEVER. Many users choose a longer

time period or even NEVER for convenience. If an unlocked device is lost/stolen, the user's sensitive information is fully accessible to whoever possesses the device.

Continuous authentication aims to continuously verify the identity of the user using a mobile device and is naturally a candidate defense against device losses/thefts. This line of work aims to verify the behavioral biometrics of the user exhibited in his keystrokes [4], finger touches on the screen [5], or app usage [6]. In addition to their relatively high false positives and negatives, these approaches often require a relatively long time window to collect sufficient data for capturing the behavioral biometrics. The attacker, however, may quickly access the user's private data and then completely wipe out the device for reinstallation, rather than using the device for an extended period of time.

1.2 Crowdsourced Indoor Positioning Systems

Indoor positioning systems (IPSes) have received tremendous attention from the academia and industry. IPSes target large indoor environments where GPS signals are unavailable, unreliable, or inadequate. They can actively locate mobile users (devices), provide ambient location context, or navigate them through an indoor venue of interest. They can also enable many location-based services, e.g., location-based proximity advertising inside a shopping mall. There are many competing IPS technologies with great commercial potential. For example, WiFi-based IPS applications are expected to generate revenues up to $2.5 billion by 2020 [7].

A typical WiFi-based IPS depends on the ubiquity of indoor WiFi APs and works in three phases. In the training phase, the IPS collects Received Signal Strength (RSS) fingerprints for all possible indoor positions to build a fingerprint database. In the operating phase, the IPS searches its fingerprint database for the closest match to a submitted RSS fingerprint and then returns the corresponding position to the

requesting user. A calibration phase is also needed for the IPS to dynamically update its fingerprint database to deal with noisy WiFi signal measurements and physical environment changes such as indoor layout changes and additions/deletions of WiFi APs.

There is significant research on crowdsourced techniques such as [8, 9, 10, 11] to reduce the calibration effort in WiFi-based IPSes. The common idea is to outsource the collection of RSS fingerprints to indoor smartphone users. These techniques combine the IMU sensor data and RSS fingerprints in various ways and demonstrate significant success. The security issue, however, is overlooked in existing studies. In particular, a crowdsourcing worker can submit fake data to induce a low-fidelity fingerprint database at the IPS for various motives. For example, the worker may want to claim rewards without actually collecting data, be hired by a malicious business competitor of the IPS operator, or misbehave just for fun.

1.3 Crowdsourced Indoor Navigation Systems

There is a strong need for usable navigation services in large indoor environments such as shopping malls, hospital, and museums. For example, shoppers often spend much time looking for the stores they are interested in. Although some shopping malls provide indoor maps at a few locations, it is still not easy for shoppers to understand the maps and reach the target stores. In particular, people with poor spatial awareness, children, senior citizens, and visually impaired users are more in need of a user-friendly indoor navigation system.

We use the shopping-mall example in Fig. 1.1 to illustrate how a usable indoor navigation system works in our opinion. Alice is now standing beside the GAP store, but her target store is Nike. Since Alice may not know the directions in the shopping mall, it would be confusing for her if we give instructions such as “Go north" or “Go

east". Instead, a more usable instruction can be “Go towards the next Coach store and then keep walking". The Coach store is adjacent to GAP and is easy to find. When she approaches the Apple store, we can give an instruction like “Turn right at the Apple store and keep walking". When she approaches the Nike store, we can give the last instruction as “The Nike store is on your right". All the instructions can be given as audio signals through the smartphone. Alice needs not to take time to understand the physical mall map or watch her phone screen while walking. Such voice instructions are particularly useful for visually impaired users to reach their target stores and also be well aware of the mall environment to gain similar shopping experience enjoyed by sighted people.

To provide the above navigation service, the system needs to keep tracking the shopper in a floor plan with context-aware information such as store names. Most existing indoor localization systems like [12], [13] focus on the localization accuracy and must be combined with a labeled floor plan with location-store mappings to provide the usable navigation service. Such labeled floor plans are often unavailable or quite difficult to obtain. For example, it would be infeasible or incur a prohibitive cost for a national indoor navigation service provider to obtain a labeled floor plan for each indoor environment it intends to cover. While rough floorplans may be available in some scenarios (e.g. shopping malls), they are often only accessible through a static PDF or JPG file with outdated store information. Without accurate dimension and up-to-date store information, they are inadequate to provide the localization and navigation services that we envision. Shen *et al.* [11] and Wang *et al.* [9] explore crowdsourcing to build an indoor pathway map which nonetheless does not contain any store label. To the best of our knowledge, automatic construction of a context-aware indoor floor plan with labeled information is still an open challenge.

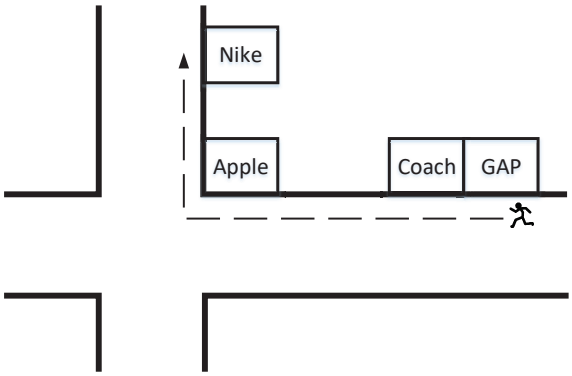


Figure 1.1: An Indoor Navigation Example.

1.4 Secure Home Alarm IoT Systems

With people's increasing attention on home security and the development of IoT technology, home security devices—such as smart cameras, smart alarms, smart locks, and smart doorbells—are flooding into the market. Compared with traditional security devices, these smart devices can provide better protection for your home and are more user-friendly. The home security market is estimated to reach 74.75 billion dollars by 2023 from 45.58 billion dollars in 2018 [14].

The home alarm system is a very popular home security product which has been provided by companies such as Ring (an Amazon company), Google, and Honeywell. Fig. 1.2 illustrates a typical home alarm system consisting of a base station, contact sensors, and extenders. Each contact sensor is associated with a magnet to monitor the OPEN or CLOSE state of the door or window. [1] The contact sensor is often installed on the door, and the magnet is usually installed on the door frame. When we open the door and separate the sensor from the magnet, the reed switch in the sensor detects low magnetic field strength and then triggers the sensor to report an OPEN event to the base station which in turn reports the event to the user's

[1] We use the door as an example in the rest of this paper.

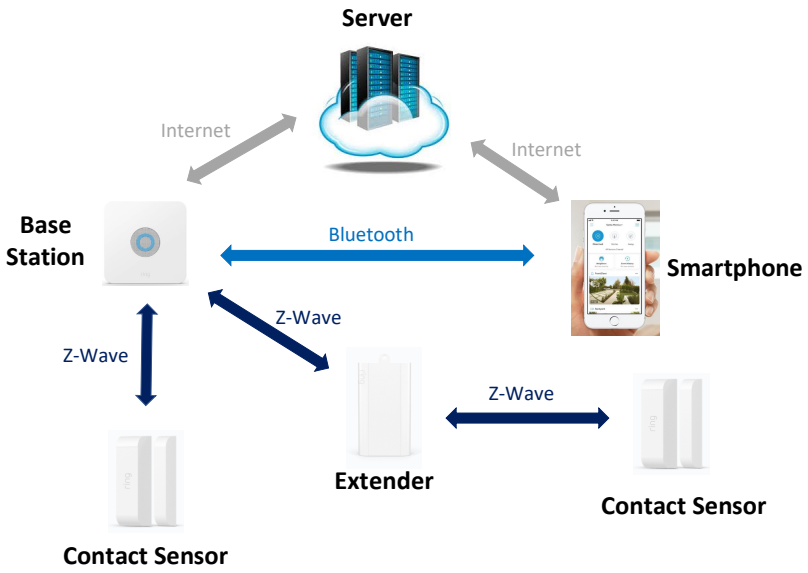


Figure 1.2: A Typical Home Alarm System.

smartphone and the alarm service provider if any. The extender is used to forward packets between the base station and contact sensors when they are too far away from each other. Communications between the base station and contact sensors are usually based on some low-cost protocols such as Z-Wave, Bluetooth Low Energy, and Zigbee. Normally, contact sensors are powered by a small battery, and the base station and extenders are connected to outlets.

The security of home alarm systems, however, is not well studied. Most attacks reported so far exploit vulnerabilities in the networking protocols. For example, Lamb [15] presented jamming and replay attacks to eliminate legitimate alarms and cause false alarms for multiple home alarm systems. Fouladi and Ghanoun [16] used a flaw in the Z-Wave protocol to reset the encryption key to a chosen value so that the attacker can inject unauthorized commands. Rouch *et al.* [17] and Fuller and Ramsey [18] introduced techniques to inject a fake controller (base station) into the network to control the home IoT devices. To the best of our knowledge, nobody has studied

the security issues arising from the home alarm system's physical components such as the reed switch in contact sensors.typical

Chapter 2

IMMEDIATE AND AUTOMATIC LOCKING OF MOBILE DEVICES AGAINST DATA THEFT

2.1 Overview

In this chapter, we present iLock, a secure and usable defense against device losses/thefts. iLock immediately and automatically locks a mobile device once it leaves the vicinity of its user. The key motivation behind iLock is that the departure of a user from his device causes the physical environment to change and thus noticeable changes in nearby wireless signals. So we can let the mobile device automatically, quickly, and accurately recognize its physical separation from its owner by detecting and analyzing the changes in wireless signals. Once significant physical separation from its user is detected, the device can immediately and automatically lock itself. iLock cannot help retrieve a lost/stolen device, but it can help prevent data theft. Specially, after iLock locks the device, the user can use various apps such as Find My Phone to track the device, remotely disable it, and even completely erase it.

iLock relies on acoustic signals and requires at least one speaker and one microphone that are available on most COTS mobile devices, such as smartphones, tablets, laptops, and all-in-one PCs. Once a user-defined vulnerable context (e.g., out of home) is automatically detected, the speaker keeps transmitting high-frequency acoustic signals inaudible to human ears. The signals are reflected by the user's body and finally reach the microphone after some delay. The device can then estimate its distance from the user based on the received signals and automatically lock itself once the distance estimation exceeds a user-defined threshold.

How could the user-device distance be estimated? One may simply let the speaker transmit an acoustic signal, which reaches the microphone via the speaker-user-microphone path. After computing the time-of-flight (ToF) as the difference between signal transmission and reception time, the device can estimate the user-device distance as $c \times \text{ToF}/2$, where c denotes the speed of sound about 340 m/s. This seemingly simple method unfortunately does not work because of very coarse-grained timestamps on mobile devices, which can be due to many reasons such as various delays between the application and physical layers [19]. For example, an error of 0.01 s may cause a distance-measurement error about 1.7 m which is obviously not acceptable for device locking.

iLock adopts a technique called FMCW (frequency modulated carrier wave) [20] to avoid computing the ToF directly based on inaccurate timestamps on mobile devices. FMCW transforms the time differences to frequency shifts between transmitted and received signals. With FMCW, the speaker changes the acoustic signal frequency linearly. The device computes Δf, the frequency difference between the signal transmitted at the speaker and the signal received by the microphone at the same time. Since the slope of the linear FMCW function is known, the ToF is roughly $\frac{\Delta f}{\text{slope}}$, and the user-device distance can still be estimated as $c * \text{ToF}/2$.

Implementing FMCW-based iLock on COTS mobile devices faces two critical challenges. First, the device must compute the frequency drift Δf as the frequency difference between the signals simultaneously transmitted at the speaker and arriving at the microphone. This seemingly simple requirement is difficult to fulfill on COTS mobile devices because the timestamps obtained from the OS are highly inaccurate. Second, the signal arriving at the microphone is actually a linear combination of multi-path signals coming from the direct speaker-microphone path, the speaker-user-microphone path, and other paths involving many other physical objects. The

device thus should be able to separate the signal from the speaker-user-microphone path from other multi-path signals.

2.2 Adversary Model and Design Goals

In this section, we first talk about the adversary model of the system and then present the design goals we want to achieve.

Adversary Model. We assume that the mobile device to protect is unlocked. This can be because the auto-lock option is disabled or has not taken effect if a long time window (e.g., 5 min) is chosen. The attacker possesses the device and tries to access sensitive information stored there. We consider three types of attackers according to their initial distance from the device relative to the (legitimate) user.

- Type-I attacker: This kind of attackers find the device the legitimate user accidentally lost in public places such as streets, restrooms, coffee shops, and subways. Type-I attackers are initially much farther away from the device than the user.

- Type-II attacker: Such attackers are still farther away from the device than the user, but the distance difference is very small. For example, the attacker can be a thief trying to steal the device from the user on a crowded bus/subway, and the attacker may also be a malicious coworker who just sat with the user for a meeting and saw the user leave without taking the device on the conference table.

- Type-III attacker: These attackers are closer to the device than the user. For example, the user may accidently put the device closer to the malicious coworker on the conference table and leave the meeting without taking the device.

Since iLock relies on acoustic signal transmissions and receptions, one may think about defeating iLock by letting the attacker jam the acoustic channel. Such jamming attacks are very easy to detect and mitigate. So we focus on dealing with the three types of attackers above.

Design Goals. iLock cannot help retrieve a lost/stolen device, but it can help prevent data theft on a lost/stolen device. We have the following design goals.

- iLock should be *device-free* and does not rely on any auxiliary device. It should also be applicable to most COTS mobile devices.
- iLock should *immediately* lock the device once the user-device distance exceeds a pre-defined threshold to minimize the time opportunity for data theft.
- iLock should be *automatic* and *user-friendly*. It should not require any explicit interaction between the user and device. Nor does the user's device-use habit need to be changed.
- iLock should be very *accurate* in detecting the user-device distance, which can translate into very low false positives and negatives for triggering device locking.

2.3 iLock

This section details the iLock design. We start by introducing FMCW in Section 2.3.1. Then we discuss how to defend against Type-I, Type-II, and Type-III attackers in Sections 2.3.2, 2.3.3, and 2.3.4, respectively.

2.3.1 Frequency-Modulated Carrier Waves

Fig. 2.1 gives a high-level overview of FMCW, and we refer the reader to [20] for a more detailed illustration. FMCW operations proceed in rounds. In each round

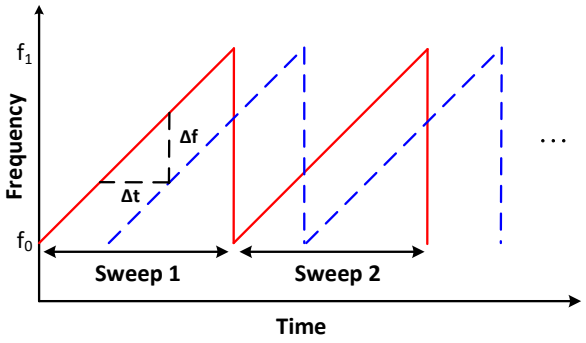


Figure 2.1: FMCW Illustration.

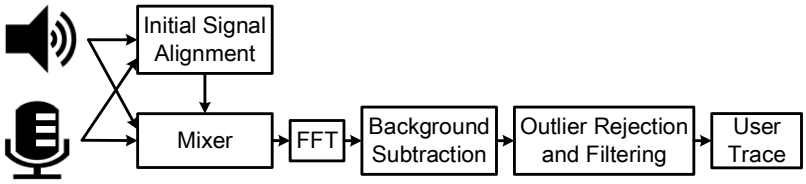


Figure 2.2: The System Framework of iLock.

referred to as a sweep, the transmitter linearly increases the transmission frequency from f_0 to f_1, where f_0 and f_1 are predetermined minimum and maximum frequencies. Each signal arrives at the receiver after some delay Δt (the so-called ToF). The transmitted and received signal frequencies for each sweep are depicted by red solid and blue dashed lines in Fig. 2.1, respectively. According to Fig. 2.1, it is clear that $\Delta t = \frac{\triangle f}{f_1 - f_0} T_{\text{sweep}}$, where T_{sweep} is the duration of each sweep. Finally, we can estimate the signal-travel distance $d = c\Delta t$, where c is the signal propagation speed.

2.3.2 Defeating Type-I Attackers: When Attackers Are Initially Farawy

iLock relies on FMCW to dynamically estimate the user-device distance and automatically locks the device once the user-defined safe distance is exceeded. iLock uses acoustic signals so that it can work on most COTS mobile devices with standard build-in microphones and speakers. Thus c is the speed of sound of about 340 m/s. The minimum FMCW frequency f_0 is set to be sufficiently high (e.g., 18 kHz) so that

the signal is almost inaudible to human ears, and the maximum FMCW frequency f_1 can be set to half the highest sampling frequency of the microphone. For example, most COTS smartphones support the sampling frequency up to 44.1 kHz, so we can set f_1 equal to 22 kHz. T_{sweep} is a design parameter dictating the tradeoff between maximum detection range and frequency drift resolution, which becomes clear shortly.

The implementation of FMCW-based iLock on COTS mobile devices faces two critical challenges. **First**, the device must compute the frequency drift Δf as the frequency difference between the signals simultaneously transmitted at the speaker and arriving at the microphone, as shown in Fig. 2.1. To do so, the transmitted and received signals for the same sweep should be properly aligned. This seemingly simple goal is difficult to achieve on COTS mobile devices because the timestamps obtained from the OS are highly inaccurate in contrast to the short sweep duration. Specifically, there are many reasons for the skew between the sending timestamp and actual signal-emission time [19]. For example, the transmission instructions have to be transferred from the application layer to the physical layer, which may be delayed by many system events such as system interrupts. Similar reasons can also account for the skew between the receiving timestamp got from the OS and the actual receiving time by the microphone circuit. More accurate time measurements can be obtained from the kernel, but this option is not feasible on mobile devices. **Second**, the signal arriving at the microphone is actually a linear combination of multi-path signals coming from the direct path between the speaker and microphone, the speaker-user-microphone path, and other paths involving many other physical objects. The device thus should be able to separate the signal from the speaker-user-microphone path from other signals.

Below we illustrate how iLock tackles these two challenges with the system diagram in Fig. 2.2. We assume Type-I attackers in this section such that the signals are reflected by only one human object (the user him/herself).

The Signal Alignment module is designed to deal with the first challenge. Specifically, the speaker transmits acoustic signals with the frequencies sweeping from f_0 to f_1, which arrive at the microphone after some delay. In ideal situations with accurate timestamps and static signal propagation environments, the time gap between transmitted and received signal vectors for the sweep that can be obtained from the transmitted and received timestamps should be constant, as shown in Fig. 2.1. Such gaps, however, may vary a lot across each sweep mainly due to inaccurate timestamps.

Our design leverages the observation that the physical distance between the speaker and microphone is fixed and usually very short relative to the user-device distance, [1] so the signals arriving from the direct speaker-microphone path dominate other multi-path components. If the sweep duration is so short that signal propagation environments are approximately static, the time gap between transmitted and received signal vectors on the direct path should be constant across each sweep regardless of inaccurate timestamps. Let $\sin(f_{\text{tx}}t)$ and $\sin(f_{\text{rx}}t)$ denote the transmitted and received signals at the same timestamp, respectively. The Signal Alignment module computes $\sin(f_{\text{tx}}t)\sin(f_{\text{rx}}t) = \frac{1}{2}(\cos[(f_{\text{tx}} - f_{\text{rx}})t] - \cos[(f_{\text{tx}} + f_{\text{rx}})t])$ and then uses a low-pass filter to get $\cos[(f_{\text{tx}} - f_{\text{rx}})t]$. Then we advance the received signal vector by an offset k to minimize the frequency difference $f_{\text{tx}} - f_{\text{rx}}$. If the microphone only receives the signals from the direct speaker-microphone path, there can be an almost perfect overlap between the transmitted and received signal vectors after the shifting with $f_{\text{tx}} - f_{\text{rx}} \approx 0$. Due to the presence of the user and other physical objects, the

[1] For example, the distances of the speaker to two microphones on a Samsung Galaxy S5 are 4.5 cm and 12.3 cm, respectively.

transmitted and received signals cannot overlap each other. Finally, the transmitted signals correspond to the red solid line in Fig. 2.1, and the advanced received signals correspond to the blue dashed line in Fig. 2.1.

Then the Mixer module is invoked to compute $\cos[(f_{\text{tx}} - f_{\text{rx}})t]$ in the same way as in the Alignment module for the transmitted and received signals at the same instant in the same sweep. Different physical objects lead to different reflection paths, each corresponding to a different time shift. So the FFT module is subsequently used in each sweep to extract these different frequency shifts. Since each frequency shift corresponds to a different ToF measurement and thus a different signal-travel distance, we plot the received signal powers at different distances in Fig. 2.3a, which are obtained from a microphone on a Samsung Galaxy S5 with $f_0 = 18$ kHz, $f_1 = 22$ kHz, and $T_{\text{sweep}} = 20$ ms. There are many horizontal strips with each corresponding to a different path the signal traveled from the speaker to microphone. Some strips are not stable with time, as user movements change the multi-path propagation environment. The strips around distance zero are the brightest, corresponding to the direct speaker-microphone path.

We then use the Background Substraction module to highlight the effect of user movements. Specifically, the physical objects other than the user (e.g., doors and walls) can be assumed to be static relative to user movements, which generally holds given the very short duration to detect user movements and then lock the device. Therefore, the reflection paths due to these static objects are static across the sweeps, so we can easily remove their effects via subtraction. Fig. 2.3b shows the substraction result, where the signal power decreases as the distance increases.

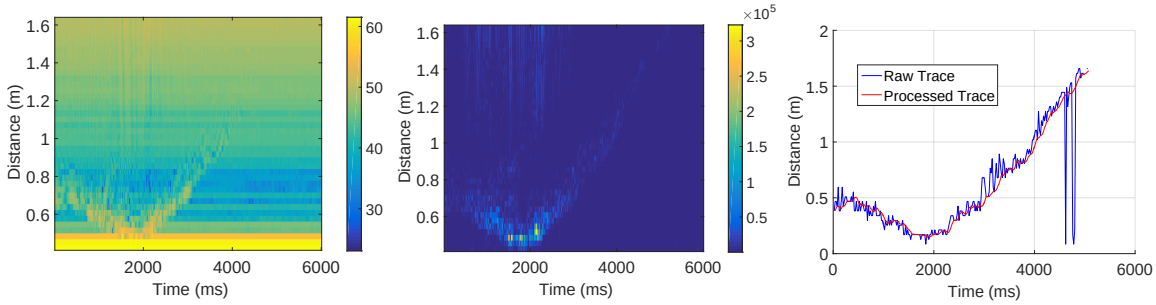
Next, we use the Kalman filter in the Outlier Rejection and Filtering module to smooth out the data. Fig. 2.3c shows the user's movement trace before and after outlier rejection and filtering. In this experiment, the user initially sits on the chair

with the smartphone on the table. Then he stands up and turns around to move away from the table and thus his smartphone. As we can see, his distance to the smartphone decreases when he stands up (around 2,000 ms) and increases when he moves away (after 2,000 ms).

When should the device be locked? In everyday life, the device is often placed within the arm's reach, so the user can set a threshold δ_1 about the arm length when installing iLock. We also define another distance threshold δ_2, beyond which the user can hardly put his device. iLock immediately and automatically locks the device when the user-device distance starts below δ_1 and then exceeds δ_2. We set $\delta_1 = 60$ cm and $\delta_2 = 1$ m in the experiments, and the user can freely adjust them in practice.

How accurate are the distance measurements in iLock? The resolution of distance measurements relies on that of ToF measurements which further depends on that of frequency measurements. The minimum frequency drift in iLock equals $1/T_{\text{sweep}}$ (i.e., the size of one FFT bin), which translates into a ToF resolution of $\frac{1/T_{\text{sweep}} * T_{\text{sweep}}}{f_1 - f_0}$. So the user-device distance resolution can be derived as $\frac{c}{2(f_1 - f_0)}$, for which we assume that the user-device distance is half of the speaker-user-microphone path length. With $f_1 = 22$ kHz, $f_0 = 18$ kHz, and $c = 340$ m/s, the user-device distance resolution is about 4.25 cm, which is sufficient to detect the user's significant departure from the device.

The maximum detection range for the user-device distance depends on both the sweep duration and also the speaker volume. Considering the sweep duration alone, we can compute the maximum user-device distance as $cT_{\text{sweep}}/2$, which equals 3.4 m if $T_{\text{sweep}} = 20$ ms. The speaker volume corresponds to transmission power and thus distance: the larger the speaker volume, the larger the transmission power consumption,



(a) Spectrogram (b) Spectrogram after Background Subtraction (c) User Leaving Trace

Figure 2.3: Single User Tracking with FMCW. Figure (a) Plots the Spectrogram after FFT on Each Sweep. Figure (b) Eliminates Static Multipath by Subtracting the Power of a Previous Sweep from the Current Sweep. Figure (c) Illustrates the User's Moving Traces before and after Outlier Rejection and Filtering.

the larger the detectable user-device distance, and vice versa. In our experiments, the 71% volume level leads to a maximum detection range at about 1.5 m.

Another issue worth mentioning is the impact of initial signal alignment on distance measurements. The net effect of initial signal alignment is to virtually place the speaker and microphone together. So each subsequent microphone-object-speaker distance measurement is actually $d^{'} = d - d_{\text{sm}}$, where d is the actual signal travel distance, and d_{sm} means the distance between the speaker and microphone. For most portable mobile devices, d_{sm} is relatively small in contrast to user movements and can be safely ignored. For larger mobile devices such as laptops and all-in-one PCs, d_{sm} can be easily estimated and then used to obtain d.

2.3.3 Defeating Type-II Attackers: When Attackers Get Closer

The basic iLock design in Section 2.3.2 assumes that the attacker is initially faraway from the device, so only the movement of the user him/herself needs to be

tracked. In this section, we discuss how to defeat Type-II attackers which are initially also close to the device but still at a greater distance than the user-device distance. There are many such scenarios in daily life. For example, the user leaves a conference room without taking his/her device on the table, where malicious coworkers or conference attendees try to access sensitive data on the user's device. The device may also slip out of the user's pocket or suitcase on public transport tools and be picked up by malicious passengers nearby. The existence of multiple persons (including the target user) nearby causes the target device to detect multiple movement traces. So the essential challenge is to identify the movement trace associated with the legitimate user, based on which to make salient device-locking decisions.

To begin with, we consider a common scenario that only one person near the device moves away from it. Even if other persons do not move, they may still have minor body movements which can be detected by the device. Since the target user is assumed to be initially closer to his/her device than other persons, his/her movement trace can be easily singled out based on the initial closer distance measurement. Fig. 2.4 shows an exemplary scenario where the target user leaves but the attacker stays, and Fig. 2.5 corresponds to the case that the attacker leaves but the user stays. It is very clear that the target user's movement trace can be easily identified, based on which the device can determine whether to lock itself according to the same rules in 2.3.2.

There can be ambiguity if the user-device distance is not much smaller than the attacker-device distance, especially when there are more than two persons near the device who may leave or stay with the device around the same time. For example, multiple passengers (including/excluding the target user) may exit at the same bus stop. As a result, there can be multiple movement traces corresponding to leaving persons and also multiple ones for staying persons. Leaving traces are easier to be

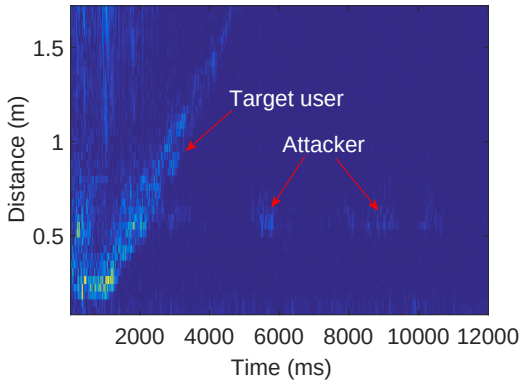


Figure 2.4: The Scenario Where the User Leaves and Attacker Stays.

distinguished from staying traces because the latter correspond to relative stable and smaller distances. But the leaving traces themselves may intersect, so may the staying traces themselves. The limited resources on COTS mobile devices make it impossible to accurately identify the movement trace for each individual person. Fortunately, our goal is to preserve data security in the case of device thefts/losses, so it makes more sense to weigh false positives over false negatives. Under the assumption that the target user is initially closer to the device than other persons nearby, we can take an aggressive approach as follows. We first construct a set of candidate leaving traces from the distance measurements. For example, if two persons leave the device with their leaving traces intersecting each other, we can construct four candidate leaving traces. Among the candidate traces satisfying the locking condition (i.e., starting below δ_1 and exceeding δ_2), we select the one whose minimum distance measurement is the smallest, denoted by d_{L}. Similarly, we construct a set of candidate staying traces, from which to select the one whose minimum distance measurement is the smallest, denoted by d_{S}. Let ω denote the maximum possible distance measurement error. As long as $d_{\mathrm{L}} \leq d_{\mathrm{S}} + 2\omega$, iLock associates the leaving trace with the target user and immediately locks the device.

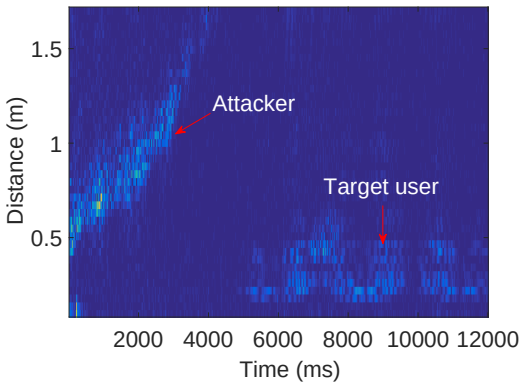


Figure 2.5: The Scenario Where the User Stays and Attacker Leaves.

