ABSTRACT

LOCATION RELIABILITY AND GAMIFICATION MECHANISMS FOR MOBILE CROWD SENSING

by
Manoop Talasila

People-centric sensing with smart phones can be used for large scale sensing of the physical world by leveraging the sensors on the phones. This new type of sensing can be a scalable and cost-effective alternative to deploying static wireless sensor networks for dense sensing coverage across large areas. However, mobile people-centric sensing has two main issues: 1) Data reliability in sensed data and 2) Incentives for participants. To study these issues, this dissertation designs and develops McSense, a mobile crowd sensing system which provides monetary and social incentives to users.

This dissertation proposes and evaluates two protocols for location reliability as a step toward achieving data reliability in sensed data, namely, ILR (Improving Location Reliability) and LINK (Location authentication through Immediate Neighbors Knowledge). ILR is a scheme which improves the location reliability of mobile crowd sensed data with minimal human efforts based on location validation using photo tasks and expanding the trust to nearby data points using periodic Bluetooth scanning. LINK is a location authentication protocol working independent of wireless carriers, in which nearby users help authenticate each other's location claims using Bluetooth communication. The results of experiments done on Android phones show that the proposed protocols are capable of detecting a significant percentage of the malicious users claiming false location. Furthermore, simulations with the LINK protocol demonstrate that LINK can effectively thwart a number of colluding user attacks.

This dissertation also proposes a mobile sensing game which helps collect crowd sensing data by incentivizing smart phone users to play sensing games on their phones. We design and implement a first person shooter sensing game, “Alien vs. Mobile User”, which employs techniques to attract users to unpopular regions. The user study results show that mobile gaming can be a successful alternative to micro-payments for fast and efficient area coverage in crowd sensing. It is observed that the proposed game design succeeds in achieving good player engagement.

LOCATION RELIABILITY AND GAMIFICATION MECHANISMS FOR MOBILE CROWD SENSING

by
Manoop Talasila

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

January 2015

APPROVAL PAGE

LOCATION RELIABILITY AND GAMIFICATION MECHANISMS FOR MOBILE CROWD SENSING

Manoop Talasila

BIOGRAPHICAL SKETCH

Author:	Manoop Talasila
Degree:	Doctor of Philosophy
Date:	January 2015
Date of Birth:	November 11, 1982
Place of Birth:	Musunuru, Andhra Pradesh, India

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, New Jersey, 2015
- Master of Science in Computer Science,
  Western Kentucky University, Bowling Green, Kentucky, 2006
- Bachelor of Technology in Electrical & Electronics Engineering,
  Jawaharlal Nehru Technological University, India, 2004

Major: Computer Science

Presentations and Publications:

M. Talasila, R. Curtmola, and C. Borcea, "Alien Invasion Enables Crowdsensing", *under submission, IEEE Pervasive Computing Magazine*, 2014

M. Talasila, R. Curtmola, and C. Borcea, "Alien vs. Mobile User Game: Fast and Efficient Area Coverage in Crowdsensing", *in Sixth International Conference on Mobile Computing, Applications and Services (MobiCASE '14)*, November 2014

M. Talasila, R. Curtmola, and C. Borcea, "Mobile Crowd Sensing", *Chapter in the Handbook of Sensor Networking: Advanced Technologies and Applications, CRC Press*, Boca Raton, FL, 2014

M. Talasila, R. Curtmola, and C. Borcea, "Collaborative Bluetooth-based Location Authentication on Smart Phones", *in Elsevier Pervasive and Mobile Computing Journal*, 2014

M. Talasila, R. Curtmola, and C. Borcea, “ILR: Improving Location Reliability in Mobile Crowd Sensing”, *in International Journal of Business Data Communications and Networking, Special Issue 9(4)*, October 2013

G. Cardone, L. Foschini, C. Borcea, P. Bellavista, A. Corradi, M. Talasila, and R. Curtmola, “Fostering ParticipAction in Smart Cities: A Geo-Social CrowdSensing Platform”, *in IEEE Communications Magazine, Special Issue on Smart Cities*, June 2013

M. Talasila, R. Curtmola, and C. Borcea, “Improving Location Reliability in Crowd Sensed Data with Minimal Efforts”, *in Proceedings of the 6th Joint IFIP/IEEE Wireless and Mobile Networking Conference (WMNC ’13)*, April 2013

M. Talasila, R. Curtmola, and C. Borcea, “LINK: Location verification through Immediate Neighbors Knowledge”, *in 7th International ICST Conference on Mobile and Ubiquitous Systems (MOBIQUITOUS ’10)*, December 2010

I dedicate this thesis to my always encouraging mom: Uma Rani and to my wife: Sumana for all her love and support at each step of the way.

कर्मण्येवाधिकारस्ते मा फलेषु कदाचन।
मा कर्मफलहेतुर्भुर्मा ते सङ्गोऽस्त्वकर्मणि॥

This quote from Bhagawadgita has always helped me focus on my work. This quote means, "you have the right to perform your actions, but you are not entitled to the fruits of the actions. Do not let the fruit be the purpose of your actions, and therefore you won't be attached to not doing your duty".

ACKNOWLEDGMENT

Foremost, I would like to express my heartfelt gratitude to my Dissertation Advisor, Dr. Cristian Borcea for the continuous support of my Ph.D study and research, for his patience, motivation, enthusiasm, and immense knowledge. I have been amazingly fortunate to have an advisor who always gave me the freedom to chose the schedules for my research meetings which fit with my full-time job.

My co-advisor, Dr. Reza Curtmola, has been always there to listen and give advice. I am deeply grateful to him for the long discussions that helped me sort out the technical details of my work. I am also thankful to him for encouraging the use of correct grammar and consistent notation in my writings and for carefully reading and commenting on every manuscript I worked.

My sincere thanks also goes to Dr. David Nassimi, Dr. Guiling Wang, and Dr. Nishkam Ravi for being a part of my dissertation committee.

I am grateful to Dr. Luca Foschini from University of Bologna, Italy for his great efforts of collaborating with me to accomplish this fine work. I thank Giuseppe Cardone for all his contribution with Android development and for improving my research.

I thank my fellow lab-mates Juan (Susan) Pan, Hillol Debnath, Mohammad Khan, Daniel Boston and others in the Networking Laboratory for the stimulating discussions and for all their support while recruiting students for my research study.

Special thanks to Western Kentucky University classmates and my friends from New Jersey. Their support and care helped me overcome setbacks and stay focused on my graduate study. I greatly value their friendship and I deeply appreciate their belief in me.

Most importantly, none of this would have been possible without the love and patience of all my family members. I am indebted to my mom, Uma Rani Talasila who had sacrificed all her life till date to see me successful. I would like to thank my wife, Sumana Nelakurthi for her understanding and love during all these years. I am also thankful to my sister, Mahathi Talasila who supported my financial needs during my master studies which paved the path towards my research. My grandma, Maruthi Devi Kota, receive my deepest gratitude and love for many years of support during my undergraduate studies that provided the foundation for this work.

TABLE OF CONTENTS

Chapter **Page**

1 INTRODUCTION 1

1.1 Mobile Sensing and Mobile Crowd Sensing 2

1.2 Location Reliability in Mobile Crowd Sensing 5

1.2.1 Trusted Platform Modules in Smart Phones 6

1.2.2 Current Methods of Location Verification 7

1.3 Incentives for Participating in Mobile Sensing 8

1.4 Problem Statement 9

1.5 Contributions of Dissertation 9

1.6 Structure of Dissertation 13

2 RELATED WORK 15

2.1 Mobile Sensing 15

2.2 Mobile Crowd Sensing 16

2.3 Mobile Device's Trusted Hardware and Software 16

2.4 Existing Location Authentication Techniques 17

2.5 Incentives for Mobile Users in Mobile Crowd Sensing 20

2.6 Gamification Mechanisms in Crowdsourcing 21

2.7 Chapter Summary 23

3 MCSENSE SYSTEM OVERVIEW AND IMPLEMENTATION 24

3.1 Basic System Design 24

3.1.1 System Architecture and Processes Involved 24

3.1.2 Prototype Implementation 27

3.2 Tasks Developed for McSense 28

3.3 Field Study 29

3.4 Results 31

TABLE OF CONTENTS
(Continued)

Chapter **Page**

3.4.1 Correlation of User Earnings and Fake Photos 32

3.4.2 Correlation of Location and Fake Photos 32

3.4.3 Malicious User: Menace or Nuisance? 33

3.5 Chapter Summary 34

4 ILR: IMPROVING LOCATION RELIABILITY IN CROWD SENSED DATA . 35

4.1 Assumptions 35

4.2 Adversarial Model 36

4.3 Proposed Scheme 37

4.3.1 An Example of ILR in Action 37

4.3.2 The ILR Scheme 39

4.3.3 Validation Process 41

4.4 Security Analysis 42

4.5 Experimental Evaluation: Field Study 42

4.6 Simulations 44

4.6.1 Simulation Setup 44

4.6.2 Simulation Results 45

4.7 Chapter Summary 47

5 LINK: LOCATION AUTHENTICATION THROUGH IMMEDIATE NEIGHBORS KNOWLEDGE 48

5.1 Assumptions 48

5.2 Adversarial Model 50

5.3 Protocol Design 51

5.3.1 Basic Protocol Operation 51

5.3.2 LCA Decision Process 55

5.3.3 Use of Historical Data in LCA Decision 59

5.4 Security Analysis 62

TABLE OF CONTENTS
(Continued)

Chapter **Page**

5.5 Simulations . . . 67
5.5.1 Simulation Setup . . . 68
5.5.2 Simulation Results . . . 69
5.6 Implementation . . . 75
5.6.1 Client API . . . 76
5.6.2 LCA Server . . . 78
5.7 Experimental Evaluation . . . 79
5.7.1 Measurements Methodology . . . 79
5.7.2 Micro-Benchmark Results . . . 80
5.7.3 End-to-End Performance Results . . . 81
5.8 Chapter Summary . . . 85
6 A GAME-BASED INCENTIVE MECHANISM FOR MOBILE SENSING . . . 86
6.1 Motivation and Main Idea . . . 86
6.2 Alien vs. Mobile User Game . . . 87
6.2.1 Design . . . 89
6.2.2 Prototype Implementation . . . 91
6.3 Alien Movement Strategies . . . 92
6.3.1 Assumptions and Definitions . . . 92
6.3.2 Localized Movement Strategy . . . 94
6.3.3 Random Movement Strategy . . . 95
6.3.4 Progressive Movement Strategy . . . 96
6.3.5 Number of Aliens in the Game . . . 97
6.4 Simulations . . . 99
6.4.1 Entropy of the Area Coverage . . . 99
6.4.2 Simulation Setup . . . 100

TABLE OF CONTENTS
(Continued)

Chapter **Page**

6.4.3 Simulation Results 101

6.5 User Study Evaluation 104

6.6 Design Guidelines for Mobile Sensing Games 112

6.7 Chapter Summary 113

7 CONCLUSION 115

REFERENCES 117

LIST OF TABLES

Table **Page**

3.1 Demographic Information of the Students 30

4.1 Photo Task Reliability 43

4.2 Number of False Location Claims 44

4.3 Simulation Setup for the ILR Scheme 44

5.1 Simulation Setup for the LINK Protocol 68

5.2 Latency Table for Individual LINK Tasks 80

5.3 Energy Consumption for Individual LINK Tasks 80

5.4 Battery Life for Different WiFi and Bluetooth Radio States 83

5.5 Average RTT and Energy Consumption per Claim for Multi-claimer Case vs. Single Claimer Case 84

6.1 Player Types 101

6.2 Number of Players from each Player Type Assigned to the Three Behaviors/ scenarios 101

6.3 Players' Engagement based on Achievements Designed in the Game 107

LIST OF FIGURES

Figure **Page**

3.1 McSense Architecture. 25

3.2 McSense Android Application showing tabs (left) and task screen for a photo task (right). 26

3.3 Correlation of Earnings and Fake photos. 31

3.4 Correlation of user location and Fake photos. 32

3.5 Photo counts of 17 cheating people. 33

4.1 Example of McSense collected Photo tasks [A-I] and Sensing tasks [1-15] on the campus map, grouped using Bluetooth discovery co-location data. . . . 38

4.2 The phases of the ILR scheme. 40

4.3 ILR performance as function of the percentage of photos manually validated in phase 1. Each curve represents a different percentage of photos with fake locations. 45

4.4 ILR performance as function of the percentage of photos manually validated in phase 1. Each curve represents a different network density represented as average number of neighbors per node. 47

5.1 Basic Protocol Operation (where C = claimer, V_i = verifiers, LBS = Location-Based Service, LCA = Location Certification Authority). 52

5.2 False negative rate over time for individual malicious claimers with mixed behavior. The claim generation rate is 1 per minute, 15% of the users are malicious, and average speed is 1m/s. 70

5.3 Trust score of malicious users with mixed behavior over time. The claim generation rate is 1 per minute, 15% of the users are malicious, and average speed is 1m/s. Error bars for 95% confidence intervals are plotted. 71

5.4 False positive rate as a function of the percentage of malicious verifiers for different node densities. The claim generation rate is 1 per minute and average speed is 1m/s. Error bars for 95% confidence intervals are plotted. 72

5.5 False positive rate as a function of the percentage of malicious verifiers for different claim generation rates. The average speed is 1m/s. Error bars for 95% confidence intervals are plotted. 72

LIST OF FIGURES
(Continued)

Figure **Page**

5.6 False positive rate over time for different percentages of malicious verifiers. The claim generation rate is 1 per minute and the average speed is 1m/s. Error bars for 95% confidence intervals are plotted. 73

5.7 False negative rate over time for colluding users. Each curve is for a different colluding group size. Only 50% of the colluding users participate in each verification, thus maximizing their chances to remain undetected. The claim generation rate is 1 per minute and the average speed is 1m/s. Error bars for 95% confidence intervals are plotted. 74

5.8 False negative rate over time when punishing and not punishing colluding verifiers. The size of the colluding group is 12, and 50% of these users participate in each verification. The claim generation rate is 1 per minute and the average speed is 1m/s. 74

5.9 False positive rate over time when punishing and not punishing colluding verifiers. All parameters are the same as in Figure 5.8. 75

5.10 Coupon application on Android phone. 76

5.11 LINK RTT and total energy consumed by claimer per claim function of the number of verifiers. 81

6.1 Cost comparison between crowd sensing enabled by mobile gaming and crowd sensing enabled by micro-payments. 88

6.2 Alien vs. Mobile User app: finding the alien (left); alien trail (middle); achievements (right). 92

6.3 Localized Movement strategy: Example of CGS moving the alien A1 locally in the assigned highlighted region when A1 is shot by the player. As part of the greedy process, CGS moves the healthy alien A2 from an unpopular region to a popular region. 93

6.4 Random Movement strategy: Example of CGS swapping alien A1 (which is shot) with healthy alien A2 from the unpopular region. 94

6.5 Progressive Movement strategy: Example of CGS moving the alien to the closest unpopular region. 96

6.6 The area coverage latency of the three alien movement strategies. 102

6.7 Entropy of the area coverage for the three alien movement strategies. 102

6.8 The entropy and latency over time for the three alien movement strategies in BH3 scenario. 103

LIST OF FIGURES
(Continued)

Figure **Page**

6.9 Area coverage over time: crowd sensing with mobile gaming vs. crowd sensing with micro-payments for the first four weeks of the studies. 104

6.10 Coverage of the University campus in the first four weeks of the studies: 46% crowd sensing with micro-payments (left) vs. 87% crowd sensing with mobile gaming (right). A number of areas have been removed from the map as they are not accessible to students. 105

6.11 Average number of active players in a given week: mobile gaming vs. micro-payments. 106

6.12 Impact of the number of registered players and the number of active players on area coverage over time in the game. 106

6.13 Ranking players based on the number of covered squares in the area (area is divided in 10mx10m squares). 107

6.14 Distribution of square coverage using the mobile game. 108

6.15 Correlation between number of aliens killed and area coverage (we started to record this data in day 7). 109

6.16 Efficacy of Progressive Movement strategy: total number of aliens killed vs. number of aliens tracked by the players over time. 109

6.17 Correlation between number of bullets and aliens killed. The calculated correlation value is 0.98 (high positive correlation). 109

6.18 Average number of covered squares per player per day. 110

6.19 Average daily pattern for the number of active players. 110

6.20 Correlation of active players and the number of squares covered at different floor levels over time in last two weeks of the user study. 112

CHAPTER 1

INTRODUCTION

Mobile sensors such as smart phones and vehicular systems represent a new type of geographically distributed sensing infrastructure that enables mobile people-centric sensing [1]. According to a forecast for global smart phone shipments from 2010 to 2017, more than 1.5 billion phones are expected to be shipped worldwide [2]. These mobile sensing devices can be used to enable a broad spectrum of applications, ranging from monitoring pollution or traffic in cities to epidemic disease monitoring or real-time reporting from disaster situations. This new type of sensing can be a scalable and cost-effective alternative to deploying static wireless sensor networks for dense sensing coverage across large areas. Compared to the tiny, energy constrained sensors of static sensor networks, smart phones and vehicular systems can support more complex computations, have significant memory and storage, and offer direct access to the Internet.

However, mobile people-centric sensing has two main issues: 1) Data Reliability in sensed data and 2) Incentivizing the participants. Regarding the data reliability issue, the sensed data submitted by participants is not always reliable as they can submit false data to earn money without executing the actual task. Therefore, it is important to validate the sensed data. Regarding the incentives issue, it is not always economical for organizations (e.g. local/state government agencies or research communities) to provide monetary compensation for every type of sensing task to access valuable data which help improve public services. Hence, it is necessary to find an effective solution to incentivize the participants to perform sensing. The thesis of my dissertation is *that mobile crowd sensing can tap its* *true potential if supported by systems and protocols that improve sensed data reliability and offer various incentives to participants.*

The rest of this chapter presents an overview of mobile sensing and mobile crowd sensing in Section 1.1. Section 1.2 presents the issues on location reliability in mobile crowd sensing. Section 1.3 discusses on the incentives for participating in mobile sensing. Section 1.4 presents the problem statement addressed by the dissertation. The contributions of this dissertation are presented in Section 1.5. Finally, Section 1.6 details the structure of this dissertation.

1.1 Mobile Sensing and Mobile Crowd Sensing

The latest smart phones now come with many embedded sensors enabling a plethora of *mobile sensing* applications [3–6] in gaming, smart environments, surveillance, emergency response and social networks. Specially, activity recognition through mobile sensing and wearable sensors has lead to many healthcare applications, such as fitness monitoring, elder-care support and cognitive assistance [7]. The expanding sensing capabilities of mobile phones have gone beyond the sensor networks focus on environmental and infrastructure monitoring where people are now the carriers of sensing devices, and the sources and consumers of sensed events [8–12].

Mobile crowd sensing plays a similar role with the one played by Amazon's Mechanical Turk (MTurk) [13] or ChaCha [14] in *crowdsourcing* [15, 16]: it allows individuals and organizations (clients) to access a sheer number of people (providers) willing to execute simple sensing tasks for which they are paid. Unlike the MTurk's tasks which are executed on personal computers and always require human work, mobile sensing tasks are executed on mobile devices that satisfy certain context/sensing requirements (e.g., location, time, specific sensors) and many times do not require human work (i.e., automatic sensing tasks).

Smart phones and mobile platforms available on the consumer market already permit accurate tracing of world-related information and (physical) activities of citizens by taking advantage of people willing to collaborate toward a continuous data harvesting process, called *crowd sensing* [1, 17–19]. Smart phones already have several sensors: camera, microphone, GPS, accelerometer, digital compass, light sensor, Bluetooth as proximity sensor [20, 21], and in the near future they are envisioned to include health and pollution monitoring sensors [22–24]. Vehicular systems have access to several hundred sensors embedded in cars, and recent vehicles come equipped with new types of sensors such as radar and camera. We believe that researchers in many fields of science and engineering as well as local, state, and federal agencies could greatly benefit from this new sensing infrastructure as they will have access to valuable data from the physical world. Additionally, commercial organizations may be very interested in collecting mobile sensing data to learn more about customer behavior.

In the following, we present several domains that can benefit from people-centric sensing as well as a number of applications (some of them already prototyped) for each domain:

- *Road Transportation*: Departments of transportation can collect fine grain and large scale data about traffic patterns in the country/state using location and speed data provided by GPS sensors embedded in cars. These data can then be used for traffic engineering, construction of new roads, etc. Drivers can receive real-time traffic information based on the same type of data collected from smart phones [25]. Drivers can also benefit from real-time parking data collected from cars equipped with ultrasonic sensors [26]. Transportation agencies or municipalities can efficiently collect pothole data using GPS and accelerometer sensors [27] in order to quickly repair the roads. Similarly, photos (i.e., camera sensor data) taken by people during/after snow storms can be analyzed automatically to prioritize snow cleaning and removal.

- *Healthcare & Wellbeing*: Wireless sensors worn by people for heart rate monitoring [22] and blood pressure monitoring [24] can communicate their information to the owners' smart phones. Typically, this is done for both real-time and long-term health monitoring of individuals. Mobile sensing can leverage these existing data into large scale healthcare studies that seamlessly collect data from various groups of people, which can be selected based on location, age, etc. A specific example involves collecting data from people who eat regularly fast food. The phones can perform activity recognition and determine the level of physical exercise done by people, which was proven to directly influence people's health. For example, as a result of such a study in a city, the municipality may decide to create more bike lanes to encourage people to do more physical activities. Similarly, the phones can determine the level of social interaction of certain groups of people (e.g., using Bluetooth scanning, GPS, or audio sensor). For example, a university may discover that students (or students from certain departments) are not interacting with each other enough; consequently, it may decide to organize more social events on campus. The same mechanism coupled with information from "human sensors" can be used to monitor the spreading of epidemic diseases.
- *Marketing/Advertising*: Real-time location or mobility traces/patterns can be used by vendors/advertisers to target certain categories of people [28, 29]. Similarly, they can run context-aware surveys (function of location, time, etc.). For example, one question in such a survey could ask people attending a concert what artists they would like to see in the future.

The applications developed so far have been mostly university prototypes, tested with student volunteers. To make such mobile crowd sensing applications usable in real life, we need to overcome several concerns such as: How to find enough people willing to provide data to satisfy the spatial and temporal constraints of the application? How to trust that the sensed data is reliable? What do users get if they consume the time and resources to participate in these applications? How can they control when, where, and what data is collected from their mobile device? What guarantees does they have that their privacy is not violated? Is the sensed data useful in inferring some interesting facts? How is the data impacted with varying incentives? Are people willing to trade off their location privacy in exchange for monetary incentives? One question that subsumes all these concerns is: Can a mobile people-centric sensing system be designed to achieve a good balance between usability, privacy, performance, and reliability?

1.2 Location Reliability in Mobile Crowd Sensing

By leveraging smart phones, we can seamlessly collect sensing data from various groups of people at different locations using mobile crowd sensing. As the sensing tasks are associated with monetary incentives, participants may try to fool the mobile crowd sensing system by providing false data in order to earn more money than they deserve. Therefore, there is a need for mechanisms to validate the collected data efficiently. In the following, we motivate the need for such a mechanism by presenting several scenarios involving malicious behavior.

Traffic jam alerts [30, 31]: Suppose that the Department of Transportation uses a mobile crowd sensing system to collect alerts from people driving on congested roads and then distributes the alerts to other drivers. In this way, drivers on the other roads can benefit from real-time traffic information. However, the system has to ensure the alert validity because malicious users may try to pro-actively divert the traffic on roads ahead in order to empty these roads for themselves or may try to earn money by submitting fake traffic data while sitting at home.

Citizen-journalism [32, 33]: Citizens can report real-time data in the form of photos, video, and text from public events or disaster areas. In this way, real-time information from anywhere across the globe can be shared with the public as soon as the event happens. But, malicious users may try to earn easy money by claiming that an event is happening at a certain location while being somewhere else.

Environment [23, 34]: Environment protection agencies can use pollution sensors installed in the phones to map with high accuracy the pollution zones around the country. The participants may claim “fake” pollution to hurt business competitors by submitting the sensed pollution data associated with false locations.