2.3.4 Defeating Type-III Attackers: When Attackers Are Closer than the User

Now we illustrate how iLock withstands a Type-III attacker, the strongest one who is even closer to the device than its legitimate user (e.g. two scenarios in Fig. 2.6). Such attack scenarios are not unusual. For example, the user sits very close to the attacker in a conference room and accidentally puts the device closer to the attacker. The previous defenses against Type-I and Type-II attackers thus fail.

The fact that more and more COTS mobile devices have two or more microphones enables possible defenses against Type-III attackers. For example, Fig. 2.6 shows dual microphones on one smartphone, where Mic2 at the bottom is mainly used for voice recording, and Mic1 at the top is designed for noise cancellation. Such dual-microphone configurations are very typical on current smartphones. The left sub-figure in Fig. 2.6 depicts a scenario where the user and attacker are closer to Mic2 and Mic1, respectively. In this scenario, the user's significant departure from the device can still be identified based on the distance measurements at the two microphones, in which case the device can be immediately locked. In contrast, the right sub-figure in Fig. 2.6 corresponds to a scenario in which the attacker and target user have similar distance to both microphones. The system will also lock the device

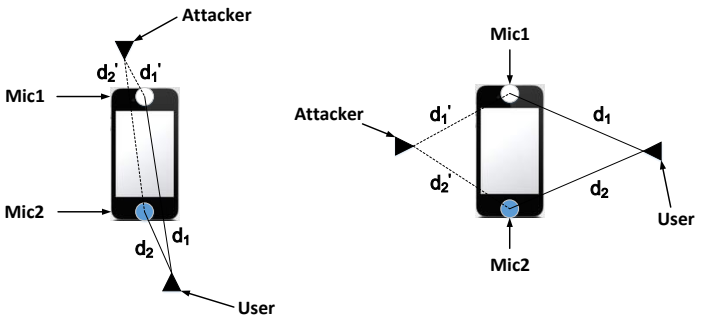


Figure 2.6: Two Scenarios in Which the Attacker is Closer to the Device than the Target User.

immediately to ensure strong data security when there is an ambiguity in the right scenario.

Relying on dual microphones, our solution applies to Type-III attackers with arbitrary locations with regard to the microphones and the user. We additionally assume that the relative orientation changes between the device and user before user movements can be automatically estimated with high precision through existing techniques. For example, the latest result we are aware of [21] can reach a precision of 5° based on IMU sensors. Since the initial relative orientation when the user is using the device is known (i.e., either landscape or portrait mode), we can calculate the final relative orientation when the user stop using the device. As a result, we just need to compare the orientation of candidate leaving user measured by two microphones with the orientation of target user calculated by IMU sensors. We also notice that the relative user-device orientation is approximately fixed, as a normal user typically walks along a straight line with a short distance from the device instead of in a zigzag fashion.

Our solution uses the distance measurements at Mic1 and Mic2 in a cohesive way. Specifically, every moving physical object near the device can lead to a speaker-object-microphone distance measurement at both Mic1 and Mic2 according to the FMCW

technique. Let $d_1(t)$ and $d_2(t)$ denote the distance measurements of Mic1 and Mic2 at time t, respectively. Note that COTS devices allow these two measurements to be perfectly aligned in time, i.e., with the same sampling clock. Consecutive distance measurements of the same object at the same microphone lead to a movement trace, either staying or leaving. Since Mic1 and Mic2 are very close to each other on the device in contrast to the user-device distance, they produce highly correlated movement traces for the same object. Assume that iLock finds two such correlated traces, so the next step is to determine whether these leaving traces should be associated with the user and triggers device lock if so. However, the distance measurement isn't accurate and stable enough to discover the orientation of candidate leaving trace, so we introduce a new metric as follows,

$$\eta(t) = \begin{cases} -1 & \text{if } d_1(t) - d_2(t) > \delta_{\text{dual}}, \\ 0 & \text{if } |d_1(t) - d_2(t)| \leq \delta_{\text{dual}}, \\ 1 & \text{if } d_2(t) - d_1(t) > \delta_{\text{dual}}, \end{cases}$$

where δ_{dual} is a system threshold and set to the theoretical distance resolution of 4.25 cm. We proceed to compute $\hat{\eta} = \frac{1}{N}\sum_{t=1}^{N}\eta(t)$, where N denotes the number of distance measurements. Obviously, $\hat{\eta}$ always belongs to $[-1, 1]$. When $\hat{\eta}$ is closer to 1 (-1), the object is closer to Mic1 (Mic2). If $\hat{\eta}$ is closer to 0, the object is about the same distance from Mic1 and Mic2.

We conjecture that $\hat{\eta}$ is closely tied to the device-object orientation and confirm it by experiments on a Samsung Galaxy S5. As shown in Fig.2.7, we fix the user's moving direction and evaluate $\hat{\eta}$ in eight different orientations (45° separation) by rotating the phone around its fixed center. 20 experiments are done for each orientation, and the distribution of $\hat{\eta}$ is shown in Fig. 2.8. We can observe that the data for symmetric orientations with regard to the vertical axis (e.g., 225° vs. 135°) overlap. So do the data for adjacent orientations (e.g., 45° vs. 90°). This observation is anticipated due

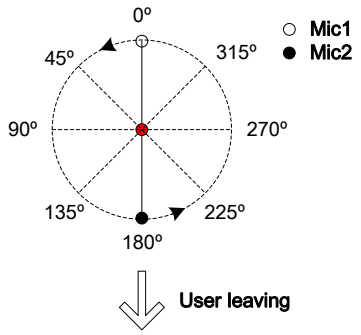


Figure 2.7: Different Orientations of the Phone When the User is Leaving.

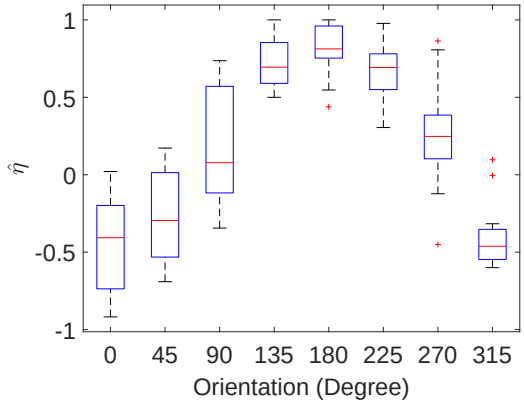


Figure 2.8: The Measured $\hat{\eta}$ in Eight Different Orientations as Illustrated in Fig. 2.7.

to distance measurement errors and also because $\hat{\eta}$ relates to only relative distance measurements. But there is a clear distinction between the data for orientations far apart (e.g., 0° vs. 180° and 45° vs. 225°).

The above observation can be explored as follows. First, we obtain a more fine-grained $\hat{\eta}$-orientation distribution than that in Fig. 2.8, which can be device-specific. The obtainment of this distribution is a one-time process and can be done when the user installs and enrolls into iLock. Once two correlated leaving traces are detected, iLock computes $\hat{\eta}$ as above, based on which to find the most probable orientation $\hat{\eta}$ corresponds to. If the likelihoods for multiple orientations are sufficiently close, all of them are candidate orientations. Recall that the initial user-device orientation can be

precisely obtained beforehand, and the user normally works in the same orientation within a short distance where iLock targets. If any candidate orientation is within a predefined threshold from the initial user-device orientation, the leaving traces are determined to be associated with the legitimate user, so iLock immediately locks the device.

2.4 Implementation and Evaluation

We implement iLock and obtain similar evaluation results in several COTS Android devices such as Samsung Galaxy S5 and Xiaomi Redmi 2. For lack of space, only the experimental data on Samsung Galaxy S5 are reported in this report. The Samsung Galaxy S5 phone has a Quad-core 2.5 GHz Krait 400 CPU, 2 GB RAM, and a 5.1-inch display. There are also two microphones, Mic1 at the top and Mic2 at the bottom. The speaker-Mic1, speaker-Mic2, and Mic1-Mic2 distances are 4.5 cm, 12.3 cm, and 14 cm, respectively. By default, the FMCW frequencies range from $f_0 = 18$ kHz to $f_1 = 22$ kHz; the sweep duration is $T_{\text{sweep}} = 20$ ms; and the speaker volume is 71%. One experiment is done in the university library, and all the others are done in a typical $12' \times 24'$ research office with desks, cabinets, computers, and six students. Unless specifically noticed, our experiment below is done on a table of 72cm height in our office with the orientation 0°; and the user stands up, turns around, and walks away with normal speed about 1.51 steps/second. Below we report the performance of iLock against Type-I, Type-II, and Type-III attackers, respectively.

2.4.1 Evaluation with Type-I Attackers

Recall that Type-I attackers are far away from the device when the user moves away. iLock in this scenario just needs to recognize the movement trace of the user alone and then locks the device if the trace starts below the near-distance threshold

δ_1 and exceeds the far-distance threshold δ_2. The experiments are conducted in a $12' \times 24'$ office with six PhD students. We set $\delta_1 = 0.6$ m (a typical arm's reach) and $\delta_2 = 1$ m beyond which a typical user does not put the device. In our experiments, a male user uses the phone for a while and then leaves it unlocked on the table, in which case iLock is automatically activated. Note that the triggering events for iLock can be automatically detected by many existing methods, e.g., through detecting when the user stops touching/holding the unlocked phone via inertial gyroscope and accelerometer sensors.

False Negatives. We first evaluate the false-negative rate of iLock through 400 experiments. In each experiment, the user puts his phone in a random position and an arbitrary orientation within δ_1. The user leaves the device in his usual way. As soon as the user-phone distance exceeds 1 m (i.e., δ_2), iLock theoretically should lock the phone. The results are quite encouraging. Specifically, the phone is successfully locked 395 times, which lead to a locking rate (true-positive rate) of 98.75% or a false-negative rate of 1.25%.

False Positives. We then evaluate the false-positive rate of iLock. In this experiment, we put the unlocked phone randomly on the desk just besides the user (within δ_1). Instead of leaving the desk and phone, the user performs regular minor movements such as typing, writing, drinking, rotating his head/shoulder, and swinging back-and-forth. Zero false device locking occurs in the entire 15 minutes, indicating an extremely low false-positive rate in practice.

Impact of Phone Orientations. The next experiment is to investigate the effect of phone orientations. We change the phone's relative orientation to the user by rotating it according to Fig. 2.7. For each orientation, the user moves away from the phone 50 times in his own way, for which each movement starts from a random position within δ_1 and goes beyond δ_2 from the phone.

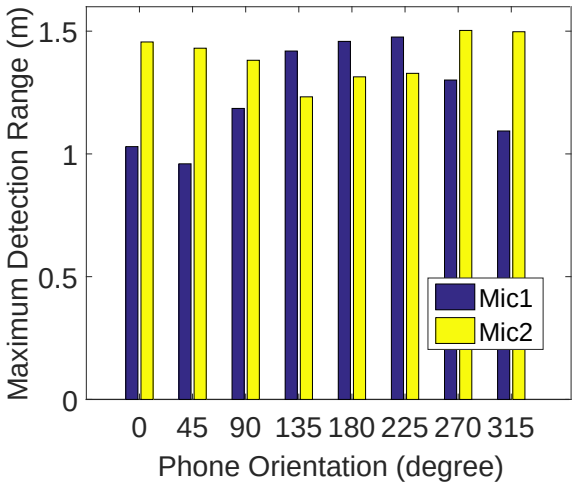


Figure 2.9: Maximum Detection Range vs. Orientations.

Fig. 2.9 illustrates the maximum detection ranges of Mic1 and Mic2 for different phone orientations. When the phone orientation is around 0° (180°), Mic2 (Mic1) yields a larger maximum detection range due to the closer distance between the user and Mic2 (Mic1). On the Samsung Galaxy S5, Mic2 is the master microphone, and Mic1 is designed for noise cancellation. So we can see that the average maximum detection range of Mic2 is larger than that of Mic1. Finally, combing the distance measurements from Mic1 and Mic2, iLock can always detect the user movement up to 1.4 m for any orientation.

Fig. 2.10 plots the true-positive rates for each orientation based on Mic1, Mic2, and their combination Mic1+Mic2. As expected, the peak performance for using Mic1 alone and Mic2 alone occur around 180° and 0° orientations, respectively. In addition, Mic2 shows better performance overall due to its higher capability as the master microphone. Finally, if we lock the phone as long as either one microphone detects a leaving trace, the true-positive rate is always above 90% regardless of initial phone orientations.

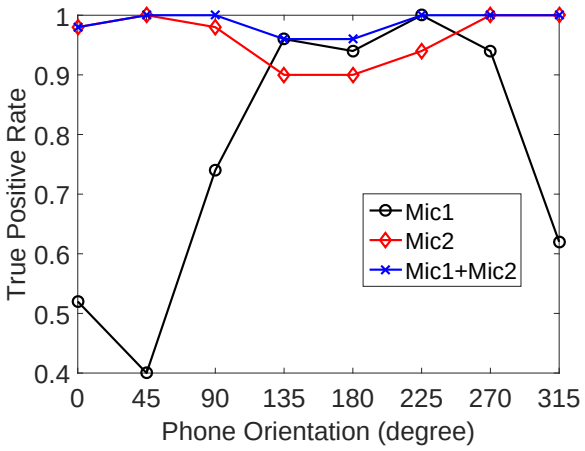


Figure 2.10: True-positive Rates vs. Orientations.

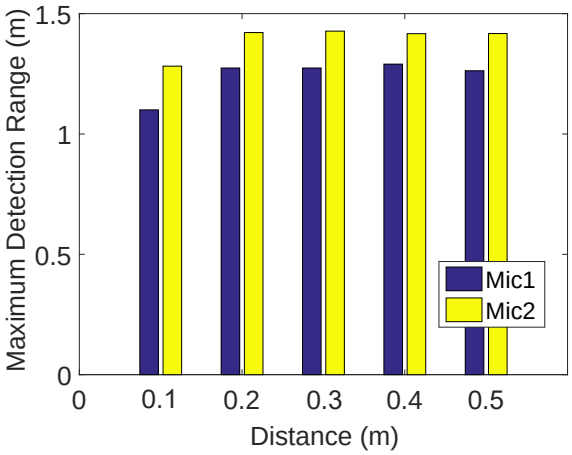


Figure 2.11: Maximum Detection Range vs. Phone-user Distance.

Impact of Initial Phone Positions. We also evaluate the impact of initial phone positions. In this experiment, the initial phone-user distance changes from 10 cm to 20 cm, 30 cm, 40 cm, and 50 cm, and the phone orientation is fixed to 0°. Fig. 2.11 and Fig. 2.12 show the maximum detection ranges and true-positive rates, respectively. We can see that the true-positive rate with Mic2 alone or Mic2 and Mic1 together can yield very high true-positive rates up to 100% for all distance settings. So initial phone positions have very little impact on iLock.

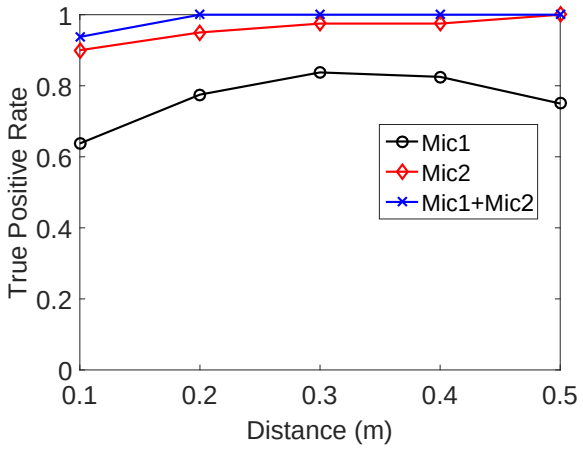
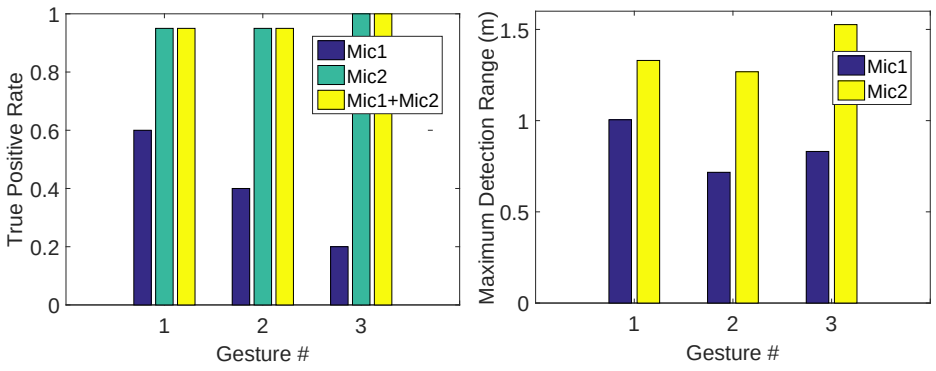


Figure 2.12: True-positive Rate vs. Phone-user Distance.

Impact of Departing Gestures. The user may leave the device with different gestures. Intuitively speaking, the departing gesture should not affect the detection performance, as iLock only measures the user-device distance. We confirm this intuition by experimenting three common gestures. In the first gesture which is the default in our experiments, the user stands up, turns around, and walks away. In the second gesture, the user initially stands facing the phone and then steps back to leave. In the final gesture, the user rotates the chair, stands up, and then moves away. Each gesture is performed 20 times, and the average maximum detection ranges and true-positive rates are shown in Fig. 2.13. We can see that Mic2 and Mic1+Mic2 produce very high and stable true-positive rates for all three gestures.

Impact of Departing Speeds. To evaluate the impact of moving speeds, we let the user perform the second gesture above with slow, normal, and fast speeds, corresponding to about 1.15, 1.51, and 2.0 steps/second, respectively. In this experiment, the user leaves 20 times for each speed setting, while the phone is initially 20 cm away at the 0° relative orientation. As we can see from Fig. 2.14, the performance of iLock becomes non-satisfactory when the user steps back at 2.0 steps/second. The main reason is that the fast speed reduces the time span for the same distance range, which in



(a) True-positive Rate (b) Maximum-detection Range

Figure 2.13: Performance of Three Leaving Gestures.

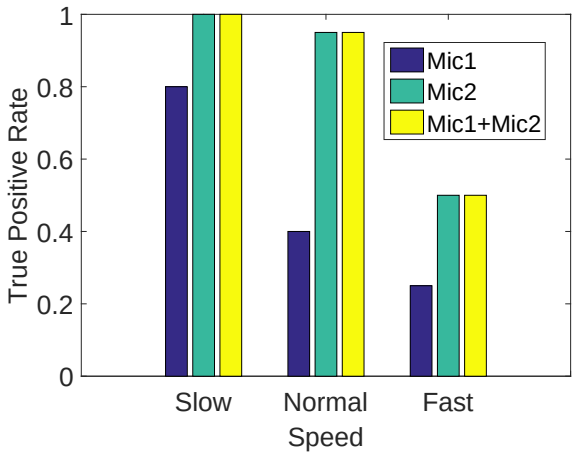


Figure 2.14: True-positive Rate vs. Leaving Speeds.

turn reduces the number of distance measurements given that the microphones have the constant sampling frequency. Fortunately, a normal user does not step back as fast as 2.0 steps/second. So the true performance of iLock is more reflected under the relatively slow and normal speeds.

Impact of Vertical Positions. The phone's vertical position may be different in various scenarios. For example, we tend to leave the phone on the desk around 70 cm high while in an office, on a chair about 40 cm high while on a subway, and the bar table about 100 cm high while in a bar. Fig. 2.15 shows the performance of iLock

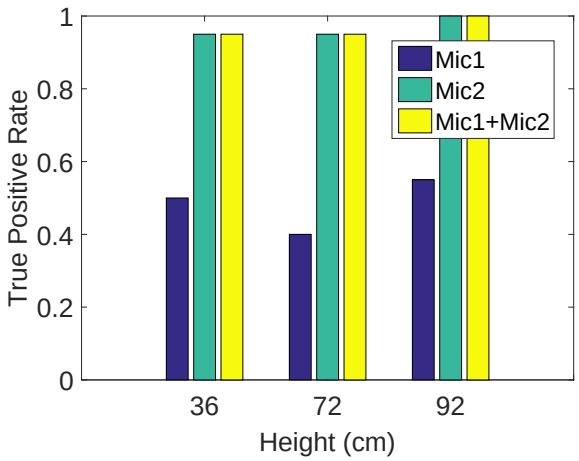


Figure 2.15: True-positive Rates vs. Phone Heights.

under different heights: 36 cm, 72 cm, 92 cm. For each height, the user moves away with the second gesture above for 20 times. We can see that different heights have very little impact on the true-positive rates of iLock.

Impact of Speaker Volumes. iLock detects the leaving movement by signal reflections, so the signal strength can potentially affect its performance. We conduct the experiment under three volume levels corresponding to three signal strengths: low (26%), medium (52%), and large (71%). From Fig. 2.16, it is of no surprise to see that the performance via Mic2 alone or Mic1+Mic2 are quite high for medium and high volume settings.

Impact of Different Users. We also ask six PhD students to use iLock. Each student leaves in his own way for 20 times with the gesture and speed he likes. As shown in Fig. 2.17, iLock achieves a true-positive rate of 85% for student 2, 95% for student 5, and 100% for the rest. It is worth noting that student 2 walks much faster than others in the experiments, leading to the similar observation as in Fig. 2.14

Impact of Experimental Environments. We finally evaluate iLock in the lobby of the university library. The lobby is about 32,000 square feet and contains many

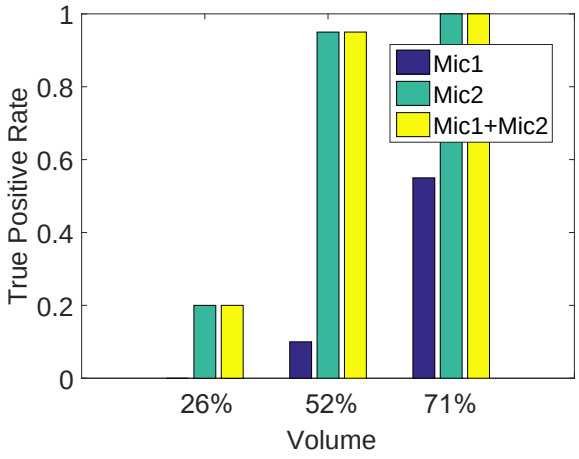


Figure 2.16: True-positive Rates vs. Different Volumes.

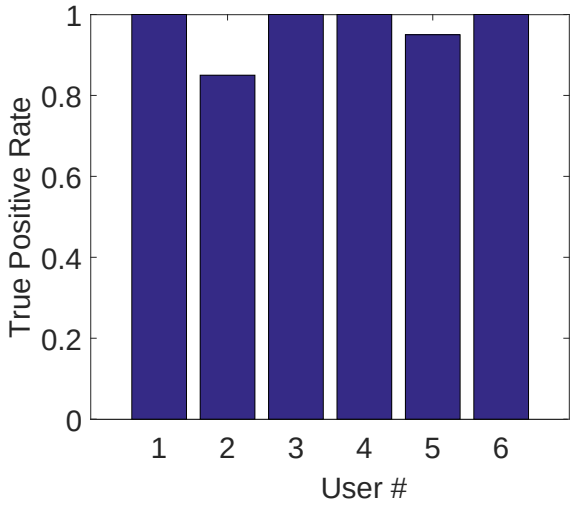


Figure 2.17: True-positive Rates vs. Different Users.

tables, sofas and public desktop computers. During our experiment, there is a lot of noise from the vending machines, public computers, and student talks. In addition, the students walk around without our control, but we make sure that they are at least 1 m from the phone. The user puts the phone randomly on a table and leaves it 20 times with a normal speed under gesture 2. We obtain a true-positive rate of almost 100% by using Mic2 alone or Mic1+Mic2. So iLock can work very well in noisy and uncontrolled environments.

2.4.2 Evaluation with Type-II Attackers

We also evaluate iLock against Type-II attackers who get closer to but are still farther away from the device than the legitimate user. With the presence of Type-II attackers, iLock can detect multiple movement traces and needs to decide which trace is associated with the user. For this experiment, we use the Precision and Recall metrics defined as follows,

$$\texttt{Precision} = \frac{\#\texttt{TP}}{\#\texttt{TP} + \#\texttt{FP}} \text{ and } \texttt{Recall} = \frac{\#\texttt{TP}}{\#\texttt{TP} + \#\texttt{FN}}, \tag{2.1}$$

where #TP is the number of user departures correctly associated with the user, #FP is the number of other users' departures incorrectly associated with the user, and #FN refers to the number of user departures not associated with the user by mistake.

The experiment involves the user and one attacker, and their distance difference to the device varies from 20 cm to 30 cm, 40 cm, 50 cm, and 60 cm. For each distance difference, the user leaves 20 times while the attacker stays, and then the attacker leaves 20 times while the user stays. The Precision and Recall results based on Mic1+Mic2 are shown in Fig. 2.18. We can see that precision is always above 95%, corresponding to very low false-alarm rates. In contrast, the recall increases from 80% to 95% when the distance difference becomes larger, as larger distance difference makes it easier to distinguish the user's trace from the attacker's.

2.4.3 Evaluation with Type-III Attackers

Now we report the performance of iLock against Type-III attackers. This experiment involves the user and one attacker who is always closer to the phone than the user. As shown in Fig. 2.20, we use five representative scenarios in which the user and attacker are in different positions and orientations relative to the phone. In each scenario, the user leaves the device 20 times while the attacker stays, and then the

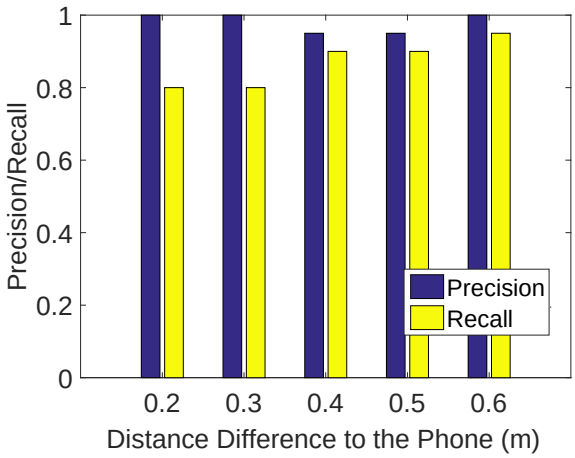


Figure 2.18: Precision and Recall with a Type-II Attacker.

attacker leaves 20 times while the user stays. In addition, the initial orientation of the device relative to the user can be accurately estimated with existing techniques [21]. Once two highly correlated leaving traces are detected, the metric $\hat{\eta}$ is computed according to the description in Section 2.3.4. Then we find the most probable orientation for $\hat{\eta}$ based on a fine-grained $\hat{\eta}$-orientation distribution, which we obtain beforehand for the Samsung Galaxy S5. Next, we compare the discovered orientation with the device's initial orientation relative to the user. Note that, in Fig. 2.8, $\hat{\eta}$ distributions of adjacent orientations overlap with each other, so we associate the traces discovered in nearby orientations to the target user to improve true positive rate. For example, if the device's initial orientation relative to the user is 180°, the leaving traces discovered between [135°, 225°] will be associated to the target user and the system locks the device immediately to ensure data security. Users can devise their own mechanism to balance Precision and Recall.

As we can see in Fig. 2.20, the Precision and Recall results are overall quite acceptable for all five scenarios. The worst performance is observed when there is a small orientation difference between the user and attacker relative to the phone (i.e.,

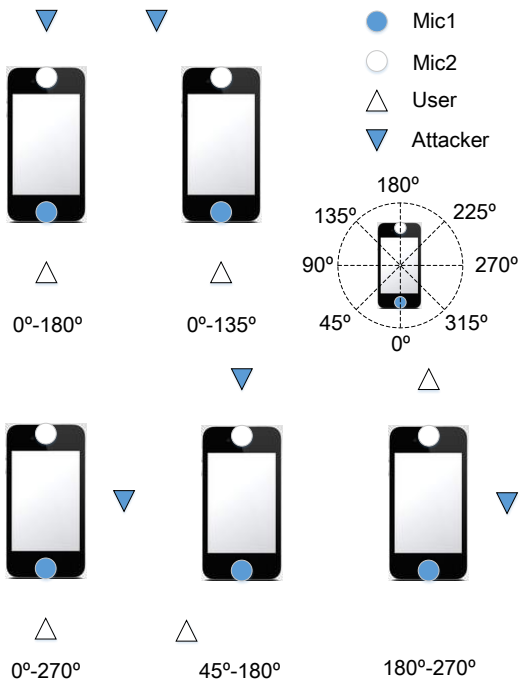


Figure 2.19: Representative Scenarios with Type-III Attackers, Where x-y Corresponds to the User's Orientation x and Attacker's Orientation y in the Shown Orientation Graph.

0°-270° and 180°-270°). This result is expected, as the smaller orientation difference makes it harder to distinguish the user's movement from the attacker's.