Ultimately, sensed data validation is important in a mobile crowd sensing system to provide confidence to its clients who use the sensed data. However, it is challenging to validate each and every sensed data point of each participant because sensing measurements are highly dependent on context. One approach to handle the issue is to validate the location associated with the sensed data point in order to achieve a certain degree of reliability on the sensed data. Still, we need to overcome a major challenge: how to validate the location of data points in a scalable and cost-effective way without help from the wireless carrier? (wireless carriers may not help with location validation for legal reasons related to user privacy or even commercial interests)

1.2.1 Trusted Platform Modules in Smart Phones

To achieve reliability on participant’s location data, there are a few traditional solutions such as using Trusted Platform Modules (TPM) [35] on smart phones or a platform such as Trustzone [36] or M-Shield [37]. Most ARM processors support TrustZone, and several mid-range and high-end Nokia phones utilize M-Shield. The modules included in the trusted code base on a provider’s device ensure the correctness of the provider’s actions. The Sensor API and Sensor Aggregation modules ensure the correctness of the sensed data and of basic data aggregation on the device. The Reputation Enforcer module ensures that a provider cannot accept new sensing tasks before the reputation score has been updated according to data feedback and data validation results. However, these cannot be used directly as it may not be cost-effective to have TPM modules on every smart phone.

1.2.2 Current Methods of Location Verification

Another solution is to verify location through the use of secure location verification mechanisms [38–40] in real time when the participant is trying to submit the sensing data location. Although a significant number of publications tackled the location authentication problem, all of them assumed support from the wireless network infrastructure [41, 42] or from a deployed localization infrastructure using distance-bounding techniques [43, 44].

Typically, these solutions are based on signal measurements between the mobile devices and fixed, trusted beacons or base stations (e.g., cell towers, WiFi access points) with known locations [45]. We argue that a solution that works without any support from the network/localization infrastructure is important because wireless carriers may refuse to authenticate user location for third-party Location Based Services (LBSs) due to legal and commercial reasons: they may not be allowed by laws to share any type of user location data, and they may not want to help their competition in the LBS arena (the network carrier itself may be providing LBS services).

Furthermore, using carrier support and existing privacy preserving methods [46] may not work as desired to solve the problem of location verification. The first reason is commercial as mentioned above. The second reason is the low accuracy of the authenticated location. If the privacy preserving methods involve k-anonymity or similar methods, the accuracy of the claimed location may not be very good. For example, the carrier may provide coarse-grained location authentication in order to satisfy k-anonymity constraints (e.g., authenticate that the user is in a certain city or a large region of that city). Such authentication may not be useful for many LBSs, which may require fine-grained location authentication.

1.3 Incentives for Participating in Mobile Sensing

A major challenge for broader adoption of the mobile crowd sensing systems is how to incentivize people to collect and share sensor data from a targeted area. Mapping an area with sensor data is a tedious effort when performed manually, and it is possible to collect a disproportionate amount of data from regions that are very popular compared to regions that are less popular [47]. Furthermore, sharing the sensed data may raise privacy concerns as it may require the sharing of private information such as location. Therefore, there is a need to find an efficient solution to incentivize the participants in collecting sensor data from an entire target area.

Many of the proposed mobile crowd sensing systems provide monetary incentives to smart phone users to collect sensing data. There are solutions based on micro-payments [48] in which small tasks are matched with small payments. The participants in mobile crowd sensing systems may require significant incentives to go out of their way and cover unpopular regions. Other techniques were also explored to motivate individuals to participate in sensing. For example, beneficial personal analytics are provided as incentives to participants through sharing bicycle ride details in Biketastic [49]. Another variety of incentive is enabling data bartering to obtain additional information, such as bargain hunting through price queries in LiveCompare [50].

In addition, there are gamification techniques proposed for crowd-sourced applications [51, 52]. However, to the best of our knowledge, no work has been done on using mobile games for incentivizing the participants in a mobile crowd sensing system to cover a targeted area uniformly. General gamification techniques for crowd-sourced applications cannot be directly applied in the context of uniform area coverage because we need to answer specific questions such as: What coverage strategies work best? What incentives mechanisms work best?

1.4 Problem Statement

This dissertation addresses the problem of data reliability in sensed data from mobile crowd sensing and that of finding a cost-effective solution to incentivize the users to participate in sensing. We argue that clients need guarantees from mobile sensing systems that collected data is valid. One simple way to address data reliability and provide certain guarantees is to generate duplicate tasks. However, this solution may work only if the two tasks run at the same location and time because sensed data is context-dependent. Therefore, additional mechanisms are needed to improve data reliability. Moreover, validating the context of every sensed data point of each participant is not a scalable solution. One alternative is to first validate the location associated with the sensed data points in order to achieve a certain degree of reliability about the sensed data. However, location validation without support from the wireless carriers is difficult.

To successfully achieve reliability in sensed data collected from mobile crowd sensing system, we need software and protocols that addresses the following questions: how to provide location reliability for sensed data? And how to use gamification in mobile sensing to incentivize user participation?

1.5 Contributions of Dissertation

The main contributions of this dissertation are:

- The design and development of McSense, a platform for mobile crowd sensing which incorporates location reliability and incentive mechanisms.
- The design and analysis of ILR (Improving Location Reliability), a scheme in which we utilize participatory sensing itself to achieve data reliability in mobile crowd sensing.
- The design, analysis, and implementation of LINK (Location authentication through Immediate Neighbors Knowledge), a location authentication protocol working independent of wireless carriers, in which nearby users help authenticate each other's location claims using Bluetooth communication. This protocol allows to achieve real-time location data reliability in mobile crowd sensing.
- The design and implementation of a first person shooter mobile sensing game, "Alien vs. Mobile User", for automatically and uniformly collecting crowd sensing data across large areas based on incentivizing smart phone users to play mobile sensing games.

To evaluate these protocols, we have built a McSense mobile platform [53] which can run three mobile sensing tasks on Android phones and a simple backend service that receives client requests to collect sensor data. Specifically, we developed one manual sensing task (capturing photos at events on campus) and two automatic sensing tasks: the first collects accelerometer and GPS data for a longer period in order to estimate student's activities on campus (accelerometer data tell us if they are walking, running or driving; GPS data tells us their significant places); the second collects Bluetooth-based co-presence data in order to estimate the level of the students' social engagement. The McSense mobile application, has been implemented in Android and is compatible with smart phones having Android OS 2.2 or higher. The McSense Android application was deployed to Google Play to make it available for campus students. The user study ran for 2 months, and the students were paid to perform crowd sensing tasks (50 cents to a few dollars per task). A total of 50 students participated in this study, and all of them were registered from the first day of the study.

To improve the location reliability in the sensed data that is collected from McSense, we propose the ILR (Improving Location Reliability) scheme. In the ILR scheme, we bootstrap the trust in the system by first manually or automatically using image processing techniques validating a small number of photos submitted by participants. Based on these validations, the location of these photos is assumed to be trusted. Second, we extend this location trust to co-located sensed data points found in the Bluetooth range of the devices that provided the validated photos. This transitive trust is extended until all the co-located tasks are trusted or no new data points are found.

In addition, the ILR scheme also helps to detect false location claims associated with sensed data. We applied ILR on data collected from our McSense prototype deployed on Android phones used by students on our campus and detected a significant percentage of the malicious users. ILR was able to detect 40% of the users submitting photos from false locations; for ground truth validation, we manually inspected these photos. Simulation results demonstrate that ILR works well at various densities and helps detect the false location claims based on a minimal number of validations. At the end of the field study, we requested each user to fill a survey in order to understand the participants' opinion on location privacy and on usage of phone resources.

To achieve stronger guarantees about the sensed data, we propose a second protocol, LINK (Location authentication through Immediate Neighbors Knowledge) to verify the participant's location in real time when the participant is trying to submit the sensing data location. LINK does not require cooperation from the wireless network carrier and thus works for any third-party service. For each user's location claim, a centralized Location Certification Authority (LCA) receives a number of verification messages from neighbors contacted by the claimer using short-range wireless networking such as Bluetooth. The

LCA decides whether the claim is authentic or not based on spatio-temporal correlation between the users, trust scores associated with each user, and historical trends of the trust scores. Simulation results demonstrate that LINK thwarts individual user attacks and a number of colluding users attacks.

We have implemented LINK on Android-based Motorola Droid 2 phones. Mobile applications can use a simple location authentication API provided by the LINK package. To participate as verifiers in the system, users have to start a LINK background process. The LCA implementation helps the mobiles to avoid Bluetooth inquiry clashes when simultaneous claims are performed by multiple claimers. For testing purposes, we implemented a coupon LBS service which distributes location-based electronic discount coupons to people passing by a shopping mall. Both this LBS and the LCA are implemented in Java.

We also performed an experimental evaluation to quantify the response latency and battery consumption associated with LINK. The results from a test-bed with six phones demonstrate that the response latency is low enough (typically 10-12s) for a static claimer to successfully complete the protocol before its verifiers, moving at walking speed, would go out of the Bluetooth transmission range. In terms of power consumption, a fully charged phone is capable of running thousands of claims and tens of thousands of certifications. Thus, users are not expected to turn off LINK due to power concerns.

We leverage gamification technique for mobile crowd sensing to achieve an economical solution without paying for the sensing tasks. We argue that a mobile sensing system requires sufficient user participation to achieve good quality sensed data for mobile sensing applications. Essentially, the system needs a cost-effective way to incentivize users to participate in sensing such as gamification techniques instead of monetary incentives. However, providing good sensing coverage of an entire area may prove difficult. It is possible to collect a disproportionate amount of data from very popular regions in the area, while the unpopular regions remain uncovered.

We propose a model for automatically collecting crowd sensing data based on incentivizing smart phone users to play sensing games, which provide in-game incentives to convince participants to cover all the regions of a target area uniformly. We designed and implemented a first person shooter sensing game, "Alien vs. Mobile User", which employs techniques to attract users to unpopular regions. Our prototype Android game collects WiFi data to create a campus coverage map. This study ran for 35 days, the students used their Android devices to play our game and collect WiFi data both outdoors and indoors throughout the campus of our institution. A total of 53 players participated in this study; the users were continuously registered after the first day of the study (i.e., they joined the game when they found out about it). The results from the user study show that mobile gaming ensures fast and efficient area coverage compared to micro-payments (McSense user study), and we observe that the proposed game design succeeds in achieving good player engagement. Furthermore, we compare three strategies for area coverage in terms of coverage time and coverage effort for users. The simulation results demonstrate that Progressive Movement is the best strategy because it manages to quickly entice users from popular regions to unpopular ones with a reasonable coverage effort.

1.6 Structure of Dissertation

The subsequent sections of this thesis dissertation are structured as follows: Chapter 2 reviews related work. Chapter 3 describes the McSense system overview and implementation. Chapter 4 introduces the ILR scheme and its experimental results. Chapter 5 describes the LINK protocol and presents the experimental results and their analysis. Chapter 6 presents the “Alien vs. NJIT” mobile user game and the results of a user study that shows the benefits of gaming for incentivizing users to uniformly cover large areas. The dissertation concludes in Chapter 7.

CHAPTER 2

RELATED WORK

This chapter presents background and related work literature in the domain of mobile sensing (Section 2.1), mobile crowd sensing (Section 2.2), mobile device's trusted hardware and software (Section 2.3), existing location authentication techniques (Section 2.4), incentives for mobile users in mobile crowd sensing (Section 2.5) and gamification mechanisms in crowdsourcing (Section 2.6). The chapter concludes in Section 2.7.

2.1 Mobile Sensing

The idea of mobile people-centric sensing was introduced fairly recently, but a lot of progress was made already. MetroSense [54], Participatory Sensing [55], and Urbanets [1] were among the first projects to demonstrate the feasibility of the idea. Following these initial projects, the community focused on developing platforms and applications for people-centric sensing. For example, MyExperience [56] is a system that captures and shares both user-specific and device-specific data on smart phones. SenseWeb [57] provides a Web-based platform and tools that let people easily publish and query sensor data. MicroBlog [58] is a participatory sensing application that allows people to use their smart phones to generate and share geo-tagged multimedia. ParkNet [26] is an application that informs drivers about on-street parking availability using a vehicular sensing system. Activity recognition for people-centric sensing has also become a major research issue [59]. For example, CenceMe infers "facts" about a user from the sensors embedded in the smart phone [10], while the Pothole Patrol [27] identifies the pot holes on the roads. All these projects advanced significantly the field, but it is still not clear how to extend people-centric sensing to real-world, beyond lab tests and small scale prototypes with selected volunteers.

2.2 Mobile Crowd Sensing

Recently, several mobile crowdsourcing projects [60,61] tried to leverage traditional crowdsourcing platforms for mass adoption of people-centric sensing: Twitter [62] has been used as a publish/subscribe medium to build a crowdsourced weather radar and a participatory noise-mapping application [63]; mCrowd [61] is an iPhone based platform that was used to build an image search system for mobile phones which relies on Amazon MTurk [13] for real-time human validation [64]. This has the advantage of leveraging the popularity of existing crowdsourcing platform (tens of thousands of available workers), but does not allow for truly mobile sensing tasks to be performed by workers (i.e., tasks which can only be performed using sensors on mobile phones). Moreover, as explained in introduction (Chapter 1), an autonomous mobile crowd sensing platform introduces additional concerns that must be addressed, such as the privacy of the participants and the reliability of the sensed data.

2.3 Mobile Device's Trusted Hardware and Software

To address the data reliability issues in sensed data on smart phones, trusted hardware represented by the Trusted Platform Module (TPM) [35, 65–67] has been leveraged to design new architectures for trustworthy software execution on mobile phones [68–70]. Recent work has also proposed architectures to ensure that the data sensed on mobile phones is trustworthy [71,72]. When untrusted client applications perform transformations on the sensed data, YouProve [66] is a system that combines a mobile device's trusted hardware with software in order to ensure the trustworthiness of these transformations and that the meaning of the source data is preserved. YouProve describes three alternatives to combine the trusted hardware with software: the first two require to extend the trusted codebase to include either the code for the transformations or the entire application, whereas the third one requires building trust in the code that verifies that transformations preserve the meaning of the source data.

Relying completely on TPM is insufficient to deal with attacks in which a provider is able to "fool" the sensors (e.g., using the flame of a lighter to create the false impression of a high temperature). Recently, there have also been reports of successful spoofing of civilian GPS signals [73].

2.4 Existing Location Authentication Techniques

Location authentication for mobile users has been studied extensively so far. Hence, there is a chance to leverage existing location authentication techniques to achieve data reliability by verifying the participant's location at real-time when the sensed data is submitted. To the best of our knowledge, all existing solutions employ trusted network/localization infrastructure [38, 43, 44, 74–78] to detect malicious users claiming false locations. Most of these solutions use distance bounding techniques, in which a beacon acting as verifier challenges the mobile device and measures the elapsed time until the receipt of its response.

None of these solutions, however, can be directly applicable to scenarios that involve interaction between mobile users and third-party services (i.e., services that do not have direct access to the network/localization infrastructure). The main novelty of one of the protocol presented in this dissertation (LINK protocol) comes from employing mobile users (more exactly their mobile devices) to certify the location claimed by other users.

The closest infrastructure-based solution to our research work is [79], which uses mobile base stations to authenticate location in sensor networks. This solution authenticates location by leveraging verifier mobility: a mobile base station sends a verification request to a sensor from one position and then waits for the response at another position. We share the idea of using mobile verifiers, but this solution cannot be applied for our problem because it is expensive and does not scale.

In the same paper [79], the authors propose that nodes in a mobile ad hoc network (MANET) provide verifications for each other, which is similar to our idea. However, the authors do not consider cases when verifiers are malicious, when the claimer is alone, or when several nodes collude with each other. Our work, on the other hand, provides solutions to all these issues. Furthermore, the proposed solution requires all nodes to have passive ranging capabilities (e.g., ultrasonic interface), while our proposed protocol works with the existing interfaces on the phones.

In [80], the authors propose location verification for VANETs in which a node can detect the malicious nodes after exchanging neighbor grouping information with other vehicles. However, the system model and design considered in this work vastly differs from our system model and design: Directional antennas are used to perform relative position verification with help of neighboring nodes, whereas our work uses Bluetooth communication to verify the user's absolute location claim. Thus, the protocol in our work is more practical when considering the available technologies on existing mobile phones (i.e., Bluetooth vs. directional antennas). Also, most LBSs require absolute location verification, not relative positioning verification. Furthermore, the very nature of VANETs/ MANETs makes it difficult to maintain accurate global information about the users in the system due to network partitioning and bandwidth limitations for large networks. Unlike these solutions, our research leverages Internet connectivity and the centralized LCA to achieve global knowledge about the users in the system (i.e., registered with the LCA) and, thus, accurate location authentication.

Other cooperative location verification protocols in VANETs are mostly dependent on distance bounding techniques or based on challenging the Time-of-Flight of the signals which involves additional infrastructure support. The differences between our protocols and these protocols are similar to the differences with the infrastructure-based solutions mentioned above.

Similar to our work, SMILE [81] and Ensemble [82] use information collected by mobile devices (keys from nearby users or received signal strength – RSS – values) to provide mutual co-location verification for mobile users. However, they do not provide location verification. RSS signatures in conjunction with RSS fingerprinting could be used for location verification, but such solutions do not scale due to the very dense fingerprinting required to achieve good accuracy.

In [83], the authors propose a protocol similar to our work in which neighbor nodes use Bluetooth communication to provide location proofs for claimers. Since this protocol focuses mostly on location privacy, it presents only a discussion of potential solutions against malicious claimers and colluding users. On the other hand, our work describes, implements, and evaluates the success of solutions that make it resilient to many types of attacks.

As one of our protocol (LINK) is based on trust scores, our work shares a number of similarities with work on reputation systems for P2P and MANETs. For example, CONFIDANT is a protocol [84] that avoids node misbehavior by establishing trust relationships between nodes based on direct and indirect observations reported by other nodes.

The CORE protocol [85] takes a similar approach and uses reputation to enforce node cooperation. In contrast with CONFIDANT, CORE requires reputation values received from indirect observations, thus preventing malicious nodes from wrongfully accusing legitimate nodes. Unlike these systems which were designed for MANET, the system in [86] was designed for P2P networks. The management of the reputation values of peer nodes is similar to the way our work manages the trust scores: when a peer is determined to be malicious, its reputation is cut in half; when a peer provides good service, its reputation is additively increased.

There are two main differences between this type of solution and our work. First, our protocol (LINK) cannot monitor indirectly additional user actions (such as packet forwarding or file sharing) to assess the trust. Second, our work employs the centralized LCA to have a global view of the the entire system. As such, it is able to detect malicious trust score trends and collusion attacks.

2.5 Incentives for Mobile Users in Mobile Crowd Sensing

Micro-payments have been studied as an incentive for users to complete tasks in crowd-sourcing (Amazon MTurk [13]), to control "free-riders" in peer-to-peer networks [87, 88], and to meter web content usage [89]. In the context of participatory sensing, micro-payments have been examined in [48, 90, 91]. Some of the key findings are that incentives can be highly beneficial in recruiting participants and that micro-payments have the potential to extend participant coverage both spatially and temporally. Other crowd sensing systems have relied on micro-payments as well [64]. Various micro-payment schemes have been shown to have different effects on data quality and participant compliance and retention [48, 91], which indicates that the parameters of an incentive scheme based on micro-payments should be tailored to specific sensing tasks. We plan to leverage the findings of these prior studies in our system. However, attracting people to unpopular places could be difficult and expensive. For example, the results from a recent crowd sensing study [47] show that many places will be infrequently visited. Our game, on the other hand, focuses on attracting players to such infrequently visited places and succeeds in covering the targeted area.

Task pricing also helps in improving the data quality which is an orthogonal work to our research. A recent paper [92] presents pricing incentive mechanisms to achieve quality data in participatory sensing application. In this work, the participants are encouraged to participate in sensing system through a reverse auction based dynamic pricing incentive mechanism in which users can sell their sensing data with their claimed bid price.

2.6 Gamification Mechanisms in Crowdsourcing

In search of cost-effective ways to incentivize users to participate in sensing, we leverage gamification techniques in our research. There is a significant literature on using gamification techniques in crowdsourcing, however there is very little in terms of applying gamification techniques in mobile sensing. A paper on Serious Games [93] motivate mobile users to participate in a game that involves the recording of as many audible signals as possible from different traffic lights. This collected data can be processed and integrated with Google maps which improve crossroad accessibility for blind pedestrians by providing information about the accessibility of the route/path. The authors mention the automatic data gathering from smart phones' sensors, but did not discuss any specific model or architecture in detail. The authors mostly focused on a single game where mobile users record traffic light audible signals.

There are a few crowdsourcing games [94, 95] that accomplish sensing tasks by requiring the explicit participation of the players using the phone's keypad. Our crowd sensing game performs automatic sensing and does not require players to provide manual input nor to complete tasks not related to the game story. Like in any other mobile game, players can simply enjoy playing. Thus, our game has a higher probability to maintain the players' interest over time.

The authors of a crowdsourced BioGame [51] show that in cases where the diagnosis is a binary decision (e.g., positive vs. negative, or infected vs. uninfected), it is possible to make accurate diagnosis by crowdsourcing the raw data (e.g., microscopic images of specimens/cells) using entertaining digital games (i.e., BioGames) that are played on PCs, tablets or mobile phones. This paper mainly focuses on the problem of handling large quantities of data and finally solves the problem through crowdsourcing based solution.

Another crowdsourcing game called "BudBurst" proposed in [52], it is a smart phone application for an environmental Participatory Sensing project that focuses on observing plants and collecting plant life stage data. The main goal is "flora-caching" where players gain points and levels within the game by finding and making qualitative observations on plants. This game is also an example of motivating participatory sensing.

One more participatory sensing game is proposed in [96], where a game named "Who" is used to extract relationship and tag data about other employees. It was found useful for rapid collection of large volumes of high-quality data from "the masses".

"Treasure" [97] is a mobile game that collects the same data with our game. In "Treasure", selected players using PDAs play against their opponents in a specific open space such as a large lawn where players have to collect coins that are scattered in the game area and upload the collected coins back to the server when they find WiFi connectivity.

This game is not intended for sensing, and it has not been designed for collecting sensing data; the WiFi data is just used to help players quickly upload the collected coins. Furthermore, this game did not attempt to study area coverage efficiency. The main focus of this game is on player's gaming experience and their tactics and strategies in a multi-player game. In addition, the players are compensated to participate in the game. Our game is designed to enable fast crowd sensing coverage of large areas, and the players are not compensated: the fun of playing the game is the only incentive.

Existing work in mobile health such as BeWell [98] utilizes smart phone sensing to assess the mobile users' wellbeing through scores based on their daily activities. In BeWell, an animated aquatic ecosystem is shown with three different animals, the behavior of each being affected by changes in the user's wellbeing. Thus, the users are motivated to maintain a healthy lifestyle. In a similar direction, our mobile game focuses on utilizing simple game graphics and in-game incentives to motivate smart phone users for achieving cost-effective crowd sensing.

2.7 Chapter Summary

In this chapter, we discussed the existing studies applied or related to mobile crowd sensing and their downsides in addressing the potential issues. We first discussed the solutions based on using trusted hardware and software in mobile devices for addressing the data reliability issue. We then presented the existing location authentication techniques to address the location data reliability issue. Subsequently, previous work addressing the problem of incentivizing mobile users in Mobile Crowd Sensing was presented. Finally, existing work on gamification mechanisms in crowdsourcing was also reviewed.

CHAPTER 3

MCSENSE SYSTEM OVERVIEW AND IMPLEMENTATION

This chapter first provides a general overview of the basic centralized design of our Mobile Crowd Sensing system and its interacting entities (Section 3.1), and its architecture & protocols (Section 3.1.1). The chapter then describes the prototype implementation (Section 3.1.2) and the tasks developed for McSense (Section 3.2). In Section 3.3 we present a field study and describe its results in Section 3.4 respectively. The chapter ends with a summary (Section 3.5).

3.1 Basic System Design

We have designed and implemented McSense [99], a mobile crowd sensing platform that allows clients to collect many types of sensing data from smart phones carried by mobile users. The interacting entities in our mobile crowd sensing architecture are:

- *McSense:* A centralized mobile crowd sensing system which receives sensing requests from clients and delivers them to providers; these entities are defined next.
- *Client:* The organization or group who is interested in collecting sensing data from smart phones using the mobile crowd sensing system.
- *Provider:* A mobile user who participates in mobile crowd sensing to provide the sensing data requested by the client.

3.1.1 System Architecture and Processes Involved

The architecture of McSense, illustrated in Figure 3.1, has two main components: (1) the server platform that accepts tasks from clients and schedules the individual tasks for

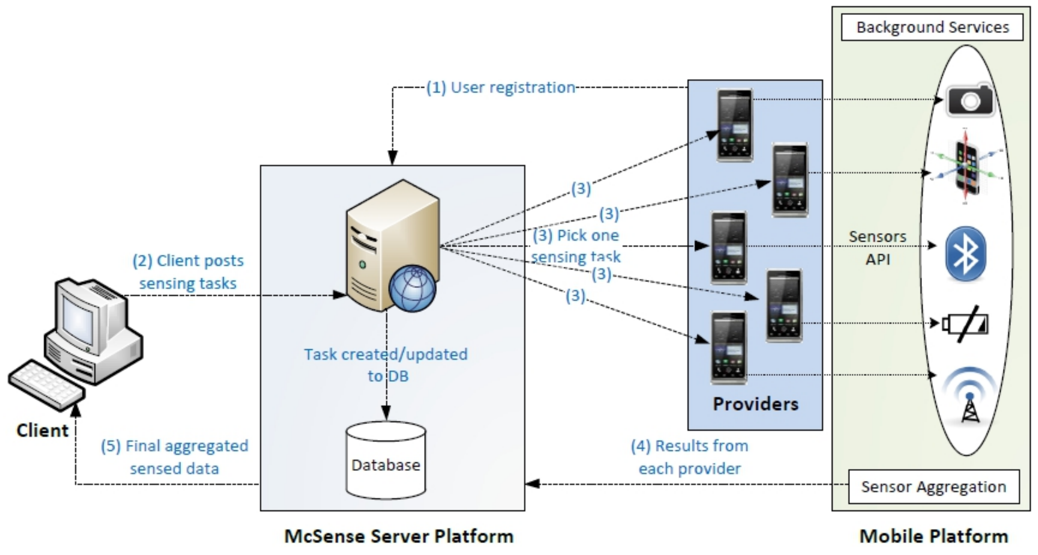
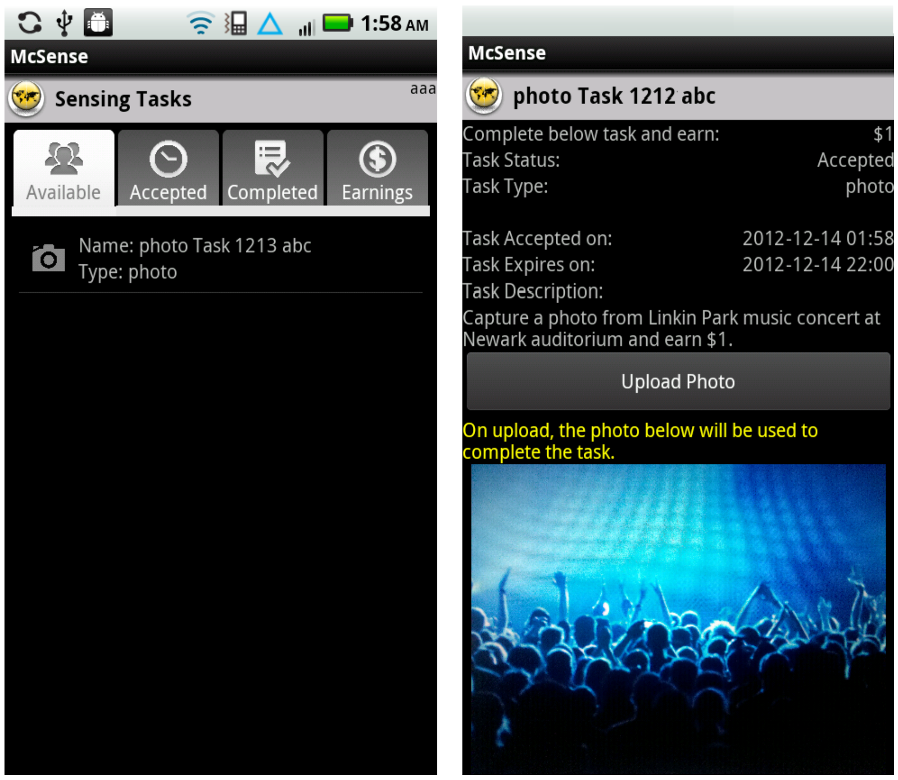

Background Services
(1) User registration
(2) Client posts sensing tasks
(3)
(3) Pick one sensing task
Task created/updated to DB
Database
Client
(5) Final aggregated sensed data
(4) Results from each provider
Providers
Sensors API
Sensor Aggregation
McSense Server Platform
Mobile Platform

Figure 3.1 McSense Architecture.

execution at mobile providers; and (2) the mobile platform (at the providers) that accepts individual tasks from the server, performs sensing, and submits the sensed data to the server. The communication among all these components takes place over the Internet. Next we discuss the overall process in more detail.

User registration: The McSense application on the smart phones shows a registration screen for first time users, prompting them to enter an email address and a password. During the registration process, the user phone's MEID (Mobile Equipment IDentifier) is captured and saved in the server's database along with the user's email address and password. We chose to store the phone's MEID in order to restrict one user registration per device. In addition, the server also avoids duplicate registrations when users try registering with the same email address again.

Posting new sensing tasks: New sensing tasks can be posted by clients using a web interface running on the McSense server. The sensing task details are entered on this web page by the client and submitted to the server's database. Once a new task is posted, the background notification service running on the provider's phone identifies the new



Name: photo Task 1213 abc
Type: photo

McSense

photo Task 1212 abc

Complete below task and earn:	$1
Task Status:	Accepted
Task Type:	photo
Task Accepted on:	2012-12-14 01:58
Task Expires on:	2012-12-14 22:00

Task Description:
Capture a photo from Linkin Park music concert at Newark auditorium and earn $1.

On upload, the photo below will be used to complete the task.

Figure 3.2 McSense Android Application showing tabs (left) and task screen for a photo task (right).

available tasks and notifies the provider with a vibrate action on the phone. Providers can check the notification and can open the McSense application to view the new available tasks. When the application is loaded, the providers can see four tabs (Available, Accepted, Completed and Earnings). The providers can view the list of tasks in the respective tabs (Figure 3.2, left) and can click on each task from the list to view the entire task details (type, status, description, accepted time, elapsed time, completion time, expiration time, payment amount).

Life cycle of a task: The life cycle starts from the Available tasks tab. When a provider selects an available task and clicks on the Accept button, the task is moved to the Accepted tab. Once a task is accepted, then that task is not available to others anymore (Figure 3.2, right). When the accepted task is completed according to its requirements, the task is moved to the Completed tasks tab. Finally, the providers view their aggregated total dollars earned for successfully completed tasks under the Earnings tab. If the accepted task expires before completing successfully according to its requirements, it is moved to the Completed tasks tab and marked as unsuccessfully completed. The providers do not earn money for the tasks that are completed unsuccessfully.

Background services on phone: When the network is not available, a completed task is marked as pending upload. A background service on the phone periodically checks for the network connection. When the connection becomes available, the pending data is uploaded and finally these tasks are marked as successfully completed. If the provider phone is restarted manually or due to the mobile OS crash, then all the in-progress sensing tasks are automatically resumed by the Android's BroadcastReceiver service registered for the McSense application. Furthermore, the Accepted and the Completed tab's task lists are cached locally and are synchronized with the server. If the server is not reachable, the users can still see the tasks that were last cached locally.

3.1.2 Prototype Implementation

The McSense application, shown in Figure 3.2, has been implemented in Android and is compatible with smart phones running Android OS 2.2 or higher. The application was tested successfully using Motorola Droid 2 phones which have 512 MB RAM, 1 GHz processor, Bluetooth 2.1, Wi-Fi 802.11 b/g/n, 8 GB on board storage, and 8 GB microSD storage. The McSense [100] Android application was deployed to Google Play [101] to make it available for campus students. The server side of McSense is implemented in Java/J2EE using the MVC (Model View Controller) framework. The Derby database is used to store the registered user accounts and assigned task details. The server side Java code is deployed on the Glassfish Application Server, which is an open-source application server.

3.2 Tasks Developed for McSense

The sensing tasks that we choose to develop for our research fall into two categories:

1. Manual tasks, e.g., photo tasks
2. Automated tasks, e.g., sensing tasks using accelerometer and GPS sensors; sensing tasks using Bluetooth.

Manual Photo Sensing Task: Registered users are asked to take photos from events on campus. Once the user captures a photo, she needs to click on the "Complete Task" button to upload the photo and to complete the task. Once the photo is successfully uploaded to the server, the task is considered successfully completed. These uploaded photos can be used by the university news department for their current news articles. On click of "Complete Task" button, if network is not available, the photo task is marked as completed and waiting for upload. This task is shown with a pending icon under completed tasks tab. Then a background service takes care of uploading the pending photos when the network becomes available. If a photo is uploaded to server after the task expiration time, then the photo is useless for the client. Therefore, the task will be marked as "Unsuccessfully completed", and the user do not earn money for this task.

Automated Sensing Task using Accelerometer and GPS Sensors: The accelerometer sensor readings and GPS location readings are collected at 1 minute intervals. The sensed data is collected along with the userID and timestamp, and it is stored into a file in the phone's internal storage which can be accessed only by the McSense application. This data will be uploaded to the application server on completion of the task (which consists of many data points). Using the collected sensed data of accelerometer readings and GPS readings, we can identify users activities like walking, running, or driving. By observing the daily activities, we could find out how much exercise each student is getting daily and derive interesting statistics such as which department has the most active and healthy students.

Automated Sensing Task using Bluetooth Radio: In this automated sensing task, the user's Bluetooth radio is used to perform periodic (*every 5mins*) Bluetooth scans until the task expires; the task reports the discovered Bluetooth devices with their location back to the McSense server on its completion. The sensed data from Bluetooth scans can provide interesting social information such as how often McSense users are near to each other. Also, it can identify groups who are frequently together to determine the level of social interaction of certain people.

Automated Resources usage Sensing Task: In this automated sensing task, the usage of user's smart phone resources is sensed and reported back to the McSense server. Specially, the report contains the mobile applications' usage, the network usage, the periodic WiFi scans, and the battery level of the smart phone. While logging the network usage details, this automated task also logs overall device network traffic (tx/rx) and per-application network traffic.

3.3 Field Study

The providers (students shown in Table 3.1) registered with McSense and submitted data together with their userID. On the application server, we periodically posted various sensing tasks. Some tasks had high monetary value associated with its successful completion. But,

Table 3.1 Demographic Information of the Students

Total participants	58
Males	90%
Females	10%
Age 16-20	52%
Age 21-25	41%
Age 26-35	7%

few tasks are offered with very low monetary incentives and mostly relied on volunteering students. As tasks are submitted to the application server, they appear on the participant's phones where our application has been installed. Each task will contain a task description, its duration and a certain amount of money. The students can then use the available tasks tab to view and accept any available task. Upon successful completion of the task, the students will accumulate credits (paid in cash at the end of the study). We conducted the study for approximately 2 months.

Details on the automated accelerometer and GPS sensing tasks posted on server for our research:

1. We posted automated tasks only between 6AM - 12PM. Users can accept these tasks when they are available. When user accept the auto sensing task, then the task starts running in the background automatically. Furthermore, users must have WiFi and GPS radios switched on for accepting Automated Sensing tasks.

2. Each day auto sensing tasks expire at night 10pm. Server checks for threshold (6hrs) for the total sensing time of the task. If it is below 6hrs, then the task is marked as "Unsuccessfully Completed" otherwise "Successfully Completed".

3. Auto sensing tasks always run as a background service. On start or resume of this service, the service always check the current time from the server. Thus, even when the user sets wrong time on his mobile, the task will always know the correct current time and will stop sensing after night 10PM.

4. Long term auto sensing tasks are posted for multiple days. Users are paid only for the number of days they successfully complete the task. Same threshold logic is applied to each day for these multi-day tasks.

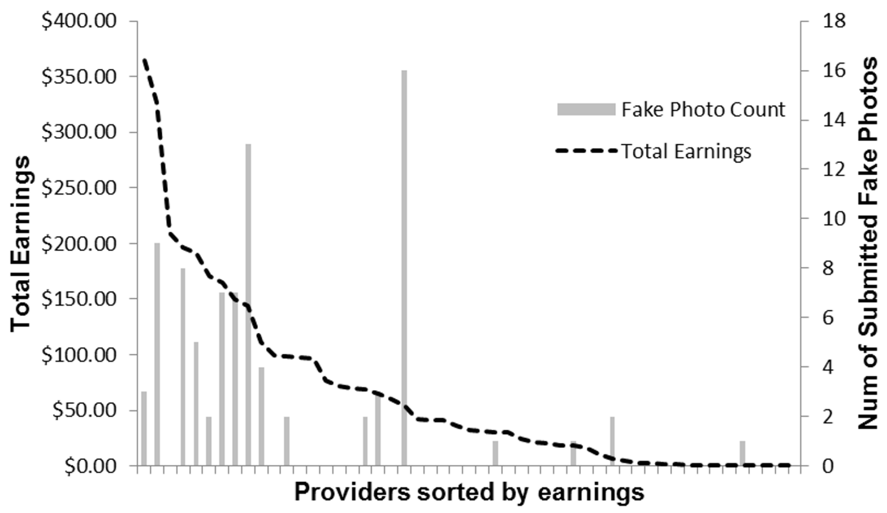


Figure 3.3 Correlation of Earnings and Fake photos.

Manual tasks like photo tasks are not completed when user simply accepts it. Users have to finish the task manually from the Accepted Tasks Tab by taking the photo from the requested location. Users were asked to take general photos from events on the NJIT campus. Once the photos are successfully uploaded to the application server, the task is considered successfully completed after a basic validation is performed (photos are manually validated for ground truth).

3.4 Results

In this section, we present our insights from the analysis of the data collected from the field study. In addition, we present observations of the user survey that was collected from users at the end of the field study to understand the participant's opinion on location privacy and usage of phone resources.

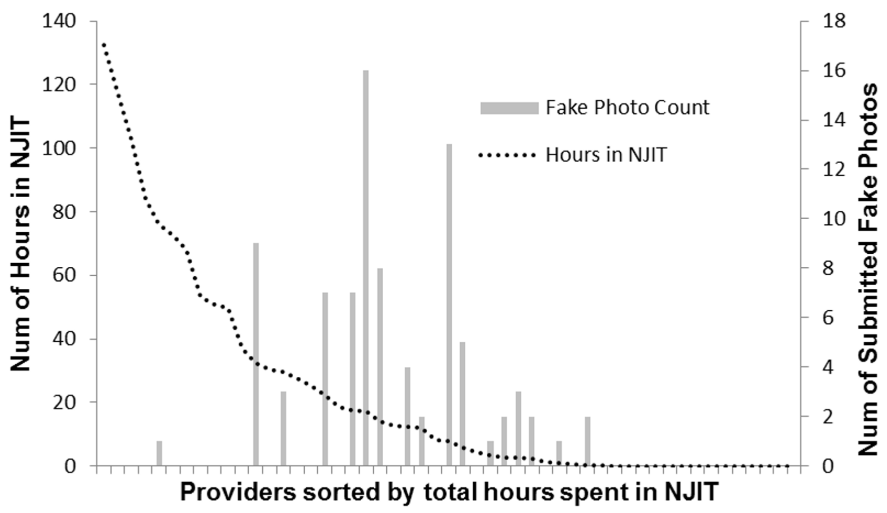


Figure 3.4 Correlation of user location and Fake photos.

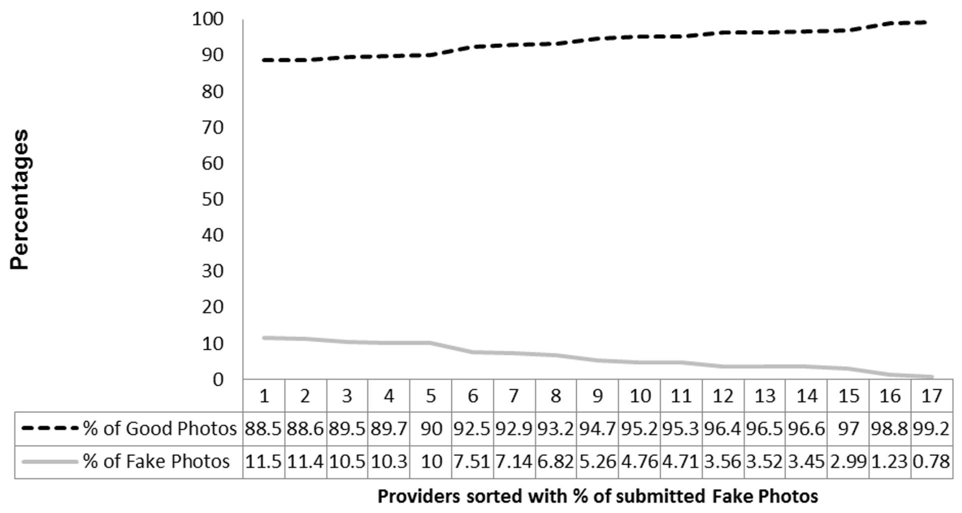
3.4.1 Correlation of User Earnings and Fake Photos

To understand the correlation between the user earnings and the number of fake photos submitted, we plot the data collected from the McSense crowd sensing field study. The experimental results in Figure 3.3 show that the users who submitted most of the fake photos are among the top 20 high earners (with an exception of 4 low earning users who submitted fake photo once or twice). This is an interesting observation that can be leveraged to improve the sensed data quality.

3.4.2 Correlation of Location and Fake Photos

We ask the question "Is there any correlation between the amount of time spent by users on campus and the number of submitted fake photos?". As suspected, the users who spent less time on campus have submitted more fake photos. This behavior can be observed in Figure 3.4.

Figure 3.4 shows the number of fake photos submitted by each user, with the users sorted by the total hours spent on the NJIT campus. The participants' total hours recorded



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
╌ % of Good Photos	88.5	88.6	89.5	89.7	90	92.5	92.9	93.2	94.7	95.2	95.3	96.4	96.5	96.6	97	98.8	99.2
— % of Fake Photos	11.5	11.4	10.5	10.3	10	7.51	7.14	6.82	5.26	4.76	4.71	3.56	3.52	3.45	2.99	1.23	0.78

Figure 3.5 Photo counts of 17 cheating people.

at NJIT campus are the hours that are accumulated from the sensed data collected from “Automated Sensing task” described in the “Tasks Developed for McSense” Section 3.2. The NJIT location is considered to be a circle with a radius of 0.5 miles. If the user is in circle, then she is considered to be at NJIT. For most of the submitted fake photos with the false location claim, the users claimed that they are at a campus location where the photo task is requested, but actually they are not frequent visitors on the campus.

3.4.3 Malicious User: Menace or Nuisance?

The photos submitted by malicious users are plotted in Figure 3.5. The data show that malicious users have submitted good photos at a very high rate than compared to the fake photos. These malicious users are among the high earners, so they are submitting more data than the average user. Thus, it may not be a good idea to remove the malicious users from the system as soon as they are caught cheating.

3.5 Chapter Summary

In this chapter, we outlined the general overview of the basic centralized design of our Mobile Crowd Sensing system. We firstly defined the interacting entities and the system architecture and protocols. We then discussed the prototype implementation. Subsequently, the tasks developed for McSense are presented. Finally, our field study and its results are also presented.

CHAPTER 4

ILR: IMPROVING LOCATION RELIABILITY IN CROWD SENSED DATA

This chapter presents ILR [102, 103], a scheme which Improves the Location Reliability of mobile crowd sensed data with minimal human efforts. The scheme also detects false location claims associated with the sensed data. We evaluate the proposed scheme on real-world data by developing McSense, a mobile crowd sensing system which is deployed on the Android market. Based on security analysis and simulation results, we show that ILR works well at various node densities.

The rest of the chapter is organized as follows. Section 4.1 defines the assumptions and Section 4.2 introduces the adversarial model. Section 4.3 describes the ILR scheme and Section 4.4 presents the security analysis. Section 4.5 presents the evaluation of field study and the simulation setup and results are presented in Section 4.6. The chapter concludes in Section 4.7.

4.1 Assumptions

We consider that McSense posts tasks to collect sensing data on behalf of clients. Providers execute any available task and report the sensed data back to McSense, which delivers it to clients pending validation. We assume that every provider performs Bluetooth scans at each location where it is collecting sensing data. We also assume that the sensed data reported by providers for a given task always includes location, time, and a Bluetooth scan. Note that Bluetooth scans can have a much lower frequency than the sensor sampling frequency. In the context of this chapter, we use the terms “data point” and “task” interchangeably.

4.2 Adversarial Model

We assume all the mobile devices are capable of determining their location using GPS. We also assume McSense is trusted and the communication between mobile users and McSense is secure. In our threat model, we consider that any provider may act maliciously and may lie about their location.

A malicious provider can program the device to spoof a GPS location [73] and start providing wrong location data for all the crowd sensing data requested by clients. Regarding this, we consider three threat scenarios, where 1) The provider does not submit the location and Bluetooth scan with a sensing data point; 2) The provider submits a Bluetooth scan associated with a sensing task, but claims a false location; 3) The provider submits both a false location and a fake Bluetooth scan associated with a sensing data point. In Section 4.4, we will discuss how these scenarios are addressed by ILR.

We do not consider colluding attack scenarios, where a malicious provider colludes with other providers to show that she is present in the Bluetooth co-location data of others. It is not practically easy for a malicious provider to employ another colluding user at each sensing location. Additionally, these colluding attacks can be reduced by increasing the minimum node degree requirement in co-location data of each provider (i.e., a provider P must be seen in at-least a minimum number of other providers' Bluetooth scans at her claimed location and time). Therefore, it becomes difficult for a malicious provider to create a false high node degree by colluding with real co-located people at a given location and time.

Finally, the other class of attacks that are out of scope for our current scheme are attacks in which a provider is able to "fool" the sensors to create false readings (e.g., using the flame of a lighter to create the false impression of a high temperature), but submits the right location and Bluetooth scan associated with this sensing task.

4.3 Proposed Scheme

In this section we present the ILR scheme which Improves the Location Reliability of mobile crowd sensed data with minimal human efforts. We also describe the validation process used by McSense to detect false location claims from malicious providers.

Before going into the details of the scheme, we assume that the sensed data is already collected by the McSense system from providers at different locations. However, this sensed data is awaiting validation before being sent to the actual clients who requested this data.

For ILR, we will assume that the sensed data includes location, time and a Bluetooth scan performed at the task's location and time. The main idea of our scheme is to corroborate data collected from manual (photo) tasks with co-location data from Bluetooth scans. We describe next an example of how ILR uses the photo and co-location data.

4.3.1 An Example of ILR in Action

Figure 4.1 maps the data collected by several different tasks in McSense. The figure shows 9 photo tasks [marked as A to I] and 15 sensing tasks [marked as 1 to 15] performed by different providers at different locations. For each of these tasks, providers also report neighbors discovered through Bluetooth scans (i.e., Bluetooth scans). All these tasks are grouped into small circles using co-location data found in Bluetooth scans within a time interval t. For example, Photo task A and sensing tasks (1, 2, and 3) are identified as

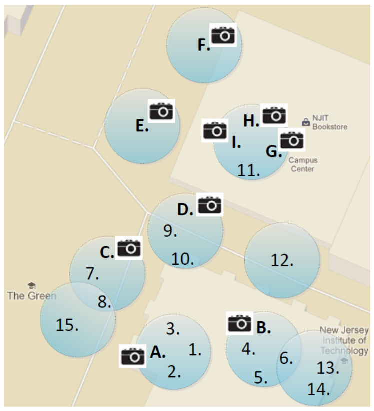


Figure 4.1 Example of McSense collected Photo tasks [A-I] and Sensing tasks [1-15] on the campus map, grouped using Bluetooth discovery co-location data.

co-located and grouped into one circle because they are discovered in each others Bluetooth scans.