2.5 Discussion

2.5.1 Energy Consumption

iLock incurs additional energy consumption on a mobile device in two main aspects. First, iLock needs to transmit high-frequency modulated acoustic signals and also record the signals reflected by physical objects. It is shown [22] that such acoustic transmitting and recording on Samsung Galaxy S5 may incur an energy consumption of about 800mW with Monsoon Power Monitor. Secondly, iLock consumes energy

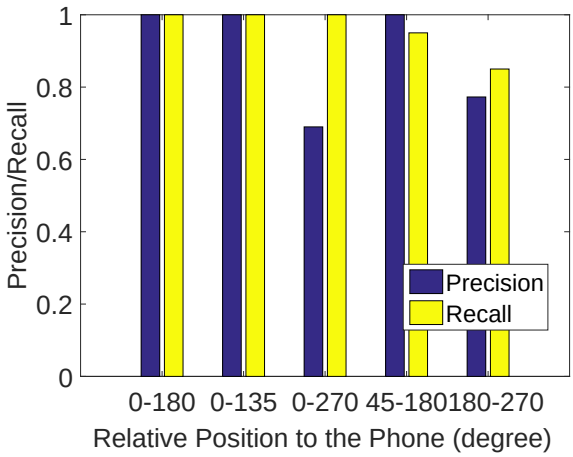


Figure 2.20: Precision and Recall with a Type-III Attacker.

in data processing such as filtering, FFT, and mixing. In practice, iLock does not need to be activated all the time. In particular, iLock can only be activated when the device enters a vulnerable context. One such context is when the user stops using the device while the screen is still unlocked, and it is can be easily detected by exploring inertial sensors such as touchscreen, gyroscope, and accelerometer. Also note that many users spend most of the time in a safe zone such as home and office. Sophisticated localization techniques allow the device to accurately determine whether it is in a predefined safe zone. iLock is only activated when the device is out of the safe zone. So the energy consumption of iLock is quite amenable in contrast to its potentially huge benefits.

2.5.2 Other Potential Solutions

We also investigate and experiment other potential solutions. The most intuitive alternative is to directly analyze the received signals which can be perturbed by leaving movements. In the experiment, we indeed find some potential signal patterns for specific leaving gestures. So one may think about training a classifier to detect a user's leaving gesture. However, different users have different gestures, so every user

who wants to use the system has to train a classifier, a time-consuming and clumsy process. In addition, even the same user may leave the device in a different way in different scenarios. As a result, it is almost impossible to train a classifier that can differentiate all possible gestures of the same user. So we give up this method.

Another candidate approach is to rely on the Doppler effect caused by user movements. In particular, the speaker transmits acoustic signals with a fixed high frequency f_0, and the microphone records the reflected signals with frequency f_r. It follows that $f_r = \frac{c-v_r}{c-v_s} f_0$, where v_s is the speed of the reflection object (user), and c is the speed of sound. Since the receiver is stationary, $v_r = 0$. Then we can do an integration over v_s to get the distance the user moves. The Doppler shift, however, is very sensitive and can be induced by any body movement. Also, the frequency shifts by different body movements at different distances to the device are mixed together. As a result, we can hardly extract the user's movement pattern based on the Doppler effect and give up this idea as well.

Finally, one may think about implementing iLock based on WiFi or Bluetooth signals rather than acoustic signals. There are two primary reasons for not doing so. First, WiFi and Bluetooth interfaces are often very busy and occupied for data communications, while the speaker and microphone have much more idle time. Second, WiFi and Bluetooth signals propagate in the speed of light and have much higher requirement for time/frequency measurement accuracy, which is not attainable on COTS mobile devices. This is also the reason why existing FMCW implementations on WiFi signals use complicated and customized hardware not available on COTS mobile devices.

2.6 Related Work

There are three ways to prevent the attackers' illegal access to mobile devices and the sensitive data therein. The first one is one-time authentication that authenticates users when they try to unlock and use the device. The second one is to authenticate users continuously when they are using the device. The third one is to lock the device immediately once the current user has left. We will analyze advantages and disadvantages of each method in what follows.

There are significant research and practice related to one-time authentication. Typically, one-time authentication schemes can be classified into three categories: *Something-You-Know*, *Someone-You-Are*, and *Something-You-Have*. In the *Something-You-Know* paradigm, users are asked to input a simple PIN, an alphanumeric password, or a gesture/graphical password. This method is vulnerable to shoulder-surfing attacks. The *Something-You-Have* paradigm requires auxiliary hardware (e.g. Signet Ring [23]) which is possessed only by the legitimate user. We note that the non-COTS hardware is a potential obstacle for the wide adoption of this paradigm. A growing body of work follows the *Someone-You-Are* paradigm [24, 25, 26]. This approach relies on physiological or behavioral biometrics which are unique to each person. Common physical features consist of fingerprints, facial features, retina patterns, etc. Physiological authentication methods may be vulnerable to spoofing attacks [24]. Behavioral biometrics may include keystroke patterns [27, 28], touching gestures [29, 30], gaits [31, 32], etc. As said, a significant number of mobile users do not password-protect their devices, not to mentioning adopting more advanced one-time authentication techniques. In addition, the time window for a password-protected device going from the unlocked mode to the locked mode may be long enough for a capable attacker to

access all the sensitive information on the lost/stolen device. If an unlocked device is missing or stolen, the user's sensitive information is completely exposed.

Continuous authentication can complement one-time authentication by continuously authenticating the current user. In this way, after the attacker uses the device for a while, the device can detect the unauthorized user and log out. In [4], the user needs to wear a bracket with a built-in accelerometer, a gyroscope, and a radio. When using a desktop computer(typing the keyboard and using the mouse), the bracket records and sends the movement data to the computer. The computer checks whether the input to the computer matches the data from the bracket. A recent paper [33] points out attacks on the technique in [4]. The technique in [34] continuously authenticates users based on behavioral biometrics with 30 features. The equal error rates drop to 2%-3% with 11 to 12 strokes. Similar techniques based on behavioral biometrics are also presented in [35, 36]. We note that continuous authentication can only detect the attacker after he has used the device for a while. As a result, the attacker still has a good chance to obtain the victim's sensitive data before being logged out. In addition, if the attacker just watches content (e.g. photos and messages) on the screen and does not use the device, he would not be detected by continuous authentication methods at all.

Our method falls into the last category that the device locks itself immediately when the user leaves. If our method is combined with one-time and continuous authentication mechanisms, the attacker can hardly get any opportunity to access the user's sensitive data even if he possesses the missing mobile device. Our work is the first in this category to the best of our knowledge.

iLock is also related to recent work on object tracking and ranging. In particular, FMCW is used in WiTrack [37] for RF-based indoor localization and achieves the positioning accuracy of centimeter. WiTrack 2.0 [38] uses more antennas to support

multi-user localization based on FMCW. Their methods are based on WiFi signals and customized transceivers that are not available on COTS mobile devices. In addition, the techniques in [39, 40] use FMCW with audio signals to track the chest motion and finger movement, respectively. Finally, the work in [41, 19, 42] work on acoustic ranging between devices. iLock differs from these work in the research problem and also system implementation.

2.7 Conclusion

In this chapter, we presented the design and evaluation of iLock, a secure and usable defense against data theft on a lost/stolen mobile device. iLock automatically, quickly, and accurately detects the user's physical separation from his/her device. Once significant physical separation is detected, iLock immediately locks the device to thwart data theft. Relying on acoustic signals, iLock can be deployed on most COTS mobile devices with standard built-in microphones and speakers. Extensive experiments on Samsung Galaxy S5 confirmed the high efficacy of iLock with negligible false positives and negatives.

Chapter 3

SECURE CROWDSOURCED INDOOR POSITIONING SYSTEMS

3.1 Overview

The goal of this chapter is to discover the vulnerabilities of crowdsourced WiFi-based IPSes and present some countermeasures. For this purpose, we prototype a WiFi-based IPS to evaluate potential attacks and the corresponding defenses. Although this study focuses on WiFi-based IPSes, the rationals can easily extend to other types of crowdsourced IPSes. We hope that our study can promote security considerations in the early phase of designing and deploying crowdsourced IPSes.

We identify three attacks based on the information the attackers have. In the first attack, the attackers know the indoor floor plan but have no knowledge about real indoor APs. So they can generate acceptable mobility traces fitting the floor plan, which are submitted along with totally random fake RSS fingerprints. In the second attack, the attackers know both the indoor floor plan and legitimate RSS fingerprints, e.g., by walking in the indoor environment. They add noise to RSS fingerprints before submitting them. In the third attack, the attackers have the same knowledge as in the second attack. But they purposefully change the mappings between the floor plan and RSS fingerprints instead of polluting RSS fingerprints.

We also propose the corresponding defenses based on the observation that there are often some trusted indoor users such as the employees in a shopping mall. Even if the IPS operator cannot produce a high-fidelity fingerprint database based on limited trusted users alone, their data are much more trustworthy and can be used to verify the data from untrusted crowdsourcing workers. We defend against the first attack

by comparing the set of APs in an untrusted submission with those in a trusted submission for the same position. To deal with the second and third attacks, we present two novel metrics to evaluate the trustworthiness of data submissions from untrusted crowdsourcing workers. The first metric considers the correlation between the RSS fingerprints for adjacent positions in the same signal trace, while the second considers the correlation between the RSS fingerprints for the same positions in different signal traces. Finally, we combine the two metrics and design an algorithm to build a high-fidelity fingerprint database resilient to malicious crowdsourcing workers.

Our contributions can be summarized as follows.

- We are the first to study the security issues in crowdsourced WiFi-based IPSes. Our principles can be easily extended to other crowdsourced IPSes.
- We present three attacks and evaluate their performance in a prototype system. We show that the attacker can induce a localization error up to 20m under the most powerful attack.
- We propose the corresponding defenses that can safeguard a crowdsourced WiFi-based IPS from malicious data injections. We experimentally show that our technique is highly resilient to the identified attacks even if the majority of crowdsourcing workers are malicious.

3.2 Basics of Crowdsourced WiFi-Based IPS

In this section, we introduce the basic operations of a WiFi-based IPS to help understand the proposed attacks and defenses. WiFi-based IPSes depend on the ubiquitous WiFi infrastructure in indoor environments and the penetration of WiFi-capable smartphones into people's everyday life. No additional hardware is needed to augment the network infrastructure or equip mobile users. When there is need

for indoor positioning, the user turns on the IPS app on his smartphone. Assume that there are n APs in a given indoor environment, denoted by $\mathrm{AP}_1, \ldots, \mathrm{AP}_n$. At any specific indoor location, the smartphone can detect n RSS values (denoted by $\mathrm{rss}_1, \ldots, \mathrm{rss}_n$), one for each AP. If some APs are not discoverable, the corresponding RSS values are set to default system values. We refer to $\langle \mathrm{rss}_1, \ldots, \mathrm{rss}_n \rangle$ as an RSS fingerprint (or just fingerprint for short) for that position. Assume that the IPS operator has maintained a fingerprint database composed of the mappings between fingerprints and indoor locations. Upon receiving the user's fingerprint, the IPS operator finds the closest match in its fingerprint database and then returns the corresponding location to the user.

Radar [43] is the most classical WiFi-based indoor positioning method. It uses deterministic fingerprinting and matching based on Euclidean distance. To find the closest match in the database for a received fingerprint $\langle \mathrm{rss}_1, \ldots, \mathrm{rss}_n \rangle$, Radar minimizes the distance $\sqrt{(\mathrm{rss}'_1 - \mathrm{rss}_1)^2 + \cdots + (\mathrm{rss}'_n - \mathrm{rss}_n)^2}$ for an arbitrary record $\langle \mathrm{rss}'_1, \ldots, \mathrm{rss}'_n \rangle$ in the database.

Horus [44] improves Radar by employing probabilistic techniques to find a maximum likelihood fingerprint in the database. Given that RSS fingerprints are highly dependent on time, locations, and even devices, Horus maintains the RSS fingerprint distribution at every position x_i as

$$P(\langle \mathrm{rss}_1, \ldots, \mathrm{rss}_n \rangle | x = x_i) = \prod_{k=1}^{n} P(\mathrm{rss}_k | x = x_i) \ .$$

Based on Bayesian inference, Horus derives the maximum likelihood location for each received fingerprint.

Traditional WiFi-based IPSes face two key challenges. First, it is very time-consuming and labor-intensive for the IPS operator to record the RSS fingerprints at every position in such a large indoor environment as a shopping mall. Second,

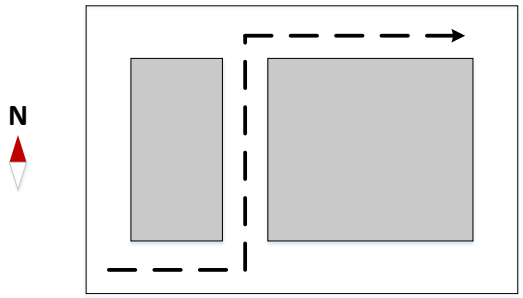


Figure 3.1: Only One Trace Exists in the Floor Plan.

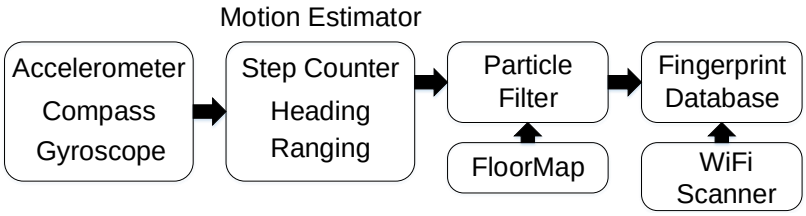


Figure 3.2: The Architecture of the Prototype System.

dynamic calibration is needed to update the fingerprint database to deal with noisy WiFi signals, physical environment changes, and AP additions/deletions.

3.3 A Prototype for Crowdsoured WiFi-Based IPS

Mobile crowdsourcing has great potential to facilitate the construction of fingerprint databases in WiFi-based IPSes. The key idea is to leverage the sensors (e.g., accelerometer, gyroscope, and magnetometer) embedded in today's smartphones in a crowdsourcing manner to automatically construct/update the fingerprint database. In such a system, a crowdsourcing worker submits the RSS fingerprints and corresponding IMU sensor data at unknown locations to the IPS operator. Different techniques such as [8, 9, 10, 11] have been proposed to explore the IMU sensor data to map the fingerprints collected at unknown locations to the correct locations.

We build a prototype system based on Zee [8], a representative crowdsourced WiFi-based IPS, to illustrate the proposed attacks and defenses. Our attacks and

defenses can easily extend to other systems after simple adaptations. The prototype system explores the fact that the indoor layout imposes constraints on the human mobility. For example, the user has to walk along the corridor and cannot walk through the walls or other barriers. When the user's mobility trace is known (e.g., a zigzag path), we may find only one pathway in the floor plan to accommodate the mobility trace. The more the user walks, the higher probability a single possible pathway exists. As illustrated in Fig. 3.1, the user walks east, turns left, turns right, and walks to the end. There is only one pathway which can accommodate such a mobility trace. If the user's mobility trace can be mapped uniquely to the floor plan, the fingerprints collected when the user walks can also be mapped to the floor plan based on timestamps.

Fig. 3.2 illustrates the architecture of our prototype system. The system first estimates the user motion (mobility trace) from accelerometer, compass, and gyroscope data. Then the particle filter explores the mobility trace to eliminate the particles that violate the floor constraints (e.g., it is impossible for the user to walk through the wall). Finally, the system uniquely maps the mobility trace and associated RSS fingerprints to the floor plan. The fingerprint database is built and dynamically updated in this way. In the operating phase, the prototype system uses Horus [44] for fingerprint matching.

We implement our prototype in Google Nexus 6 based on Java. The Google Nexus 6 phone has a Quad-core 2.7 GHz Krait 450 CPU, 3 GB RAM, a 5.96-inch display, and four relevant IMU sensors (magnetometer, compass, accelerometer, and gyroscope). The sampling frequency for IMU sensors and the WiFi module are about 16.7 Hz and 0.67 Hz, respectively. We deploy the prototype on a rectangular 135m-by-35m floor of a university building with the floor plan shown in Fig. 3.3.

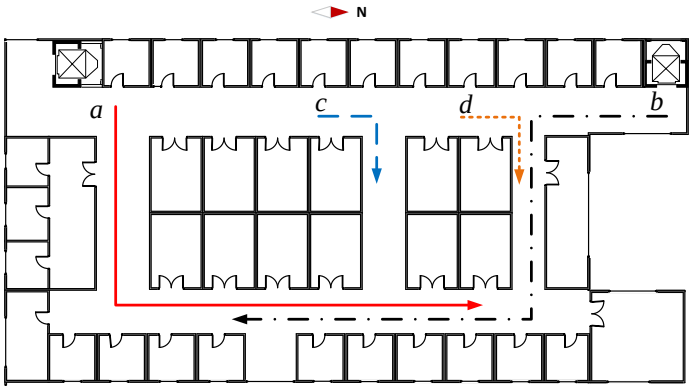


Figure 3.3: Indoor Floor Plan for the Experiment.

To validate the fidelity of our prototype system, we let two trusted users walk 20 times to form the initial fingerprint database with walking traces ranging between 96m to 110m long. Each user walks arbitrarily in the floor plan while carrying the smartphone in any way and orientation. In some cases, users prefer to carry the smartphone in the pocket for convenience. The sensor and RSS data are collected in two weeks to capture the RSS variations due to time. Based on the initial database, we emulate location queries at 11 random positions in the floor plan and get an average error of 2.48m and a median error of 1.72m. These results are quite consistent with the results in Zee [8].

3.4 Adversary Model and Attacks

In this section, we first outline the adversary model. Then we present three attacks and report their efficacy with experimental results in our prototype system.

3.4.1 Adversary Model

Crowdsourced WiFi-based IPSes depend on the voluntary participation of many mobile users that can be recruited in various channels such as Amazon Mechanical Turk. Each crowdsourcing worker submits timestamped RSS fingerprints and con-

current IMU sensor data via an app from the IPS operator, which can be part of the actual IPS app. The IPS operator may be offering indoor positioning services for many indoor venues. In this case, it can easily associate crowdsourced data with the correct indoor venue, e.g., by checking the GPS location of the crowdsourcing worker before he enters the indoor venue. Crowdsourcing workers normally receive some rewards for their participation.

Crowdsourcing workers can misbehave for various reasons. For example, he may submit fake sensor/RSS data to claim rewards without performing the actual WiFi sensing which can be time-consuming or quickly drain his phone battery. Or he can be hired by a malicious competitor to ruin the business of the IPS operator, and such instances are not uncommon in the business world. He may also extort the IPS operator or misbehave just for fun.

The adversary can control many crowdsourcing workers, e.g., by registering many sybil accounts, teaming up with other attackers, or compromising many smartphones via malware. We are aware of the rich literature on sybil defenses which, however, still cannot eliminate fake accounts in practice. We also assume that the adversary knows our defenses.

As the first work on securing crowdsourcing-based IPSes, this chapter does not have the ambition to thwart the attacks other than fake data injection. For example, one may deploy fake APs to interrupt the IPS operations or other common attacks against mobile crowdsourcing systems. These attacks deserve serious treatment in separate work.

3.4.2 Attacks

We first consider the naive attack that the attacker has no knowledge about the indoor floor plan and submits fake IMU sensor and RSS data. The mobility trace

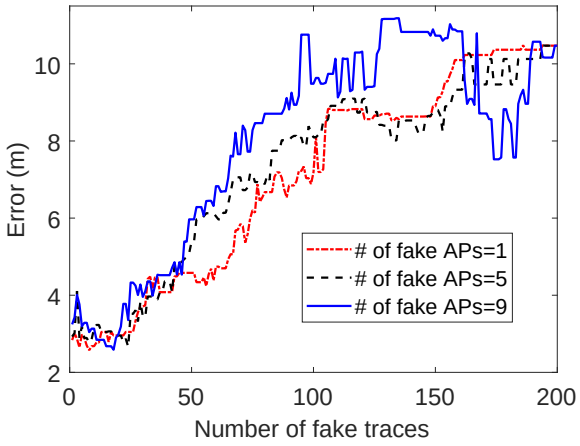


Figure 3.4: Localization Errors Induced by Attack-I.

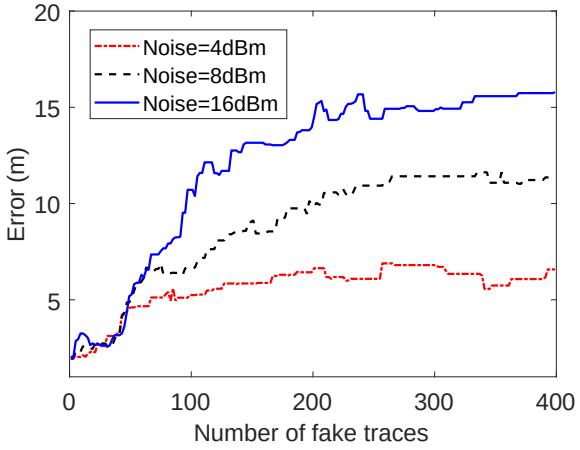


Figure 3.5: Localization Errors Induced by Attack-II with Constant Noise.

generated from fake IMU sensor data can hardly fit the floor plan, so the system can easily detect and reject the fake IMU sensor and RSS data.

Then we consider an attacker who knows the floor plan and thus can generate IMU sensor data resulting in a valid mobility trace fitting the floor plan. So the attacker merely needs to generate fake RSS fingerprints associated with valid mobility traces. Consider the floor plan in Fig. 3.3. We first emulate the attacker to generate a list of mobile traces which range from 40m to 400m long and can fit the floor plan (e.g., traces a and b). The starting position of each trace is random in the floor plan. As

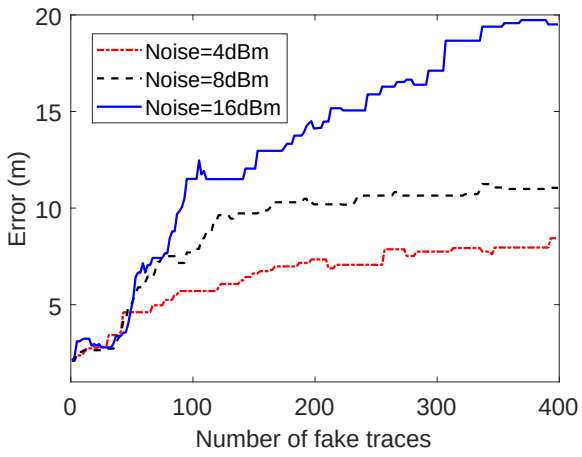


Figure 3.6: Localization Errors Induced by Attack-II with Alternating Noise.

in [8, 9, 10, 11], the traces that have ambiguous fittings in the floor plan (e.g., traces c and d) are not considered. We consider the following three scenarios in which the attacker tries to generate fake RSS fingerprints in different ways.

Attack-I: the attacker does not know genuine RSS fingerprints and generates fake RSS traces purely at random.

In this attack, the attacker knows neither the true RSS fingerprints for any valid mobile trace nor available APs in the indoor venue. So the attacker submits the RSS fingerprints corresponding to fake APs for each valid mobile trace. In a large indoor environment such as the shopping mall, the APs are controlled by different parties who can add, remove, replace, or move their owned APs. The IPS operator can only learn available APs from crowdsourced RSS reports and accordingly adjust the length and format of the RSS fingerprint. Specifically, the IPS operator always maintains an ordered list of APs, appends to the list any new AP learned from crowdsourced RSS fingerprints, and also removes any AP that is not seen in the RSS fingerprints from either crowdsourcing workers or IPS users for a while. For example, when a new AP is discovered, the IPS operator increases the fingerprint length by one by appending

a default value (say, 0) to each existing fingerprint. Therefore, the RSS fingerprints the attacker submits for fake APs will be accepted by the IPS operator and used to update the fingerprint database.

Fig. 3.4 illustrates the localization errors induced by Attack-I. We have about 10 genuine RSS fingerprints for each position in the database before the attack. When fake fingerprints are inserted into the database, the fingerprint distribution changes at the positions covered by fake fingerprints, so the maximum likelihood location derived by the system changes as well (see Section 3.2). Therefore, the localization error changes dramatically with the increase of fake traces. The attacker cannot exactly control the induced errors without knowing the fingerprint database, and he can only cause random changes to the existing fingerprint distribution. As a result, we can see some temporary error fluctuations especially for the traces with more fake APs.

Attack-II: the attacker knows legitimate RSS fingerprints and adds noise to them.

In this attack, the attacker knows legitimate RSS fingerprints, e.g., by visiting the indoor venue in person or getting them from an accomplice. He then submits them after adding noise. Fig. 3.5 shows the localization error when a constant noise (in dBm) is added to each RSS value which ranges from -50 dBm to -90 dBm in our experiments. Again, the localization errors dramatically increase with the number of fake traces and noise strength. The errors stop quickly increasing when there are too many fake traces that start to dominate the fingerprint distribution. The attacker can also add random noise to legitimate RSS fingerprints. Fig. 3.6 shows a simple example, where $+r$ dBm and $-r$ dBm noises are alternately added to adjacent RSS values in a fingerprint. We can clearly see larger localization errors due to larger changes in the fingerprint distribution.

Attack-III: The attacker changes the mappings between fingerprints and indoor locations.

In this attack, the attacker knows genuine RSS fingerprints. Instead of adding noise, the attacker changes the mappings between RSS fingerprints and indoor locations. For example, let $\langle p_1, \ldots, p_\alpha \rangle$ denote the valid mobility trace that can be inferred from the attacker's IMU sensor data, where p_i ($\forall i \in [1, \alpha]$) denotes the ith position. Also assume that the genuine RSS trace corresponding to $\langle p_1, \ldots, p_\alpha \rangle$ is denoted by $\langle f_1, \ldots, f_\alpha \rangle$, where f_i is the RSS fingerprint for position p_i ($\forall i \in [1, \alpha]$). In the simplest case, the attacker maps f_i to position $p_{i+k \mod \alpha}$, where $k \in [1, \alpha - 1]$ denotes an arbitrary offset. So the attacker submits $\langle f_{\alpha-k+1}, \ldots f_\alpha, f_1, \ldots, f_{\alpha-k} \rangle$ along with his IMU sensor data (or equivalently the mobility trace $\langle p_1, \ldots, p_\alpha \rangle$) to the IPS operator. Fig. 3.7 shows that the average localization error under Attack-III increases dramatically with the number of fake traces for three position offsets. An IPS normally has discretized locations with constant distance (2m in our experiments) between adjacent positions. The integer-valued offset k thus has been translated into the corresponding physical distance in Fig. 3.7.

Attack-III is much more powerful than the previous two attacks due to its more organized nature, as it purposefully misleads existing fingerprints towards the offset direction. As we can see in Fig. 3.7, the attacker just needs to inject about 40 fake traces to attain the maximum localization error achievable under Attack 1 or Attack 2. In addition, Fig. 3.8 shows that the localization error increases linearly with the offset for a fixed number of traces, which is quite expected.

Summary of Attacks: The three attacks above are all in their basic forms. The adversary can conceive many variations of each attack or an arbitrary combination of the three attacks to disrupt the IPS operations. There is thus a pressing need

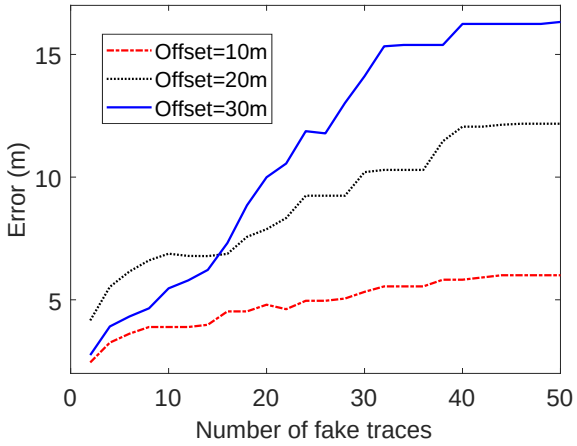


Figure 3.7: Localization Errors Induced by Attack-III.

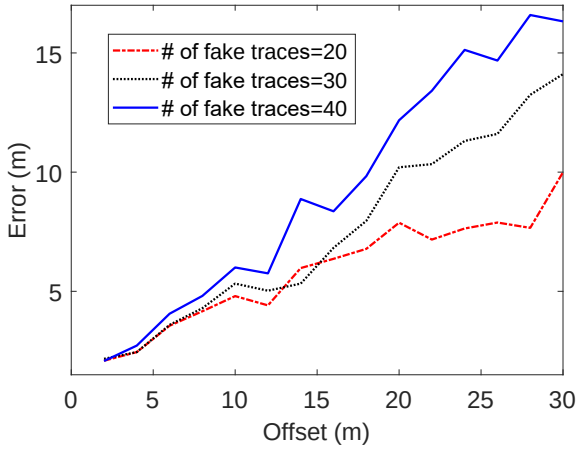


Figure 3.8: Localization Errors Induced by Attack-III.

to develop sound defenses to safeguard crowdsourced WiFi-based IPSes from these attacks.

3.5 Defenses

In this section, we present some effective defenses against the three attacks reported above. Our defenses depend on the observation that there are always some trusted users in many indoor environments (e.g., the employees in a shopping mall) who can act as trustworthy crowdsourcing workers for the IPS operator. Even though

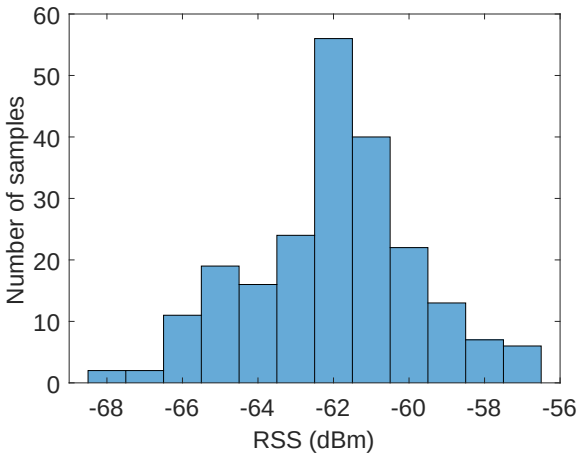


Figure 3.9: Histogram of RSS Values Collected from an AP When the User is Static for About 5 Minutes.

these trusted users are far from enough to build a high-fidelity fingerprint database, their data can be explored to infer the trustworthiness of RSS fingerprints submitted by unknown crowdsourcing workers.