In this example, McSense does not need to validate all the photo tasks mapped in the figure. Instead, McSense will first consider the photo tasks with the highest node degree ($NodeDegree$) by examining the co-located groups for photo task providers who have seen the highest number of other providers in Bluetooth scans around them. In this example we consider $NodeDegree \geq 3$. Hence, we see that photo tasks A, B, C, D, and G have discovered the highest number of providers around their location. Therefore, McSense will choose these 5 photo tasks for validation. These selected photo tasks are validated either manually or automatically (we discuss this in detail in Section 4.3.2). When validating these photo tasks, if the photo is not valid then its photo is rejected and McSense ignores its Bluetooth scans. If the photo is valid then McSense will consider the location of the validated photo as trusted because the validated photo is actually taken from the physical location requested in the task. However, it is very difficult to categorize every photo as a valid or a fake photo. Therefore some photos will be categorized as "unknown" when a decision cannot be made.

In this example, we assume that these 5 selected photos are successfully validated through manual verification. Next, using the transitivity property, McSense will extend the location trust of validated photos to other co-located providers' tasks which are found in the Bluetooth scans of the A, B, C, D, and G photo tasks. For example, A will extend trust to the tasks 1, 2, and 3, and B will extend trust to tasks 4, 5, and 6. Now task 6 will extend its trust to tasks 13 and 14. Finally, after the end of this process, McSense system will have 21 successfully validated tasks out of a total of 24 tasks. In this example, McSense required manual validation for just 5 photo tasks, but using the transitive property it was able to extend the trust to 16 additional tasks automatically. Only 3 tasks (E, F, and 12) are not validated as they lack co-location data around them.

4.3.2 The ILR Scheme

The ILR scheme has two phases as shown in Figure 4.2. "Phase 1: Photo Selection" elects the photo tasks to be validated. And "Phase 2: Transitive Trust" extends the trust to data points co-located with the tasks elected in Phase 1.

Phase 1 - Photo Selection Using collected data from Bluetooth scans of providers, we construct a connected graph of co-located data points for a given location and within a time interval t (these are the same groups represented in circles as discussed in the above example). From these graphs, we elect the photo tasks that have node degree greater than a threshold (N_{th}).

These selected photo tasks are validated either by humans or by applying computer vision techniques. For manual validation, McSense could rely on other users recruited from Amazon MTurk [13] for example. In order to apply computer vision techniques, first we

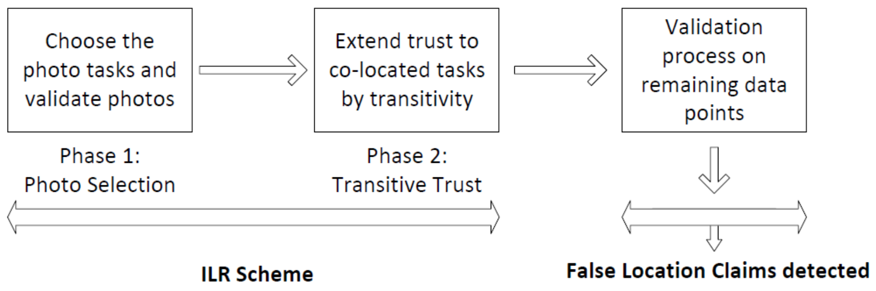

Choose the photo tasks and validate photos
Extend trust to co-located tasks by transitivity
Validation process on remaining data points
Phase 1: Photo Selection
Phase 2: Transitive Trust
ILR Scheme
False Location Claims detected

Figure 4.2 The phases of the ILR scheme.

need to collect ground truth photos to train image recognition algorithms. One alternative is to have trusted people collect the ground truth photos. However, if the ground truth photos are collected through crowd sensing, then they have to be manually validated as well. Thus, reducing the number of photos that require manual validation is an important goal for both manual and automatic photo recognition. Once the validation is performed, the location of the validated photo task is now considered to be reliable because the validated photos have been verified to be taken from the physical location requested in the task. For simplicity, we will refer to the participants who contributed valid photo tasks with reliable location and time as “Validators”.

Phase 2 - Transitive Trust In this phase, we rely on the transitive property and extend the trust established in the Validator’s location to other co-located data points. In short, if the photo is valid, the trust is extended to co-located data points found in Bluetooth discovery of the validated photo task. In current scheme, trust is extended until all co-located tasks are trusted or no other task is found, alternately McSense can set a TTL (Time To Live) on extended trust. The following two steps are performed in this phase:

- (Step 1) Mark co-located data points as trusted: For each task co-located with a validated photo task, mark the task’s location as trusted.

- (Step 2) Repeat Step 1 for each newly validated task until all co-located tasks are trusted or no other task is found.

Algorithm 1 ILR Validation Pseudo-Code

Notation:

```
TList:  Tasks List which are not yet marked trusted after completing first two
phases of ILR scheme.
T: Task submitted by a Provider.
L: Location of the Photo or Sensing Task (T).
t:  Timestamp of the Photo or Sensing Task (T).
hasValidator(L, t):  Function to check, if already there exist any valid data
point at task T's location and time.
```

validationProcess():
run to validate the location of each task in TList

```
1: for each task T in TList do
2:     if $hasValidator(L, t) == TRUE$ then
3:         Update task T with false location claim at (L, t)
```

4.3.3 Validation Process

After executing the two phases of ILR scheme, all the co-located data points are validated successfully. If any malicious provider falsely claims one of the validated task's location at the same time, then the false claim will be detected in the validation step. Executing the validation process shown in algorithm 1 will help us detect wrong location claims around the already validated location data points. For instance, if we consider task 12 from Figure 4.1 as a malicious provider claiming a false location exactly at photo task A's location and time, then task 12 will be detected in the validationProcess() as it is not co-located in the Bluetooth scans of photo task A. In addition to the validation process, McSense will also do a basic spatio-temporal correlation check to ensure that the provider is not claiming location at different places at same time.

4.4 Security Analysis

The goal of the ILR scheme is to establish the reliability of the sensed data by validating the claimed location of the data points. In addition, ILR seeks to detect false claims made by malicious participants.

ILR is able to handle all the three threat scenarios presented in our adversarial model (Section 4.2). In the first threat scenario, when there is no location and Bluetooth scan submitted along with the sensed data, the sensed data of that task is rejected and the provider will not be paid by McSense.

In the second threat scenario, when a provider submits its Bluetooth discovery with a false location claim, then McSense will detect the provider in it neighbors' Bluetooth scans at a different location using the spatio-temporal correlation check and will reject the task's data.

Finally, when a provider submits fake Bluetooth discovery with a false location claim, then the scheme looks for any validator around the claimed location and if it finds anyone, then the sensed data associated with the false location claim is rejected. But, if there is no validator around the claimed location, then the data point is categorized as "unknown".

As discussed in our adversarial model (Section 4.2), sensed data submitted by malicious colluding attackers could be filtered to a certain extent in McSense by setting the node degree threshold (N_{th}) to the minimum node degree requirement requested by the client.

4.5 Experimental Evaluation: Field Study

The providers (students shown in Table 3.1) registered with McSense and submitted data together with their userID. Both phases of ILR and the validation process are executed on

Table 4.1 Photo Task Reliability

	Number of photo tasks
Total photos	1784
Num of photos with Bluetooth scans (manually validated in ILR)	204
Trusted data points added by ILR	148

data collected from the providers, and we acted as the clients collecting the sensed data in these experiments.

The location data is mostly collected from the university campus (0.5 miles radius). The main goal of these experiments is to determine how efficiently the ILR scheme can help McSense to validate the location data and detect false location claims. ILR considers the Bluetooth scans found within 5min interval of measuring the sensor readings for a sensing task.

Table 4.1 shows the total photo tasks that are submitted by people; only 204 photo tasks are having Bluetooth scans associated with them. In this data set, we considered the $NodeDegree \geq 1$, therefore we used all these 204 photo tasks with Bluetooth scans in Phase-1 to perform manual validation, and then in Phase-2 we are able to automatically extend the trust to 148 new location data points through the transitive closure property of ILR.

To capture the ground truth, we manually validated all the photos collected by McSense in this study and identified that we have a total of 45 fake photos submitted to McSense from malicious providers, out of which only 16 fake photo tasks are having Bluetooth scans with false location claims. We then applied ILR to verify how many of these 16 fake photos can be detected.

We were able to catch 4 users who claimed wrong locations to make money with fake photos, as shown in Table 4.2. Since the total number of malicious users involved in the 16

Table 4.2 Number of False Location Claims

	Detected by ILR scheme	**Total**	**Percentage Detected**
Tasks with False Location claim	4	16	25%
Cheating People	4	10	40%

Table 4.3 Simulation Setup for the ILR Scheme

Parameter	**Value**
`Number of nodes`	200
`% of tasks with false location claims`	10, 15, 30, 45, 60
`Bluetooth transmission range`	10m
`Simulation time`	2hrs
`User walking speed`	1m/sec
`Node Density`	2, 3, 4, 5
`Bluetooth scan rate`	1/min

fake photo tasks is 10, ILR was able to detect 40% of them. Finally, ILR is able to achieve this result by validating only 11% of the photos (i.e., 204 out of 1784).

4.6 Simulations

This section presents the evaluation of the ILR scheme using the NS-2 network simulator. The two main goals of the evaluation are: (1) Estimate the right percentage of photo tasks needed in Phase 1 to bootstrap the ILR scheme, and (2) Quantify the ability of ILR to detect false location claims at various node densities.

4.6.1 Simulation Setup

The simulation setup parameters are presented in Table 4.3. Given a simulation area of 100m x 120m, the node degree (i.e., average number of neighbors per user) is slightly higher than 5. We varied the simulation area to achieve node degrees of 2, 3, and 4. We consider low walking speeds (i.e., 1m/sec) for collecting photos. In these simulations, we

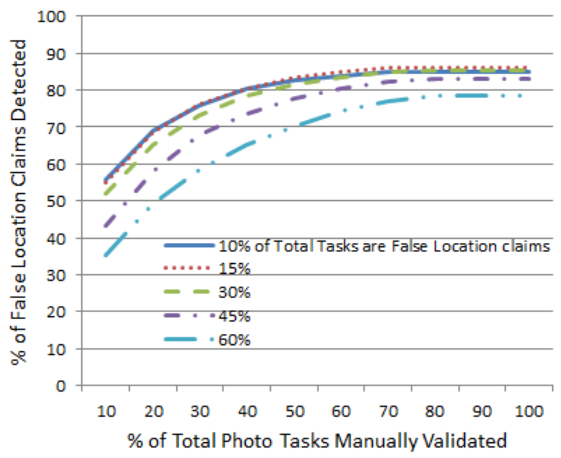


Figure 4.3 ILR performance as function of the percentage of photos manually validated in phase 1. Each curve represents a different percentage of photos with fake locations.

considered all tasks as photo tasks. A photo task is executed every minute by each node. Photo tasks are distributed evenly across all nodes. Photo tasks with false location claims are also distributed evenly across several malicious nodes. We assume the photo tasks in ILR's phase 1 are manually validated.

After executing the simulation scenarios described below, we collect each photo task's time, location, and Bluetooth scan. As per simulation settings, we will have 120 completed photo tasks per node at the end of the simulation (i.e 24,000 total photo tasks for 200 nodes). Over this collected data, we apply the ILR validation scheme to detect false location claims.

4.6.2 Simulation Results

Varying percentage of false location claims. In this set of experiments, we vary the percentage of photo tasks with false location claims. The resulting graph, plotted in Figure- 4.3, has multiple curves as a function of the percentage of photo tasks submitting false location. This graph is plotted to gain insights on what will be the right percentage of photo tasks needed in Phase 1 to bootstrap the ILR scheme. Next, we analyze Figure 4.3:

- **Low count of malicious tasks submitted:** When 10% of total photo tasks are submitting false location, Figure 4.3 shows that just by using 10% of the total photo tasks validated in Phase1, the ILR scheme can detect 55% of the false location claims. This figure also shows that in order to detect more false claims, we can use up to 40% of the total photo tasks in Phase 1 to detect 80% of the false location tasks. Finally, Figure 4.3 shows that increasing the percentage of validated photo tasks above 40% will not help much as the percentage of detected false tasks remains the same.
- **High count of malicious tasks submitted:** When 60% of the total photo tasks are submitting false location, Figure 4.3 shows that ILR can still detect 35% of the false claims by using 10% of the total photo tasks in Phase 1. But in this case, ILR scheme requires more validated photo tasks(70%) to catch 75% of the false claims. This is because by increasing the number of malicious tasks, the co-location data is reduced and therefore ILR cannot extend trust to more location claims in its Phase 2.

Therefore, we conclude that the right percentage of photo tasks needed to bootstrap the ILR scheme is proportional to the expected false location claims (which can be predicted using the history of the users' participation).

Node density impact on the ILR scheme. In this set of experiments, we assume that 10% of the total photo tasks are submitting false locations. In Figure 4.4 we analyze the impact of node density on the ILR scheme. We seek to estimate the minimum node density required to achieve highly connected graphs to extend the location trust transitively to more co-located nodes.

- **High Density:** When simulations are run with node density of 5, Figure 4.4 shows the ILR scheme can detect the highest percentage (>85%) of the false location claims. The figure also shows similarly high results even for a node density of 4.
- **Low Density:** When simulations are run with node density of 2, we can see that the ILR scheme can still detect 65% of the false location tasks using 50% of the total photo tasks in Phase 1. For this node density, even after increasing the number of validated photo tasks in Phase 1, the percentage of detected false claims does not increase. This is because of there are fewer co-located users at low node densities.

Therefore, we conclude that the ILR scheme can efficiently detect false claims with a low number of manual validations, even for low node densities.

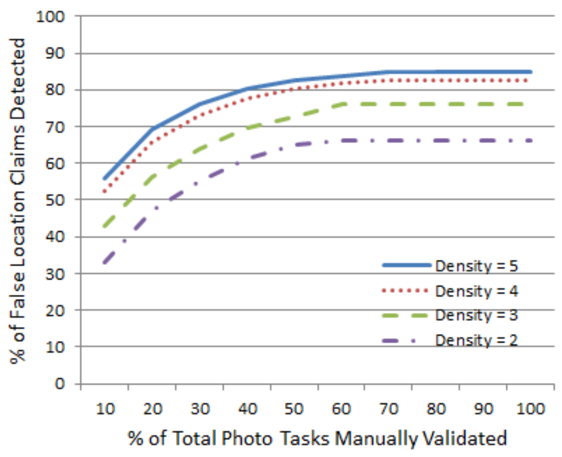


Figure 4.4 ILR performance as function of the percentage of photos manually validated in phase 1. Each curve represents a different network density represented as average number of neighbors per node.

4.7 Chapter Summary

This chapter presented ILR, a scheme which increases the reliability of mobile crowd sensed data with minimal human efforts. ILR also detects false location claims associated with the sensed data. Based on security analysis and simulation results, we show that ILR works well at varying node densities. We evaluated the proposed scheme on real data, by developing McSense – a mobile crowd sensing system – which is deployed in the Android market. The analysis on sensed data collected from over 50 users during a two-month period demonstrated that ILR is efficient in attaining location data reliability and in detecting a significant percentage of false location claims.

CHAPTER 5

LINK: LOCATION AUTHENTICATION THROUGH IMMEDIATE NEIGHBORS KNOWLEDGE

This chapter proposes LINK (Location authentication through Immediate Neighbors Knowledge) [104], a secure location authentication protocol in which users help authenticate each other's location claims. LINK associates trust scores with users and nearby mobile devices belonging to users with high trust scores play similar roles with the trusted beacons-/base stations in existing location authentication solutions. The main idea is to leverage the neighborhood knowledge available through short-range wireless technologies, such as Blue-tooth which is available on most cell phones, to verify if a user is in the vicinity of other users with high trust scores.

The rest of the chapter is organized as follows. Section 5.1 defines the assumptions and the adversarial model is presented in Section 5.2. Section 5.3 describes the LINK protocol, and Section 5.4 analyzes its security. Section 5.5 presents the simulation results. The implementation and experimental evaluation are presented in Sections 5.6 and 5.7. The chapter concludes in Section 5.8.

5.1 Assumptions

This section defines the interacting entities in our environment, and the assumptions we make about the system for LINK protocol.

Interacting entities. The entities in the system are:

- *Claimer*: The mobile user who claims a certain location and subsequently has to prove the claim's authenticity.
- *Verifier*: A mobile user in the vicinity of the claimer (as defined by the transmission range of the wireless interface, which is Bluetooth in our implementation). This user receives a request from the claimer to certify the claimer's location and does so by sending a message to the LCA.
- *Location Certification Authority (LCA)*: A service provided in the Internet that can be contacted by location-based services to authenticate claimers' location. All mobile users who need to authenticate their location are registered with the LCA.
- *Location-based Service (LBS)*: The service that receives the location information from mobile users and provides responses as a function of this location.

We assume that each mobile device has means to determine its location. This location is considered to be approximate, within typical GPS or other localization systems limits. We assume the LCA is trusted and the communication between mobile users and the LCA occurs over secure channels, e.g., the communication is secured using SSL/TLS. We also assume that each user has a pair of public/private keys and a digital certificate from a PKI. Similarly, we assume the LCA can retrieve and verify the certificate of any user. All communication happens over the Internet, except the short-range communication between claimers and verifiers.

We choose Bluetooth for short-range communication in LINK because of its pervasiveness in cell phones and its short transmission range (10m) which provides good accuracy for location verification. However, LINK can leverage WiFi during its initial deployment in order to increase the network density. This solution trades off location accuracy for number of verifiers.

LCA can be a bottleneck and single point of failure in the system. Currently, we do not address this issue, but standard distributed systems techniques can be used to improve the LCA's scalability and fault-tolerance. For example, an individual LCA server/cluster can be assigned to handle a specific geographic region, thus reducing the communication overhead significantly (i.e., communication between LCA servers is only required to access user's data when she travels away from the home region). LINK also needs significant memory and storage space to store historic data about each pair of users who interact in the system. To alleviate this issue, a distributed implementation of the LCA could use just the recent history (e.g., past month) to compute trust score trends, use efficient data intensive parallel computing frameworks such as *Hadoop* [105] to pre-compute these trends offline, and employ distributed caching systems such as *memcached* [106] to achieve lower latency for authentication decisions.

5.2 Adversarial Model

Any claimer or verifier may be malicious. When acting individually, malicious claimers may lie about their location. Malicious verifiers may refuse to cooperate when asked to certify the location of a claimer and may also lie about their own location in order to slander a legitimate claimer. Additionally, malicious users may perform stronger attacks by colluding with each other in order to verify each other's false claims. Colluding users may also attempt two classic attacks: mafia fraud and terrorist fraud [107].

We do not consider selfish attacks, in which users seek to reap the benefits of participating in the system without having to expend their own resources (e.g., battery). These attacks are solved by leveraging the centralized nature of LCA, which enforces a tit-for-tat mechanism, similar to those found in P2P protocols such as BitTorrent [108], to incentivize nodes to participate in verifications. Only users registered with the LCA can participate in the system as claimers and verifiers. The tit-for-tat mechanism requires the verifiers to submit verifications in order to be allowed to submit claims. New users are allowed to submit a few claims before being requested to perform verifications.

Finally, we rely on the fact that a user cannot easily obtain multiple user IDs because the user ID is derived from a user certificate and obtaining digital certificates is not cheap; this deters Sybil attacks [109]. Further, techniques such as [110, 111], complimentary to our protocol, can be used to address these attacks.

5.3 Protocol Design

This section presents the basic LINK operation, describes the strategies used by LCA to decide whether to accept or reject a claim, and then details how trust scores and verification history are used to detect strong attacks from malicious users who change their behavior over time or collude with each other.

5.3.1 Basic Protocol Operation

All mobile users who want to use LINK must register with the LCA. During registration, the LCA generates a userID based on the user's digital certificate. Users of the system do not have to register with the LBS because they submit their LCA-generated userID to the LBS together with their requests. By not requiring the users to register with each LBS, we simplify the protocol.

At registration time, the LCA assigns an initial trust score for the user (which can be set to a default value or assigned based on other criteria). Trust scores are maintained and used by the LCA to decide the validity of location claims. A user's trust score is additively increased when her claim is successfully authenticated and multiplicatively decreased otherwise in order to discourage malicious behavior. This policy of updating the scores is

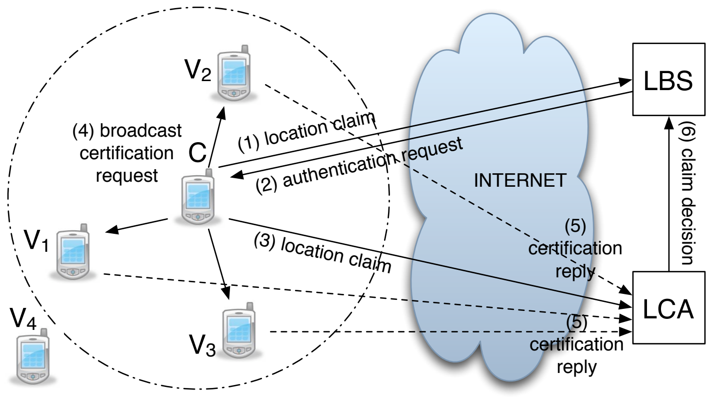


Figure 5.1 Basic Protocol Operation (where C = claimer, V$_i$ = verifiers, LBS = Location-Based Service, LCA = Location Certification Authority).

demonstrated to work well for the studied attacks, as shown in Section 5.5. The values of all trust score increments, decrements, and thresholds are presented in the same section. A similar trust score updating policy has been shown to be effective in P2P networks as well [86].

LCA also maintains a verification count of each user to determine whether the user is participating in verifications or simply using the system for her own claims. A user needs to perform at least VC$_{th}$ verifications in order to be allowed to submit one claim (VC$_{th}$ is the verification count threshold). Each time a verifier submits a verification, her verification count is incremented by 1, and each time a claimer submits a request to LCA, her verification count is decreased by VC$_{th}$. If a claimer's verification count reaches below VC$_{th}$, the claimer is informed to participate in other claimers verifications and her claims are not processed until her verification count reaches above VC$_{th}$.

Figure 5.1 illustrates the basic LINK operation. The pseudo-code describing the actions of claimers, verifiers, LCA, and LBS is presented in Algorithm 2. LINK messages are signed. When we say that a protocol entity sends a signed message, we mean that the entity computes a digital signature over the entire message and appends this signature at the end of the message. In step 1, a user (the claimer) wants to use the LBS and submits her location (Claimer Pseudo-code, line 5). The LBS then asks the claimer to authenticate her location (step 2) (LBS Pseudo-code, line 3). In response, the claimer will send a signed message to LCA (step 3) which consists of *(userID, serviceID, location, seq-no, serviceID, verifiers' IDs)* (Claimer Pseudo-code, line 12). The sequence number (*seq-no*) is used to protect against replay attacks (to be discussed in Section 5.4). The *serviceID* is an identifier of the LBS. The verifiers' IDs consists of the list of verifiers discovered by the claimer's Bluetooth scan; in this way, LCA will ignore the certification replies received from any other verifiers (the purpose of this step is to defend against mafia fraud attacks as detailed in Section 5.4). Furthermore, the LCA timestamps and stores each newly received claim.

The claimer then starts the verification process by broadcasting to its neighbors a location certification request over the short-range wireless interface (step 4). This message is signed and consists of *(userID, serviceID, location, seq-no)*, with the same sequence number as the claim in step 3. The neighbors who receive the message, acting as verifiers for the claimer, will send a signed certification reply message to LCA (step 5) (Verifier Pseudo-code, line 8). This message consists of *(userID, location, certification-request)*, where the userID and location are those of the verifier and certification-request is the certification-request broadcasted by the claimer. The certification-request is included to allow the LCA to match the claim and its certification messages. Additionally, it proves that indeed the certification-reply is in response to the claimer's request.

The LCA waits for the certification reply messages for a short period of time and then starts the decision process (described next in Section 5.3.2). Finally, the LCA informs the LBS about its decision (step 6) (LCA Pseudo-code, line 9), causing the LBS to provide or deny service to the claimer (LBS Pseudo-code, line 8).