A naive idea is to directly compare the fingerprints from unknown workers with those from trusted users for the same positions. However, wireless signal variations make such direct comparisons unreliable. For example, Fig. 4.2 shows the histogram of RSS values collected from an AP when the user is static for about 5 minutes. The histogram covers a large range of 11 dBm. In the crowdsourcing scenario, the histogram range can be even larger because of user mobility, inaccurate mapping from fingerprints to positions, and so on. An effective defense thus must tolerate RSS variations.

In what follows, we first present a simple method in data preprocessing to defend against Attack-I. Then we present two metrics to evaluate the trustworthiness of RSS traces from crowdsourcing workers. Finally, we present an iterative algorithm based on the two metrics to build a high-fidelity fingerprint database even in the presence of attacks. Throughout the discussion, we consider a candidate RSS trace $\langle f_1, \ldots, f_n \rangle$,

where $f_i = (x_{i1}, x_{i2}, ..., x_{im})$ is the fingerprint at position i ($\forall i \in [1, n]$), and x_{ij} is the RSS value from AP_j ($\forall j \in [1, m]$) collected at position i.

3.5.1 Preprocessing against Attack-I

The AP set sensed by different users in the same position should not differ too much in a short time window, or most IPSes would not work. To defend against Attack-I, we compare the APs detected by an unknown worker and a trusted user in the same position. For example, consider a worker who submitted an RSS trace $\langle f_1, \ldots, f_n \rangle$ for n positions. Let A_i and $A_i^{'}$ denote the APs detected by the trusted user and the worker at position i, respectively, for which the detection-time difference is smaller than a system threshold (say, 24 hours). The IPS operator computes the average intersection ratio

$$\delta = \frac{1}{n} \sum_{i=1}^{n} \frac{|A_i \bigcap A_i^{'}|}{|A_i|} \ .$$

If δ is larger than a system threshold, the trace $\langle f_1, \ldots, f_n \rangle$ is temporarily considered trustworthy for further processing.

3.5.2 Metric 1: Temporal Correlation within an RSS Trace

We also observe that fingerprints collected by different users tend to exhibit a similar RSS trend. For example, when the user walks towards an AP, the RSS increases gradually; when the user walks away from the AP, the RSS decreases gradually. Fig. 3.10 exemplifies this observation with the RSS trends collected by five different users when passing by the same AP in our prototype system. A fake trace with totally random RSS fingerprints will not be consistent with genuine traces related to the same AP. In other words, the attacker will be forced to generate fake traces with RSS trends similar to those of genuine traces.

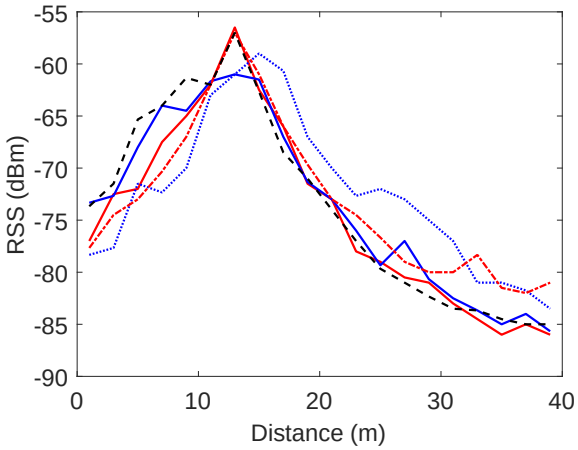


Figure 3.10: RSS Values Collected by Five Users When Passing the Same AP.

Based on this observation, we design a temporal likelihood metric to evaluate the temporal correlation between trusted traces and crowdsourced traces. Consider a candidate RSS trace $\langle f_1, \ldots, f_n \rangle$, where $f_i = (x_{i1}, x_{i2}, ..., x_{im})$. When deriving the likelihood of observing the whole trace, we should take into account the temporal correlations among adjacent RSS fingerprints in the trace. For example, when x_{ij} is known, the range of $x_{(i+1)j}$ is largely determined because of the RSS trends illustrated in Fig. 3.10.

As a result, we can calculate the likelihood of observing $x_{(i+1)j}$ from AP_j as

$$\mathcal{L}(x_{(i+1)j}) = \mathcal{L}(x_{(i+1)j}|x_{ij})\mathcal{L}(x_{ij}) \ ,$$

where $\mathcal{L}(x_{(i+1)j}|x_{ij})$ is the likelihood of observing $x_{(i+1)j}$ given that x_{ij} is observed from AP_j. We can extract a distribution for adjacent RSS variations from trusted traces and then easily compute $\mathcal{L}(x_{(i+1)j}|x_{ij})$. It is also fairly easy to calculate $\mathcal{L}(x_{1j})$ based on the distribution extracted from the RSS values in trusted traces. So we can compute the likelihood for observing a sequence of RSS values $(x_{1j}, x_{2j}, \ldots, x_{nj})$ from AP_j as

$$\mathcal{L}(x_{1j}, x_{2j}, \ldots, x_{nj}) = \mathcal{L}(x_{1j})\mathcal{L}(x_{2j}|x_{1j}) \ldots \mathcal{L}(x_{nj}|x_{(n-1)j}) \ .$$

The temporal likelihood for the whole trace can then be computed as

$$\mathcal{L}_{\text{temporal}}(\langle f_1, \ldots, f_n \rangle) = \prod_{j=1}^{m} \mathcal{L}(x_{1j}, x_{2j}, \ldots, x_{nj}),$$

which is normalized as $\frac{1}{n} \sum_{j=1}^{m} \log \mathcal{L}(x_{1j}, x_{2j}, \ldots, x_{nj})$.

3.5.3 Metric 2: Spatial Correlation with Other Traces

Although the RSS data collected by different users may differ because of many reasons such as channel variations, phone orientation, and phone model, they generally follow a Gaussian Distribution which is exemplified in Fig. 4.2. We can use the distribution formed by trusted users to infer the likelihood of the RSS submitted by a crowdsourcing worker. The second metric (called *spatial likelihood*) is designed to capture the spatial RSS correlation between the fingerprints from the same position in different traces. According to [44], the RSS values from different APs collected in the same position are independent from each other. So we can estimate the likelihood of the fingerprint in position i as

$$\mathcal{L}(f_i) = \mathcal{L}(x_{i1}, x_{i2}, ..., x_{im}) = \prod_{j=1}^{m} \mathcal{L}(x_{ij}),$$

where $\mathcal{L}(x_{ij})$ refers to the likelihood of observing the RSS value x_{ij} from AP_j at position i. The spatial likelihood of the trace $\langle f_1, \ldots, f_n \rangle$ is represented as the product of likelihood in every position as

$$\mathcal{L}_{\text{spatial}}(\langle f_1, \ldots, f_n \rangle) = \prod_{i=1}^{n} \mathcal{L}(f_i) = \prod_{i=1}^{n} \prod_{j=1}^{m} \mathcal{L}(x_{ij}).$$

After normalization for different trace lengths, we can rewrite

$$\mathcal{L}_{\text{spatial}}(\langle f_1, \ldots, f_n \rangle) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} \log \mathcal{L}(x_{ij}) \ .$$

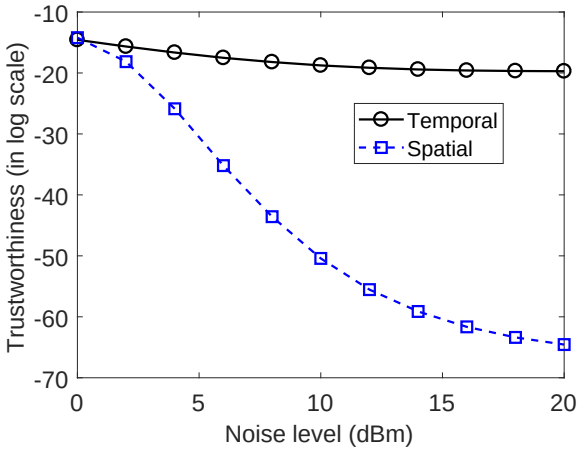


Figure 3.11: Temporal and Spatial Trustworthiness of Fake Traces under Attack-II with Equal Noise.

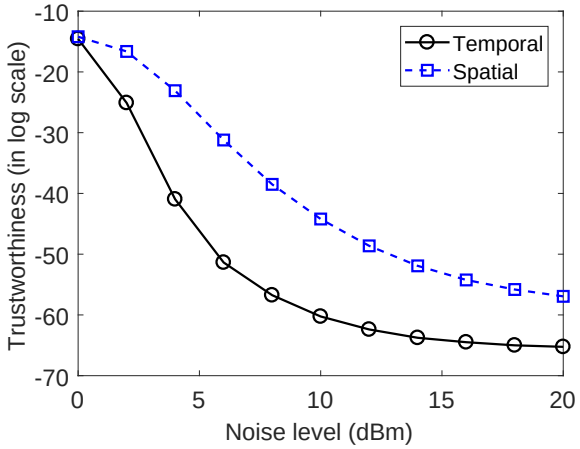


Figure 3.12: Temporal and Spatial Trustworthiness of Fake Traces under Attack-II with Alternating Noise.

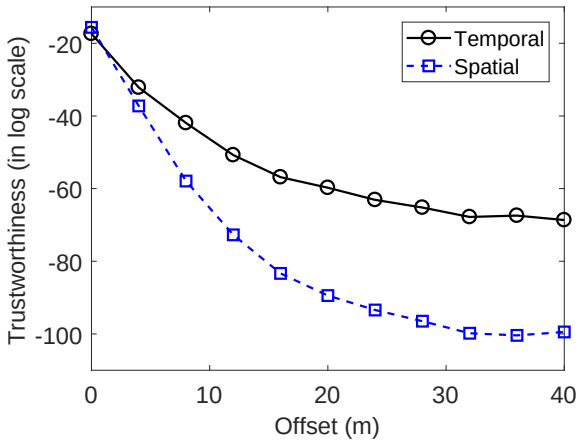


Figure 3.13: Temporal and Spatial Trustworthiness of Fake Traces under Attack-III vs. Offset

3.5.4 Iterative Fingerprint-Database Construction

It is well known that the RSS fingerprint database in a WiFi-based IPS needs to be periodically calibrated to deal with wireless channel variations, indoor layout changes, AP changes, and many dynamic factors in a large, complex indoor environment [8, 9]. So we present an iterative algorithm to build and maintain the RSS fingerprint database. In each updating interval (say, daily or weekly or biweekly), the IPS server always accepts new RSS traces from the trusted users first and then uses them to evaluate the trustworthiness of the RSS traces from crowdsourcing workers.

Our algorithm treats each crowdsourcing worker with equal suspicion in each updating interval. Specifically, a crowdsourcing worker may exhibit dynamic behavior by alternating between "good" and "bad" states. Reputation systems are traditional defenses against such dynamic behavior, but there are also well-documented attacks on reputation systems. For lack of space, we leave the integration of a sound reputation system in our algorithm to future work. Our algorithm applies equally to

each received RSS trace without considering the past behavior of the crowdsourcing worker.

Our algorithm is designed to work even if the majority of crowdsourcing workers in a given updating interval are malicious. In particular, we rank each crowdsourced RSS trace with the two metrics above, and higher ranks indicate more trustworthiness. Only the traces with sufficient trustworthiness are added and used to update the database.

Algorithm 1 summarizes the main steps the IPS server takes in each updating interval. The IPS server first adds the traces from trusted users in the current updating interval and checks if these new trusted traces indicate any major change in the indoor environment. Specifically, the IPS server updates the fingerprint distribution for each AP at every position and compares it with the previous distribution. If the mean of any fingerprint distribution changes more than ε_1 or its variance changes more than ε_2, we consider that the distribution has significantly changed since last updating interval, where ε_1 and ε_2 are two system parameters. If none of the fingerprint distribution has changed, then there is no need to add additional unknown traces.

If the IPS server determines that there has been any significant change to any fingerprint distribution from the last updating interval, it selects more trustworthy traces to add to the database. The traces which cannot fit the floor plan will be discarded before the likelihood (trustworthiness) evaluation. Then the IPS server discards the traces subject to Attack I by checking whether the traces contain sufficiently common APs to those of trusted traces. Next, the IPS server evaluates the trustworthiness of each remaining trace by combining its temporal and spatial likelihoods in a weighted fashion. Finally, the IPS server discards all the traces with combined trustworthiness (i.e., l_U) lower than η and uses the remaining top-K trustworthy traces to update

the database. The impacts of system parameters such as η and K are evaluated in Section 4.6.

Alternatively, the IPS operator can iteratively integrate the remaining traces into the database in the descending order of their trustworthiness until the database quality is sufficient or all the traces are used up. We ignore this option in this chapter for lack of space.

3.6 Countermeasure Evaluation

In this section, we report the experimental performance of our countermeasures in the prototype system. In our experiments, no fake trace generated under Attack-I passes the AP-correlation test in trace preprocessing. So we focus on the resilience of our countermeasures against Attack-II and Attack-III in this section. The floor plan for all the experiments remains the same as Fig. 3.3.

3.6.1 Evaluation of Metrics

In this subsection, we evaluate the two metrics on Attack-II and Attack-III, respectively. The evaluation is based on the database we build in Section 3.3 which contains 20 walking traces from trusted users. The attacker walks in the floor plan for 200m and then generates all kinds of fake traces from the learned legitimate trace.

Resilience to Attack-II. We first evaluate the performance of the temporal and spatial trustworthiness metrics under Attack-II. Fig. 3.11 shows the temporal and spatial trustworthiness varying with the amount of the noise added to each RSS value. As we can see, the temporal trustworthiness is relatively insensitive to the change in the amount of the noise added. This is because adding equal amount of noise to every RSS does not change the RSS trend across adjacent fingerprints for the same APs. For example, if a sequence of RSS values for the same AP in a genuine

trace exhibit an ascending trend, we can still observe the same trend after the same amount of noise is added to each RSS value. In contrast, the spatial trustworthiness decreases with the increase in the noise added, as larger noise induces larger deviations of the fake fingerprints from genuine ones, and vice versa.

Fig. 3.12 shows the temporal and spatial trustworthiness when the attacker adds $+r$ dBm and $-r$ dBm noise alternately to adjacent RSS values in a fingerprint. As we can see, both temporal and spatial trustworthiness decrease as the amount of noise added increases, which is expected. In addition, the temporal trustworthiness decreases more rapidly than spatial trustworthiness as the amount of noise increases because the fake trace exhibits a very different temporal pattern (trend) from trusted traces considering the alternating noise added to adjacent RSS values.

Resilience to Attack-III. Fig. 3.13 compares the temporal and spatial trustworthiness of fake traces generated under Attack-III, where the attacker introduces different position offsets. As we can see, both metrics can provide good discrimination between fake and legitimate traces. Attack III induces dramatic differences between the fake and legitimate traces for the same positions. In contrast, the temporal trend in a legitimate trace can still be preserved to some extent in a fake trace. So we can see that the spatial trustworthiness metric outperforms the temporal one.

3.6.2 Evaluation of Fingerprint-Database Updating Algorithm

We now evaluate our fingerprint-database updating algorithm when both temporal and spatial trustworthiness metrics are employed under the following settings. In the initial phase, the database only contains four traces submitted by trusted users. The IPS server receives two legitimate traces and 40 fake traces in each updating period, as well as one trusted trace every two updating periods. The algorithm is executed in each updating period, in which the IPS server relies on the trusted traces to evaluate

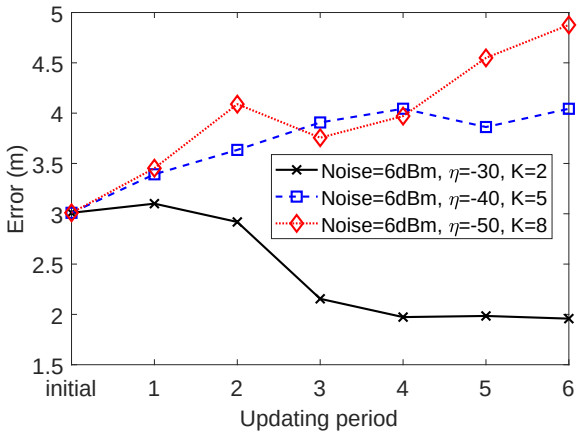


Figure 3.14: Average Localization Error under Attack-II with 6 dBm Equal Noise.

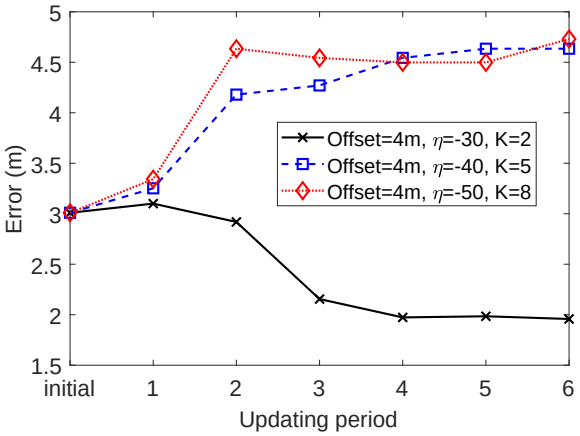


Figure 3.15: Average Localization Error under Attack-III with 4m Offset.

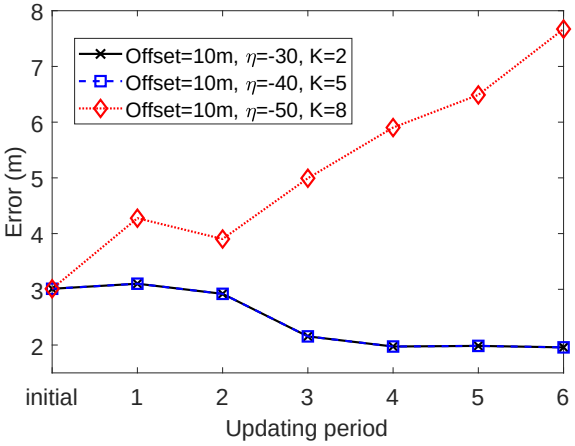


Figure 3.16: Average Localization Error under Attack-III with 10m Offset.

the trustworthiness of each unknown trace to distinguish legitimate traces from fake ones. Experiments for all parameter scenarios below use the same list of (trusted, legitimate, and fake) walking traces which we generate in the same way we talked in Section 3.3.

In our experiments, we set $\theta = 0.7$, corresponding to the assumption that the APs detected by legitimate users cannot differ by more than 70% in a short time window. We also set $\alpha = 0.4$, indicating the better overall performance of spatial trustworthiness over temporal trustworthiness. In addition, we set $\eta = -30$ (log scale), because the trustworthiness (likelihood) of over 90% legitimate traces is over -30. These parameters (θ, α, and η) can be learned in practice through machine learning. The IPS server only accepts the top-K trustworthy RSS traces, where K can be estimated based on the number of trusted users and indoor traffic volume. Larger K can accelerate the convergence of database construction but also increase the vulnerability to fake traces, and vice versa.

Fig. 3.14 shows how the average localization error changes under Attack-II when an equal noise of 6 dBm is added to every RSS value. Our algorithm updates the database based on the top-K trustworthy traces, each with a trustworthy value $\geq \eta$. In our experiments, the two legitimate traces always have higher trustworthiness than the fake traces and are always in the top-K list for different K. The remaining $K-2$ traces in the top-K list are filled with fake traces only when there are fake traces with an trustworthy value higher than η. In general, the smaller K is, the fewer fake traces ($K-2$) added to the database, the lower the localization error, and vice versa. This trend is clearly seen in Fig. 3.14. When we enforce a strict criterion ($\eta = -30$ and $K = 2$), no fake trace can achieve trustworthiness equal to or over η, so only the two legitimate traces are accepted; the localization error keeps decreasing as more legitimate and trusted traces are added to the database in each updating interval

and then becomes stable around the performance limit of the IPS. If we relax the criterion (e.g., $\eta = -50$ and $K = 8$), more and more fake traces are accepted, leading to increasing localization errors as time goes by.

Fig. 3.15 shows the average distance error under Attack-III when the attacker adds an offset of 4m to the traces. When the most restricted parameters ($\eta = -30$ and $K = 2$) are used, no fake trace is accepted in any updating interval. The average distance error keeps decreasing as more legitimate and trusted traces are included until reaching the system limit. When we relax the criteria by using smaller η or larger K, more fake traces are inserted into the database, leading to the increase in the distance errors before it becomes stable around the 4m.

Fig. 3.16 shows the average distance error under Attack-III when the attacker increases the offset to 10m. Under such larger offset, none of the fake traces can achieve a trustworthiness value greater than -40, so all fake traces are rejected by the IPS operator. Therefore, we can see the lines corresponding to ($\eta = -40, K = 5$) and ($\eta = -30, K = 2$) overlap with each other. On the other hand, if we decrease η to -50 and increase K to 8, some fake traces will be accepted by the database, leading to gradual increase in distance errors.

3.7 Related Work

This section discusses some most germane work.

Fingerprint-based indoor positioning techniques are the most popular approaches for indoor positioning. As probably the first work along this line, Radar [43] is a deterministic localization method that employs RSS for indoor localization. Horus [44] improves Radar by keeping a fingerprint distribution for every position in the floor plan and then finding a maximum likelihood match in the database. The work [45] introduces Channel Frequency Response as a new feature for localization. Surround-

Sense [46] introduces more indoor features (such as light and sound) in addition to RSS fingerprints. All these methods rely on labor-intensive calibration where the IPS operator has to collect and update fingerprints for every position in the floor plan.

Model-based indoor positioning techniques estimate indoor locations using statistic models. A popular approach is to build a relation between RSS and signal propagation distance based on the RF propagation model (e.g., the log-distance path loss (LDPL)) [47, 48]. Model-based methods can dramatically decrease the need for RSS measurements but at the cost of accuracy. For example, the work in [49] evaluates some self-calibrating algorithms in office environments and finds that the median errors are consistently greater than 5m. In addition to LDPL-based schemes, there are other techniques based on Angle of Arrival (AoA) [50, 51], Time of Arrival (ToA) [52], and Time Difference of Arrival (TDoA) [53].

More recently, researchers start to explore visible light for indoor positioning. Most such techniques [54, 55, 56, 57, 58] rely on customized smart LEDs which send identification beacons for localization. Although some techniques can achieve submeter precision [58], it incurs significant cost to retrofit current illuminating systems. LiTell [59] first enables visible light localization on unmodified existing light hardware, but the method only applies to tube lights and the camera of smartphone must be held flat.

Simultaneous Localization and Mapping (SLAM) is a technique originating from the robotics community. SLAM relies on a robot to explore the space of interest with discrete landmarks or obstacles. Based on the laser ranging and cameras in the robot, we can determine the relative locations of the landmarks, and the robot can infer its relative location. WiFi-SLAM [60] uses a Gaussian process to model the relation of WiFi signal strengths. With more sensors embedded in the smartphone, many techniques combine IMU sensor data with human movements to realize SLAM. For

example, Unloc [9] uses smartphones to sense the natural landmarks in the floor plan such as elevators and stairs, which are then connected via dead reckoning. Similar to our prototype system, Zee [8] uses the indoor constraints to map crowdsourced human mobility traces to the floor plan and then generates the fingerprint database. LiFS [10] maps fingerprints by comparing the similarity between the high-dimensional fingerprint space and the stress-free floor plan. Walkie-markie [11] presents a method for generating the floor plan based on the RSS trend when the user passes the AP.

There are also some studies on the false data injection attack in other crowdsourcing systems. For example, Zhang *et al.* [61] studies false spectrum report injection attack in crowdsourcing-based spectrum sensing. More recently, the work in [62] introduces a mechanism that explores user proximity to detect false data submitted by sybil users in crowdsourced map systems. These techniques do not consider the unique features of RSS-fingerprint-based IPSes and are not applicable to the attacks identified in this chapter.

3.8 Conclusion

In this chapter, we presented the first systematic study about the security issues in crowdsourced WiFi-based IPSes. We presented three attacks and evaluated their performance in a prototype system. We also designed an algorithm based on novel temporal and spatial trustworthiness metrics to generate high-fidelity fingerprint databases even if most crowdsourced RSS traces are fake. Thorough experiments confirmed that our algorithm has strong resilience to the reported attacks. Both the attacks and defenses developed in this chapter can be easily extended to other crowdsourced IPSes.

Algorithm 1: Iterative Fingerprint-Database Construction

input : Fingerprint database $\mathcal{F}$, traces $\mathcal{T}$ submitted by trusted users, traces $\mathcal{U}$ submitted by crowdsourcing workers.

output: Updated fingerprint database $\mathcal{F}$.

Fit all the traces in $\mathcal{T}$ to the floor plan and add the corresponding fingerprints to $\mathcal{F}$;

if *No fingerprint distribution has major change* **then**

  return $\mathcal{F}$;

Calculate RSS distribution $\mathcal{N}_i$ and the set A_i of APs for every position i in the floor plan;

foreach *trace* U *in* $\mathcal{U}$ **do**

  if U *does not fit the floor plan* **then**

    $\mathcal{U} \longleftarrow \mathcal{U} \setminus \{U\}$;

  else

    Calculate $A_i^{'}$ for every position i in trace U;

    $r \longleftarrow \frac{1}{n} \sum_{i=1}^{n} \frac{|A_i \bigcap A_i^{'}|}{|A_i|}$;

    $l_U \longleftarrow \alpha \mathcal{L}_{\text{temporal}}(U) + (1-\alpha)\mathcal{L}_{\text{spatial}}(U)$;

    if $r < \theta$ *or* $l_U < \eta$ **then**

      $\mathcal{U} \longleftarrow \mathcal{U} \setminus \{U\}$;

Rank all the traces in $\mathcal{U}$ according to l_U;

Add the K most trustworthy traces to $\mathcal{F}$;

return $\mathcal{F}$;

Chapter 4

A CROWDSOURCING-BASED CONTEXT-AWARE INDOOR NAVIGATION SYSTEM

4.1 Overview

In this chapter, we present IndoorWaze, a crowdsourcing-based usable indoor navigation system. We use the shopping-mall example throughout this chapter for convenience, but IndoorWaze can easily apply to any large complex indoor environment where a usable indoor navigation service is needed. Core to IndoorWave is a novel crowdsourcing-based technique to automatically construct an accurate indoor floor plan with labeled stores. Our technique is motivated by the observation that shoppers often walk around in the shopping mall, while store employees mostly stay in their respective stores. The shoppers help geometrically connect the stores they pass by. When a shopper passes a particular store, the Wi-Fi fingerprints he senses there can be very similar to those measured by the store employees. In addition, the employees of each store can and are motivated to provide a store label. We can then construct a high-fidelity context-aware indoor floor plan by correlating the shoppers' Wi-Fi fingerprints with Wi-Fi fingerprints and store labels offered by store employees.

IndoorWaze is a lightweight crowdsourcing system that involves little effort from participating shoppers and store employees. The tasks of shoppers should be simple enough, as otherwise they may lack incentives to participate. In our system, the only thing shoppers need to do is to allow access to the IMU and RSS data on their smartphones. All the data are collected implicitly while they shop in their usual way. They do not need to take photos [63] or check in manually [64] in the stores.

Note that such low requirements on crowdsourcing workers have been proved quite feasible and effective in Waze, Google Map, and other crowdsourcing-based traffic and navigation apps. In contrast, store employees can do slightly more work because they want shoppers to more easily find their stores. In IndoorWaze, store employees are required to collect fingerprints at a few locations near the store according to the store dimension. This one-time work can be easily done within a few minutes, which is quite acceptable.

We make the following contributions. First, we present the first crowdsourcing-based indoor navigation system that can automatically generate a context-aware indoor floor plan. Our system infers the shoppers' walking traces from the IMU sensors on their smartphones and then geometrically connects the stores by mapping them to the walking traces. By combining the walking traces from different shoppers, we can get a high-fidelity floor plan which accurately delineates the labeled stores, pathways, and turning positions. Second, we develop techniques to conquer RSS signal fluctuations which may cause large errors when inferring relative store positions. We also present techniques to extract useful walking traces from the complex data submitted by crowdsourcing shoppers. Third, we implement the system on Android smartphones and evaluate it in a large shopping mall. Our system can generate a high-fidelity labeled floor plan, in which all the stores are correctly labeled and arranged, all the pathways and crossings are correctly shown, and the median estimation error for the store dimension is below 12%.

4.2 System Model and Architecture

4.2.1 System Model

There are three entities in the IndoorWaze system: the IndoorWaze service provider, store employees, and shoppers. The service provider releases an app that store employees and shoppers can download and install to their smartphones. The shoppers and employees then register in the app and are required to allow access to the IMU sensors (accelerometer, compass, and gyroscope) in the smartphones. In addition, store employees need to provide their store names in the registration. Store employees are responsible for collecting Wi-Fi RSS fingerprints at a few locations near their store entrances and submit them to the service provider. The sampling locations are picked by the employees themselves and can be a few meters apart from each other. The larger the store, the more sampling locations needed. RSS data collection is a one-time task for employees and takes only a few minutes. The fingerprints provided by the employees of each store act as labeled samples for the store. After that, the employees work as normal. In contrast, the shoppers just walk and shop as usual with their smartphones and do not need to do anything else.

We do not consider security, privacy, and incentive problems which are associated with any crowdsourcing-based system and deserve to be explored in separate papers [65, 66, 62, 67].

4.2.2 System Architecture

Fig. 4.1 illustrates the architecture of our system. IndoorWaze first constructs the shoppers' walking traces based on their IMU sensors. It then extracts the walking traces which are useful for later floor plan construction. Next, we compare the RSS fingerprints from shoppers and store employees to infer the stores shoppers passed

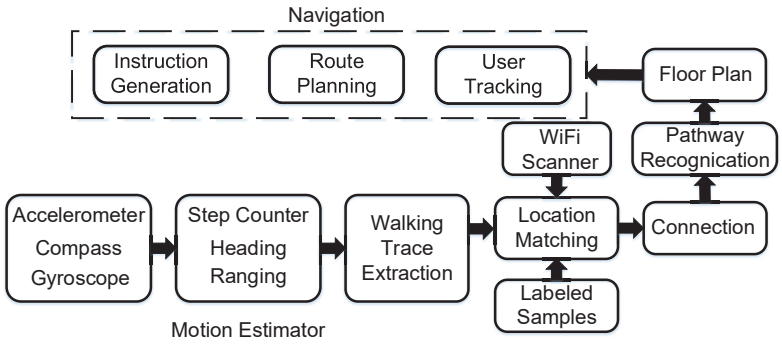


Figure 4.1: IndoorWaze System Architecture.

in the walking traces. The system then connects and combines walking traces from different shoppers to form a labeled floor plan. IndoorWaze also recognizes pathways in the floor plan and connects stores along the two sides of each pathway to improve the connectivity of the graph representing the floor plan. When we combine all the walking traces to form a labeled floor plan, the RSS fingerprints are mapped to the floor plan to form a fingerprint map as well. After a labeled fingerprint map is constructed, IndoorWaze starts to accept navigation requests from users. The user inputs the target store name and submits his current RSS fingerprint readings. IndoorWaze calculates the user's realtime position using his RSS data and then finds a path to his destination. At last, the system gives audio instructions along the way to the user's destination store according to his changing RSS fingerprints and thus locations.

In Section 4.3, we assume that the shoppers only walk along one side of a pathway for simplicity of descriptions. In Section 4.4.4, we relax this assumption and present methods to deal with more complex walking traces.

4.3 Constructing a Rough Floor Plan

In this section, we introduce how to construct a rough floor plan using the RSS fingerprint samples collected by store employees and shoppers.

We assume that there are n stores in the shopping mall. According to the dimension of each store, its employees pick a few locations along the exterior perimeter of the store to collect RSS fingerprint samples. Let $\langle S_1, S_2, \cdots, S_{n'} \rangle$ denote the n' ($n' \geq n$) sampling positions on the floor plan. At each sampling position, the employee of each store uses the smartphone to collect a set of RSS fingerprints. Each RSS fingerprint is represented by $(rss_1, rss_2, \ldots, rss_m)$, where each rss_i is the received signal strength (RSS) for the ith Wi-Fi access point (AP) for all $1 \leq i \leq m$, and m is the number of APs in the environment.

The RSS value at each sampling position typically exhibits fluctuation. For example, Fig. 4.2 shows the histogram of RSS values for a single AP at the same location during a period of five minutes. Although the RSS values fluctuates over a large range of 11 dBm, they generally follow a Gaussian distribution. As a result, we represent the RSS value at each sampling position S_j for each AP_i using a Gaussian distribution $\mathcal{G}(\mu_{i,j}, \sigma_{i,j}^2)$ fitted from the employee's RSS fingerprint samples, where $\mu_{i,j}$ and $\sigma_{i,j}$ are the mean and standard deviation, respectively. Let $x_1, ..., x_s$ are the RSS samples measured at sampling positon S_j from AP_i by the employee. The mean and standard deviation of the Gaussian distribution are computed as

$$\begin{aligned} \mu_{i,j} =& \frac{1}{s} \sum_{k=1}^{s} x_k, \\ \sigma_{i,j} =& \sqrt{\frac{1}{s-1} \sum_{k=1}^{s} (x_k - \mu_{i,j})^2}. \end{aligned} \tag{4.1}$$

We can then estimate the store passed by a shopper using the maximum likelihood estimation. Specifically, for an RSS fingerprint $f_u = (rss_{u,1}, \ldots, rss_{u,m})$ submitted by a shopper, the likelihood of RSS value $rss_{u,i}$ generated by the Gaussian distribution $\mathcal{G}(\mu_{i,j}, \sigma_{i,j}^2)$ is given by

$$\mathcal{L}_{i,j}(rss_{u,i}) = \frac{1}{\sqrt{2\pi\sigma_{i,j}^2}} e^{-\frac{(rss_{u,i}-\mu_{i,j})^2}{2\sigma_{i,j}^2}}, \tag{4.2}$$

where we set $rss_{u,i}$ to -100 dbm if the user does not detect any Wi-Fi signal from AP_i for any $1 \leq i \leq m$ [68]. Considering all m RSS values, the likelihood of RSS fingerprint f_u being measured at sampling position S_j is then given by

$$\mathcal{L}_j(f_u) = \prod_{i=1}^{m} \mathcal{L}_{i,j}(rss_{u,i}), \tag{4.3}$$

where $\mathcal{L}_{i,j}(rss_{u,i})$ is given in Eq. (4.2). The sampling position passed by the shopper is then estimated as

$$j^* = \arg \max_{j \in \{1,\ldots,m\}} \mathcal{L}_j(f_u). \tag{4.4}$$

For an RSS fingerprint trace $\langle f_1, \ldots, f_l \rangle$ submitted by a shopper when walking in the mall, we can infer the shopper's location in realtime based on each RSS fingerprint along the trace.

Fig. 4.3 illustrates a simple floor plan which has two rows of stores and a pathway in the middle. Each store has a sampling position (red dot). The shopper walks from store 1 to store 5 along the upper side of the pathway. The server calculates the shopper's realtime location by comparing the RSS fingerprints from the shoppers and employees based on likelyhood. Fig. 4.4 shows a sequence of inferred locations when the shopper walks, where each location is represented by an inferred store index. Normally, when the shopper passes a store, the system ought to pick the store as his current location. Due to signal fluctuations, the system may nevertheless associate

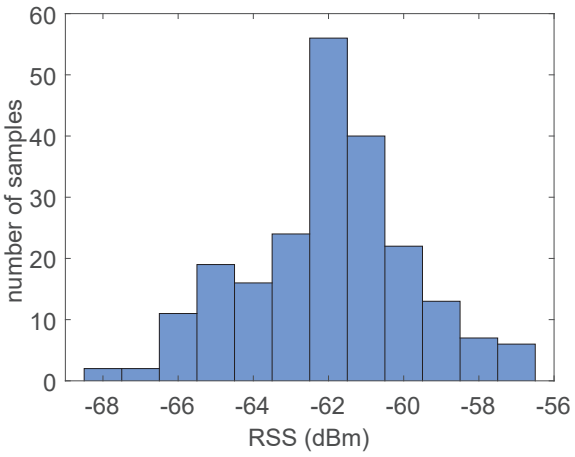


Figure 4.2: Histogram of RSS Values Collected from an AP When the User is Static for About 5 Minutes.

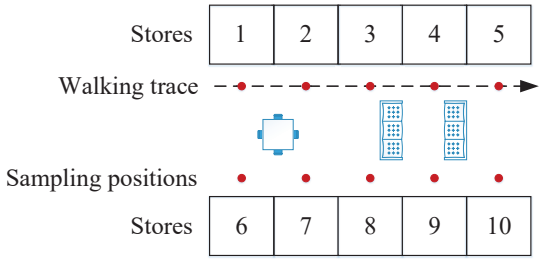


Figure 4.3: An Exemplary Floor Plan.

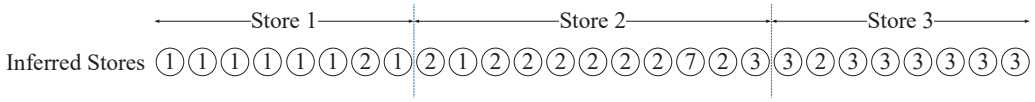


Figure 4.4: The Inferred Stores of Each Fingerprint Sample When the User Walks from Store 1 to Store 3.

Figure 4.5: A Simple Graph Corresponding to the Walking Trace in Fig. 4.3

the shopper's position with an adjacent store especially when the shopper is in the middle of two stores. The system may also pick the stores on the opposite side of the pathway by mistake. Our remedy comes from the observation that it takes a normal shopper a few seconds to pass a store. By contrast, the mistakes caused by instant signal fluctuations do not last long. For example, assume that w is the store dimension, v is the walking speed, and s is the RSS signal sampling rate. Then the shopper can get $\frac{w}{v} \cdot s$ RSS samples when passing the store. Common shoppers walk slowly in the shopping mall with speed v less than 1 m/s, and most stores are wider than 4 meters. If the RSS sampling rate is 1 Hz as in our prototype, the shopper can at least collect 4 RSS fingerprints which are most similar to the fingerprints collected by the store employees. So we search in Fig. 4.4 for the stores that appear continuously for at least four times with a sliding window. Finally, we can get a simple graph like Fig. 4.5.

It is possible that a store may get a wrong adjacent store from a single walking trace for many reasons. For example, the shopper walks so fast that the system cannot capture enough fingerprint samples especially when he passes a small store. In addition, a crowd of people in the mall may cause long-time signal fluctuations which lead to large localization errors. We solve these problems by considering the walking traces from different shoppers. As long as the results from most shoppers are correct, the overall system performance is satisfactory.

4.4 Constructing an Accurate Floor Plan

The rough graph generated in the above section only gives the label of each store and the relative positions between adjacent stores. To find the target store with the rough labeled floor plan, the shopper has to look for stores one by one, which is still not user-friendly enough. To make the navigation system more usable, we should give

more instructions as described in Section 2.1. As a result, we need a more accurate labeled floor plan which contains the shape and dimension of each store as well as the pathway information. In this section, we combine IMU sensors and the walking traces submitted by multiple shoppers to achieve the above ambitious goal.

4.4.1 Dealing with a Single Walking Trace

With the help of IMU sensors (accelerometer, gyroscope, and compass), we can recover the shopper's walking trace. In particular, the accelerometer can capture the motion caused by the shopper's walking. After recognizing each step, we can count the steps to infer the shopper's walking distance. The walking direction can be extracted from the electronic compass, but the magnetic signals needed are not stable due to indoor magnetic interference. In contrast, the gyroscope can provide an accurate short-term angle estimation which is free from indoor signal interference, but it can introduce large cumulative errors in the long term. Our system thus combines gyroscope and compass to get a more accurate and stable direction estimation.

Fig. 4.6 illustrates the walking trace submitted by a shopper, which corresponds to part of the floor plan in a large shopping mall in Fig. 4.7. The shopper starts from store "PANDORA", walks around, and stops at store "NOBILITEA". The server compares the fingerprints submitted by the shopper with the data from the employees of each store using the method in Section 4.3. Fig. 4.6 shows that we can correctly infer each store the shopper passed in the walking trace. In addition, we can recover the shape of the original floor plan, so we are able to give more accurate instructions like "Turn right at store APEX". By counting the steps of the shopper when passing the stores, the system can know the dimension of each store as well, which is particularly useful for visually impaired users.

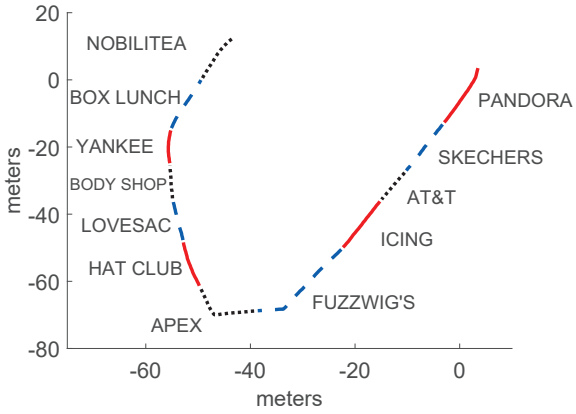


Figure 4.6: A Walking Trace with a Sequence of Stores the Shopper Passed.

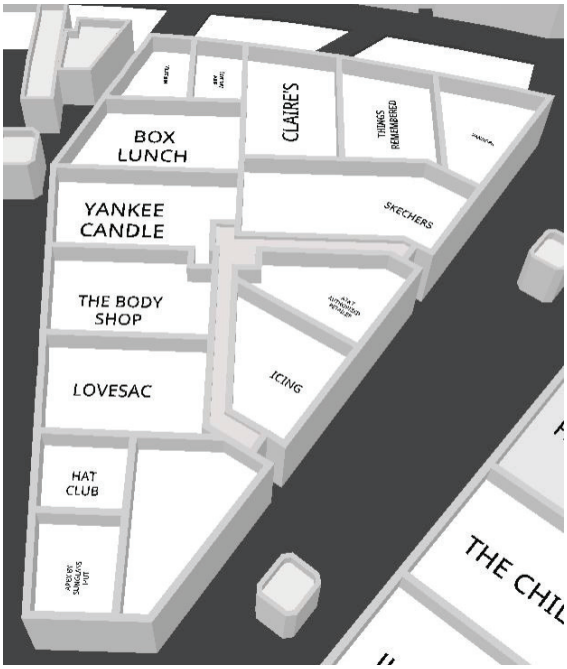


Figure 4.7: The Original Floor Plan Where the Walking Trace in Fig. 4.6 was Extracted.

4.4.2 Combining Walking Traces from Multiple Users

For a large shopping mall, a single shopper may only walk on part of the floor plan, so we need to connect walking traces from multiple shoppers to cover the whole floor plan. Even for the same part of the floor plan, we can combine the data from multiple users to improve performance. The sampling locations of each store provide good landmarks to combine or connect walking traces from different shoppers.

We design an iterative algorithm similar to the one in Shen *et al.* [11] to combine data from multiple shoppers to construct a more accurate floor plan. Note that the graph in Section 4.3 only shows the connectivity between different stores, while Algorithm 1 adds displacements between stores in the graph. Let G denotes the graph that represents the floor plan. In graph G, there are n' vertexes which represent the n' sampling positions in n stores. The edges between vertexes represent displacements between stores.

Let c_i be the current coordinate for vertex i and $\vec{d}_{i,j} = c_i - c_j$ the current displacement between vertexes i and j in the graph. Assume there are N edges (displacements) between vertexes in the graph. By combining the data from multiple shoppers, the system may receive $N_{i,j}$ measurements $\{\vec{r}_{i,j,k} | 1 \leq k \leq N_{i,j}\}$ for every displacement $\vec{d}_{i,j}$, where measurement $\vec{r}_{i,j,k}$ is extracted from the smartphone's IMU sensor readings. We first initialize all the vertex coordinates to the origin (Lines 1 to 2), so all initial displacements are also zero. We then iteratively update the vertex coordinates based on the measurements. In each iteration (Lines 4 to 18), we first average all the measurements for each current displacement as $\vec{\eta}_{i,j} = \frac{1}{N_{i,j}} \sum_{k=1}^{N_{i,j}} \vec{r}_{i,j,k}$ (Lines 12 to 14). We then sum up the adjustment vectors over all N_i neighboring vertexes as $\vec{S}_i = \sum_j^{N_i} \vec{\eta}_{i,j} - \vec{d}_{i,j}$ (Line 15). The coordinate for vertex i is updated as $c_i = c_i + \frac{1}{N_i} \vec{S}_i$, where $\frac{1}{N_i} \vec{S}_i$ is the average adjustment vector across N_i neighboring ver-

texes (Line 16). The iteration terminates if the average of $S = \frac{1}{n'} \sum_{i=1}^{n'} \frac{|\vec{S}_i|}{N_i}$ (Lines 17 to 18) is below a threshold α at which point the algorithm outputs the coordinates of each of the n' sampling positions (Line 19).

The running time of Algorithm 1 largely depends on how fast the algorithm converges, which further depends on how consistent the input walking traces are. For each iteration, the running time increases as the number of stores, i.e., the number of vertexes n', and the number of edges N increase. Given the same set of stores, the more walking traces from shoppers, the less uncertainty of the graph G and thus the higher accuracy of constructed floor plan, and vice versa.

When the shopper passes two adjacent sampling positions for a store, the system may not locate the shopper correctly especially in a large environment with a very small RSS fingerprint database. If we incorporate the adjacent sampling positions in the algorithm, the results would not be good. However, it is relatively easy to locate the shopper between two sampling positions of adjacent stores because of the large interval. As a result, we only use one sampling position for a store in the construction algorithm and explore other sampling positions as well to infer the store dimensions.

4.4.3 Pathway Recognition

Fig. 4.8 illustrates a representive distribution of stores in a shopping mall. The stores can be adjacent to each other like stores 1 and 2, on the opposite sides of a pathway like stores 3 and 8, or back-to-back with each other like stores 7 and 12. The adjacency relations have been inferred by our techniques in previous sections. The back-to-back stores are not directly reachable and thus do not affect the performance of IndoorWaze. The stores on the opposite sides of the pathway are often directly reachable, but their relations cannot be captured by previous techniques. For example, stores 3 and 8 are physically directly reachable from each other, but they

are not directly connected in the graph formed with our previous techniques. The disconnectivity of stores on the opposite sides of the pathway may cause problems in the navigation. For example, a shopper is now at store 3, but his destination is store 10. The system may navigate the shopper along the blue solid route in Fig. 4.8, because the system thinks by mistake that stores 1 and 6 are adjacent to each other, but stores 3 and 8 are not directly reachable. Actually, the shortest path from store 3 to store 10 should be the red dashed line.

From the discussion above, it is important to recognize each pathway in the graph and correctly associate the stores on its two sides. For example, we need to know there is a pathway between stores 1 and 6 but not between stores 6 and 11. We use some simple criteria to identify such relations with stores 1, 6, and 11 as an example, which are situated around the intersection of pathways. First, there exist straight (e.g., north-south in Fig. 4.8) walking traces between two stores. Second, there are walking traces that pass either store in other directions (e.g., west-east in Fig. 4.8). The walking traces under these two criteria should cross with each other. Third, the likelihood of collecting similar fingerprints along the walking trace between the two stores is relatively low. For example, the walking trace from store 6 to store 1 meets the third criterion, but the one from store 6 to store 11 does not because there is a big space gap between stores 1 and 6 but not between stores 6 and 11. Our previous techniques have identified adjacent stores along either side of a pathway. After we know the pathway between stores 1 and 6, we can correctly associate the other stores along the two sides of the pathway. Fig. 4.9 is the new graph after we add edges (dotted lines) to stores on the opposite sides of the pathway. Sometimes two stores on the opposite sides may not be directly reachable in practice, so we need to double-check if there is a direct walking trace between each pair of opposite stores in the original walking traces.

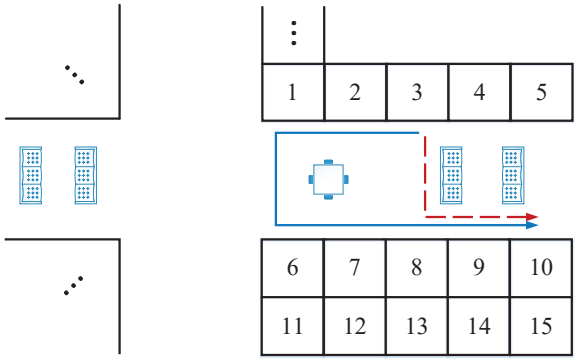


Figure 4.8: An Exemplary Floor Plan with Three Common Geometric Store Relations.

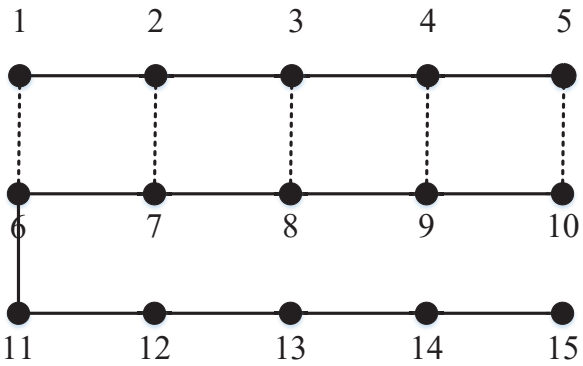


Figure 4.9: A Refined Graph Representation of the Floor Plan in Fig. 4.8.

4.4.4 Dealing with Complex Walking Traces

Previous sections only consider the shoppers walking along one side of the pathway, but some shoppers may walk between stores on the opposite sides of the pathway. As a result, the walking traces from many shoppers become too complex to construct a floor plan. Now we present methods to extract walking traces which are useful in the floor-plan construction. We observe that the indoor layout imposes different patterns on the shoppers' walking traces. For example, many people tend to make a turn at the intersection stores (e.g., stores 1 and 6 in Fig. 4.8) rather than around the stores in the middle of one row (e.g., stores 2 to 4). As a result, we can count the

number of shoppers who turn at each store to infer the stores at turning positions. Then we search for straight walking traces between turning stores. For example, a walking trace from store 1 to 5 is a valid trace which can be used in the floor-plan construction. By contrast, a walking trace that passes store 1, 2, 3, 8, 9, and 10 is rejected because there is a turn at store 3. Other walking traces in the original data set are not useless and can still be used to infer the connectivity between stores on the two sides of the pathway.

4.4.5 Navigation

With the aforementioned techniques, the system can construct a labeled indoor map with Wi-Fi RSS fingerprints to provide usable indoor navigation services. Let us continue the previous shopping-mall example. The shopper first inputs his target store into the IndoorWaze app. The system then estimates the shopper's current location based on his current fingerprint readings and the fingerprint map. After that, the system searches in the floor plan and finds a shortest path from his current location to the target store. Based on the path and the shopper's current location, the system provides context-aware audio instructions to the shopper. The system also tracks the shopper's realtime positions using his RSS fingerprints and provides new instructions if he deviates from the path. The RSS fingerprints the shopper measures may not be stable when walking. If we infer his realtime locations based on only the most recent RSS fingerprints, the calculated positions would be back and forth because of signal fluctuations. Similar to what we have proposed previously, we can determine whether the shopper is passing a store only when he can report at least 4 continuous RSS fingerprints which are most similar to those collected by the store employees. In addition, we can combine the shopper's IMU sensors and the floor plan to detect false position estimates. For example, the shopper in Fig. 4.8 started from

store 2 and wants to reach store 5, and he is currently at store 3. From his IMU sensors, the system knows that he walks in a straight line. If the position estimate changes from store 3 to 2, our system can immediately catch this wrong location estimate. Such navigation details are well considered in our system.

4.4.6 RSS Fingerprint Update

The RSS distribution at a sampling position may change due to indoor layout update. While the employee of nearby stores can easily notice the change and recollect the RSS fingerprint samples, such change may not be noticed by the employee of stores that are further away. It is possible to design and develop an automatic mechanism for detecting RSS fingerprint change to keep the RSS distribution up-to-date at each sampling location. Specifically, we can maintain the RSS distribution using the most recent RSS fingerprints from users at the same sampling location using a sliding window. For example, the most recent RSS distribution can be fitted from the 100 most recent ones or the ones received within the last 24 hours. If the difference between the most recent RSS distribution and the original RSS distribution exceeds certain threshold, we can update the RSS distribution at the sampling location using the most recent RSS fingerprint samples and let the app inform the nearby store's employee to collect additional samples if needed. We leave the detailed investigation of this issue as our future work.

4.5 System Implementation

We implemented IndoorWaze with Java on a Google Nexus 6 smartphone which has a Quad-core 2.7 GHz Krait 450 CPU, 3 GB RAM, a 5.96-inch display, and four relevant IMU sensors (magnetometer, compass, accelerometer, and gyroscope). The sampling frequencies for IMU sensors and the Wi-Fi module are 16.7 Hz and 1

Hz, respectively. Below, we briefly describe some implementation details about step detection, turn detection, and AP filtering.

4.5.1 Step Detection

The system estimates the walking distance of shoppers by counting their steps. The accelerometer can capture the motion caused by walking and recognize each step. For example, let $(a_x,\ a_y,\ a_z)$ denote an accelerometer reading which represents the acceleration along the three axes. In each step, the amplitude $\sqrt{a_x^2 + a_y^2 + a_z^2}$ reaches a maximum when the shopper's heel strikes the ground. As a result, we just need to count the amplitude peaks in a sequence of readings to infer the number of steps.

Fig. 4.10(a) illustrates the amplitudes of a sequence of acceleration readings. It is difficult to count the peaks caused by each step because the shopper's random movements can introduce many peaks in the signal. We first process the data using a moving average filter to obtain Fig. 4.10(b) which is much smoother than Fig. 4.10(a). We then use a low-pass filter to remove some high-frequency components caused by random phone movements. Fig. 4.10(c) illustrates the signal after filtering, from which we can easily identify the peak caused by each step. To count the peaks, we have to know a rough period of the signal, which can be inferred by doing a signal autocorrelation. After that, we use a sliding window to search and count the peaks in the signal. The length of the sliding window t_w can be equal to the period of the signal. The ith data sample is a peak if it is larger than all the samples in $[t(i) - t_w/2, t(i) + t_w/2]$, where $t(i)$ is the time stamp of the ith data sample.

4.5.2 Turn Detection

The compass in the smartphone works well in the outdoor environment. However, the indoor magnetic environment is so complex that the compass cannot provide

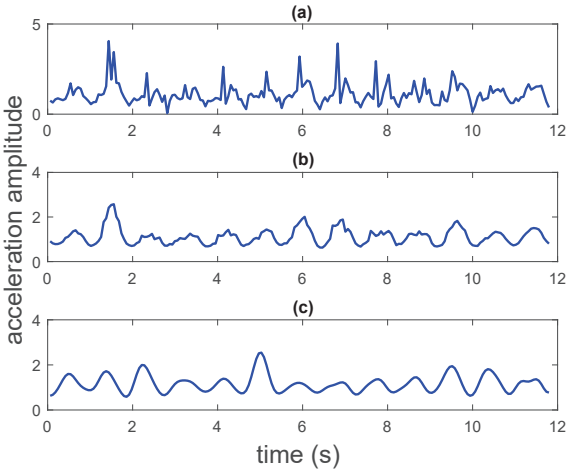


Figure 4.10: Acceleration Data Processing for Step Detection

reliable direction estimation. Fortunately, the gyroscope can provide accurate short-term angle changes and is not affected by indoor magnetic signals. In our system, we combine the accelerometer and gyroscope to provide stable turn detection.

The gyroscope reading (g_x, g_y, g_z) represents realtime phone angle changes in the 3-axis smartphone coordinate system. To detect the direction changes of walking users, we need to convert the gyroscope readings to those in the Earth coordinate system. Before the axis rotation, we have to know the realtime altitude of the phone. Assume that (ϕ, θ, ψ) represents the phone's altitude angles in the three axes of the Earth coordinate system. Based on the gravity sensor (i.e., the vertical component of the accelerometer), we can only infer ϕ and θ which seem not enough to calculate the exact altitude in the 3-axis coordinate system. Fortunately, the user normally walks in a two-dimensional $x - y$ plane, so what we care about is not the exact yaw angle ψ but an angle change around the z axis. Therefore, we can calculate the gyroscope data in the Earth coordinate system as $(e_x, e_y, e_z) = R_y(\theta) \cdot R_x(\phi) \cdot (g_x, g_y, g_z)$, where

$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & sin(\phi) \\ 0 & -sin(\phi) & cos(\phi) \end{bmatrix}$ and $R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & -sin(\theta) \\ 0 & 1 & 0 \\ sin(\theta) & 0 & cos(\theta) \end{bmatrix}$ are the rotation matrices around the x and y axes, respectively. After the axis rotation, we use e_z to measure the angle changes of the walking user.

4.5.3 Filtering Access Points

The Wi-Fi environment in the shopping mall is very complex. In addition to the APs set up by the shopping mall, every store can install its own APs. Sometimes we may even find some hotspots served by mobile devices which are not stable signal sources. In the experiment area, we found 487 APs in total. If we incorporate all the APs into the experiments, the performance would be bad. We therefore consider three possible criteria to classify the APs as shown in Table 4.1 based on the information we collected. In particular, we observed that the signals from the APs with names are more stable, because these APs are often installed by shopping mall operators and stores (e.g. AT&T store) and have fixed locations. In contrast, the signals from those APs without names exhibit large fluctuation and are thus less stable. In addition, the signals of the APs classified under the other two criteria do not differ too much. We therefore only use named APs in our experiments.

4.6 System Evaluation

IndoorWaze was evaluated in a large shopping mall in our metropolitan area. The mall covers 120,000 m^2 and consists of 213 stores. In the experiment, we mimicked both shoppers and store employees. The mall is so large that we cannot cover the whole area or all the stores. Therefore, the experiment was done in part of the shopping mall as illustrated in Fig. 4.11. The experiment area is about 6,000 m^2 and

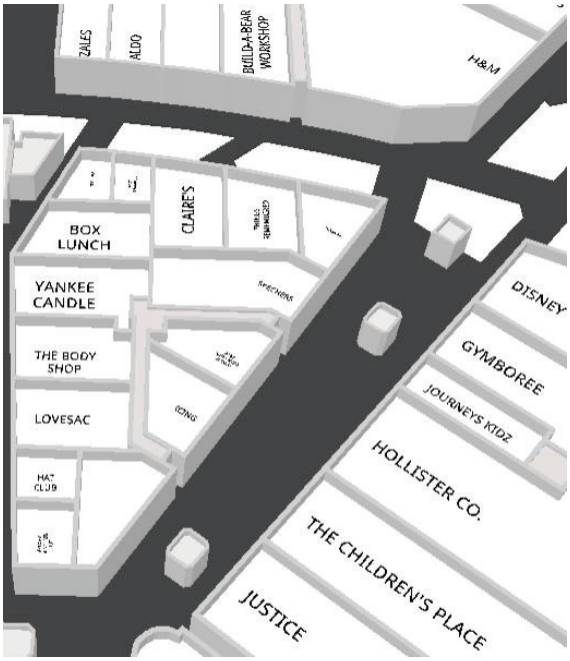


Figure 4.11: The Real Floor Plan in the Experiment.

contains 25 stores. We chose this area because it contains all possible indoor features such as pathways and crossings.

Our evaluation mainly focuses on the construction of the labeled floor plan which is the main contribution of this chapter. For completeness, we also test the room level localization and navigation based on the generated floor plan using Horus [44]. It is difficult to compare the IndoorWaze with prior work quantitatively, since different works use different data sources, provide different functions, or work in different settings. In addition, quantitative comparison will be affected by many factors and parameters in real experiment that we cannot control. Instead, we provide a qualitative comparison with prior work in Table 4.2.