Algorithm 2 LINK Pseudo-Code

Claimer Pseudo-code:

```
1: // KC$_{pr}$ = private key of claimer
2: // KC$_{pu}$ = public key of claimer
3: // seq-no = provided by LCA before each claim
4: // LBS-request = [userID, location]
5: send(LBS, LBS-request)
6: receive(LBS, response)
7: if $response == Authenticate$ then
8:     certification-request = [userID, serviceID, location, seq-no];
9:     certification-request += sign(KC$_{pr}$, certification-request);
10:    verifiers = BluetoothDiscoveryScan();
11:    signed-verifiers = verifiers + sign(KC$_{pr}$, verifiers);
12:    send(LCA, certification-request, signed-verifiers);
13:    broadcast(verifiers, certification-request);
14:    receive(LBS, response);
```

Verifier Pseudo-code:

```
1: // KV$_{pr}$ = private key of verifier
2: // KV$_{pu}$ = public key of verifier
3: // Verifier enables Bluetooth Radio to discoverable mode
4: claimer = BluetoothDiscovered();
5: receive(claimer, certification-request);
6: certification-reply = [userID, location, certification-request];
7: certification-reply += sign(KV$_{pr}$, certification-reply);
8: send(LCA, certification-reply);
9: // Potential request from LCA to authenticate its location
```

LCA Pseudo-code:

```
1: receive(claimer, certification-request, verifiers)
2: if $verify(KC_{pu}, certification - request)$ and $verify(KC_{pu}, verifiers)$ then
3:     for $v = 1$ to verifiers.size() do
4:         receive(v, certification-reply[v]);
5:         if $verify(KV_{pu}, certification{-}reply[v])$ and $verify(KC_{pu}, getCertificationRequest(certification{-}reply[v]))$ then
6:             storeDataForDecision(certification-request, certification-reply)
7: decision = decisionProcess(claimer, getClaimLocation(certification-request));
8: LBS = getServiceID(certification-request);
9: send(LBS, decision);
```

LBS Pseudo-code:

```
1: receive(claimer, LBS-request);
2: if $locationAuthentication == Required$ then
3:     send(claimer, Authenticate);
4:     receive(LCA, decision);
5:     if $decision == Reject$ then
6:         send(claimer, ServiceDenied)
7:         return
8: send(claimer, response);
```

5.3.2 LCA Decision Process

In the following, we present the pseudo-code (Algorithm 3) and description of the LCA decision process. For the sake of clarity, this description skips most of the details regarding the use of historical data when making decisions, which are presented in Section 5.3.3.

Claimer lies The LCA first checks the user spatio-temporal correlation by comparing the currently claimed location with the location of the user's previously recorded claim (lines 1-3 in the algorithm). If it is not physically possible to move between these locations in the time period between the two claims, the new claim is rejected.

If the claimer's location satisfies the spatio-temporal correlation, the LCA selects only the "good" verifiers who responded to the certification request and who are in the list of verifiers reported by the claimer (lines 5-12). These verifiers must include in their certification reply the correct certification request signed by the claimer (not shown in the code) and must satisfy the spatio-temporal correlation themselves. Additionally, they must have trust scores above a certain threshold. We only use "good" verifiers because verifiers with low scores may be malicious and may try to slander the claimer. Nevertheless, the low score verifiers respond to certification requests in order to be allowed to submit their own certification claims (i.e., tit-for-tat mechanism) and, thus, potentially improve their trust scores.

Colluding users help claimer After selecting the "good" verifiers, the LCA checks if they are colluding with the claimer to provide false verifications (lines 13-15), and it rejects the claim if that is the case. This *collusion check* is described in detail in Section 5.3.3.

Contradictory verifications If the LCA does not detect collusion between the claimer and verifiers, it accepts or rejects the claim based on the difference between the sums of the trust scores of the two sets of verifiers (lines 16-23), those who agree with the location submitted by the claimer (Y_{sum}) and those who do not (N_{sum}). Of course, the decision is easy as long as all the verifiers agree with each other. The difficulty comes when the verifiers do not agree with each other. This could be due to two causes: malicious individual verifiers or verifiers colluding with the claimer who have escaped detection.

Notation of Algorithm 3:

c: claimer
$V = \{v_0, v_1, ... v_n\}$: Set of verifiers for claimer c
N_{set}: Set of verifiers who do not agree with c's location claim
v_i: The i-th verifier in V
T_{v_i}: Trust score of verifier v_i
T_c: Trust score of claimer
W_{v_i}: Weighted trust score of verifier v_i
L_c: Location claimed by claimer
L_{v_i}: Location claimed by verifier v_i
IND_{tr}: Individual threshold to eliminate the low-scored verifiers
AVG_{tr}: Average threshold to ensure enough difference in averages
VRF_{cnt}=0: Variable to hold recursive call count
INC=0.1: Additive increment
DEC=0.5: Multiplicative decrement
secVer[]: Array to hold the response of second level verifications

If the difference between the trust score sums of two sets of verifiers is above a certain threshold, the LCA decides according to the "winning" set. If it is low, the LCA does not make a decision yet. It continues by checking the trust score trend of the claimer (lines 24-27): if this trend is poor, with a pattern of frequent score increases and decreases, the claimer is deemed malicious and the request rejected. Otherwise, the LCA checks the score trends of the verifiers who disagree with the claimer (lines 28). If these verifiers are deemed

Algorithm 3 Decision Process Pseudo-Code

decisionProcess(c, L$_c$):

:run to validate the location $\mathtt{L}_c$ claimed by $\mathtt{c}$

```
1:  if $SpatioTempCorrelation(c) == FALSE$ then
2:      $T_c$ = $T_c$*DEC
3:      return Reject
4:  $VRF_{cnt}$++
5:  if $hasNeighbors(c) == TRUE$ then
6:      for $i = 0$ to $n$ do
7:          if $SpatioTempCorrelation(v_i) == FALSE$ then
8:              $T_{v_i}$ = $T_{v_i}$*DEC
9:              remove $v_i$ from set V
10:         $W_{v_i}$ = getUpdatedWeightScore($v_i$,c)
11:         if $W_{v_i} \leq IND_{tr}$ then
12:             remove $v_i$ from set V
13:     if $V.notEmpty()$ then
14:         if $checkCollusion(V, c) == TRUE$ then
15:             return Reject
16:         (absAVGDiff,$Y_{sum}$,$N_{sum}$) = getTrustScore(V,c)
17:         if $absAVGDiff \geq AVG_{tr}$ then
18:             if $Y_{sum} \geq N_{sum}$ then
19:                 $T_c$ = $T_c$+INC
20:                 return Accept
21:             else
22:                 $T_c$ = $T_c$*DEC
23:                 return Reject
24:         else
25:             if $trend(c) == POOR$ then
26:                 $T_c$ = $T_c$*DEC
27:                 return Reject
28:             else if $verifyScoreTrends(N_{set}) == POOR$ then
29:                 if $T_c \leq IND_{tr}$ then
30:                     return Ignore
31:                 else
32:                     $T_c$ = $T_c$-INC
33:                     return Accept
34:             else if $VRF_{cnt} == 2$ then
35:                 return Ignore {//2nd level claim is ignored}
36:             else
37:                 for $i = 0$ to $N_{set}$.size() do
38:                     secVer[i] = decisionProcess($v_i$,$L_{v_i}$)
39:                 if $Majority(secVer) == Ignore$ or $Reject$ then
40:                     $T_c$ = $T_c$+INC
41:                     return Accept
42:                 else
43:                     $T_c$ = $T_c$*DEC
44:                     return Reject
45: if $VRF_{cnt} == 2$ then
46:     return Ignore {//2nd level claim is ignored}
47: else if $trend(c) == POOR$ then
48:     $T_c$ = $T_c$*DEC
49:     return Reject
50: else if $T_c \leq IND_{tr}$ then
51:     return Ignore
52: else
53:     $T_c$ = $T_c$-INC
54:     return Accept
```

malicious, the claim is accepted. Otherwise, the claim is ignored, which forces the claimer to try another authentication later.

Note that even if the claim is accepted in this phase, the trust score of the claimer is preventively decremented by a small value (lines 32-33). In this way, a claimer who submits several claims which are barely accepted will receive a low trust score over time; this trust score will prevent future "accepts" in this phase (lines 29-30) until her trust scores improves.

If the trend scores of both the claimer and the verifiers are good, the verifiers are challenged to authenticate their location (lines 34-44). This second level verification is done through a recursive call to the same *decisionProcess()* function. This function is invoked for all verifiers who do not agree with the claimer (lines 37-38). If the majority of these verifiers cannot authenticate their location (i.e., Ignore or Reject answers), the claim is accepted (lines 39- 41). Otherwise, the claim is rejected. The VRF_{cnt} variable is used to keep track of the recursive call count. Since we perform just one additional verification level, the function returns when its value is 2.

Claimer with no verifiers The LCA deals with the case when no "good" verifiers are found to certify the claim in lines 45-54 of the algorithm (this includes no verifiers at all). If the claimer's trust score trend is good and her trust score is higher than a certain threshold, the claim is accepted. In this situation, the claimer's trust score is decreased by a small value $INC = 0.1$ as shown at line 53 in Algorithm 3 to protect against malicious claimers who do not broadcast a certification request to their neighbors when they make a claim. Over time, a user must submit claims that are verified by other users; otherwise, all her claims will be rejected.

5.3.3 Use of Historical Data in LCA Decision

The LCA maintains for each user the following historical data: (1) all values of the user's trust score collected over time, and (2) a list of all users who provided verifications for this user together with a verification count for each of them. These data are used to detect and prevent attacks from malicious users who change their behavior over time or who collude with each other.

Trust score trend verification. The goal of this verification is to analyze the historical trust values for a user and find malicious patterns. This happens typically when there are no good verifiers around a claimer or when the verifiers contradict each other with no clear majority saying to accept or reject the claim.

For example, a malicious user can submit a number of truthful claims to improve her trust score and then submit a malicious claim without broadcasting a certification request to her neighbors. Practically, the user claims to have no neighbors. This type of attack is impossible to detect without verifying the historical trust scores. To prevent such an attack, the LCA counts how many times has a user's trust score been decreased over time. If this number is larger than a certain percentage of the total number of claims issued by that user (10% in our implementation), the trend is considered malicious. More complex functions or learning methods could be used, but this simple function works well for many types of attacks, as demonstrated by our experiments.

Colluding users verification. Groups of users may use out-of-band communication to coordinate attacks. For example, they can send location certification messages to LCA on behalf of each other with agreed-upon locations. To mitigate such attacks, the LCA maintains an NxN matrix M that tracks users certifying each other's claims (N is the total number of users in the system). M[i][c] counts how many times user *i* has acted as verifier for user c. The basic idea is that colluding users will certify frequently each other's claims compared with the rest of the users in the system. However, identifying colluding users based solely on this criterion will not work because a spouse or a colleague at the office can certify very frequently the location of certain users. Furthermore, a set of colluding malicious users can use various permutations of subsets of malicious verifiers to reduce the chances of being detected.

Therefore, we propose two enhancements. First, the LCA algorithm uses weighted trust scores for verifiers with at least two verifications for a claimer. The weighted trust score of a verifier v is $W_v = T_v/\log_2(M[i][c])$, where T_v is the actual trust score of v. The more a user certifies another user's claims, the less its certifying information will contribute in the LCA decision. We choose a log function to induce a slower decrease of the trust score as the count increases. Nevertheless, a small group of colluding users can quickly end up with all their weighted scores falling below the threshold for "good" users, thus stopping the attack.

This enhancement is used until enough verification data is collected. Then, it is used in conjunction with the second enhancement, which discriminates between colluding malicious users and legitimate users who just happen to verify often for a claimer. LINK rejects a claim if the following conditions are satisfied for the claimer:

1. The number of claims verified by each potentially colluding user is greater than a significant fraction of the total number of claims issued by the claimer, and
2. The number of potentially colluding users who satisfy the first condition is greater than a significant fraction of the total number of verifiers for the claimer.

The pseudo-code for this function is presented in Algorithm 4. The function uses a dynamic threshold, W_{max}, which is a percentage of the total number of claims issued by

Algorithm 4 Collusion Check Pseudo-Code

Notation:

```
M: NxN matrix for keeping count of certifications for pairs (claimer, verifier)
W$_{max}$:  Threshold for count $w$
k:  Number of verifiers with $w \geq W_{max}$ for claimer c
V: Set of active verifiers for claimer c
T$_c$:  Trust score of claimer
T$_{v_i}$:  Trust score of i-th verifier in V
m:  Number of verifiers with $w > 0$ for claimer c
NumCl$_c$:  Total number of claims made by claimer c
W$_{rst}$:  reset value
```

checkCollusion(V, c):

```
1: $W_{max} = \beta * NumCl_c$
2: for $i = 0$ to $M.size$ do
3:     if $M[i][c] \geq W_{max}$ then
4:         k++;
5:     if $M[i][c] > 0$ then
6:         m++;
7: if $k/m \geq \alpha$ then
8:     $T_c = T_c * DEC$
9:     for $i = 0$ to $M.size$ do
10:        if $(M[i][c] \geq W_{max})$ and $((i \in V)$ or $(i.punished == FALSE))$ then
11:            $T_i = T_i * DEC$
12:            $i.punished = TRUE$
13:    return TRUE
14: else
15:    for $i = 0$ to $V.size$ do
16:        if $M[V[i]][c] \geq W_{max}$ then
17:            $M[V[i]][c] = W_{rst}$
18:    return FALSE
```

the claimer over time (in our implementation, this percentage, β, is set to 30%). Since W$_{max}$ is dynamically set, the algorithm can adapt its behavior over time. The function then computes the percentage of the verifiers who already reached W$_{max}$ (lines 3-6). If a significant number of verifiers reached this threshold, the algorithm considers them malicious and punishes them together with the claimer. Simulation results presented in Section 5.5 demonstrate the advantage of punishing the verifiers as well vs. a method that would punish only the claimer.

A higher percentage of users verifying often for the same claimer is a strong indication of malicious behavior (the parameter α, set to 10% in our implementation, is used for this purpose). The underlying assumption is that a legitimate user going about her business is verified by many users over time, and only a few of them would verify often (e.g., family, lab mates).

Lines 7-13 show how the decision is made. If the number of potentially colluding verifiers is greater than α, the claimer and those verifiers are punished. Note that we do not punish a verifier who did not participate in verifications for this claimer since last time she was punished (line 10). In this way, the verifiers can redeem themselves, but at the same time, their contribution is still remembered in M. Finally, as shown in lines 14-18, if the percentage of potentially colluding users is less than α, the counts for those users are reset to allow them to have a greater contribution in future verifications for the claimer (this is correlated with the weighted trust score described previously).

5.4 Security Analysis

LINK's goal is to prevent malicious users from claiming an incorrect location and to accept truthful location claims from legitimate users. The decision made by the LCA to accept or reject a claim relies on the trust scores of the users involved in this claim (i.e., claimer and verifiers). Thus, from a security perspective, the protocol's goal is to ensure that *over time* the trust score of malicious users will decrease, whereas the score of legitimate users will increase. LINK uses an additive increase and multiplicative decrease scheme to manage trust scores in order to discourage malicious behavior.

There are certain limits to the amount of adversarial presence that LINK can tolerate. For example, LINK cannot deal with an arbitrarily large number of malicious colluding verifiers supporting a malicious claimer because it becomes very difficult to identify the set of colluding users. Similarly, LINK cannot protect against users who accumulate high scores and very rarely issue false claims while pretending to have no neighbors (i.e., the user does not broadcast a certification request). An example of such situation is a "hit and run" attack, when the user does not return to the system after issuing a false claim. This type of behavior cannot be prevented even in other real-world systems that rely on user reputation, such as Ebay. Thus, we do not focus on preventing such attacks. Instead, we focus on preventing users that *systematically* exhibit malicious behavior. Up to a certain amount of adversarial presence, our simulation results in Section 5.5 show that the protocol is able to decrease over time the scores of users that exhibit malicious behavior consistently and to increase the scores of legitimate users.

All certification requests and replies are digitally signed, thus the attacker cannot forge them, nor can she deny messages signed under her private key. Attackers may attempt simple attacks such as causing the LCA to use the wrong certification replies to verify a location claim. LINK prevents this attack by requiring verifiers to embed the certification request in the certification reply sent to the LCA. This also prevents attackers from arbitrarily creating certification replies that do not correspond to any certification request, as they will be discarded by the LCA.

Another class of attacks claims a location too far from the previously claimed location. In LINK, the LCA prevents these attacks by detecting it is not feasible to travel such a large distance in the amount of time between the claims.

The LCA's decision making process is facilitated when there is a clear difference between the trust scores of legitimate and malicious users. This corresponds to a stage in which the user scores have stabilized (i.e., malicious scores have low scores and legitimate users have high scores). However, there may be cases when this score difference is not significant and it becomes challenging to differentiate between a legitimate verifier vouching against a malicious claimer and a malicious verifier slandering a legitimate claimer. In this case, the LCA's decision relies on several heuristic rules. The true nature of a user (malicious or legitimate) may be reflected in the user's score trend and the LCA can decide based on the score trends of the claimer and verifiers. The LCA may also potentially require the verifiers to prove their location. This additional verification can reveal malicious verifiers which are certifying a position claim (even though they are not in the vicinity of the claimed position), because the verifiers will not be able to prove their claimed location.

Replay Attack. Attackers may try to slander other honest nodes by intercepting their certification requests and then replaying them at a later time in a different location. However, the LCA is able to detect that it has already processed a certification request (extracted from a certification reply) because each such request contains a sequence number and the LCA maintains a record of the latest sequence number for each user. Thus, such duplicate requests will be ignored.

Individual Malicious Claimer or Verifier Attacks. We now consider individual malicious claimers that claim a false location. If the claimer follows the protocol and broadcasts the certification request, the LCA will reject the claim because the claimer's neighbors provide the correct location and prevail over the claimer. However, the claimer may choose not to broadcast the certification request and only contact the LCA. If the attacker has a good trust score, she will get away with a few false claims. The impact of this attack is limited because the attacker trust score is decreased by a small decrement for each such claim, and she will soon end up with a low trust score; consequently, all future claims without verifiers will be rejected. Accepting a few false claims is a trade-off we adopt in LINK in order to accept location claims from legitimate users that occasionally may have no neighbors.

An individual malicious verifier may slander a legitimate user who claims a correct location. However, in general, the legitimate user has a higher trust score than the malicious user. Moreover, the other (if any) neighbors of the legitimate user will support the claim. The LCA will thus accept the claim.

Colluding Attack. A group of colluding attackers may try to verify each other's false locations using out-of-band channels to coordinate with each other. For example, one attacker claims a false position and the other attackers in the group support the claim. LINK deals with this attack by recording the history of verifiers for each claimer and gradually decreasing the contribution of verifiers that repeatedly certify for the same claimer (see Section 5.3.3). Even if this attack may be successful initially, repeated certifications from the same group of colluding verifiers will eventually be ignored (as shown by our simulations in Section 5.5).

Mafia Fraud. In this attack, colluding users try to slander honest claimers without being detected, which may lead to denial-of-service. For example, a malicious node M_1 overhears the legitimate claimer's certification request and relays it to a remote collaborator M_2; M_2 then re-broadcasts this certification request pretending to be the legitimate claimer. This results in conflicting certification replies from honest neighbors of the legitimate claimer and honest neighbors of M_2 from a different location. This attack is prevented in LINK because the LCA uses the list of verifiers reported by the legitimate claimer from its Bluetooth scan. Therefore, LCA ignores the certification replies of the extra verifiers who are not listed by the legitimate claimer. These extra verifiers are not punished by LCA, as they are being exploited by the colluding malicious users. Furthermore, it is difficult for colluding users to follow certain users in order to succeed in such an attack.

Limitations and Future Work. The thresholds in the protocol are set based on our expectations of normal user behavior. However, they can be modified or even adapted dynamically in the future.

LINK was designed under the assumption that users are not alone very often when sending the location authentication requests. As such, it can lead to significant false positive rates for this type of scenario. Thus, LINK is best applicable to environments in which user density is relatively high.

Terrorist fraud is another type of attack in which one attacker relays the certification request to a colluding attacker at a different location, in order to falsely claim the presence at that different location. For example, a malicious node M_1 located at location L_1 relays its certification request for location L_2 to collaborator M_2 located at L_2. M_2 then broadcast M_1's request to nearby verifiers. Verifiers certify this location request, and as a result the LCA falsely believes that M1 is located at L_2. This attack is less useful in practice and is hard to mount, as it requires one of the malicious users to be located at the desired location. From a philosophical perspective, attacker M_1 *has a physical presence* at the falsely claimed location through its collaborator M_2, so this attack is arguably not a severe violation of location authentication. The attack could be prevented by using distance bounding protocols [112–114]. However, applying such protocols is non-trivial. Two major issues are: (a) These protocols consider the verifiers as trusted where as LINK verifiers are not trusted and (b) special hardware (i.e., radio) not available on current phones may be needed to determine distances with high accuracy.

In addition, a group of colluding attackers may attempt a more sophisticated version of the terrorist fraud attack, in which the malicious collaborators share their cryptographic credentials to sign the false location claims for each other. In practice, this scenario is unlikely because a malicious user will be reluctant to share her credentials with another malicious user [114].

We implicitly assume that all mobile devices have the same nominal wireless transmission range. One can imagine ways to break this assumption, such as using non-standard wireless interfaces that can listen or transmit at higher distances such as the BlueSniper rifle from DEFCON ’04. In this way, a claimer may be able to convince verifiers that she is indeed nearby, while being significantly farther away. Such attacks can also be prevented similar to the above solution for terrorist fraud using distance bounding protocols [112–114].

Location privacy could be an issue for verifiers. Potential solutions may include rate limitations (e.g., number of verifications per hour or day), place limitations (e.g., do not participate in verifications in certain places), or even turning LINK off when not needed for claims. However, the tit-for-tat mechanism requires the verifiers to submit verifications in order to be allowed to submit claims. To protect verifier privacy against other mobile users in proximity, the verification messages could be encrypted as well.

5.5 Simulations

This section presents the evaluation of LINK using the ns-2 simulator. The two main goals of the evaluation are: (1) Measuring the false negative rate (i.e., percentage of accepted malicious claims) and false positive rate (i.e., percentage of denied truthful claims) under various scenarios, and (2) Verifying whether LINK’s performance improves over time as expected.

Table 5.1 Simulation Setup for the LINK Protocol

Parameter	Value
Simulation area	100m x 120m
Number of nodes	200
% of malicious users	1, 2, 5, 10, 15
Colluding user group size	4, 6, 8, 10, 12
Bluetooth transmission range	10m
Simulation time	210min
Node speed	2m/sec
Claim generation rate (uniform)	1/min, 1/2min, 1/4min, 1/8min
Trust score range	0.0 to 1.0
Initial user trust score	0.5
``Good'' user trust score threshold	0.3
Low trust score difference threshold	0.2
Trust score increment	0.1
Trust score decrement - common case	0.5
Trust score decrement - no neighbors	0.1

5.5.1 Simulation Setup

The simulation setup parameters are presented in Table 5.1. The average number of neighbors per user considering these parameters is slightly higher than 5. Since we are interested to measure LINK's security performance, not its network overhead, we made the following simplifying changes in the simulations. Bluetooth is emulated by WiFi with a transmission range of 10m. This results in faster transmissions as it does not account for Bluetooth discovery and connection establishment. However, the impact on security of this simplification is limited due to the low walking speeds considered in these experiments. Section 5.7 will present experimental results on smart phone that quantify the effect of Bluetooth discovery and Piconet formation. The second simplification is that the communication between the LCA and the users does not have any delay; the same applies for the out-of band communication between colluding users. Finally, a few packets can be lost due to wireless contention because we did not employ reliable communication in our simulation. However, given the low claim rate, their impact is minimal.

To simulate users mobility, we used the Time-variant Community Mobility Model (TVCM model) [115] which has the realistic mobility characteristics observed from wireless LAN user traces. Specifically, TVCM selects frequently visited communities (areas that a node visits frequently) and different time periods in which the node periodically re-appears at the same location. We use the following values of TVCM model in our simulations: 5 communities, 3 periods, and randomly placed communities represented as squares having the edge length 20m. The TVCM features help in providing a close approximation of real-life mobility patterns compared to the often-used random waypoint mobility model (RWP). Nevertheless, to collect additional results, we ran simulations using both TVCM and RWP. For most experiments, we have seen similar results between the TVCM model and the RWP model. Therefore, we omit the RWP results. There is one case, however, in which the results for TVCM are worse than the results for RWP: it is the "always malicious individual verifiers", and this difference will be pointed out when we discuss this case.

5.5.2 Simulation Results

Always malicious individual claimers. In this set of experiments, a certain number of non-colluding malicious users send only malicious claims; however, they verify correctly for other claims.