4.6.1 Evaluation of Floor Plan Construction

Data Collection. We collected data during the normal operating hours of the shopping mall. We first acted as the store employees to collect fingerprints along each store in the floor plan. For this purpose, we picked the number of sampling positions according to the dimension of each store. Fig. 4.12 illustrates a histogram of the number of sampling positions along the stores. Most stores are not very large and just need three sampling positions. We first collected Wi-Fi RSS fingerprints at every sampling position for 20 seconds. We then held the phone and walked around for about 20 minutes at a normal speed (0.8 m/s). During the 20-minute collection period, we collected 14 fingerprint traces to cover each store at least twice.

Fingerprint collection by shoppers under IndoorWaze is the same as in Walkie-markie [11] and UnLoc [9] and thus they incur similar time cost. IndoorWaze additionally requires store employees to collect fingerprint samples at sampling locations near their stores. Such cost is very acceptable in practice, as it suffices to take samples for about 20 seconds at each sampling location. While we expect that more samples can improve the localization accuracy, the benefit from additional samples diminishes as the number of samples increases. As a result, the cost of data collection under IndoorWaze is similar to that under prior solutions Walkie-markie [11] and UnLoc [9] and is significantly lower than prior floor plan construction systems that require more complex information such as image [71], video [72], and check-in [64] that require explicit participation of shoppers.

Visual Comparison. Fig. 4.13 depicts the graph representation of the floor plan generated by our system. The stores at the crossings are represented by red dots, the edges along either side of a pathway are shown by solid lines, and edges across the pathways are represented by dotted lines. Fig. 4.13 only shows the relative positions

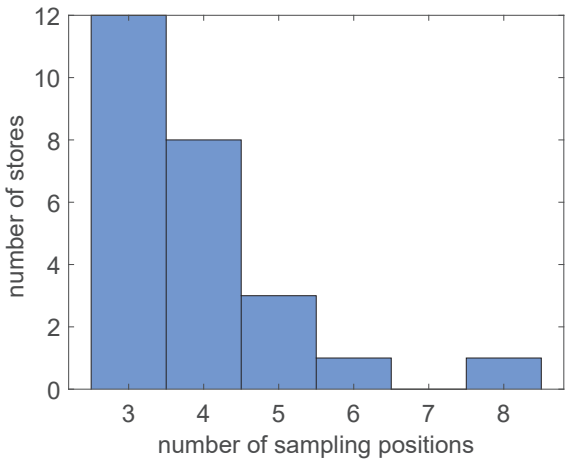


Figure 4.12: Number of Sampling Positions for Stores.

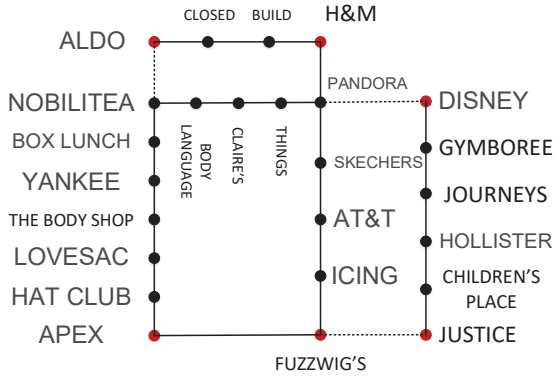


Figure 4.13: Graph Representation of the Floor Plan with IndoorWaze.

among adjacent stores, and the resulting floor plan is still too rough for usable indoor navigation. Fig. 4.14 to Fig. 4.16 show the generated floor plan after 5, 10, and 20 iterations, respectively. The floor plan in Fig. 4.16 is very similar to the ground truth in Fig. 4.11. It clearly identifies the pathways and differentiates the stores along two sides of the pathways. All the store labels are also correct.

Dimension Estimation. In addition to the relative store positions, our system can estimate the dimension of each store. Fig. 4.17 illustrates the errors in the store-dimension estimation, where the median error is about 2.5 steps. Actually, the

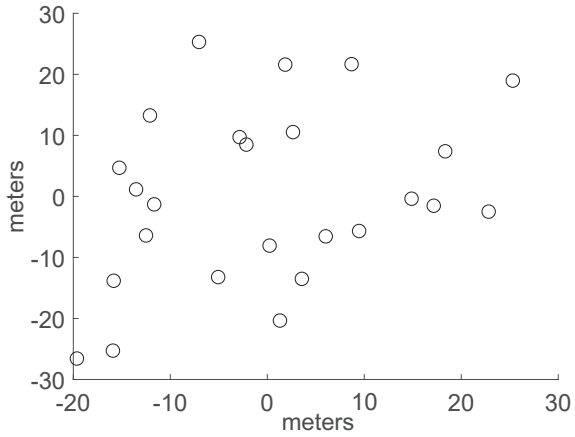


Figure 4.14: Floor Plan after 5 Iterations.

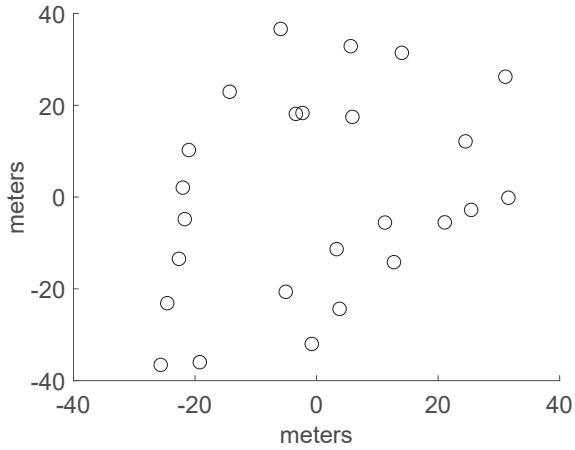


Figure 4.15: Floor Plan after 10 Iterations.

estimation errors are closely related to the true store dimensions. Fig. 4.18 illustrates the ratio of errors to the store dimension, where the median error ratio is about 12%. By accurately estimating the store dimensions, we can infer the duration for a shopper to pass a particular store.

Impact of Participation. IndoorWaze is a crowdsourcing-based system, and not all the stores may participate in data collection. In the experiment, we randomly removed a few stores from the dataset and evaluated the impact on the floor-plan construction. Fig. 4.19 shows the floor plan after we randomly removed three stores.

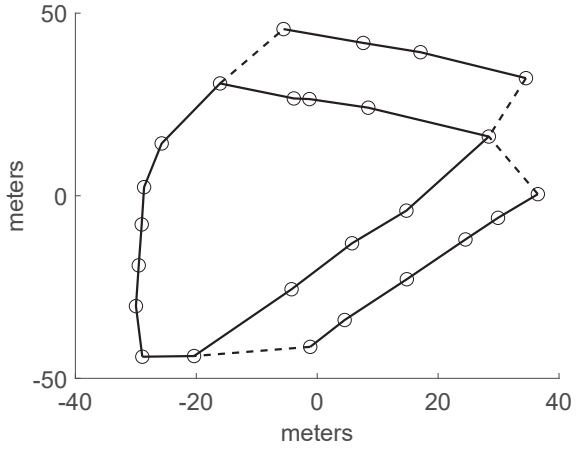


Figure 4.16: Floor Plan after 20 Iterations.

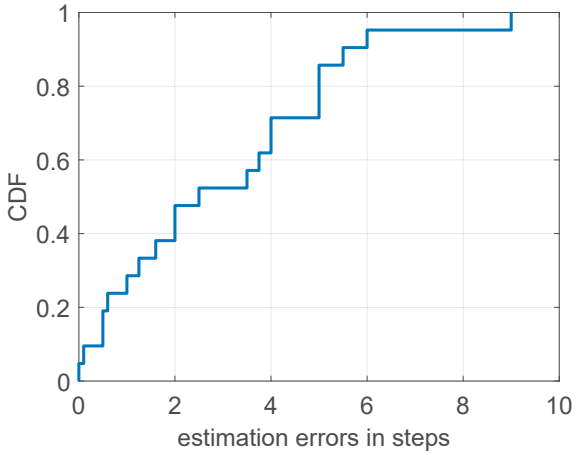


Figure 4.17: Accuracy of Store-dimension Estimation.

The floor plan is not as good as the one in Fig. 4.16, but it is still sufficient for highly usable indoor navigation.

4.6.2 Localization and Navigation

Static localization. We evaluated the localization performance of IndoorWaze when the shopper is static. In this experiment, we stood still at the center of each store's exterior perimeter for about 10 seconds. Then, we computed the average of all the

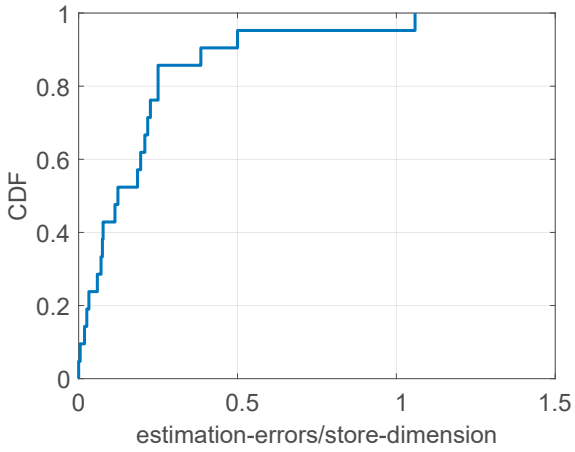


Figure 4.18: Ratio-based Accuracy of Store-dimension Estimation.

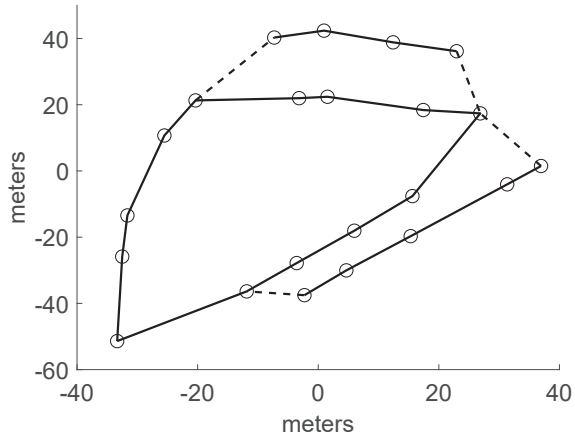


Figure 4.19: Floor Plan after 20 Iterations with 3 Non-participating Stores.

collected RSS fingerprints to find the best match in the RSS fingerprint database. Our system found the correct stores for all the 25 location queries.

Navigation. We also tested the navigation service based on the generated floor plan. In this experiment, we randomly picked four destination stores ("BUILD", "CLAIRE'S", "AT&T", and "JOURNEYS") and a starting store "LOVESAC" in Fig. 4.13. We successfully reached each destination store at ease by following IndoorWaze's audio instructions.

4.7 Related Work

Most existing indoor localization/navigation systems focus on improving the localization accuracy. To provide a user-friendly navigation service, these systems need a labeled floor plan with location-store mappings. However, such context-aware labeled floor plans are often unavailable or quite prohibitive to obtain. Our work fills this gap and generates a high-fidelity labeled floor plan that can be used with most existing indoor localization/navigation system. In what follows, we describe some representative work on indoor localization systems, indoor navigation systems, and indoor floor-plan construction. Table 4.2 summarizes some most germane work.

Fingerprint-based techniques are the most popular approaches for indoor localization. As probably the first work along this line, Radar [43] is a deterministic localization method that employs RSS for indoor localization. Horus [44] improves Radar by keeping a fingerprint distribution for every position in the floor plan and then finding a maximum likelihood match in the database. SurroundSense [46] introduces more indoor features (such as light and sound) in addition to RSS fingerprints.

Model-based indoor localization techniques estimate indoor locations using statistic models. A popular approach is to build a relation between RSS and signal propagation distance based on the RF propagation model (e.g., the log-distance path loss (LDPL)) [47]. The work in [49] evaluates some self-calibrating algorithms in office environments and finds that the median errors are consistently greater than 5m. In addition to LDPL-based schemes, there are other techniques based on Angle of Arrival (AoA) [50], Time of Arrival (ToA) [52], and Time Difference of Arrival (TDoA) [53].

Simultaneous Localization and Mapping (SLAM) is a technique originating from the robotics community. SLAM relies on a robot to explore the space of interest with

discrete landmarks or obstacles. Based on the laser ranging and cameras in the robot, we can determine the relative locations of the landmarks, and the robot can infer its relative location. WiFi-SLAM [60] uses a Gaussian process to model the relation of Wi-Fi signal strengths. With more sensors embedded in the smartphone, many techniques combine IMU sensor data with human movements to realize SLAM. For example, the schemes in [8, 73] use the indoor constraints to map human mobility traces to the floor plan and then generate the fingerprint database. LiFS [10] maps fingerprints by comparing the similarity between the high-dimensional fingerprint space and the stress-free floor plan.

There are also systems based on a leader-follower navigation model. Escort [11] navigates users based on crowd-encounter information and dead reckoning. Magnetic information is adopted by some systems [70, 69] for navigation. The leader first records the magnetic signal along the route. The user then compares the magnetic signals he collects with the signals from the leader to determine his location. Travi-Navi [63] is a vision-guided navigation system. The leader (store employee) first takes photos along the route to the store. The system then gives the instructions based on the photos and dead-reckoning techniques. This line of systems do not construct a context-aware floor plan.

People have also studied indoor floor-plan construction. SemSense [64] presents a floor-plan labeling method based on a given unlabeled floor plan and shoppers' manual check-ins in each store. Jigsaw [71] presents a floor-plan construction technique by combing IMU sensors and landmarks extracted from images taken by crowdsourcing workers. CrowdMap [72] is a sensor-rich video-based approach for indoor floor-plan construction, which furnishes the consistent video frame relation to generate spatial information for the indoor environment. Unloc [9] constructs the floor plan by sensing natural indoor landmarks such as elevators and stairs, which are then connected via

dead reckoning. Walkie-markie [11] generates the floor plan based on the RSS trend when the user passes the AP. Other IMU-based floor-plan construction methods are presented in [74, 75, 76]. None of these techniques could automatically generate a high-fidelity indoor floor plan with accurate POI labels and dimensions. In addition, our system IndoorWaze incurs minimal effort on crowdsourcing users.

Algorithm 2: Iterative Floor-Plan Construction

input : Graph $G = (V, E)$, coordinates $\{c_i | 1 \leq i \leq n'\}$, number of adjacent vertexes of ith vertex N_i, displacement measurements $\{\vec{r}_{i,j,k} | 1 \leq k \leq N_{i,j}, 1 \leq j \leq N_i, 1 \leq i \leq n^{'}\}$, number of edges N, threshold α.

output: $\{c_i | 1 \leq i \leq n'\}$.

forall $i \in \{1, \ldots, n'\}$ **do**
    $c_i \leftarrow (0, 0)$;
$S \leftarrow \infty$;
while $S > \alpha$ **do**
    $S \leftarrow 0$;
    forall $i \in \{1, \ldots, n'\}$ **do**
        $N_i \leftarrow 0$;
        $\vec{S}_i \leftarrow (0, 0)$;
        forall $j \in \{i + 1, \ldots, n'\}$ **and** $e(i, j) \in E$ **do**
            $N_i \leftarrow N_i + 1$;
            $\vec{d}_{i,j} \leftarrow c_i - c_j$;
            $\vec{\eta}_{i,j} \leftarrow (0, 0)$;
            forall $k \in \{1, \ldots, N_{i,j}\}$ **do**
                $\vec{\eta}_{i,j} \leftarrow \vec{\eta}_{i,j} + \frac{1}{N_{i,j}} \vec{r}_{i,j,k}$;
            $\vec{S}_i \leftarrow \vec{S}_i + \vec{\eta}_{i,j} - \vec{d}_{i,j}$;
        $c_i \leftarrow c_i + \frac{1}{N_i} \vec{S}_i$;
        $S \leftarrow S + \frac{1}{N_i} |\vec{S}_i|$;
    $S \leftarrow \frac{S}{n'}$;
return $\{c_i | 1 \leq i \leq n'\}$;

Table 4.1: Wi-Fi AP Analysis

Frequency	**Encrypted?**	**Has a name?**
2.4 GHz: 250	Yes: 411	Yes: 281
5 GHz: 237	No: 76	No: 206

Table 4.2: Summary of Related Work

System	**Method**	**User's task**	**Navigation**	**Floor plan**	**Granularity**	**Flexibility**	**labelling**
Travi-Navi [63], FollowMe [69], [70]	Leader-follower	Not required	✓	–	–	Low	–
SemSense [64]	Manual check-in	Explicit	–	–	–	Low	✓
Jigsaw [71], CrowdMap [72]	Image, Video	Explicit	–	✓	Room Level	High	–
Walkie-markie [11], UnLoc [9]	RSS, IMU	Implicit	–	✓	Pathway Level	High	–
IndoorWaze	RSS, IMU from both shoppers & employees	Implicit	✓	✓	Room Level	High	✓

Chapter 5

PRACTICAL ATTACKS ON HOME ALARM SYSTEMS

5.1 Overview

In this chapter, we present new *event-eliminating* and *event-spoofing* attacks on commercial home alarm systems by interfering with the reed switch in almost all contact sensors. In both attacks, the adversary uses a magnet of its own—called a malicious magnet henceforth—to control the state of the reed switch. In the event-eliminating attack, the adversary makes the malicious magnet have the same polarity as the legitimate one so that their magnetic fields strengthen each other. When the adversary opens the corresponding door to enter the house, the interfered reed switch may not trigger any alarm because the magnetic strength around the contact sensor is still maintained by the malicious magnet. In the event-spoofing attack, the adversary makes the malicious magnet have the opposite polarity to the legitimate one so that their magnetic fields weaken each other. If the magnet field strength falls below a threshold, the reed switch can trigger a false alarm even though the door is always closed. Even worse, when receiving no alarm or too many annoying false alarms from a particular contact sensor, the user or base station may consider it faulty by mistake and temporarily disable it. The door with the disabled sensor thus can become the weakest entry point until a technician responds to a service call, which may happen in a few days.

In addition to the event-eliminating and event-spoofing attacks, we present a new *battery-depletion* attack to deplete the sensor battery quickly and quietly. The basic idea is to force a contact sensor to generate large amounts of fake events and transmit

continuously to consume energy without raising alarms. The contact sensor with low battery cannot provide any security alarm unless the user manually replaces the battery, which may be infeasible if the user has no backup batteries at home or is traveling away from the home. Since the contact sensor will have to be temporarily disabled to avoid continuous low-battery warnings, attackers may have a long time window to illegally access the house and an even longer time window if the home owner is on travel. [1]

To launch the above attacks, there are some critical challenges to solve. First, attackers are outside the target house and cannot see the interior contact sensor. Attackers may not be able to achieve their goal or even trigger unexpected alarms if they cannot accurately infer the sensor's location or magnet's polarity. To solve this challenge, we present techniques for attackers to localize the contact sensor and then determine the legitimate magnet's polarity with a smartphone. Second, it is impractical for attackers to manually generate a large amount of fake events to deplete the sensor battery with a permanent magnet. To tackle this challenge, we build a system with a programmable microcontroller and an electromagnet to attack the sensor automatically. The system can be programmed to transmit magnetic signals periodically to force the sensor to generate OPEN and CLOSE events continuously until the sensor battery is dead. Finally, the triggered fake events can be received by the base station which may report the anomaly to the service provider or the home owner. To launch the attack quietly, we introduce novel jamming techniques to prevent the base station from receiving any packet while triggering the contact sensor to transmit continuously.

Our contributions can be summarized as follows.

[1] One of the authors had to remotely disable a contact sensor with low battery during his vacation to avoid continuous warnings sent to his phone and the alarm company, which motivates this work.

- We present practical event-eliminating and even-spoofing attacks on home alarm systems utilizing security flaws of the reed switch which is commonly used for proximity detection in contact sensors. Attackers can eliminate true alarms and also generate false alarms with magnetic signals of different polarities. To make the attack practical, we introduce techniques to help external attackers localize interior contact sensors and infer their magnet polarity with a COTS smartphone.

- We build a system with a programmable micro-controller and an electromagnet to make a contact sensor continuously transmit to the base station so that its battery can be quickly depleted. We also propose novel jamming techniques so that the base station cannot receive any sign of the ongoing battery-depletion attack. Same as event-eliminating and even-spoofing attacks, the battery-depleting attack is generic and can apply to almost all home alarm systems using the reed switch.

- We conduct extensive experiments to evaluate the above attacks. Our evaluation results show that the attacks are highly practical and effective. In particular, the attacker can successfully deplete a new sensor battery in 43 hours which should work for years.

5.2 Background

5.2.1 Reed Switch and Contact Sensor

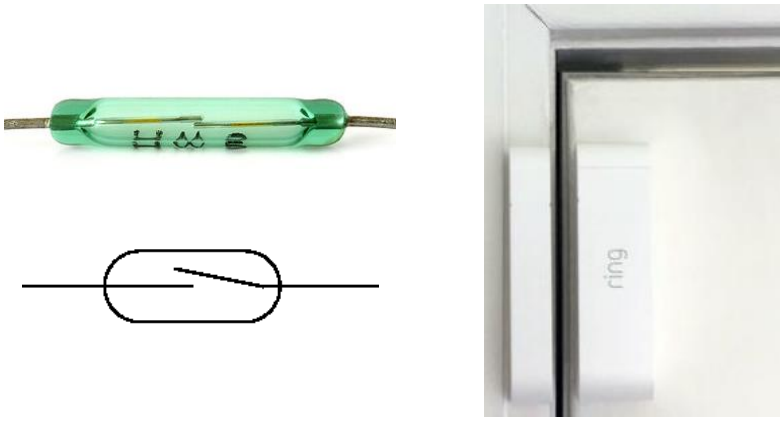
A Reed switch is a contactless electrical switch which is widely used as a proximity sensor to activate or deactivate a circuit. We can easily find reed switches in computers, alarms, and a lot of other appliances. Fig. 5.1(a) shows a reed switch used in a contact sensor. It contains two ferromagnetic contacts which are sealed in

a small glass envelope filled with unreactive gas. If there is a strong magnetic field parallel to the contacts, the two contacts are magnetized and snap together. Then the current flows through the closed reed switch to activate the circuit. When the magnetic field disappears, the two contacts are separated from each other so that the reed switch deactivates the circuit.

The contact sensor is normally installed on the interior of a window or door frame, and the associated magnet is usually installed on the interior of a window or door as illustrated in Fig. 5.1(b). When the door is closed, the sensor and magnet are very close to each other, so the reed switch keeps closed. When the door is open, the sensor and magnet are separated, so the reed switch is open and triggers an OPEN alarm to notify the base station and/or the user's smartphone app. Since each contact sensor is powered by a battery, it is normally in the sleep mode and does not respond to or forward any packet to save energy. Only some specific events (e.g., door OPEN or CLOSE) can switch the sensor into the active mode to transmit and receive messages. Since contact sensors have very limited computing resources and can be easily reached by malicious signals outside the house, they are the weakest points in the home alarm system.

5.2.2 Communication Protocol

Most home alarm systems use low-power communication protocols such as Z-Wave and Thread as summarized in Table 4.2. Z-Wave was developed by Zensys in 1999 targeting low-bandwidth communications between embedded devices such as security sensors, smart bulbs, controllers, and other home appliances. It can construct a mesh network consisting of different home embedded devices. Z-Wave devices transmit on 868.42 MHz in Europe and both 908.4 MHz and 916 MHz in North America for different purposes. Z-Wave also uses Frequency-shift Keying (FSK) as the modula-



(a) Reed switch (b) Door sensor

Figure 5.1: Examples of Reed Switches and Contact Sensors.

tion method. Thread is a new mesh network protocol developed specifically for IoT systems by Google. It is based on IEEE 802.15.4 and adopted in most IoT systems provided by Google.

Table 5.1: Technical Summary of Some Popular Home Alarm Systems

	Ring alarm	**Google nest alarm**	**Honeywell alarm**
Protocol	Z-Wave	Thread	Z-Wave
Frequency	908.6MHz, 916MHz	2.4GHz	868MHz
Modulation	FSK	O-QPSK	FSK
Proximity sensor	Reed switch	Reed switch	Reed switch
Battery of contact sensors	CR123A, about 1500 mAh	CR123, about 1500 mAh	CR2032, about 220 mAh

5.2.3 Home Alarm System

We use the Ring Alarm System as an example to describe the working principle of home alarm systems. The system mainly consists of three parts: a base station, contact sensors, and range extenders. Contact sensors monitor the OPEN or CLOSE state of the door and report such events to the base station. The base station controls

contact sensors in the range and reports events further to the user's smartphone and the alarm service company if any. It periodically broadcasts messages and is always ready to answer messages from contact sensors. After receiving an event from the contact sensor, the base station replies with an ACK. If no ACK is received in a certain period, the contact sensor retransmits the packet. The range extender serves as a signal repeater between the base station and sensors when they are far away from each other. In contrast to battery-powered contact sensors, the base station and range extenders are normally plugged to an outlet in the house, so they have no energy limitation when working.

Fig. 5.2 shows the system initialization steps of the Ring alarm system. The first step is to connect the base station to the Internet via Wi-Fi or Ethernet. Then the user installs a Ring app on the smartphone and also registers an account. The phone should keep the Bluetooth open for pairing with the base station. The user taps a button on the base station to start the pairing process. After the successful pairing, the user can manage the base station through the app. The fourth step is to add each contact sensor to the alarm system using the app. The user inputs each sensor ID to the system by scanning the QR code on the sensor using the smartphone. When the user installs the battery in the contact sensor, the sensor transmit a message to the base station to start the cryptographic key generation process. Based on the common initial key in firmware, the two devices generate two 128-bit keys for authentication and payload encryption, respectively. All the packets except the ACKs between the sensor and base station are encrypted using the 128-bit AES algorithm. The devices communicate at the frequency of 908.4 MHz in the sleep mode with a transmission rate of 40 kb/s and at the frequency of 916 MHz with a transmission rate of 100 kb/s in the active mode.

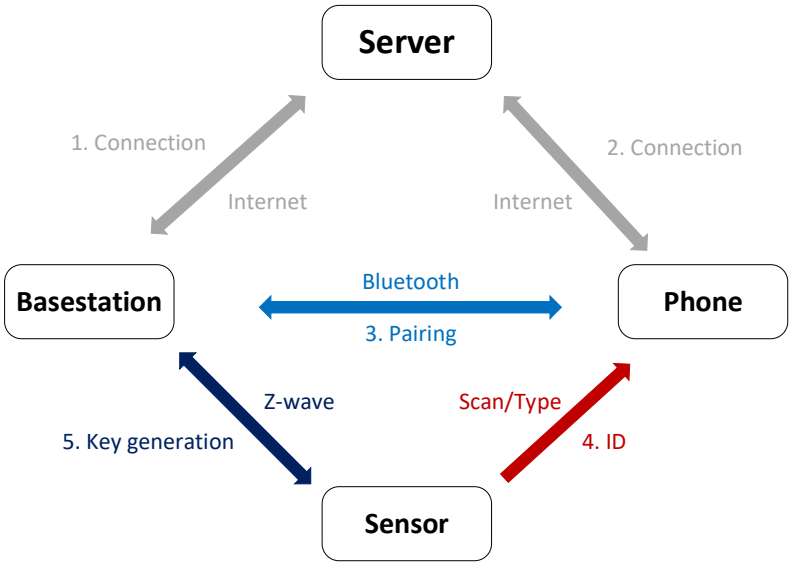


Figure 5.2: The Initialization of the Ring Alarm System.

5.3 Threat Model

We consider a realistic threat model for alarm systems in daily life. The entire alarm system—comprising the base station, contact sensors, and extenders—is installed inside the house, while attackers are outside and cannot physically access the devices. We do not consider attackers from the Internet who may hack the alarm service provider or the base station to disable the alarm system. This category of attacks deserve separate studies [77, 78]. Also, we assume that communications between devices are secure because the secret key is only known to the vendor. In our model, outside attackers use wireless signals to launch the attack in a short range.

Based on the model above, there are three types of potential attackers. Type-I attackers want to open the door to illegally access the house, but they do not want to trigger any alarm. Type-II attackers do not want to enter the house but want to trigger false alarms just for fun. Type-III attackers want to quickly deplete the battery of a selected contact sensor without arousing the attention of the user, base

Figure 5.3: The Bar Magnet Used in the System.

station, or alarm service company if any. When the battery level of the contact sensor is below a threshold, a low-battery warning is periodically sent to the user's smartphone, the base station, and the alarm service company. If the user has no backup battery at home or is away from home, he usually just disables the involved contact sensor to avoid receiving too many low-battery warnings and also enable the activation of other contact sensors. The door or window with the disabled sensor thus becomes the unguarded entry point into the house.

5.4 Inferring Location and Polarity

Outside attackers need to solve two challenges for a successful attack on the target door or window. First, they must infer the location of the contact sensor, so they can know where to launch the attack. The magnetic field generated by the paired magnet is a good location indicator. Fig. 5.3 shows a bar magnet which is commonly used in alarm systems. The sensor localization accuracy is critical to the success of the attack. Second, they need to determine the polarity of the magnet (the south or north magnetic pole), as incorrect magnet polarity may trigger unexpected alarms.

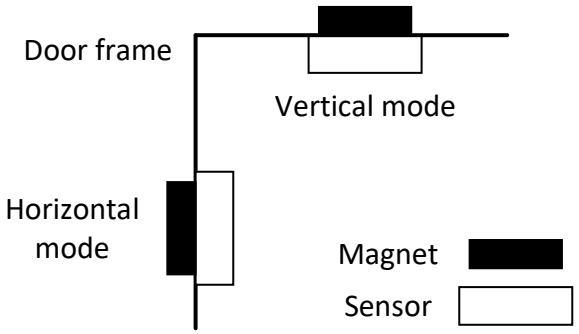


Figure 5.4: Two Install Modes.