If malicious claimers broadcast certification requests, the false negative rate is always 0. These claimers are punished and, because of low trust scores, they will not participate in future verifications. For higher numbers of malicious claimers, the observed false positive rate is very low (under 0.1%), but not 0. The reason is that a small number of good users remain without neighbors for several claims and, consequently, their trust score is decreased; similarly, their trust score trend may seem malicious. Thus, their truthful claims

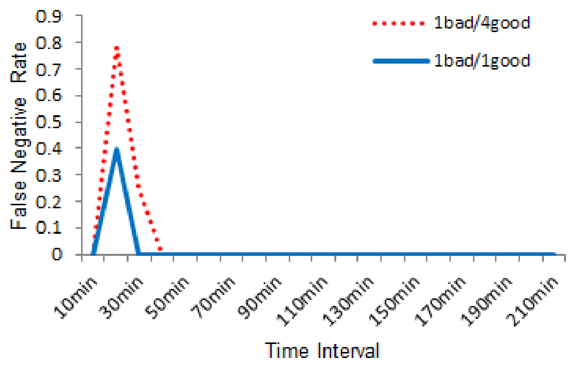


Figure 5.2 False negative rate over time for individual malicious claimers with mixed behavior. The claim generation rate is 1 per minute, 15% of the users are malicious, and average speed is 1m/s.

are rejected if they have no neighbors. The users can overcome this rare issue if they are made aware that the protocol works best when they have neighbors.

If malicious claimers do not broadcast certification requests, a few of their claims are accepted initially because it appears that they have no neighbors. If a claimer continues to send this type of claim, her trust score falls below the "good" user threshold and all her future claims without verifiers are rejected. Thus, the false negative rate will become almost 0 over time. The false positive rate remains very low in this case.

Sometimes malicious individual claimers. In this set of experiments, a malicious user attempts to "game" the system by sending not only malicious claims but also truthful claims to improve her trust score. We have evaluated two scenarios: (1) Malicious users sending one truthful claim, followed by one false claim throughout the simulation, (2) Malicious users sending one false claim for every four truthful claims. For the first 10 minutes of the simulation, they send only truthful claims to increase their trust score. Furthermore, these users do not broadcast certification requests to avoid being proved wrong by others.

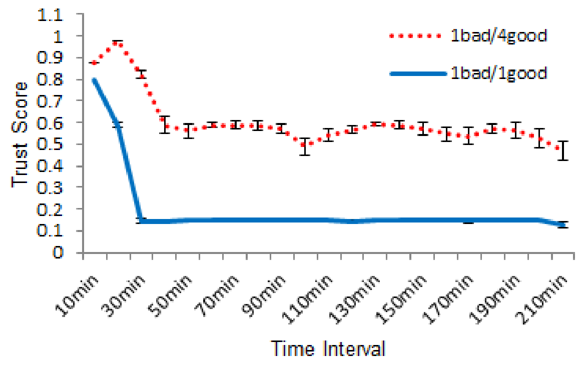


Figure 5.3 Trust score of malicious users with mixed behavior over time. The claim generation rate is 1 per minute, 15% of the users are malicious, and average speed is 1m/s. Error bars for 95% confidence intervals are plotted.

Figure 5.2 shows that LINK quickly detects these malicious users. Initially, the false claims are accepted because the users claim to have no neighbors and have good trust scores. After a few such claims are accepted, LINK detects the attacks based on the analysis of the trust score trends and punishes the attackers.

Figure 5.3 illustrates how the average trust score of the malicious users varies over time. For the first type of malicious users, the multiplicative decrease followed by an additive increase cannot bring the score above the "good" user threshold; hence, their claims are rejected even without the trust score trend analysis. However, for the second type of malicious users, the average trust score is typically greater than the "good" user threshold. Nevertheless, they are detected based on the trust score trend analysis. In these simulations, the trust score range is between 0 and 1, i.e., additive increase of a trust score is done until it reaches 1, then it is not incremented anymore. It stays at 1, until there is a claim rejection or colluding verifier punishment.

Always malicious individual verifiers. The goal of this set of experiments is to evaluate LINK's performance when individual malicious verifiers try to slander good claim-

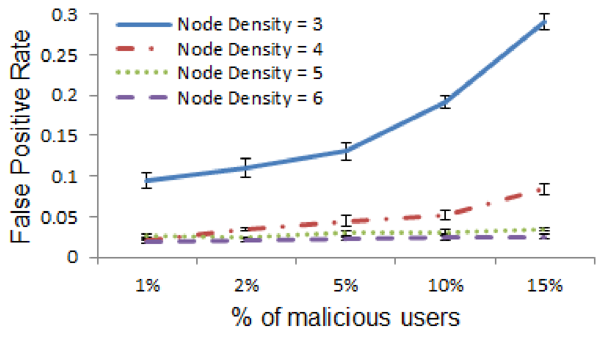


Figure 5.4 False positive rate as a function of the percentage of malicious verifiers for different node densities. The claim generation rate is 1 per minute and average speed is 1m/s. Error bars for 95% confidence intervals are plotted.

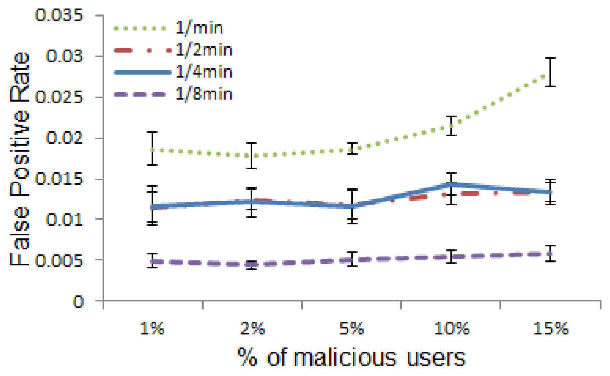


Figure 5.5 False positive rate as a function of the percentage of malicious verifiers for different claim generation rates. The average speed is 1m/s. Error bars for 95% confidence intervals are plotted.

ers. In these experiments, there are only good claimers, but a certain percentage of users will always provide malicious verifications.

Figure 5.4 shows that LINK is best suited for city environments where user density of at least 5 or higher can be easily found. We observe that for user density less than 4 LINK cannot afford more than 10% malicious verifiers, and for user density of 3 or less LINK will see high false positive rates due to no verifiers or due to the malicious verifiers around the claimer.

From Figure 5.5, we observe that LINK performs well even for a relatively high number of malicious verifiers, with a false positive rate of at most 2%. The 2% rate happens

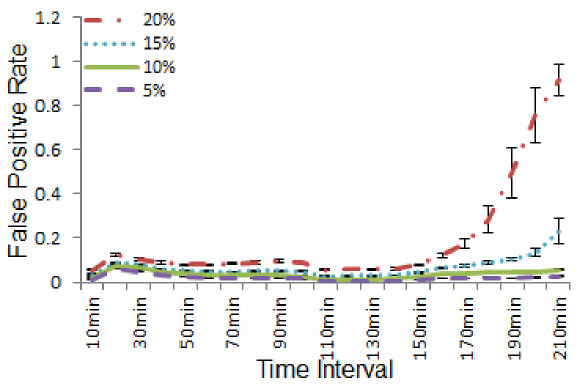


Figure 5.6 False positive rate over time for different percentages of malicious verifiers. The claim generation rate is 1 per minute and the average speed is 1m/s. Error bars for 95% confidence intervals are plotted.

when a claimer has just one or two neighbors and those neighbors are malicious. However, a claimer can easily address this attack by re-sending a claim from a more populated area to increase the number of verifiers.

Of course, as the number of malicious verifiers increases, LINK can be defeated. Figure 5.6 shows that once the percentage of malicious users goes above 20%, the false positive rate increases dramatically. This is because the trust score of the slandered users decreases below the threshold and they cannot participate in verifications, which compounds the effect of slandering. This is the only result for which we observed significant differences between the TVCM model and the RWP model, with RWP leading to better results. This difference is due to the fact that nodes in same community in TVCM move together more frequently, which compounds the effect of slandering by malicious verifiers.

Colluding malicious claimers. This set of experiments evaluates the strongest attack we considered against LINK. Groups of malicious users collude, using out-of-band communication, to verify for each other. Furthermore, colluding users can form arbitrary verification subgroups; in this way, their collusion is more difficult to detect. To achieve high trust score for the colluding users, we consider that they submit truthful claims for the

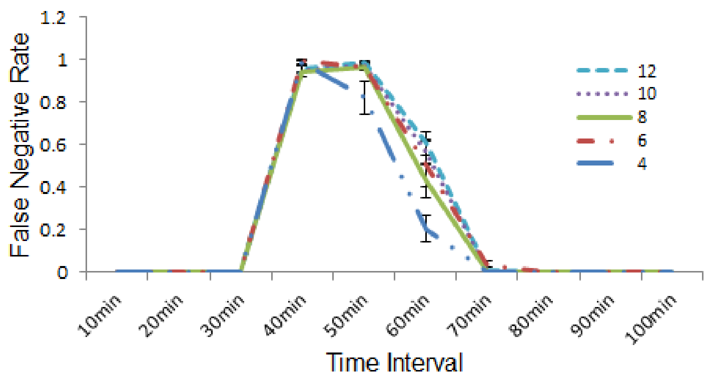


Figure 5.7 False negative rate over time for colluding users. Each curve is for a different colluding group size. Only 50% of the colluding users participate in each verification, thus maximizing their chances to remain undetected. The claim generation rate is 1 per minute and the average speed is 1m/s. Error bars for 95% confidence intervals are plotted.

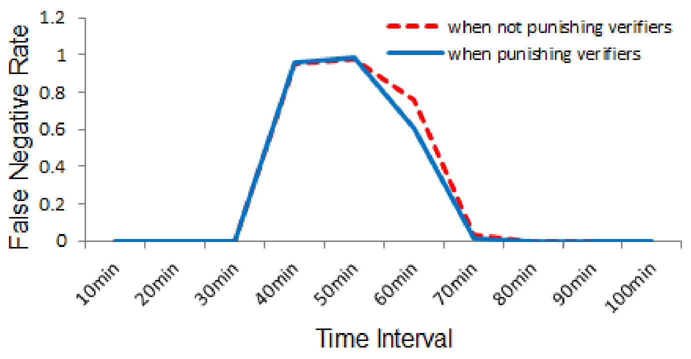


Figure 5.8 False negative rate over time when punishing and not punishing colluding verifiers. The size of the colluding group is 12, and 50% of these users participate in each verification. The claim generation rate is 1 per minute and the average speed is 1m/s.

first 30 minutes of the simulation. Then, they submit only malicious claims. As these are colluding users, there are no honest verifiers involved in these simulations.

Figure 5.7 shows that LINK's dynamic mechanism for collusion detection works well for these group sizes (up to 6% of the total nodes collude with each other). After a short period of high false negative rates, the rates decrease sharply and subsequently no false claims are accepted.

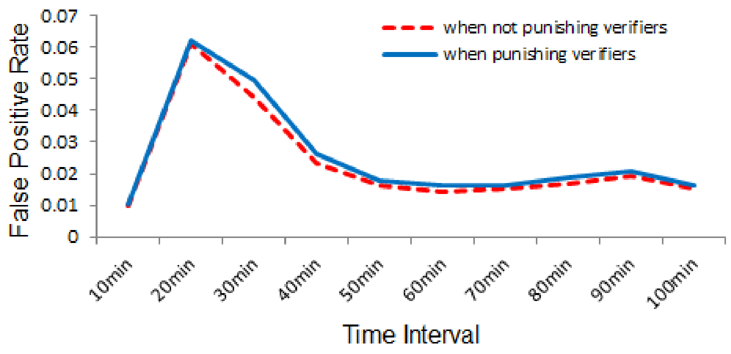
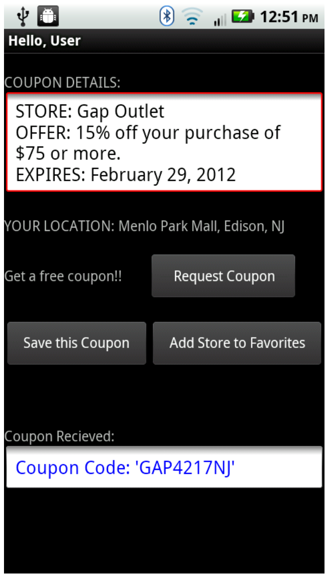


Figure 5.9 False positive rate over time when punishing and not punishing colluding verifiers. All parameters are the same as in Figure 5.8.

In LINK, all colluding users are punished when they are found to be malicious (i.e., the claimer and the verifiers). This decision could result in a few "good" verifiers being punished once in a while (e.g., family members). Figures 5.8 and 5.9 shows the false negative and positive rates, respectively, when punishing and not punishing the verifiers (i.e., the claimers are always punished). We observe that LINK takes a little more time to catch the colluding users while not punishing verifiers; at the same time, a small increase in the false positive rate is observed while punishing the verifiers. Since this increase in the false positive rate is not significant, we prefer to punish the verifiers in order to detect malicious claims sooner.

5.6 Implementation

The LINK prototype has been implemented and tested on Motorola Droid 2 smart phones installed with Android OS 2.2. These phones have 512 MB RAM, 1 GHz processor, Bluetooth 2.1, WiFi 802.11 b/g/n, 8 GB on board storage, and 8 GB microSD storage. Since we did not have data plans on our phones, all experiments were performed by connecting to the Internet over WiFi.


12:51 PM
Hello, User
COUPON DETAILS:
STORE: Gap Outlet
OFFER: 15% off your purchase of $75 or more.
EXPIRES: February 29, 2012
YOUR LOCATION: Menlo Park Mall, Edison, NJ
Get a free coupon!!

Coupon Recieved:
Coupon Code: 'GAP4217NJ'

Figure 5.10 Coupon application on Android phone.

The implementation consists of two main components: (1) an Android client side package that provides applications with a simple API to perform location authentication and allows users to start a background LINK verification process on the phones; (2) the LCA server implemented in Java. The communication between applications and LBSs can be done through any standard or custom API/protocol. To test LINK, we implemented, in Java, a Coupon LBS and its associated application that runs on smart phones (illustrated in Figure 5.10).

5.6.1 Client API

We present the client API in the context of the Coupon LBS and its corresponding application. This service distributes location-based electronic discount coupons to people passing by a shopping mall. To prevent users located farther away to receive these coupons, the service has to authenticate their location.

The corresponding application is implemented as an Android Application Project. The user is provided with a simple 'Request Coupon' button as shown in Figure 5.10. The application submits the current location of the user to LBS and waits for an answer. Upon receiving the location authentication request from the LBS, the application invokes the *submit_claim* LINK API. An optimization done in the LINK package implementation was to limit the Bluetooth discovery to 5.12s instead of the standard 10.24s. This improves significantly the response time and saves energy on the phones (as shown in Section 5.7) at the expense of rarely missing a neighboring phone. In [116], the authors show a single inquirer can locate 99% of all scanning devices within transmission range in 5.12s.

The Bluetooth discovery's call back function returns the list of discovered Bluetooth devices. This list may contain devices that do not belong to phones running LINK. Trying to establish Bluetooth connections with these devices will lead to useless consumption of resources and extra-delays. Therefore, when *submit_claim* function contacts the LCA, it submits not only the location to be authenticated, but also the list of discovered Bluetooth devices. LCA answers with the assigned transaction ID for this claim and also provides the list of registered LINK-enabled devices. LINK on the phone will now establish a Bluetooth connection with each of these devices, sequentially, to send the claim certification request for the given transaction ID. Finally, the *submit_claim* function waits for the LCA answer, which is then returned to the application. If the response is positive, the application invokes another function to receive the coupon from the LBS; the LBS is informed of the authentication decision by the LCA directly.

For the verifier's side, which is not invoked from applications, the user has to start a LINK server that makes the Bluetooth listener active (i.e., puts Bluetooth in discoverable mode). This mode allows any claimer to find the phone in order to request the certification reply for their location claim. When a verifier's Bluetooth listener receives the certification request from a claimer, it invokes the *submit_verification* API. This function reads the claimer's message, generates the verification message, and sends it to the LCA. All the messages are signed using 'SHA1withRSA' algorithm from the 'java.security.Signature' package (1024 bits key size).

5.6.2 LCA Server

LCA is a multi-threaded server that maintains the claim transaction's hashmap, list of all user's details (ID, Bluetooth device address, RSA Public key, Trust score etc.), and all users weight matrices used in the decision process. One of the important implementation decisions is how long should a thread that received a claim wait for verifications to arrive. [1] This is necessary because some verifier phones may be turned off during the verification process, go out of the Bluetooth transmission range before the connection with the claimer is made, or even act maliciously and refuse to answer. This last example could lead to a denial of service attack on the LCA. Thus, the LCA cannot wait (potentially forever) until all expected verification messages arrive. It needs a timeout after which it makes the decision based on the verification received up to that moment.

We considered a waiting function linear in the number of verifiers. The linear increase is due to the sequential establishment of Bluetooth connections between the claimer and verifiers (i.e., they cannot be done in parallel). Since such a connection takes about 1.2s, we defined the waiting time *w = number of verifiers * 2s*, where 2s is an upper bound for the connection latency. However, this fixed waiting time could lead to long delays in situations when there are many verifiers and one or two do not answer at all. Therefore,

[1]the LCA knows the number of verifiers from the submitted claim message.

we decided to adapt (i.e., reduce) this waiting time as function of the number of received verifications. Upon each received verification, $w = w * 4/5$. The value of the reduction factor can be further tuned, but so far it worked well in our experiments.

Once all the verification replies are received or the timeout expires, the LCA processes the claim through the decision process algorithm. Finally, the LCA informs the claimer and the LBS about its decision.

5.7 Experimental Evaluation

The main goals of these experiments are to understand the performance of LINK in terms of end-to-end response latency and power consumption when varying the number of claimers and verifiers. Additionally, we ran micro-benchmarks to understand the cost of individual tasks in LINK.

We used six Motorola Droid 2 smart phones for running the experiments [117]. The LCA and LBS server programs are deployed on Windows-based DELL Latitude D820 laptops, having Intel Core 2 CPU at 2.16GHz and 3.25GB RAM. Before starting the Coupon application, the smart phones' WiFi interfaces are switched on. Additionally, the background LINK processes which listen for incoming certification requests are started on each phone.

5.7.1 Measurements Methodology

For latency, the roundtrip time of the entire LINK protocol (LINK RTT) is measured in seconds at the claimer mobile's coupon application program. For battery consumption, PowerTutor [118] available in the Android market is used to collect power readings every second. The log files generated by PowerTutor are parsed to extract the CPU and WiFi

Table 5.2 Latency Table for Individual LINK Tasks

Task	**Total time taken (s)**
`WiFi communication RTT`	0.350
`Bluetooth discovery`	5.000
`Bluetooth connection`	1.200
`Signing message`	0.020
`Verifying message`	0.006

Table 5.3 Energy Consumption for Individual LINK Tasks

Task	**Energy Consumed (Joules)**
`WiFi communication RTT`	0.100
`Bluetooth discovery`	5.428
`Bluetooth connection (Claimer side)`	0.320
`Bluetooth connection (Verifier side)`	0.017
`Signing message`	0.010
`Verifying message`	0.004

power usage for our application's process ID. Separate tests are performed to benchmark the Bluetooth tasks as PowerTutor does not provide the Bluetooth radio power usage in its logs. All values are measured by taking the average for 50 claims for each test case.

5.7.2 Micro-Benchmark Results

In these experiments, we used just two phones, one claimer and one verifier. Table 5.2 show the latency breakdown for each individual task in LINK. Bluetooth Discovery and Bluetooth connection are the tasks which took the major part of the response time. Note that we limited Bluetooth discovery to 5.12s, as explained in Section 5.6, to reduce the latency. From these results, we estimate that LINK latency will be around 7s for one verifier; the latency increases linearly with the number of verifiers because the Bluetooth connections are established sequentially.

Table 5.3 shows the energy consumption breakdown for each task in LINK. The results show Bluetooth discovery consumes the most energy, while the Bluetooth connec-

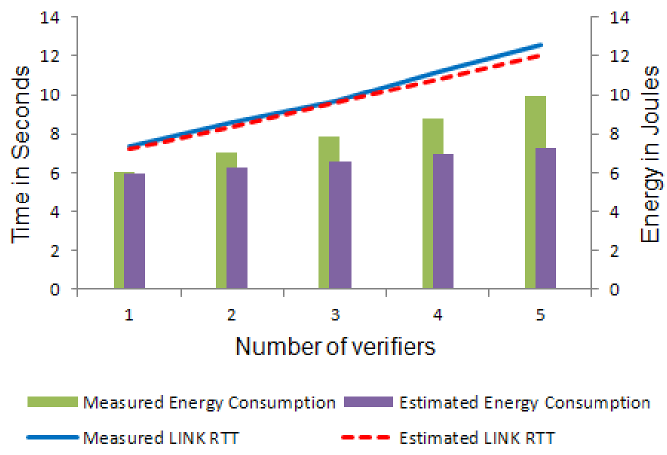


Figure 5.11 LINK RTT and total energy consumed by claimer per claim function of the number of verifiers.

tion's energy consumption will add up with the increase in the number of verifiers per claim.

5.7.3 End-to-End Performance Results

One claimer and different number of verifiers We performed this set of experiments to see whether the estimation of latency and energy consumption based on the micro-benchmark results is correct. In these experiments we measured the LINK RTT and the total energy consumption at the claimer, while varying the number of verifiers. The results in Figure 5.11 demonstrate that in terms of latency the estimations are highly accurate. In terms of power, we observe a linear increase in the difference between the estimations and the measured values. The difference becomes significant (>25%) for 5 verifiers. This difference could be explained by the extra-energy spent by Bluetooth to create and maintain Piconets with higher number of slaves.

The relevance of these results comes in the context of the two main questions we aim to answer: (1) Can LINK work well in the presence of mobility?, and (2) Can LINK run on the phones without quickly exhausting the battery?

If both the claimer and verifiers move at regular walking speed (1.2m/s), then LINK RTT should be less than 8s to have any chance to establish Bluetooth connections before users move out of each other's transmission range (i.e., Bluetooth range is 10m). This bound is the worst case scenario, and LINK could barely provide a response with just one verifier. Of course, LINK would perform better if not all users move or move in the same direction. A simple solution to avoid this worst case scenario is to make the claimer aware that she should not walk while submitting claims. This is a reasonable requirement because the claimer needs to interact with the phone to access the LBS anyway. In such a case, the bound on RTT doubles to 16s, and LINK will be able to establish connections with all verifiers (as shown in Figure 5.11). Note that a Piconet is limited to 7 slaves, which limits the maximum RTT. Finally, let us recall that LINK is robust to situations when verifiers run out of the transmission range before the Bluetooth connection is established.

To understand LINK's feasibility from an energy point of view (i.e., to answer the second question posted above), we performed an analysis to see how many claims and verifications can a smart phone execute before it runs out of battery power. The total capacity of the Motorola Droid 2 battery is: 18.5KJ. Since LINK requires WiFi and Bluetooth to be on, we first measured the effect of these wireless interfaces on the phone lifetime. Table 5.4 shows the results for different interface states (without running any applications). We observe that even when both are on all the time, the lifetime is still over 2 days, which is acceptable (most users re-charge their phones at night). The lifetime is even better in

Table 5.4 Battery Life for Different WiFi and Bluetooth Radio States

	Bluetooth and WiFi off	**Bluetooth off and WiFi on**	**Bluetooth and WiFi on**
Battery life	10Days 16Hrs	3Days 15Hrs	2Days 11Hrs

reality because Android puts WiFi to sleep when there is no process running that uses WiFi (in our experiments, we forced it to be on all the time).

Next, using this result, we estimate how many claims and verifications can a LINK do with the remaining phone energy:

Number of claims a phone can do until battery is exhausted = ***2,701 Claims***

Number of verifications a phone can do until battery is exhausted = ***20,458 Verifications***

These numbers demonstrate that a fully charged phone is capable of running LINK in real life with minimal impact on the overall battery consumption.

LINK is designed to work for cellular (over 3G) as well, which uses less battery power compared to WiFi [119]. In our experiments, we used WiFi because the phones did not have cellular service plans. In general, LINK will consume even less energy over the cellular network, but its RTT will increase because 3G has typically higher latency than WiFi.

Simultaneous nearby claimers The goal in this set of experiments is to evaluate how much the RTT and the energy consumption increase when phones act simultaneously as both claimer and verifier. We used three phones which are in the Bluetooth transmission range of each other while sending concurrent claims. For each claim, the other two phones act as verifiers.

Table 5.5 Average RTT and Energy Consumption per Claim for Multi-claimer Case vs. Single Claimer Case

	Average RTT (s)	Energy consumed (Joules)
Multi-Claimer case	15.31	9.25
Single-Claimer case	8.60	7.04

When multiple phones send claims simultaneously, hence perform Bluetooth discovery at the same time, we notice many situations when only one or even no verifier was discovered. In total, only in 35% of the verifiers were discovered for all tests. This behavior is mostly due to the well- known Bluetooth inquiry clash problem [120]. An additional problem is the shortened discovery period (5.12s instead of 10.24s). While simultaneous claims in the same transmission range are expected to be rare in practice, we decided to provide a solution for this problem nevertheless.

Since LCA is aware of all the devices and their claims, it can successfully predict and prevent the Bluetooth inquiry clash by instructing the claimers on when to start their Bluetooth discovery. Practically, a claimer is delayed (for 3s in our experiments) if another one just started a Bluetooth discovery. The delay setting can be determined more dynamically by LCA at runtime depending on factors such as the number of simultaneous claimers and the number of surrounding verifiers in the region. Furthermore, claimers are allowed to perform complete Bluetooth inquiry scans (10.24s).

Table 5.5 compares the RTT and energy consumption measured when simultaneous claims are performed by three phones with the values measured for the case with one single claimer and two verifiers. With the new settings, the claimers were able to discover all verifiers. However, as expected, this robustness comes at the cost of higher latency (the energy consumption also increases, but not as much as the latency). The good news is that we expect all three claimers to be static according to the guidelines for LINK claimers. Therefore, the increased latency should not impact the successful completion of the protocol.

5.8 Chapter Summary

This chapter presented LINK, a protocol for location authentication based on certification among mobile users. LINK can be successfully employed to provide location authentication for third-party location-based services without requiring cooperation from the network/ localization infrastructure. The simulation results demonstrated that several types of attacks, including strong collusion-based attacks, can be quickly detected while maintaining a very low rate of false positives.

LINK was implemented on Android-based smart phones. Experimental results demonstrated that LINK is feasible in real-life situations. It is fast enough for static claimers to successfully authenticate their location despite the potential mobility of the verifiers. Additionally, its energy consumption does not have a significant impact on the phone's battery life even for relatively high usage.

CHAPTER 6

A GAME-BASED INCENTIVE MECHANISM FOR MOBILE SENSING

In the previous chapters, we discussed our progress in achieving data reliability in mobile sensing. In this chapter, we argue that we also need sufficient user participation to achieve good quality sensed data for mobile sensing applications. In this work, we propose a cost-effective way to incentivize users to participate in a mobile crowd sensing system to densely cover a large targeted area uniformly with automatic sensing using gamification techniques instead of monetary incentives. We designed and implemented a first person shooter sensing game, “Alien vs. Mobile User”, which employs techniques to attract users to unpopular regions. The results from the user study [121] shows that mobile gaming can be a successful alternative to micro-payments for fast and efficient area coverage in crowd sensing.

The remainder of the chapter is organized as follows. Section 6.1 presents the proposed game model. Section 6.2 describes the game design and implementation. Section 6.3 presents three alien movement strategies to cover the area. The simulation results are discussed in Section 6.4. Section 6.5 presents the results of our user study. In Section 6.6 we discuss the general design principles for sensing games. Finally, the chapter concludes in Section 6.7.

6.1 Motivation and Main Idea

Many of the proposed mobile crowd sensing systems provide monetary incentives to the smart phone users for sensing the data. Instead of monetary incentives, we would like to provide social incentives for participation through gamification techniques. There is a significant literature on using gamification techniques in crowdsourcing, however there is very little in terms of applying gamification techniques in mobile sensing. But, specially in mobile crowd sensing platform there are no architectures or known research proposed yet on incentivizing the participants by utilizing the mobile games.

For instance, mapping an area with required sensor data is a tedious effort when performed manually and when we resort to the mobile crowd sensing system, it is possible to get a lot of data for some highly populated regions of the area, but it is difficult to cover all the regions. The users may require lots of money to go out of their way and cover the unpopular regions. Therefore, we propose to automatically collect mobile crowd sensing data by incentivizing smart phone users to play sensing games that provides game incentives to uniformly cover all the regions of an area. In addition, the mobile game environments can be improved using sensed data which give interesting real-world gaming experience to user while playing the game. To the best of our knowledge we are the first to propose the sensing games in the mobile crowd sensing platform. Even though game development initially involves a relatively significant one-time cost, choosing mobile gaming over micro-payments is expected to be the economical option in large scale crowd sensing as illustrated in Figure 6.1.

6.2 Alien vs. Mobile User Game

The game is a first person shooter game played by mobile users on their smart phones while moving in the physical environment. Since the goal of the game is to uniformly cover a large area with high density sensing data, it is essential to link the game story to

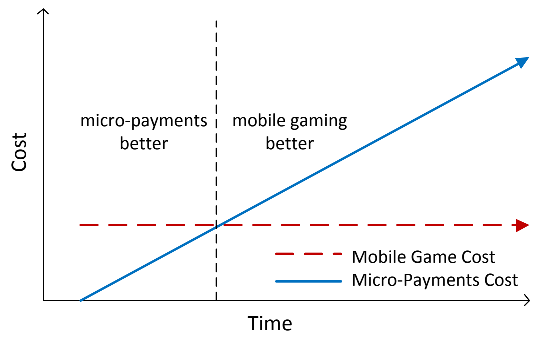


Figure 6.1 Cost comparison between crowd sensing enabled by mobile gaming and crowd sensing enabled by micro-payments.

the physical environment. Broadly speaking, in our game, the players must find aliens throughout an area (e.g., our campus) and destroy them. In the process, players collect sensing data as they move through the area. Although the game could collect any type of sensing data available on the phones, our implementation automatically collects WiFi data (*BSSID, SSID, Frequency, Signal strength*) to build a WiFi coverage map of the targeted area. The motivation to play the sensing game is twofold: 1) The game provides an exciting *real-world gaming experience* to the players, and 2) The players can *learn useful information about the environment* such as the WiFi coverage map which lists the locations having best WiFi signal strength near the player's location.

The game contains the following entities/characters:

- **CGS**: The Central Game Server (CGS) controls the sensing game environment on the mobile devices and maintains the players profiles and the sensing data collected from the sensing game.
- **Player**: Users who play the sensing game on their mobile devices. The players are rewarded with points for shooting and/or destroying the aliens. All players can see the current overall player ranking.
- **Alien**: A negative role character that needs to be found and destroyed by the game players. Aliens are controlled by CGS according to the sensing coverage strategy.

6.2.1 Design

Game story: All the aliens in the game are hiding at different locations across the targeted area. Players can see the aliens on their screens only when they are close to the alien positions. This is done in order to encourage the players to walk around the area to discover aliens; in the process, we collect sensing data. At the same time, this makes the game more unpredictable and potentially interesting. When running on the phones, the game periodically scans for nearby aliens and alerts the players when aliens are detected; the player locates the alien on the game screen and starts shooting at the alien using the game buttons. When an alien gets hit, there are two possible outcomes: if this is the first or second time the alien is shot, then the alien escapes to a new location to hide from the player. To completely destroy the alien, the player has to find and shoot the alien three times, while hints of the aliens location are provided after it was shot once. In this way, the players are provided with an incentive to cover more locations. The aliens hiding in the targeted area are common to all the players. Thus, different players can detect and shoot the same alien at different times as long as the alien was not destroyed.

The sensing side of the game: Sensing data is collected periodically when the game is on. Since this collects location traces, the players will be made aware of this potential privacy risk when they install the game. Nevertheless, the goal is to build games that are attractive enough to convince players to trade-off privacy for fun. The placement of aliens on the map will ultimately ensure full sensing coverage of the area. The challenge, thus, is how to place/move the aliens to ensure fast coverage while at the same time maintain a high player interest in the game. The CGS moves aliens on the game map to desired locations (i.e., uncovered regions) using one of the alien movement algorithms proposed in Section 6.3.

In the initial phases of sensing, CGS moves the alien to a location which is not yet covered, but later on it moves the alien intelligently from one location to another location by considering a variety of factors (e.g., less visited regions, regions close to pedestrian routes, or regions where the client who needs the data requires high sensing accuracy). CGS helps the player who shoots an alien by providing game hints such as revealing the direction in which the alien has escaped. Generally, the alien will escape to farther away regions and the player might be reluctant to follow despite the hints provided by CGS. To increase the chances that users follow the alien, we provide more points for shooting the alien for a second time, and even more for the third shot (which destroys the alien).

The bullets in the game are placed around the players and maintained individually for each players view of the game. CGS decides the number of bullets placed around each player based on three main factors: the number of bullets collected so far by the player, the total play time of the player, and the game level of the player. The count of the bullets increases linearly with the increase in total number of collected bullets by the player; this is also a function of the game level. The intuition behind adding the bullets in this fashion is that more active players should be given more bullets when they advance at the next level as an award that will keep them motivated to continue playing. But, placing too many bullets around the players could make the game less interesting. Therefore, CGS limits bullet count to a certain threshold.

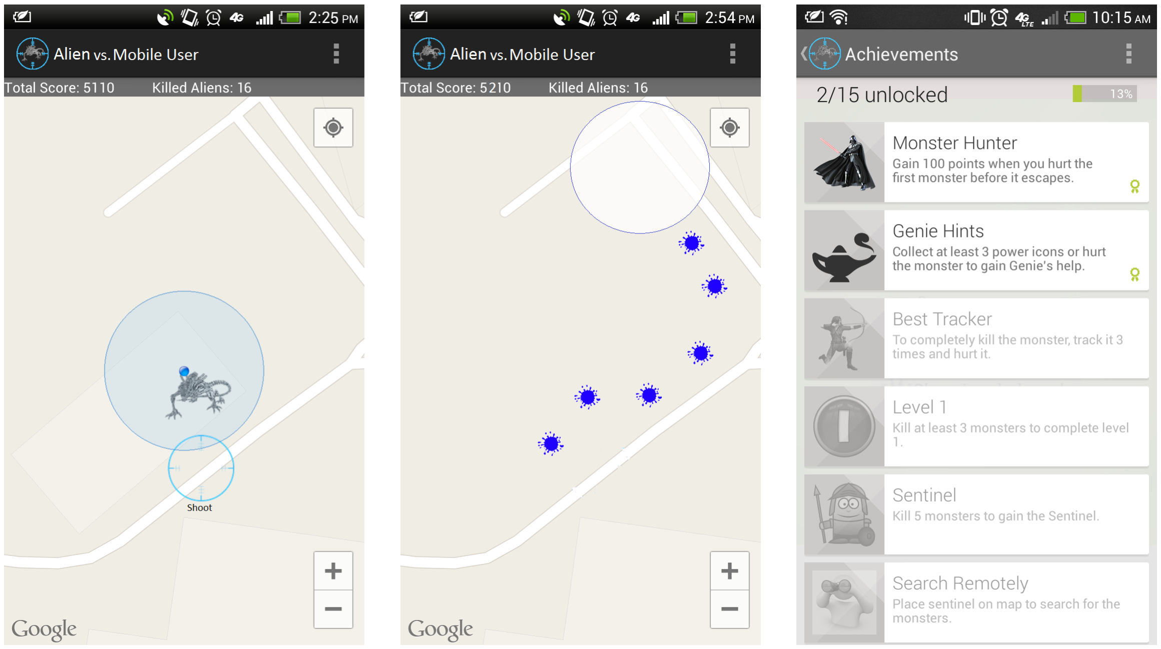
Game difficulty and achievements: We designed the game with difficulty levels based on the aliens killed, the bullets collected from around the player's location, and the total score of the player. In this way, the players have extra-incentives to cover more ground. The player has to track and kill a minimum number of aliens required to unlock specific achievements and to enter the next levels in the game. We leverage the achievements

APIs provided in Android platform as part of Google Play Game Services which allow the players to unlock and display achievements as shown in Figure 6.2 (right). The players can check their game progress and the current unlocked achievements through the game menu.

Indoor localization: This is necessary in order to cover the building floors. By default, Android uses WiFi triangulation in conjunction with GPS to estimate the location of the player in the (X, Y) plane while indoors. The error of this estimation is reasonable for our purpose. We leverage the barometric pressure sensor available in most Android phones to determine the Z coordinate, specifically the building floor. For accurate estimation of floor levels, we initially measured the altitude at ground level near each campus building. For the phones without barometric pressure sensor, we perform indoor localization based on the fingerprinting done by other players who visited the same area and whose phones have barometric pressure sensors. Specifically, these players have recorded the mappings between the estimated location and the strengths of the WiFi signals measured at that location. Thus, the players will be helped to hunt down aliens indoors by the sensing data they collect (i.e., WiFi signals).

6.2.2 Prototype Implementation

A prototype of the game has been implemented in Android and is compatible with smart phones having Android OS 2.2 or higher. When the player opens the game on her smart phone, the map of the player's current location is displayed on the game screen. An alien appears on the map when the player is close to the alien's location, as shown in Figure 6.2 (left). The player can target the alien and shoot it using the smart phone's touch screen.



Monster Hunter
Gain 100 points when you hurt the first monster before it escapes.

Genie Hints
Collect at least 3 power icons or hurt the monster to gain Genie's help.

Best Tracker
To completely kill the monster, track it 3 times and hurt it.

Level 1
Kill at least 3 monsters to complete level 1.

Sentinel
Kill 5 monsters to gain the Sentinel.

Search Remotely
Place sentinel on map to search for the monsters.

Figure 6.2 Alien vs. Mobile User app: finding the alien (left); alien trail (middle); achievements (right).

When the alien escapes to a new location, its “blood trail” on the map leading to its new location is provided to the player as a hint to track it down (as shown in Figure 6.2 (middle)). The server side of the game is implemented in Java using the Model View Controller framework and is deployed on the Glassfish Application Server.

6.3 Alien Movement Strategies

In this section, we propose three strategies for placing and moving the aliens in order to cover the targeted area efficiently.

6.3.1 Assumptions and Definitions

The target area that needs to be covered is divided into small square cells (in our user study we consider the cell size 10m x 10m). Inside buildings, the target area includes both the ground floor as well as the upper floors. Each alien can be used to cover K squares before

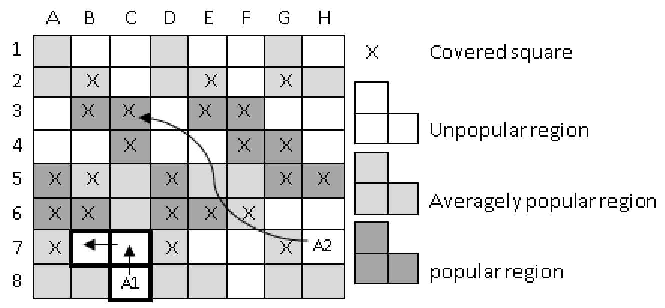


Figure 6.3 Localized Movement strategy: Example of CGS moving the alien A1 locally in the assigned highlighted region when A1 is shot by the player. As part of the greedy process, CGS moves the healthy alien A2 from an unpopular region to a popular region.

it is destroyed (in our game, aliens are destroyed after being shot 3 times, so $K = 3$). We assume there cannot be more than one alien in one square at any moment. Hence, the minimum number of aliens required to cover the area is $TotalNumberOfSquares/K$.

Clients/applications may need more than one reading per square due to reliability issues. Thus, the game has a parameter that indicates how many times each square must be covered by sensor readings.

In this section, we use the following terms:

- *Square*: The square is the smallest unit of coverage. Aliens and players move to squares.
- *Available Square*: A square for which the number of collected data points is lower than the desired number of data points.
- *Region*: A larger portion of the targeted area which contains K squares.
- *Popular Region*: A region where a substantial number of data points have been collected, but the number of data points required by the client has not been reached.
- *Unpopular Region*: A region where no data point has been collected.
- *Healthy Alien*: An alien that was never shot in the game.

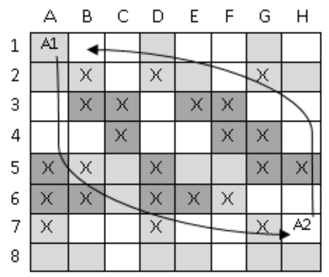


Figure 6.4 Random Movement strategy: Example of CGS swapping alien A1 (which is shot) with healthy alien A2 from the unpopular region.

6.3.2 Localized Movement Strategy

In this strategy, every alien is assigned to a single region, and it moves only in that region when shot by the player as shown in Figure 6.3 for alien A1. This strategy does not require the players to move much to hunt down the aliens, and thus, it could be successful.

CGS maintains statistics about popular and unpopular regions based on the data collected from players. Thus, CGS is able to adapt to the collected information and execute a greedy process which periodically moves the healthy aliens from unpopular regions to popular regions as shown in Figure 6.3 for alien A2. When the player shoots this healthy alien, CGS moves the alien back to its originally assigned region. A "blood trail hint is provided to the player to indicate the location where the alien has escaped. This process places the aliens at highly popular regions to increase the probability of being found by the player and helps in luring the players to unpopular regions.

Issue: The main concern with this strategy is that the aliens are restricted to one region, and thus, the players may eventually predict aliens' locations. Therefore, the Localized Strategy may lead to a simple game plan and may become boring for the players, ultimately leading to a decrease in the number of players.

6.3.3 Random Movement Strategy

In this strategy, the aliens are not restricted to a region and they can be placed in any square randomly by CGS. Thus, the strategy addresses the issue of easily predicting the alien's location. Basically, CGS swaps the alien which is shot in the game with a healthy alien from an unpopular region chosen randomly as shown in Figure 6.4. A1 is moved to a square in an unpopular region, chosen randomly from all the unpopular regions at that time. A2 is moved to an available square near to the location where A1 was shot. When there are no available unpopular regions to choose (i.e., regions with no coverage), CGS chooses the square from the regions with the minimum coverage.

The purpose of the alien swap is totake advantage of the popularregions from the starting phases of the game deployment,instead of waiting for a long period to adapt to the collecteddata (i.e., identify the top few popular regions) as performed in the Localized Movement strategy. Hence, the shot alien is moved from the popular region to an unpopular one to lure the player there. At the same time, we bring the alien from the unpopular region to an uncovered square of the popular region where there is a higher chance for players to encounter it. In the Random Movement strategy, new aliens are added at random time intervals.

Issues: Even though randomness helps CGS to place the aliens at unpredictable squares, CGS may end up placing the aliens too far from the player's current location. If this situation happens in the early phases of the game deployment, then players may lose interest in tracking the alien. The players might be more interested to track the aliens if the aliens are closer to their current region, though at a square that is not easy to predict. This should be done at least until CGS achieves a good game adoption rate.

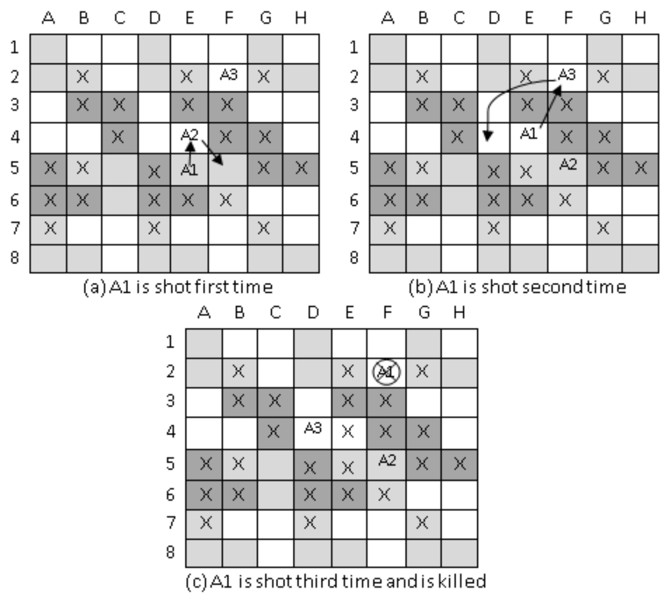

(a) A1 is shot first time
(b) A1 is shot second time
(c) A1 is shot third time and is killed

Figure 6.5 Progressive Movement strategy: Example of CGS moving the alien to the closest unpopular region.

6.3.4 Progressive Movement Strategy

In this strategy, the aliens are moved to the nearest unpopular regions when they are shot by the player. This addresses the issue of randomly placing the alien too far from the player's location. Basically, CGS swaps the alien which is shot in the game with the closest healthy alien from the unpopular region as shown in Figure 6.5. In the figure, when A1 is shot the first time, CGS chooses the closest available square from all the unpopular regions, which is A2's square. A2 is then moved to the next available square near the A1's location as shown in Figure 6.5(a). A similar swap happens between A1 and A3 when A1 is shot again. A1 is finally destroyed when it is shot for a third time. Steadily, CGS covers the targeted area in a progressive fashion by starting from the popular regions as main centers and slowly expanding the coverage toward the unpopular regions.

CGS chooses a square from the closest unpopular region at that time. When there are no available unpopular regions, CGS chooses the square from the regions with the minimum coverage. When all the regions with same number of available squares are at same distance from the alien's location, CGS selects the square with the "healthiest" alien (i.e., never shot or shot the fewest times).

The purpose of this swap process is two-fold: 1) CGS pulls the players into unpopular regions to cover the targeted area efficiently, and 2) the game is made challenging because the players cannot predict the alien's moves without knowing the overall game details such as health status of all the aliens and the coverage status of all the squares. This process avoids the issues present in the other two strategies. Since our results show that this strategy works best, we present its detailed operation in Algorithm 5.

6.3.5 Number of Aliens in the Game

In our game, it is essential to set the number of aliens in such a way as to achieve a good balance between efficient coverage and maintaining player interest in the game. Empirically, our goal is to allow players to encounter aliens every so often, while not being able to predict the alien's movement. The number of aliens depends on the number of squares, the number of players, and the stage of the game (e.g., in early stages most regions are not covered, and in late stages most regions are covered).

For each strategy, CGS adds new aliens every few minutes near the players' locations depending on the game status to keep them interested in tracking the aliens. For each player, CGS allocates G_S aliens for every T_{GS} minutes time period, where $T_{GS} = 2 \times NumOfGameStages \times T$. $NumOfGameStages$ is a parameter that counts how many times each square must be covered. This is important for clients/applications that do not want to rely on only one reading per square due to reliability reasons. Thus, our game

Algorithm 5 Progressive Movement Strategy Pseudo-Code

Notation:
A1: The alien that needs to be moved to another location by CGS.
A2: The other alien in the game to whose location the A1 is moved by CGS.
currentSquare: The square where A1 is shot by the player.
chosenSquare: The square to which A1 is moved by CGS.
unPopularRegionsList: The list of unpopular regions.
mostUnpopularRegion: The region with highest number of uncovered squares.
getCurrentSquare(): Get the current square details of the alien.
getAlien(): Get the alien from the square if any alien is at that square.
getNearestAvailableSquare(currentSquare): Get the available square closest to the currentSquare.
updateCurrentSquare(chosenSquare): Update the currentSquare of the alien with the chosenSquare.
updateCoveredSquare(currentSquare): Set the coverage indicator for the currentSquare.
getNearestSquare(mostUnpopularRegion,currentSquare): Get the square details for the square in the most unpopular region which is closest to the currentSquare.
getRegionsWithMostAvailableSquares(sqCount): Get the list of unpopular regions having available squares of count sqCount.
sortByDistanceAscend(unPopularRegionsList, currentSquare): Sort the unPopularRegionsList by the distance between each region to the currentSquare in ascending order.
getRegionWithHealthiestAlien(): Get the region which contains the healthiest alien.

moveAlien(A1):

```
1: currentSquare = A1.getCurrentSquare()
2: chosenSquare = chooseSquareFromUnpopularRegions(currentSquare)
3: if $chosenSquare == NULL$ then
4:     The targeted area is fully covered and the game is complete
5: else
6:     A2 = chosenSquare.getAlien()
7:     A1.updateCurrentSquare(chosenSquare)
8:     if $A2 \neq NULL$ then
9:         A2.updateCurrentSquare(getNearestAvailableSquare(currentSquare))
10:    updateCoveredSquare(currentSquare)
```

chooseSquareFromUnpopularRegions(currentSquare):

```
1: mostUnpopularRegion=getNearestRegion(getMostUnpopularRegionsList(), currentSquare)
2: chosenSquare = getNearestSquare(mostUnpopularRegion,currentSquare)
3: return chosenSquare
```

getMostUnpopularRegionsList():

```
1: regionSize = 3 //constant
2: for $sqCount = regionSize$ to 1 do
3:     unPopularRegionsList=getRegionsWithMostAvailableSquares(sqCount)
4:     if $unPopularRegionsList \neq NULL$ then
5:         break
6: return unPopularRegionsList
```

getNearestRegion(unPopularRegionsList, currentSquare):

```
1: unPopularRegionsList=sortByDistanceAscend(unPopularRegionsList, currentSquare)
2: if multiple regions have the same distance then
3:     mostUnpopularRegion = getRegionWithHealthiestAlien()
4: else
5:     mostUnpopularRegion = unPopularRegionsList[0]
6: return mostUnpopularRegion
```

allows multiple readings for each square. However, the game works in stages: it first attempts to cover each square once, then to cover each square twice, and so on. T is the time period after which CGS attempts to add a new alien. This formula allows CGS to add more aliens in the final stages of the game to pull the players into the most unpopular regions.