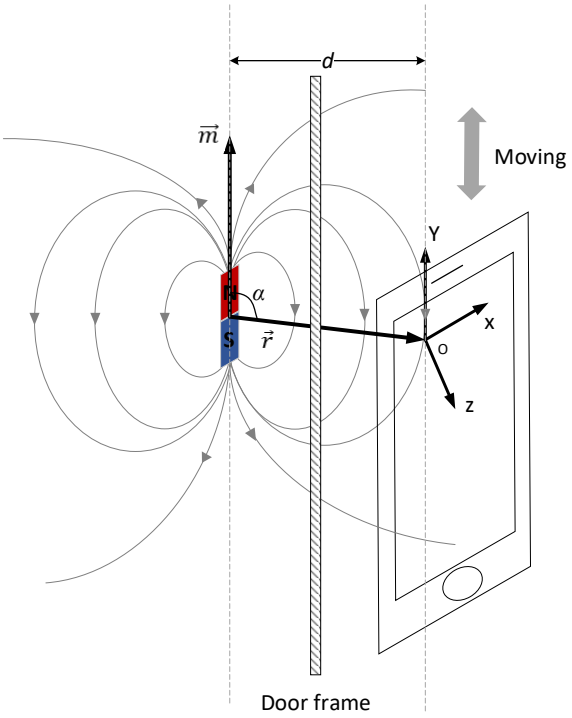


Figure 5.5: Sensor Localization Using a Smartphone.

Attackers can localize the contact sensor by measuring the magnetic field generated by the legitimate magnet. The contact sensor and magnet are always installed in the horizontal mode or the vertical mode on the door or the frame as illustrated in Fig. 5.4. In Fig. 5.5, the attacker moves the phone vertically along the door frame to collect the magnetic signal. O on the top left corner denotes the magnetometer sensor in the smartphone. The magnetometer sensor can measure a 3-dimensional magnetic field vector (MFV) based on the phone's local coordinate system. According to the magnetic field theory, the MFV received by the magnetometer can be calculated as

$$H(\vec{r}) = \frac{K}{\|\vec{r}\|^3}\left[\frac{3\vec{r}(\vec{m}\cdot\vec{r})}{\|\vec{r}\|^2} - \vec{m}\right], \tag{5.1}$$

where K is a constant related to the magnetic moment which determines the magnetic strength of the magnet, $\overrightarrow{r} = (r_x, r_y, r_z)$ represents the 3D distance vector relative to the magnetometer, $\overrightarrow{m} = (m_x, m_y, m_z)$ is the directional unit vector of the magnet, and all the variables take values in the magnetometer's coordinate system shown in Fig. 5.5. Since K is approximately a constant for a given magnet, the measured MFV is only determined by the 3D relative position and orientation between the magnet and magnetometer.

When we move the phone along the frame, the bar magnet is always parallel with the smartphone, so the horizontal distance d and the vector $\vec{m} = (0, 1, 0)$ do not change. As a result, the MFV is only related to vector $\vec{r}$. We use magnetic field strength (MFS) to denote the magnitude of the magnet field vector. Intuitively, MFS reaches its maximum when $\|\vec{r}\|$ reaches the minimum. Below we first theoretically prove the conjecture and then verify it via later experiments. In particular, we can

calculate $\|H(\vec{r})\|^2$ as

$$\begin{aligned}\|H(\vec{r})\|^2 &= \frac{K}{\|\vec{r}\|^3}\left[\frac{3\vec{r}(\vec{m}\cdot\vec{r})}{\|\vec{r}\|^2}-\vec{m}\right]\cdot\frac{K}{\|\vec{r}\|^3}\left[\frac{3\vec{r}(\vec{m}\cdot\vec{r})}{\|\vec{r}\|^2}-\vec{m}\right] \\ &= \frac{K^2}{\|\vec{r}\|^6}\left[3\cos^2\alpha+1\right].\end{aligned}$$

Since $\|\vec{r}\| = \frac{d}{\sin\alpha}$, we can substitute $\|\vec{r}\|$ and get

$$\|H(\vec{\alpha})\|^2 = \frac{K^2}{d^6}(4\sin^6\alpha - 3\sin^8\alpha),$$

where α is now the only variable in $\|H(\vec{\alpha})\|^2$. We then calculate the derivation as

$$(\|H(\vec{\alpha})\|^2)' = \frac{24K^2}{d^6}(\sin^5\alpha\cos^3\alpha).$$

We can find that $\|H(\vec{r})\|^2$ reaches the maximum when $\alpha = \pi/2$ and $\|\vec{r}\|$ reaches the minimum. So the phone can measure the maximum MFS when the magnetometer is in the same height with the magnet. Fig. 5.6 illustrates the MFS readings when we move the phone along the door frame at a uniform speed. The magnetometer reaches the same height with the magnet in the 60th sample. The experiment results are consistent with our above analysis.

When we know the position of the magnet, it is straightforward to infer its polarity. We just need to place the phone along the door frame in the same height with the legitimate magnet and check the y-axis reading of the MFV. If the reading is negative, the south pole is at the bottom of the magnet, or the north pole is at the bottom.

5.5 Events Eliminating and Spoofing

Now we present two attacks to manipulate the reactions of contact sensors to the OPEN or CLOSE action. From Section 5.2.1, the reed switch changes its state when the MFS along the contacts falls below or exceeds a threshold. Following this principle, our basic idea is to influence the magnetic field along the reed switch in

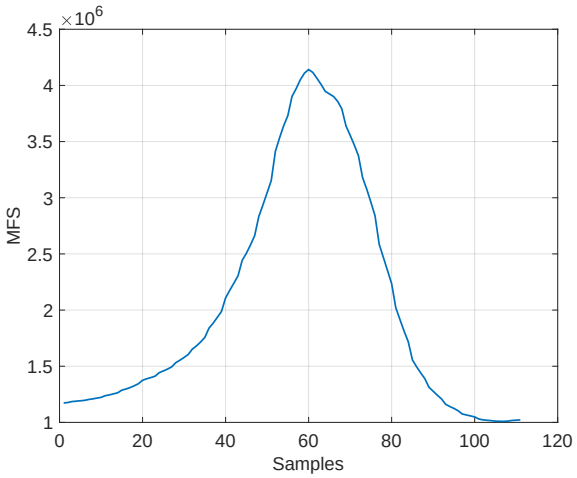


Figure 5.6: MFS Collected When Moving Magnetometer at a Uniform Speed.

the contact sensor using a malicious magnet. The attacker can launch the event-eliminating attack to disable the contact sensor which then does not raise an alarm when the attacker opens the corresponding door or window. In addition, the attacker can use the event-spoofing attack to trigger false alarms, though the door or window keeps closed.

5.5.1 Event-Eliminating Attack

The event-eliminating attack can enable the attacker to open the door "quietly" without triggering any alarm. In Fig. 5.7, the attacker moves an malicious magnet of the same polarity as legitimate one close to the contact sensor. The magnetic field vectors of the two magnets are approximately in the same direction when passing the reed switch, so the two magnetic fields are constructive to each other. At this time, if the attacker opens the door and separates the legitimate magnet from the sensor, the sensor (reed switch) reports nothing as long as the malicious magnet keeps the MFS along the reed switch above the default threshold.

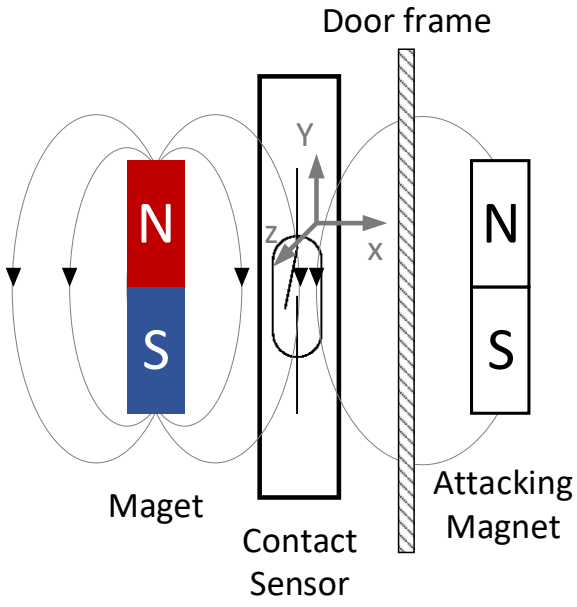


Figure 5.7: Event-eliminating Attack Using a Malicious Magnet of the Same Polarity.

We now introduce the requirement for the malicious magnet to successfully disable the contact sensor. The contact sensor and the legitimate magnet in Fig. 5.7 are installed in the horizontal mode. To generate MFV that is parallel to the reed switch in the sensor, the attacker also needs to place the malicious magnet horizontally. The strength of the magnet that the attacker needs to generate the MFS above the threshold is related to the 3D distance $\vec{r}$ and orientation $\vec{m}$ of the magnet relative to the reed switch. Given the local coordinate system in Fig. 5.7, let (x, y, z) denote the position of the malicious magnet. Then we can get $\vec{r} = (-x, -y, -z)$ and $\vec{m} = (0, 1, 0)$. So the y-axis component H_y of the MFV that is generated by the malicious magnet along the reed switch can written as

$$H_y = \frac{K(2y^2 - x^2 - z^2)}{(x^2 + y^2 + z^2)^{5/2}}. \tag{5.2}$$

Assuming that the magnetic strength threshold to change the reed switch's state is η, we can have

$$|H_y| > \eta.$$

This means that in addition to preserve certain orientation, the attacker needs a malicious magnet with a constant $K > \frac{\eta(x^2+y^2+z^2)^{5/2}}{|-x^2+2y^2-z^2|}$, which depends on η and the estimated location (x, y, z) of the malicious magnet relative to the sensor. Considering the above factors, the attacker can find an acceptable malicious magnet to launch the event-eliminating attack.

5.5.2 Event-Spoofing Attack

In addition to eliminating legitimate OPEN events, attackers can generate fake OPEN alarms using the malicious magnet. In Fig. 5.8, the attacker moves a malicious magnet of the opposite polarity close to the legitimate magnet and contact sensor. As the magnetic field generated by the malicious magnet is destructive to the magnetic field of the legitimate magnet, the MFS at the reed switch's location decreases. If the MFS falls below the default threshold, the sensor triggers an OPEN alarm while the door is still closed. We assume that the y-axis component of the MFV generated by the legitimate magnet at the reed switch is $H_y^{'}$. To change the state of the reed switch, the malicious magnet should be able to generate an MFV with the y-axis component H_y satisfying $|H_y + H_y^{'}| < \eta$, where H_y and $H_y^{'}$ have different signs. If too many false alarms are generated, the user normally considers the contact sensor faulty and temporarily disables it until the alarm service company dispatches a technician for onsite diagnosis, which may take a few days. During this period, the door or window with the disabled sensor becomes a vulnerable intrusion point.

5.6 Battery Depletion Attack

Each contact sensor is powered by a small battery which is expected to last a few years with the low-power communication protocols like Z-Wave. In this section, we present a battery-depletion attack that can deplete the sensor battery quickly and

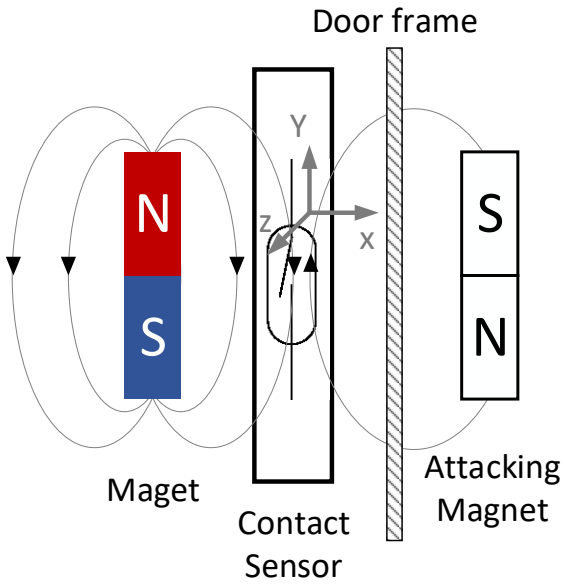
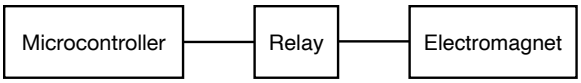


Figure 5.8: Event-spoofing Attack Using a Malicious Magnet of the Opposite Polarity.

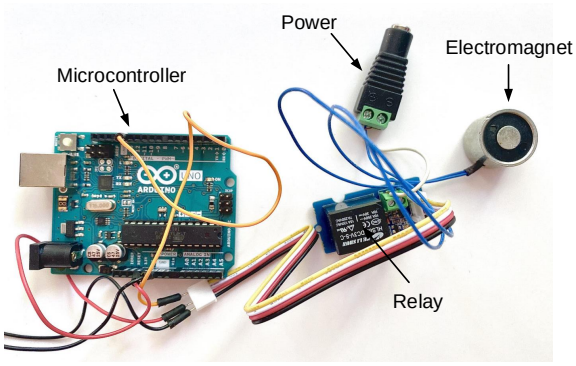
silently. The basic idea is to use an advanced event-spoofing attack to force the sensor to continuously generate and report a large amount of fake OPEN or CLOSE events without arousing the attention of the user, base station, or alarm service company. A contact sensor with a low battery level would periodically send low-battery warnings to the base station, the user's smartphone, and the alarm service company. To avoid receiving too many low-battery warnings pushed by the involved contact sensor, the user often chooses to temporarily disable the contact sensor. It may take the user many days to replace the dead battery, as he may not have a backup one at home or even on travel. So the attacker would have a longer time window to illegally enter the victim's home through the affected door or window.

5.6.1 Automatic Event Spoofing using Electromagnet

In Section 5.5.2, we use a permanent magnet to manually generate fake OPEN events. It is, however, impractical to generate a large amount of events to deplete the



(a) The Block Diagram



(b) The Circuit

Figure 5.9: The Circuit for Battery Depletion Attack.

battery in a short period. To enable automatic event spoofing, we design a system that can generate magnetic signals automatically to control the reed switch's state, as illustrated in Fig. 5.9. The microcontroller can be programmed to generate the ON-OFF square wave which is then used by the relay to control the power supply to the electromagnet. In the ON state, the current goes through a coil of copper wire in the electromagnet and creates a magnetic field; in the OFF state, the electromagnet is turned off to make the magnetic field disappear. In this way, the attacker can trigger the contact sensor to generate OPEN or CLOSE events with any time interval.

5.6.2 Quite Event Spoofing with Jamming

To deplete the sensor battery, attackers need to trigger the contact sensor to generate a large amount of OPEN or OFF events which are normally immediately reported to the base station, the user's smartphone, and the alarm service company. Abnormally manage alarms during a certain period would arouse the attention of the

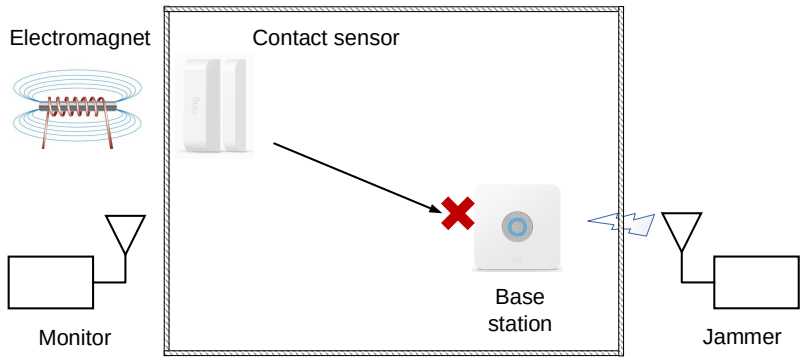


Figure 5.10: Battery Depletion Attack.

user and alarm service company. Therefore, it is necessary for the attacker to jam the channel between the sensor and base station to achieve quick and stealthy battery depletion, as shown in Fig. 5.10.

The reactions of contact sensors to jamming signals depend on the MAC (medium access control) protocol and the specific system implementation. If the system adopts carrier-sense multiple access (CSMA), the sensor waits for a clear channel after detecting the high energy noise before transmitting and may abandon the packet if the channel is always noisy in a certain duration. In contrast, if no CSMA strategy is used in the system, the sensor keeps transmitting regardless of jamming signals in the channel. The contact sensor is not programmable, and the vendor does not want to disclose implementation details. So it is difficult for attackers to infer the specific MAC strategy used in the alarm system when jamming signals exist. Therefore, we aim to devise a generic attack that can work on most alarm systems which may or may not use the carrier-sense MAC strategy. Also, attackers should be able to observe the triggered packets from the contact sensor when jamming signals exist so that they can evaluate the effect of the attack in real time.

To achieve the above goals, the attack in Fig. 5.10 should meet the following requirements. First, the base station cannot decode the packets from the contact

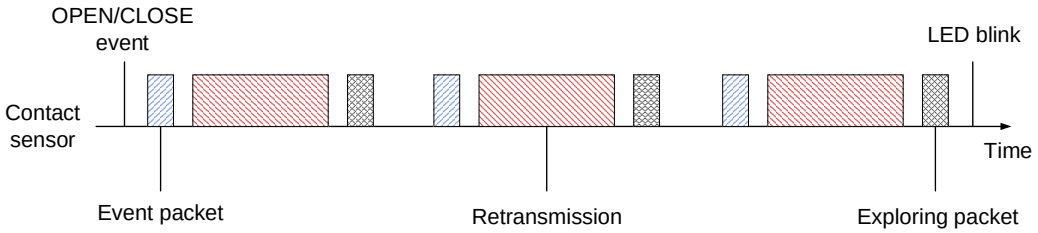


Figure 5.11: Retransmission When ACK is Not Received from the Base Station.

sensor to keep the user and the service provider unaware of the ongoing attack. Second, the noise energy level should be below a threshold so that the sensor can keep transmitting to consume energy even if it uses the carrier-sense MAC strategy. Third, the attacker's sniffer can decode the packets from the sensor, so the SNR at the sniffer should not be too low. So we need to put the jammer close to the base station and far away from both the targeted contact sensor and the sniffer.

The attacker achieves the above goals in a few steps. First, the attacker walks around the house with a handheld sniffer to measure the signals broadcast by the base station around doors and windows. [2] The contact sensor around the location with the minimum signal strength is considered the farthest away from the base station and chosen as the targeted sensor to attack; the sniffer is finally placed there as well. Also, the location with the maximum signal strength is is considered closest to the base station and is chosen as the jammer's location. Next, the attacker gradually decreases the jammer's transmission power from the maximum until the base station starts to respond with an ACK; the previous jamming power level is chosen as the optimal one. Note that the base station should acknowledge any event report, so the attacker can assume that the base station cannot decode anything if no ACK is observed.

[2] A drone can be used for this purpose as well.

After finding the optimal jamming power, the attacker uses the programmed circuit and the retransmission strategy to trigger the contact sensor to transmit as many packets as possible. Fig. 5.11 illustrates the retransmission strategy used in the Ring alarm system. When an OPEN or CLOSE event is triggered, the contact sensor first transmits a packet to report the event. Then it retransmits the original packet eight times if no ACK is received from the base station in a specific period. If still no ACK is returned, it transmits an exploring packet to show that it may lose connection with the base station. The contact sensor repeats the whole process three times, and the LED light blinks after the sensor finishes the whole retransmission process. It takes the sensor about 18 seconds to transmit all the 30 packets in the retransmission process. So we let the microcontroller automatically change the ON-OFF state of the electromagnet every 18 seconds to induce the energy-consuming retransmission process. The battery of the contact sensor failed in 43 hours in contrast to the expected battery lifetime of a few years.

With the presence of jamming signals, the base station cannot receive any event report from any sensor instead of just the one targeted by the attacker. Since the user may notice missing events from the sensor on his smartphone, the attacker can launch the attack in multiple noncontinuous periods, e.g., at late night or when the user is not at home.

5.7 Defenses

We can have the following countermeasures to thwart the above attacks. First, the attacks are based on the reed switch's vulnerability that it cannot differentiate the MFS changes caused by the real OPEN or CLOSE action from the attacker's interference. Therefore, we can add an accelerometer to the contact sensor to detect the continuous OPEN or CLOSE action. In particular, we assume that the accelerometer

is static and that the reed switch is in the CLOSE state at t_1. After a time delay δ, the reed switch transfers to the OPEN state. We can calculate the sensor's displacement as $\vec{D} = \int_{t_1}^{t_1+\delta} \vec{v}(t)dt = \int_{t_1}^{t_1+\delta} \int_{t_1}^{t_v} \vec{a}(t_a)dt_a dt_v$, where $\vec{v}(t)$ and $\vec{a}(t)$ denote the sensor's speed and acceleration at time t. Since the displacement is very small, the sensor-magnet distance $d_{\mathsf{s\text{-}m}} \approx \|\vec{D}\|$. We claim that the sensor has been separated from the magnet if $d_{\mathsf{s\text{-}m}}$ exceeds a critical value that determines the reed switch's state change. If the reed switch triggers an OPEN event while the accelerometer cannot detect the corresponding action, the event may be fake. If the reed switch does not trigger any event while the accelerometer detects the action, an event may have been eliminated by attackers.

Second, the attacker cannot launch the attacks stealthily if the base station can detect the jamming signals. There are some techniques [79, 80] to detect continuous jamming signals. One simple solution is that the base station continuously monitors the energy level in the frequency range. In each time period, if the base station detects a large percentage of time with the energy level above a threshold, there can be an ongoing jamming attack. This defense can limit the impact of the attack but would fail if the attacker know the jamming-detection parameters of the system.

5.8 Evaluation

In this section, we evaluate the efficacy of the proposed attacks with a popular Ring home alarm system.

5.8.1 Localization

We use iPhone 6S as a magnetic signal detector to localize the legitimate magnet paired with the contact sensor. The magnetometer in iPhone 6S is 2.2 cm from the

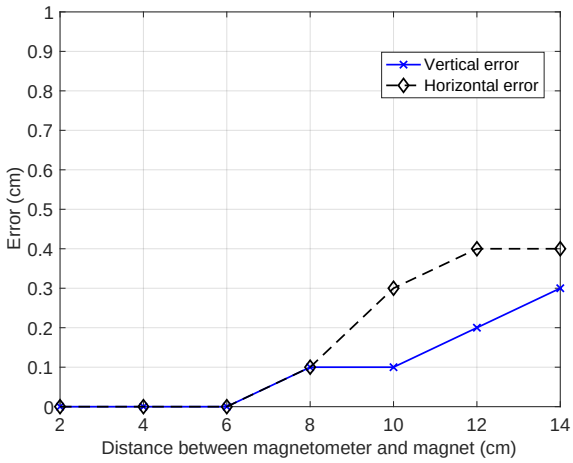


Figure 5.12: The Localization Error with Different Distances between Legitimate Magnet and the Magnetometer.

left frame and 1.6 cm from the upper frame. In addition, the Ring system uses a 3cm $\times$ 0.35cm $\times$ 0.8cm bar magnet in the contact sensor illustrated in Fig. 5.3.

We first evaluate the horizontal and vertical localization accuracy of the magnet with different distance between the reed switch in the contact sensor and the magnetometer in the phone, as shown in Fig. 5.12. For each distance configuration, we measure the magnet's position three times and then calculate the average. We get localization errors of under 0.5cm for all the distance settings and higher precision when the magnetometer is close to the magnet. Then we test the localization accuracy when there is a plank of 3.2cm thick acting as a window or door between the magnet and magnetometer. We get localization errors of 0.1cm for both horizontal and vertical accuracy. Therefore, the wood material between the magnetometer and magnet has little impact on the localization error. Note that we need to remove the background magnetic field from the readings before magnet localization.

5.8.2 Event Eliminating and Spoofing

We first evaluate the MFS threshold η that changes the reed switch's state because η is used to find acceptable malicious magnets in both event-eliminating and event-spoofing attacks. We measure η by attaching the reed switch to the magnetometer along the y-axis and then use a bar magnet to trigger the reed switch. The bar magnet is placed parallel to the reed switch as illustrated in Fig. 5.7 so that it can generate the MFV along the y-axis.

Fig. 5.13 illustrates the y-axis component of the MFV generated by the bar magnet at the reed switch. Initially, the magnet is far away from the reed switch; so the reading is small, and the reed switch is open. Then we move the magnet close to the reed switch until hearing a click which indicates that the two contacts have snapped together. At this time, the y-axis reading of the magnetometer reaches the maximum which represents the threshold for the state change as the red circle in Fig. 5.13. When we move the bar magnet away, the reading decreases until the contacts are separated from each other. We repeat the above process five times and get an average threshold of 752.92 μT. We also use a magnet with an opposite polarity to do this experiment and get an average threshold of 741.54 μT. The difference may be caused by measurement errors. When we place the bar magnet vertical to the reed switch, it cannot activate the reed switch since it cannot generate the MFV parallel to the reed switch.

We also evaluate the critical distance between the reed switch and malicious magnets of different strength to change the state in both event-eliminating and event-spoofing attacks. In particular, we build magnets of different strength by connecting different numbers of cylinder magnets together. As demonstrated in Fig. 5.14, the distance increases with the strength of the magnet so that the stronger magnet can

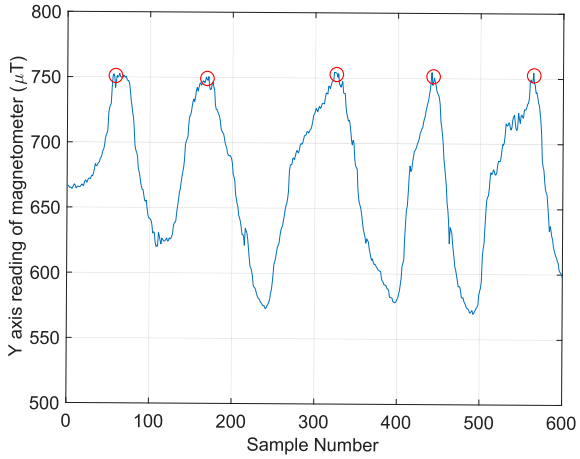


Figure 5.13: Event-eliminating Attack.

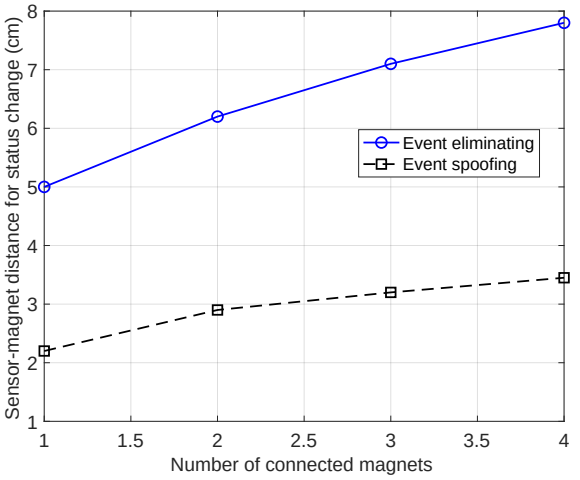


Figure 5.14: Distance to Change Status of Reed Switch.

interfere with the reed switch from a position farther away. Fig. 5.14 also shows that in the event-spoofing attack, the malicious magnet needs to be closer to the reed switch to cause state changes, as the malicious magnet needs to generate a stronger magnetic field to offset the one generated by the legitimate magnet. This means that the required magnetic strength for the reed switch's state change is low, but the cost to offset the legitimate magnetic field is high.

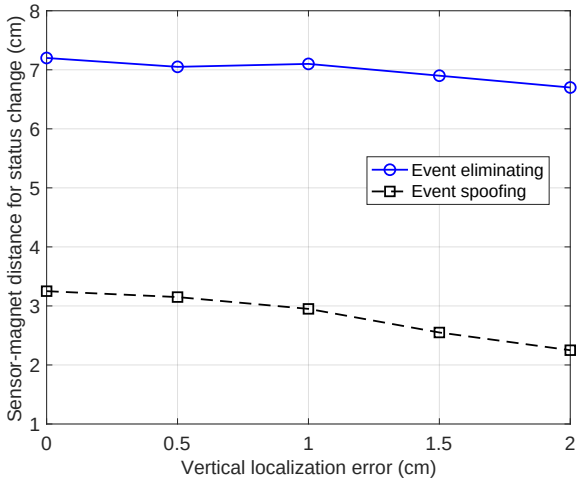
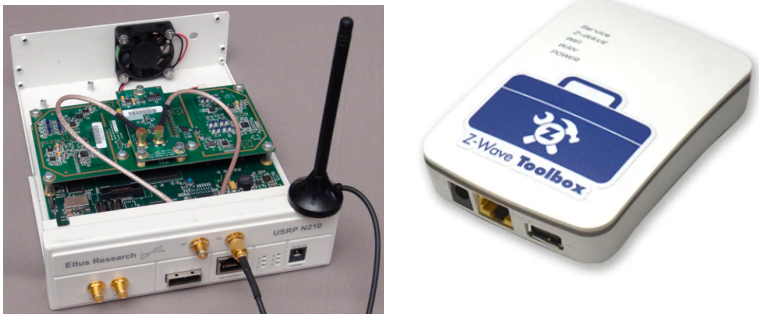


Figure 5.15: Distance to Change Status of Reed Switch with Different Localization Errors.

We then evaluate the impact of the sensor localization error on the critical distance. As we can see from Fig. 5.15, the critical distance in both the event-eliminating and event-spoofing attacks decreases slowly with the localization error. Therefore, the attacker needs to move the malicious magnet closer to generate stronger magnetic field when he cannot determine the sensor's location accurately. Also, the small errors in Fig. 5.12 has little impact on the critical distance. Similar to Fig. 5.14, the critical distance in event-spoofing attack is smaller as well.

5.8.3 Battery Depletion Attack

We evaluate the battery depletion attack with a fully furnished 10m $\times$ 4m apartment room as illustrated in Fig. 5.17. Initially, the base station is close to position F, and the target contact sensor is close to position A. We use a USRP N210 (Fig 5.16a) as the jammer to transmit Gaussian noise in the same Z-Wave communication frequency. The actual distance between the base station and the jammer is about 2 meters. We use a Z-Wave Toolbox (Fig 5.16b) as a sniffer which is placed close



(a) USRP N210 (b) Z-Wave Toolbox

Figure 5.16: Equipment for the Battery Depletion Attack.

to position A to measure the communication between the contact sensor and base station.

We first set both the jammer's initial transmission power and antenna gain to its maxima. Then, we decrease the jammer's antenna gain gradually and monitor the number of packets transmitted by the contact sensor when triggering an OPEN event, as illustrated in Fig. 5.18. At the beginning, only the sniffer can receive the packets because it is far away from the jammer. Note that the sensor transmits 30 packets if no ACK is received from the base station, but the sniffer only receives about 18 packets. The sniffer may fail to decode the rest packets because of low SNR or because the sensor may not transmit when detecting high energy in the channel. With the decrease of the antenna gain, the sniffer starts to receive more packets from the sensor with a maximum of 30. When the antenna gain falls below -4.5 dBm in Fig. 5.18, the base station starts to receive packets from the sensor after a few retransmissions. When the antenna gain is even lower, the communication between the base station and contact sensor returns to normal.

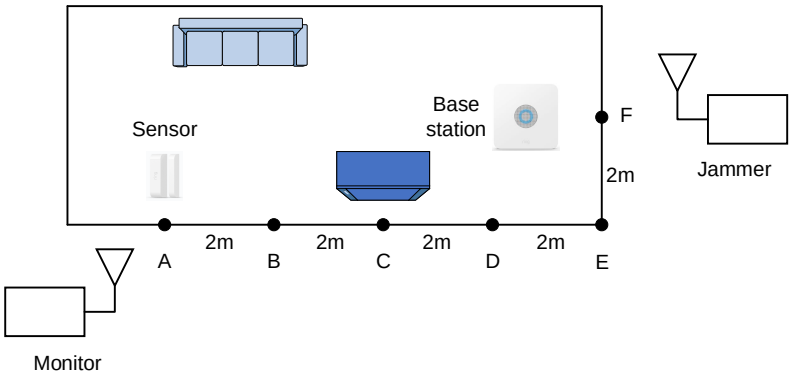


Figure 5.17: Experiment Scenario for the Battery Depletion Attack.

We call the minimum gain that can interrupt the communication between the sensor and base station *critical gain*. For example, the critical grain in Fig. 5.18 is -4.5 dBm. All the gains in the range of [-4.5, 20] is a feasible to launch the battery-depletion attack without alarming the base station. The gains in the range of [-4.5, -1.5] are optimal because the attacker can trigger the sensor to generate more packets with less energy. We can observe similar results in Fig. 5.19 when CLOSE events are triggered. When the antenna gain is set to the optimal range, we depleted the sensor battery in about 43 hours.

We also evaluate the impact of the relative positions of the jammer, base station, and contact sensor. From Fig. 5.20, we can see that the critical gain increases when we move the base station away form the jammer and keep others unchanged. Therefore, it is important to place the jammer close to the base station to achieve good jamming performance. Fig. 5.21 illustrates the critical gain when we move the contact sensor and monitor close to the jammer and keep the base station in position A. We can see that the change of critical gain is very small when the relative position between the base station and jammer is fixed. The channel between position C and position

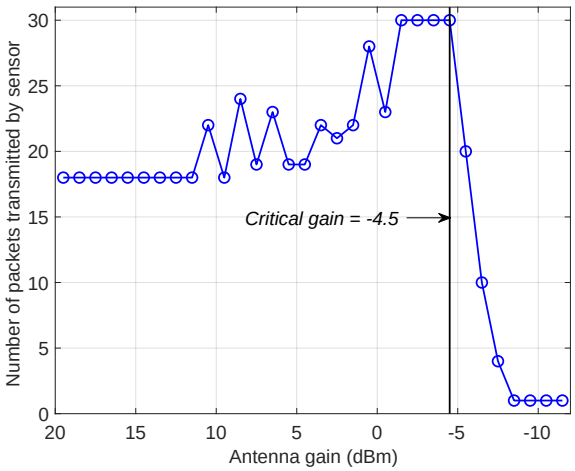


Figure 5.18: Number of Packets Transmitted by the Sensor When the Attacker Triggers an OPEN Event.

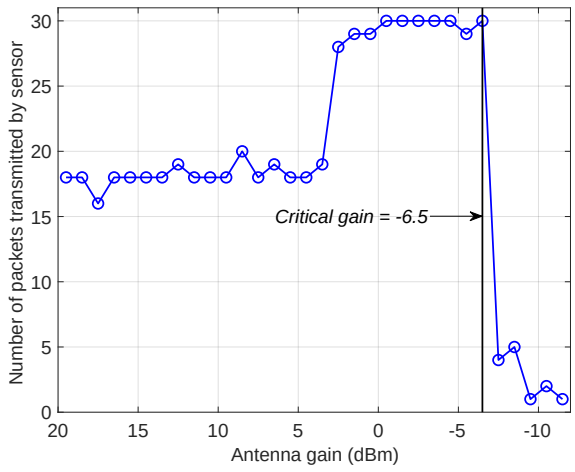


Figure 5.19: Number of Packets Transmitted by the Sensor When the Attacker Trigger an CLOSE Event.

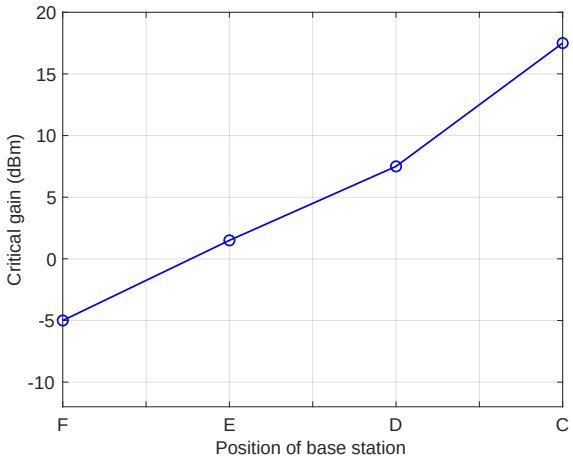


Figure 5.20: Critical Gain When Changing the Position of the Base Station.

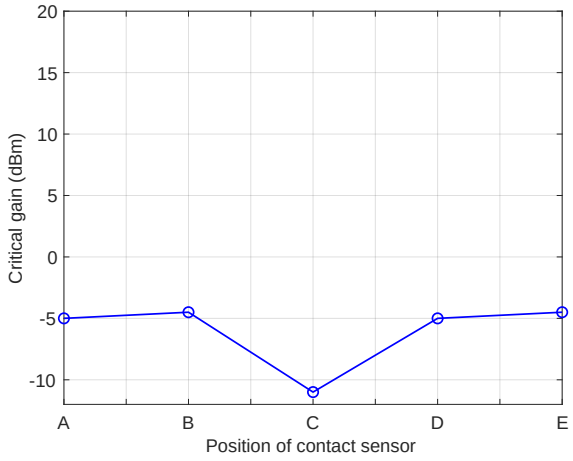


Figure 5.21: Critical Gain When Changing the Position of the Contact Sensor.

F is almost blocked by a piece of metal furniture, so the critical gain is low when the sensor is in position C.

5.9 Related Work

This section first discusses some most germane work on the security of smart home devices including home alarm systems. Then we introduce some prior work on battery-depletion attacks and jamming attacks in wireless sensor networks (WSNs).

We first discuss some attacks on home IoT systems using RF techniques. Picod *et al.* [81] present a software-defined radio framework *Scapy* for packet manipulation and security assessment and test it in a Z-Wave network. Lamb [15] presents jamming and replay attacks to eliminate legitimate alarms and cause false alarms for multiple home alarm systems. Fouladi and Ghanoun [16] use a flaw in Z-Wave protocol to reset the encryption key to a chosen value so that the attacker can inject unauthorized commands. In [82], the authors launch a sinkhole attack by deploying a malware to a legitimate device of a home Zigbee network. Rouch *et al.* [17] and Fuller and Ramsey [18] introduce techniques to inject a fake controller (base station) to the network to control home IoT devices. Badenhop *et al.*[83] provide attacks on routing protocols of Z-Wave networks. None of the above work considers possible attacks utilizing magnetic interference with the reed switch in home IoT systems.

The access point in the home IoT system also provides opportunities for attackers. After gaining access, the attacker may control the whole network. Crowley *et al.* [77] find several vulnerabilities that expose sensitive information from a Z-Wave gateway controller. By using these vulnerabilities, the attacker can create a backdoor account on the gateway. Barcena and Wueest [78] poison the gateway Address Resolution Protocol to redirect gateway firmware update requests to their own malicious server. After modifying the firmware, the gateway receives the malicious firmware as a legitimate update giving the attacker full control.

There are also some research on protecting smart home devices. Homonit [84] monitors the encrypted network traffic to detect anomaly for Samsung SmartThings. Brown *et al.* [85] jam unsolicited messages for a home automation system without impairing legitimate transmissions in neighbouring houses. We can find some countermeasures for the malicious packets injection attack on Z-Wave in [86] [87].

Battery-depletion attacks have received attention in wireless sensor networks. In [88], the authors present routing-layer attacks which exhaust energy by specifying far longer routing paths and forcing packet processing at remote network positions. Ghost-in-ZigBee [89] depletes nodes' energy in a ZigBee network by constructing bogus messages to lure nodes to do superfluous security-related computations. Raymond *et al.* [90] analyze the effects of Denial-of-Sleep attacks in WSN by considering the attacker's knowledge on the MAC protocol. In contrast, since the packets are triggered by reed switches in the home alarm network, we use magnetic signals to interfere with the reed switch to generate more communications and deplete the sensor battery.

There is also extensive research [91] on jamming techniques and countermeasures in WSNs. Xu *et al.* [92] study the feasibility of launching and detecting jamming attacks at the MAC layer. Li *et al.* [93] investigate the optimal jamming and defense techniques under energy constraints in WSNs. We jam the base station from the physical layer so that the alarm service provider and the system user are unaware of the battery-depletion attack.

Chapter 6

CONCLUSION AND FUTURE WORK

In this report, we work on two lines related to security and usability of mobile and wireless systems. Specifically, we present an automatic locking system for mobile devices against data theft, one systematic study of security issues in crowdsourced indoor positioning systems, one usable indoor navigation system, and practical attacks on home alarm IoT systems. In Chapter 2, we present a novel system called *iLock*, a secure and usable defense against data theft on a lost/stolen mobile device. iLock automatically, quickly, and accurately detects the user's physical separation from his/her device. Once significant physical separation is detected, iLock immediately locks the device to thwart data theft. Relying on acoustic signals, iLock can be deployed on most COTS mobile devices with standard built-in microphones and speakers. Extensive experiments on Samsung Galaxy S5 confirmed the high efficacy of iLock with negligible false positives and negatives. In Chapter 3, we presented the first systematic study about the security issues in crowdsourced WiFi-based IPSes. We presented three attacks and evaluated their performance in a prototype system. We also designed an algorithm based on novel temporal and spatial trustworthiness metrics to generate high-fidelity fingerprint databases even if most crowdsourced RSS traces are fake. Thorough experiments confirmed that our algorithm has strong resilience to the reported attacks. Both the attacks and defenses developed in this report can be easily extended to other crowdsourced IPSes. In Chapter 4, we present IndoorWaze, the first indoor navigation system which can automatically construct an accurate labeled indoor floor plan. We prototyped IndoorWaze on Android smartphones and evaluated it in a large shopping mall. The experiment confirmed the high

efficacy and usability of IndoorWaze. In Chapter 5, we present some attacks targeting home alarm systems by interfering with reed switch in the contact sensor using malicious magnetic signals. By generating specific magnetic signals, the attacker can eliminate the legitimate alarms and cause false alarms. Also, we combine jamming techniques and successfully drain the sensor battery in 43 hours without notice of the service provider and the home owner. We also provide potential countermeasures on the attacks. Extensive experiments on real indoor environment show that our attacks are highly practical.

For future work, we plan to advance further along these two lines. First, we would like to investigate more solutions for automatic locking of mobile devices. A candidate approach is to rely on the Doppler effect caused by user movements. Also, ilock may be implemented based on WiFi or Bluetooth signals rather than acoustic signals. Second, we would like to explore techniques to achieve privacy-preserving crowdsourced indoor positioning systems. The system is supposed to protect users' location privacy in both training and operating phases and provide acceptable localization services at the same time. Third, we would like to combine vision-based techniques and IMU sensors to provide more accurate and user-friendly navigation service for indoor users. Finally, we would like to investigate more security flaws of home alarm IoT systems and the corresponding countermeasures.

REFERENCES

[1] "Cisco visual networking index global mobile data traffic forecast update 2014-2019," http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html.

[2] http://www.channelpronetwork.com/article/mobile-device-security-startling-statistics-data-loss-and-data-breachesl.

[3] "Kaspersky lab survey," http://www.kaspersky.com/about/news/virus/2015/Quarter-of-Users-Do-Not-Understand-the-Risks-of-Mobile-Cyberthreats.

[4] S. Mare, A. Markham, C. Cornelius, R. Peterson, and D. Kotz, "Zebra: Zero-effort bilateral recurring authentication," in *IEEE S&P*, San Jose, CA, May 2014.

[5] L. Li, X. Zhao, and G. Xue, "Unobservable re-authentication for smartphones," in *NDSS*, San Diego, USA, February 2013.

[6] H. Khan, A. Atwater, and U. Hengartner, "Itus: An implicit authentication framework for android," in *ACM MobiCom*, Maui, HI, September 2014.

[7] "Wi-fi indoor location in retail worth $2.5 billion by 2020." [Online]. Available: https://www.abiresearch.com/press/wi-fi-indoor-location-retail-worth-25-billion-2020

[8] A. Rai, K. Chintalapudi, V. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *ACM MobiCom*, Istanbul, Turkey, August 2012.

[9] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *ACM MobiSys*, Low Wood Bay, UK, June 2012.

[10] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: wireless indoor localization with little human intervention," in *ACM MobiCom*, Istanbul, Turkey, August 2012.

[11] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang, "Walkie-markie: indoor pathway mapping made easy," in *USENIX NSDI*, Lombard, IL, April 2013.

[12] D. Vasisht, S. Kumar, and D. Katabi, "Decimeter-Level localization with a single WiFi access point," in *USENIX NSDI*, Santa Clara, CA, Mar. 2016.

[13] S. Kumar, S. Gil, D. Katabi, and D. Rus, "Accurate indoor localization with zero start-up cost," in *ACM MobiCom*, Maui, HI, Sep. 2014.

[14] "Home security system market." [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/home-security-system-market-205573901.html.

[15] L. Lamb, "Home insecurity: No alarms, false alarms, and sigint," in *Black Hat USA*, Las Vegas, NV, Aug. 2014.

[16] B. Fouladi and S. Ghanoun, "Security evaluation of the z-wave wireless protocol," in *Black Hat USA*, Las Vegas, NV, July 2013.

[17] L. Rouch, J. François, F. Beck, and A. Lahmadi, "A universal controller to take over a z-wave network," in *Black Hat Europe*, London, United Kingdom, Dec. 2017.

[18] J. Fuller and B. Ramsey, "Rogue z-wave controllers: A persistent attack channel," in *IEEE LCN*, Clearwater Beach, FL, Oct. 2015.

[19] C. Peng, G. Shen, Y. Zhang, Y. Li, and K. Tan, "Beepbeep: a high accuracy acoustic ranging system using cots mobile devices," in *ACM SenSys*, Sydney, Australia, November 2007.

[20] B. Mahafza, *Radar Systems Analysis and Design Using MATLAB Third Edition*. CRC press, 2013.

[21] P. Zhou, M. Li, and G. Shen, "Use it free: Instantly knowing your phone attitude," in *ACM MobiCom*, Maui, HI, September 2014.

[22] Y.-C. Tung and K. Shin, "Echotag: accurate infrastructure-free indoor location tagging with smartphones," in *ACM MobiCom*, Paris, France, September 2015.

[23] T. Vu, A. Baid, S. Gao, M. Gruteser, R. Howard, J. Lindqvist, P. Spasojevic, and J. Walling, "Distinguishing users with capacitive touch communication," in *ACM MobiCom*, Istanbul, Turkey, August 2012.

[24] "http://gizmodo.com/hackers-iphone-5s-fingerprint-security-is-notsecure-1367817697."

[25] Y. Chen, J. Sun, R. Zhang, and Y. Zhang, "Your song your way: Rhythm-based two-factor authentication for multi-touch mobile devices," in *IEEE INFOCOM*, Hong Kong, China, April 2015.

[26] J. Sun, R. Zhang, J. Zhang, and Y. Zhang, "Touchin: Sightless two-factor authentication on multi-touch mobile devices," in *IEEE CNS*, San Francisco, CA, October 2014.

[27] E. Maiorana, P. Campisi, N. González-Carballo, and A. Neri, "Keystroke dynamics authentication for mobile phones," in *ACM SAC*, TaiChung, Taiwan, March 2011.

[28] F. Monrose, M. Reiter, and S. Wetzel, "Password hardening based on keystroke dynamics," *International Journal of Information Security*, vol. 1, no. 2, pp. 69–83, 2002.

[29] M. Shahzad, A. Liu, and A. Samuel, "Secure unlocking of mobile touch screen devices by simple gestures: you can see it but you can not do it," in *ACM MobiCom*, Miami, FL, September 2013.

[30] N. Sae-Bae, K. Ahmed, K. Isbister, and N. Memon, "Biometric-rich gestures: a novel approach to authentication on multi-touch devices," in *ACM CHI*, Austin, TX, May 2012.

[31] D. Gafurov, K. Helkala, and T. Søndrol, "Biometric gait authentication using accelerometer sensor," *Journal of Computers*, vol. 1, no. 7, pp. 51–59, 2006.

[32] J. Kwapisz, G. Weiss, and S. Moore, "Cell phone-based biometric identification," in *IEEE BTAS*, Washington, D.C., September 2010.

[33] O. Huhta, P. Shrestha, S. Udar, M. Juuti, N. Saxena, and N. Asokan, "Pitfalls in designing zero-effort deauthentication: Opportunistic human observation attacks," in *NDSS*, San Diego, CA, February 2015.

[34] M. Frank, R. Biedert, E.-D. Ma, I. Martinovic, and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 136–148, 2013.

[35] W. Shi, F. Yang, Y. Jiang, F. Yang, and Y. Xiong, "Senguard: Passive user identification on smartphones using multiple sensors," in *IEEE WiMob*, Shanghai, China, October 2011.

[36] T. Feng, Z. Liu, K.-A. Kwon, W. Shi, B. Carbunar, Y. Jiang, and N. Nguyen, "Continuous mobile authentication using touchscreen gestures," in *IEEE HST*, Waltham, MA, November 2012.

[37] F. Adib, Z. Kabelac, D. Katabi, and R. Miller, "3d tracking via body radio reflections," in *USENIX NSDI*, Seattle, WA, April 2014.

[38] F. Adib, Z. Kabelac, and D. Katabi, "Multi-person localization via rf body reflections," in *USENIX NSDI*, Oakland, CA, May 2015.

[39] R. Nandakumar, S. Gollakota, and N. Watson, "Contactless sleep apnea detection on smartphones," in *ACM MobiSys*, Florence, Italy, May 2015.

[40] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, "Fingerio: Using active sonar for fine-grained finger tracking," in *ACM CHI*, San Jose, CA, May 2016.

[41] R. Nandakumar, K. Chintalapudi, and V. Padmanabhan, "Centaur: locating devices in an office environment," in *ACM MobiCom*, Istanbul, Turkey, August 2012.

[42] K. Liu, X. Liu, and X. Li, "Guoguo: Enabling fine-grained indoor localization via smartphone," in *ACM MobiSys*, Taipei, Taiwan, June 2013.

[43] P. Bahl and V. Padmanabhan, "Radar: An in-building rf-based user location and tracking system," in *IEEE INFOCOM*, Tel Aviv, Israel, March 2000.

[44] M. Youssef and A. Agrawala, "The horus wlan location determination system," in *ACM MobiSys*, Seattle, WA, June 2005.

[45] S. Sen, B. Radunovic, R. Choudhury, and T. Minka, "You are facing the mona lisa: spot localization using phy layer information," in *ACM MobiSys*, Low Wood Bay, UK, June 2012.

[46] M. Azizyan, I. Constandache, and R. Choudhury, "Surroundsense: mobile phone localization via ambience fingerprinting," in *ACM MobiCom*, Beijing, China, September 2009.

[47] H. Lim, L.-C. Kung, J. Hou, and H. Luo, "Zero-configuration indoor localization over ieee 802.11 wireless infrastructure," *Wireless Networks*, vol. 16, no. 2, pp. 405–420, 2010.

[48] K. Chintalapudi, A. Iyer, and V. Padmanabhan, "Indoor localization without the pain," in *ACM MobiCom*, Chicago, IL, September 2010.

[49] D. Turner, S. Savage, and A. Snoeren, "On the empirical performance of self-calibrating wifi location systems," in *IEEE LCN*, Bonn, Germany, October 2011.

[50] J. Xiong and K. Jamieson, "Arraytrack: A fine-grained indoor location system," in *USENIX NSDI*, Lombard, IL, April 2013.

[51] Z. Zhang, X. Zhou, W. Zhang, Y. Zhang, G. Wang, B. Zhao, and H. Zheng, "I am the antenna: accurate outdoor ap location using smartphones," in *ACM MobiCom*, Las Vegas, NV, September 2011.

[52] M. Youssef, A. Youssef, C. Rieger, U. Shankar, and A. Agrawala, "Pinpoint: An asynchronous time-based location determination system," in *ACM MobiSys*, Uppsala, Sweden, June 2006.

[53] N. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *ACM MobiCom*, Boston, MA, August 2000.

[54] B. Xie, K. Chen, G. Tan, M. Lu, Y. Liu, J. Wu, and T. He, "Lips: A light intensity–based positioning system for indoor environments," *ACM Transactions on Sensor Networks*, vol. 12, no. 4, p. 28, 2016.

[55] B. Xie, G. Tan, and T. He, "Spinlight: A high accuracy and robust light positioning system for indoor applications," in *ACM SenSys*, Seoul, South Korea, November 2015.

[56] N. Rajagopal, P. Lazik, and A. Rowe, "Visual light landmarks for mobile devices," in *ACM/IEEE IPSN*, Berlin, Germany, April 2014.

[57] Y.-S. Kuo, P. Pannuto, K.-J. Hsiao, and P. Dutta, "Luxapose: Indoor positioning with mobile phones and visible light," in *ACM MobiCom*, Maui, HI, September 2014.

[58] Z. Yang, Z. Wang, J. Zhang, C. Huang, and Q. Zhang, "Wearables can afford: Light-weight indoor positioning with visible light," in *ACM MobiSys*, Paris, France, September 2015.

[59] C. Zhang and X. Zhang, "Litell: Robust indoor localization using unmodified light fixtures," in *ACM MobiCom*, New York City, NY, October 2016.

[60] B. Ferris, D. Fox, and N. Lawrence, "Wifi-slam using gaussian process latent variable models," in *IJCAI*, Hyderabad, India, January 2007.

[61] R. Zhang, J. Zhang, Y. Zhang, and C. Zhang, "Secure crowdsourcing-based cooperative spectrum sensing," in *INFOCOM*, Turin, Italy, April 2013.

[62] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Zhao, "Defending against sybil devices in crowdsourced mapping services," in *ACM MobiSys*, Singapore, Singapore, June 2016.

[63] Y. Zheng, G. Shen, L. Li, C. Zhao, M. Li, and F. Zhao, "Travi-Navi: Self-deployable indoor navigation system," in *ACM MobiCom*, Maui, HI, Sep. 2014.

[64] M. Elhamshary and M. Youssef, "SemSense: Automatic construction of semantic indoor floorplans," in *IEEE IPIN*, Calgary, Canada, Oct. 2015.

[65] T. Li, Y. Chen, R. Zhang, Y. Zhang, and T. Hedgpeth, "Secure crowdsourced indoor positioning systems," in *IEEE INFOCOM*, Honolulu, HI, Apr. 2018.

[66] L. Yuan, Y. Hu, Y. Li, R. Zhang, Y. Zhang, and T. Hedgpeth, "Secure RSS-fingerprint-based indoor positioning: attacks and countermeasures," in *IEEE CNS*, Beijing, China, May 2018.

[67] D. Yang, G. Xue, X. Fang, and J. Tang, "Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1732–1744, June 2016.

[68] "Understanding wifi signal strength." [Online]. Available: https://www.metageek.com/training/resources/wifi-signal-strength-basics.html.

[69] Y. Shu, K. Shin, T. He, and J. Chen, "Last-mile navigation using smartphones," in *ACM MobiCom*, Paris, France, Sep. 2015.

[70] T. Riehle, S. Anderson, P. Lichter, N. Giudice, S. Sheikh, R. Knuesel, D. Kollmann, and D. Hedin, "Indoor magnetic navigation for the blind," in *IEEE EMBS*, San Diego, CA, Aug. 2012.

[71] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Bian, and X. Li, "Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing," in *ACM MobiCom*, Maui, HI, Sep. 2014.

[72] S. Chen, M. Li, K. Ren, and C. Qiao, "CrowdMap: Accurate reconstruction of indoor floor plans from crowdsourced sensor-rich videos," in *IEEE ICDCS*, Columbus, OH, June 2015.

[73] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao, "A reliable and accurate indoor localization method using phone inertial sensors," in *ACM UbiComp*, Pittsburgh, PA, Sep. 2012.

[74] M. Alzantot and M. Youssef, "CrowdInside: Automatic construction of indoor floorplans," in *ACM GIS*, Redondo Beach, CA, Nov. 2012.

[75] Y. Jiang, Y. Xiang, X. Pan, K. Li, Q. Lv, R. Dick, L. Shang, and M. Hannigan, "Hallway based automatic indoor floorplan construction using room fingerprints," in *ACM UbiComp*, Zurich, Switzerland, Sep. 2013.

[76] H. Shin, Y. Chon, and H. Cha, "Unsupervised construction of an indoor floor plan using a smartphone," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 42, no. 6, pp. 889–898, Oct. 2012.

[77] D. Crowley, D. Bryan, and J. Savage, "Home invasion v2. 0-attacking network-controlled hardware," in *Black Hat USA*, Las Vegas, NV, July 2013.

[78] M. Barcena and C. Wueest, "Insecurity in the internet of things," *Security Response*, Mar. 2015.

[79] K. Grover, A. Lim, and Q. Yang, "Jamming and anti-jamming techniques in wireless networks: a survey," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 17, no. 4, pp. 197–215, 2014.

[80] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A survey on jamming attacks and countermeasures in wsns," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, pp. 42–56, 2009.

[81] J. Picod, A. Lebrun, and J. Demay, "Bringing software defined radio to the penetration testing community," in *Black Hat USA*, Las Vegas, NV, Aug. 2014.

[82] L. Coppolino, V. DAlessandro, S. DAntonio, L. Levy, and L. Romano, "My smart home is under attack," in *IEEE CSE*, Porto, Portugal, Oct. 2015.

[83] C. Badenhop, S. Graham, B. Ramsey, B. Mullins, and L. Mailloux, "The z-wave routing protocol and its security implications," *Computers & Security*, vol. 68, pp. 112–129, July 2017.

[84] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang, and H. Zhu, "Homonit: Monitoring smart home apps from encrypted traffic," in *ACM CCS*, Toronto, Canada, Oct. 2018.

[85] J. Brown, I. Bagci, A. King, and U. Roedig, "Defend your home! jamming unsolicited messages in the smart home," in *ACM HotWiSec*, Budapest, Hungary, Apr. 2013.

[86] J. Fuller, B. Ramsey, M. Rice, and J. Pecarina, "Misuse-based detection of z-wave network attacks," *Computers & Security*, vol. 64, pp. 44–58, Oct. 2017.

[87] J. Fuller, B. Ramsey, J. Pecarina, and M. Rice, "Wireless intrusion detection of covert channel attacks in itu-t g. 9959-based networks," in *ICCWS*, Boston, MA, Mar. 2016.

[88] E. Vasserman and N. Hopper, "Vampire attacks: draining life from wireless ad hoc sensor networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 2, pp. 318–332, Feb. 2011.

[89] X. Cao, D. M. Shila, Y. Cheng, Z. Yang, Y. Zhou, and J. Chen, "Ghost-in-zigbee: Energy depletion attack on zigbee-based wireless networks," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 816–829, Oct. 2016.

[90] D. Raymond, R. Marchany, M. Brownfield, and S. Midkiff, "Effects of denial-of-sleep attacks on wireless sensor network mac protocols," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 1, pp. 367–380, Jan. 2009.

[91] A. Mpitziopoulos, D. Gavalas, C. Konstantopoulos, and G. Pantziou, "A survey on jamming attacks and countermeasures in wsns," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 4, pp. 42–56, Dec. 2009.

[92] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," in *ACM MobiHoc*, Urbana-Champaign, IL, May 2005.

[93] M. Li, I. Koutsopoulos, and R. Poovendran, "Optimal jamming attack strategies and network defense policies in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 8, pp. 1119–1133, Aug. 2010.