CGS adds a new alien at the player's location only if the player satisfies either of the following conditions. First, the player has not found the alien in the last T minutes. Second, the player tracked and killed the alien in last T minutes. The intuition behind these conditions is that in both scenarios the player is interested in tracking the alien. CGS adds the new alien only if the player has moved from her last location in T minutes. In case the player has found the alien in the last T minutes and ignored it, CGS alerts the player about the last found alien instead of adding a new alien again after T minutes.

6.4 Simulations

This section presents the evaluation of the alien movement strategies, which have a major impact on the efficiency of the area coverage. The three main goals of the evaluation are: (1) Compare the coverage latency for the three strategies, (2) Measure the coverage effort as experienced by players, and (3) Compare the area coverage over time for the strategies. We use the NS2 simulator for the experiments.

6.4.1 Entropy of the Area Coverage

The goal of the game is to cover the targeted area as fast as possible and with the least possible effort from the players. In order to measure the player effort, we propose to look at the "system utilization": a perfect utilization requires that each square be covered uniformly with the minimum number of sensing readings required by the client who will use the data. In this case, the entire effort of the players is utilized. Any additional readings will lead to wasted player effort. Therefore, we need a metric to capture this idea.

The entropy of the area coverage presented in Equation 6.1 can be used to observe the overall system utilization. The greater the entropy value, the more uniformly the area is covered, and thus the player effort is not wasted.

$$H = -\sum_{i=1}^{N} \frac{v_i}{V} \log(\frac{v_i}{V}) \qquad (6.1)$$

Where v_i is the number of visits by the players in square i; V is the total number of visits by the players of all visited squares ($V = \sum v_i$); and N is the total number of squares in the targeted area. The equation is similar to the deployment entropy calculated in [122], where a swarm of micro air vehicles are deployed to maximize the area coverage.

In the game context, the entropy is the measure of uniformity with which the players visit the squares in the targeted area. Once the game completes and the area is fully covered, the entropy of the area coverage quantifies the completeness and uniformity with which the players covered the area.

6.4.2 Simulation Setup

The simulation is setup in a 500m X 500m area with 12 players moving at 2m/sec. The simulation area is divided into 2500 squares of size 10m x 10m. We used the following settings for the parameters defined in Section 6.3: $NumOfGameStages$=5 (each square must be covered at least 5 times), T=3, and G$_S$={5..9} for game stages 1 to 5, respectively.

Table 6.1 Player Types

Player Type	Player Type Description
PT1	Active player who always tracks the alien
PT2	Tracks the alien if it is in 50 meters range or if she can deliver the third shot that destroys the alien
PT3	Tracks the alien only if it is in 50 meters range.
PT4	Player never tracks the alien; shoots the alien only when found in her travel path.

Table 6.2 Number of Players from each Player Type Assigned to the Three Behaviors/ scenarios

	BH1	BH2	BH3
PT1	0	3	6
PT2	2	3	4
PT3	4	3	2
PT4	6	3	0

Sensing is performed once per second by each player. The players get 100, 200 and 300 points, respectively, for the three shots needed to destroy the alien.

Although player behavior is very difficult to predict, we define four player types as shown in Table 6.1. These player types are allocated to three simulation scenarios as defined in Table 6.2: Behavior 1 (BH1) in which less active players are in majority, 2) Behavior 2 (BH2) in which the player types are balanced, and 3) Behavior 3 (BH3) in which highly active players are in majority.

6.4.3 Simulation Results

Area Coverage Latency. Figure 6.6 shows the coverage latency of the three strategies for the three simulation scenarios (BH1, BH2, BH3). The best coverage latency is achieved by the Progressive Movement strategy in all scenarios. This is one good reason to pick this

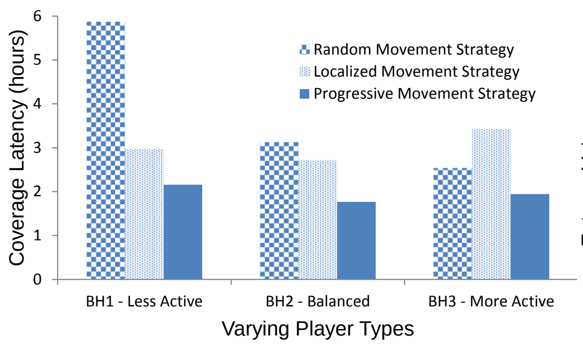


Figure 6.6 The area coverage latency of the three alien movement strategies.

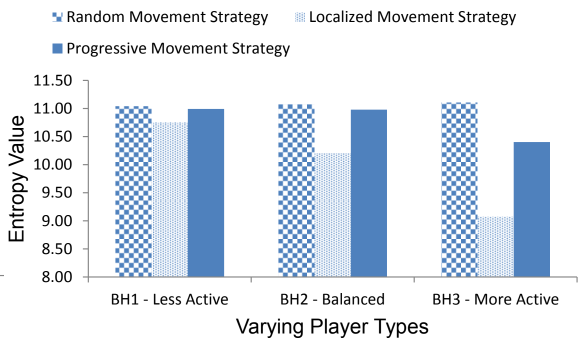


Figure 6.7 Entropy of the area coverage for the three alien movement strategies.

strategy in future deployments of the game. In addition, this strategy also has techniques to keep the players interested by making the alien movement unpredictable.

The Random Movement strategy performs the worst in BH1 and BH2. In BH1, there are no active players to play the game which is the reason it takes more time to cover the targeted area. But with the increase in the number of active players, this strategy improves until it overcomes the Localized Movement strategy in BH3. In the Progressive and Localized strategies, good results are achieved even for BH1 because these strategies move the alien closer to the player who shot it and the PT2 type players track the alien in 50 meters range.

In BH3 scenario, surprisingly, the Localized and Progressive strategies take longer to complete than in the other two scenarios. The reason is that the active players are following similar paths to the unpopular regions. Thus, the rate of new squares covered is slower in BH3 compared to BH2.

Area Coverage Entropy. Figure 6.7 shows the entropy values calculated at the end of the simulation for the three strategies. As expected, the Random strategy performs best as it achieves the most uniform coverage. In the Localized and Progressive strategies, the increase in active players leads to a decrease in entropy as the active players cover the

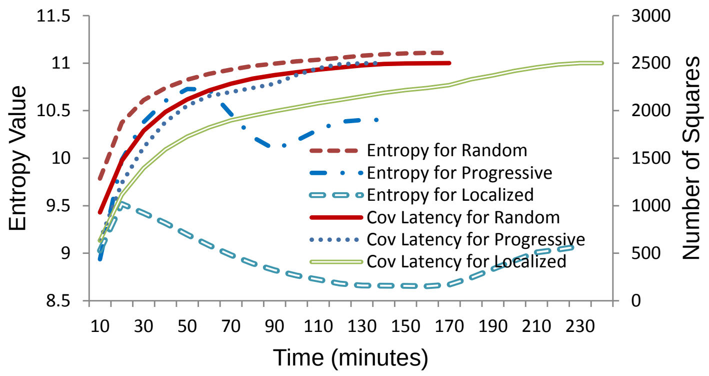


Figure 6.8 The entropy and latency over time for the three alien movement strategies in BH3 scenario.

squares faster than in the Random strategy (in BH2 and BH3). Thus, the overall coverage is not uniform which results in lower entropy. When the number of active players is low, the entropy is similar for the three strategies.

When comparing the Progressive and the Random strategies for BH3, we see that it is difficult to conclude which one is better: Progressive leads to lower latency, while Random leads to higher entropy. However, Progressive is clearly better for the other two scenarios as it achieves lower latency and comparable entropy.

Area Coverage over Time. Figure 6.8 shows the entropy and latency over time for the three strategies in the BH3 scenario, which we observed to be the most complex in terms of selecting the best strategy. The results show that the goal of covering each square at least $NumOfGameStages$ times (set to 5 in these simulations) is achieved in less time by the Progressive strategy compared to the other two strategies. Thus, we conclude that this strategy is the best overall and use it in our prototype implementation.

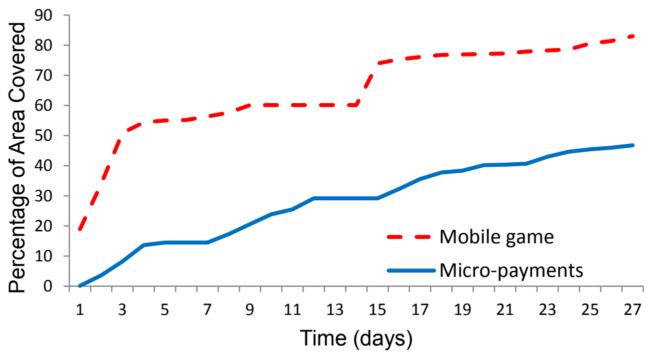


Figure 6.9 Area coverage over time: crowd sensing with mobile gaming vs. crowd sensing with micro-payments for the first four weeks of the studies.

6.5 User Study Evaluation

We ran a user study (performed with our institution's IRB approval) for 35 days, in which students use their Android devices to play our game [123] and collect WiFi data both outdoors and indoors throughout the campus of our institution. The campus area is divided into small squares of size 10m X 10m. The game is available in the Google Play Store. A total of 53 players registered, then installed and continued playing the game.

The main goals of this user study are: (1) compare the area coverage efficiency between crowd sensing enabled by mobile games and crowd sensing enabled by micro-payments, (2) determine the effectiveness and performance of mobile gaming as an incentive for crowd sensing, and (3) quantify the players' engagement in the mobile game, which helps to further improve the game.

Area coverage: mobile gaming vs. micro-payments. Figure 6.9 shows the mobile gaming approach performs substantially better than the micro-payments approach. We observe that players get highly engaged in the game from the first days, which leads to high coverage quickly (50% of the target area is covered in less than 3 days). This happens despite the fact that players have registered with the game over time during the study, while in McSense, all the users have registered with the system before the study started. The

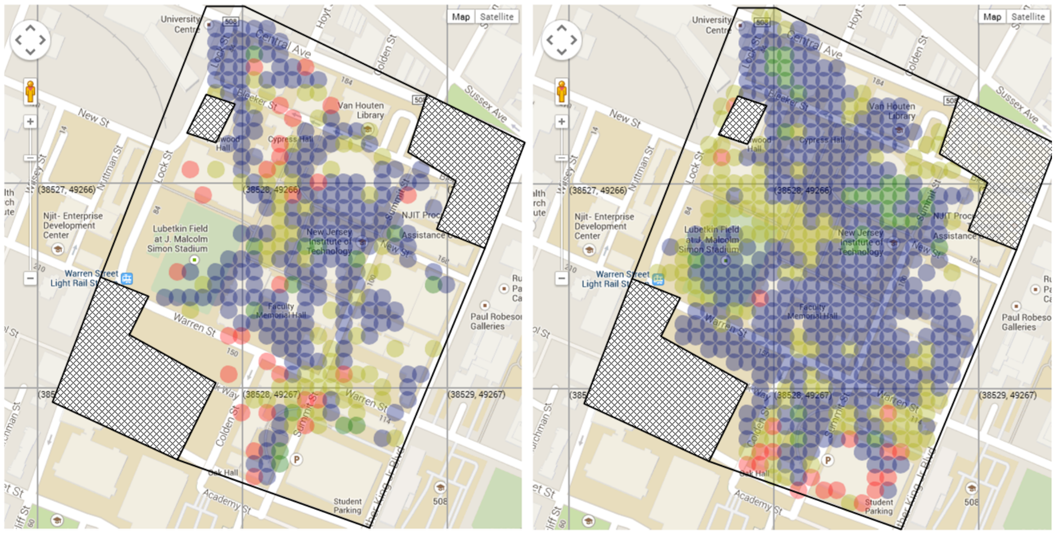


Figure 6.10 Coverage of the University campus in the first four weeks of the studies: 46% crowd sensing with micro-payments (left) vs. 87% crowd sensing with mobile gaming (right). A number of areas have been removed from the map as they are not accessible to students.

coverage progress slows down after the initial phase due to several reasons. First, in order to make the comparison fair with the McSense study, the results show only the ground level covered by the mobile game players. However, starting in the second week, many aliens have been placed at higher floors in the buildings; this coverage is not captured in the figure. Second, the slowdown is expected to happen after the more common areas are covered, as the players must go farther from their usual paths. Third, we observe that the coverage in both studies remains mostly constant over the weekends as our school has a high percentage of commuters and thus mobile users are not on campus (as we see on days 4 to 6, and 11 to 13).

Figure 6.10 overlays the collected WiFi data over our campus map for both methods. The WiFi signal strength data is plotted with the following color coding: green for areas with strong signal; blue for areas with medium signal; yellow for areas with low signal; and red for areas with no Campus WiFi signal. Overall, the mobile gaming approach doubles the area coverage compared to the micro-payment approach (87% vs. 46% of the campus).

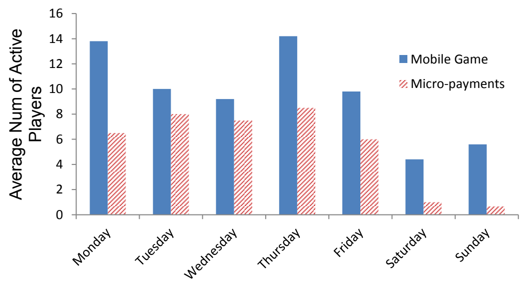


Figure 6.11 Average number of active players in a given week: mobile gaming vs. micro-payments.

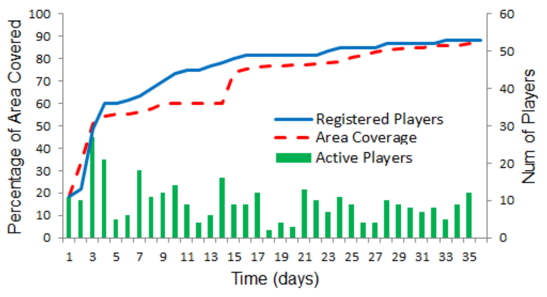


Figure 6.12 Impact of the number of registered players and the number of active players on area coverage over time in the game.

The complete details of the WiFi area coverage maps for both studies are available on-line under the game website [124].

Furthermore, user participation in the mobile game is better than the micro-payments. Figure 6.11 proves that the average number of players who participated in the mobile game is higher on any given day of the week.

Effectiveness and performance of mobile game. Figure 6.12 presents the impact of the number of registered players and the number of active players on area coverage over time in the mobile sensing game. The results show the improvement in area coverage with the increase in the number of registered players in the game. This proves that the players are interested in the game and are involved in tracking the aliens. The players are consistently active in the weekdays over the period of the study and the players are less active in the weekends. For more insights on the individual contribution of the players, Figure 6.13 presents the players' ranks based on number of covered squares in the area. We observe a power-law distribution of each player's contribution to the area coverage.

Players Engagement. Table 6.3 shows the level of overall player engagement in the sensing game. These results demonstrate that the levels and achievements designed in the game motivate the players to continue playing. The decrease in percentage of players who

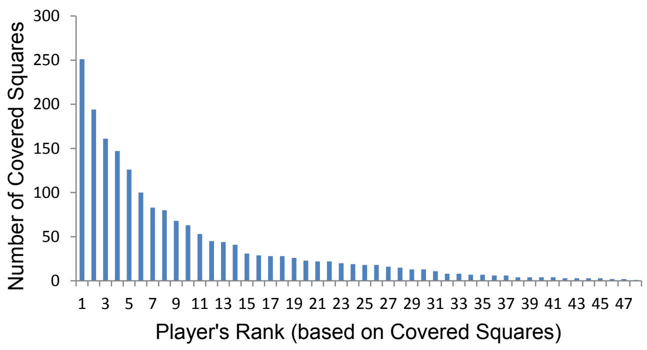


Figure 6.13 Ranking players based on the number of covered squares in the area (area is divided in 10mx10m squares).

Table 6.3 Players' Engagement based on Achievements Designed in the Game

Game Achievements	**Num of Players who unlocked this achievements**	**Percentage of Players who unlocked this achievements**	**Mean time to unlock this achievement (days)**
Genie Hints	46	87%	1.7
Monster Hunter	44	83%	0.8
Best Tracker	33	62%	0.9
Level 1	21	40%	7.4
Sentinel	15	28%	5.5
Search Remotely	14	26%	5.7
Level 2	9	17%	9.6
Bonus Power	7	13%	8.8
GRT PLAY => GRT PWR!!	6	11%	10.0
Level 3	6	11%	12.0
Double Power	5	9%	11.8
Level 4	5	9%	19.5
Extra Power	4	8%	16.4
Level 5	3	6%	12.9
Legendary Hunter	1	2%	17.2

advance to higher levels or unlock more difficult achievements shows that the game design was challenging and could sustain player interest.

To understand the effort put by players into achieving this high coverage, we compute the entropy of the area coverage. The entropy is calculated at the end of the user study using

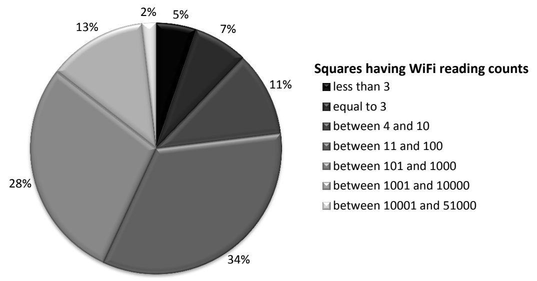


Figure 6.14 Distribution of square coverage using the mobile game.

the counts of sensor readings for each of the covered squares in the area (cf. Equation 6.1). The calculated value for our study is 6.3, while the optimal value is 9.2 (i.e., when every square in the area is uniformly covered with the same number of sensor readings). This result shows that the players' effort is substantial, and future work should focus on additional game strategies to reduce this effort. To understand better the potential for improvement, Figure 6.14 shows the distribution of square coverage across the entire area. We consider only the squares that have been covered. In the user study, we required that each square be covered with at least three WiFi readings. Nevertheless, we also plot the squares covered with only one or two readings. From the plot, we observe that 15% of the squares are covered with over 1000 readings each. We believe that these squares cover the campus center building and the labs, and the high number of readings is just a by product of the fact that students spend most of the time there. The next two categories, between 11 and 1000 readings, represent 45% of the squares. Our future goal is to devise strategies that reduce the coverage in these categories while, at the same time, increase the coverage of the remaining categories.

Analysis of Mobile Game Incentives. Figure 6.15 presents the correlation between the number of aliens killed and area coverage. The results show that our strategy of placing the aliens strategically across the area achieves its goal, as we observe a clear correlation

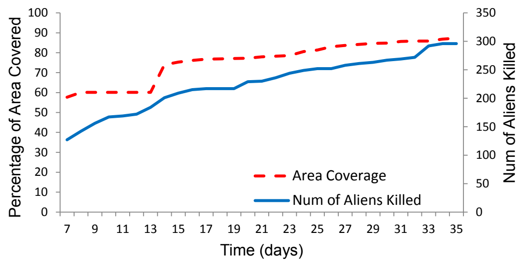


Figure 6.15 Correlation between number of aliens killed and area coverage (we started to record this data in day 7).

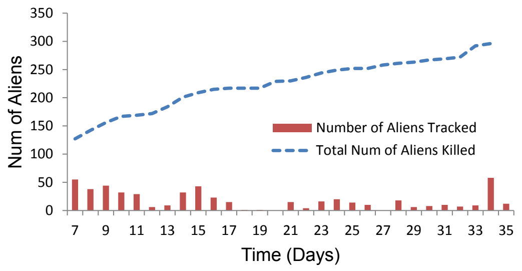


Figure 6.16 Efficacy of Progressive Movement strategy: total number of aliens killed vs. number of aliens tracked by the players over time.

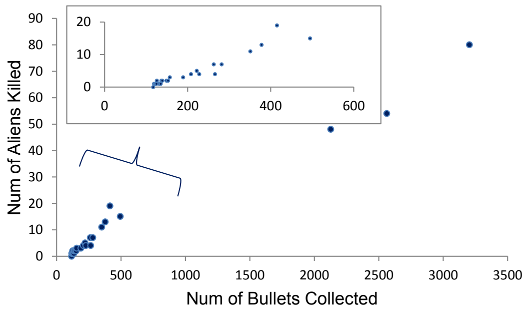


Figure 6.17 Correlation between number of bullets and aliens killed. The calculated correlation value is 0.98 (high positive correlation).

between the number of aliens killed and area coverage. For a more in-depth analysis, we investigate the performance of the Progressive Movement strategy. Figure 6.16 shows the total number of aliens killed over time and the number of aliens tracked by the players. We observe that the players have been actively tracking the aliens in the first two weeks, and this results in more aliens killed; consequently, higher area coverage was achieved. After that, since only areas located farther away from the players normal places and paths are left, we wee a decrease in the number of tracked aliens. However, the players are still interested in killing the aliens on their paths. Finally, in the last week, we announced a number of prizes for the best ranked players. The prizes were of little monetary value. However, they work well to incentivize the players to track the aliens to the least popular regions as demonstrated by the sudden spike toward the end of the period. Given the symbolic value

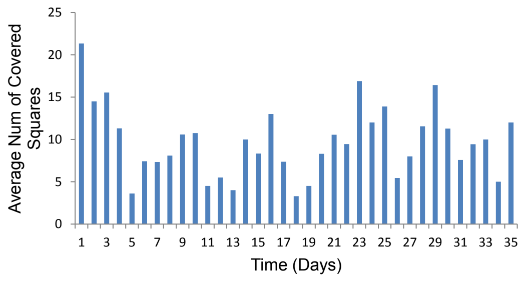


Figure 6.18 Average number of covered squares per player per day.

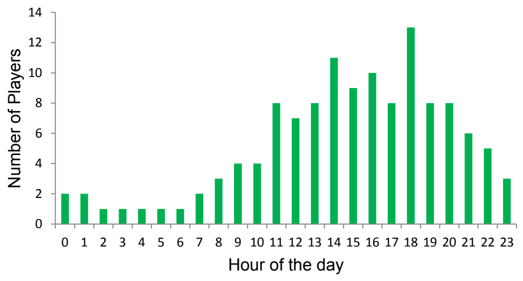


Figure 6.19 Average daily pattern for the number of active players.

of these prizes, they could be considered similar to game-based incentives. Thus, they do not change the incentive assumptions of our user study.

Another factor that is expected to incentivize the users to continue playing, is the number of bullets that can be collected from around them. Figure 6.17 presents the correlation between number of bullets and aliens killed. As already demonstrated, the number of killed aliens is a good indicator for area coverage. We observe two types of players: 1) players who collect many bullets around them, and 2) players who collect few bullets. The top 3 ranked players collected a lot of bullets, and subsequently were able to kill more aliens. These results provide two insights. First, the number of collected bullets is indeed a good, although indirect, indicator for area coverage. Second, this incentive may have to be re-designed to impact a larger number of users.

The next set of results analyze how active the players were during the study. Figure- 6.18 shows the average number of covered squares per player per day. We observe a weekly cycle, in which the players are more active during the early part of the week. This is due to two reasons: (1) at the beginning of each week, we emailed the latest ranking to all participants and this proved to be a good incentive for players; (2) the students spend less time on campus during the weekend. This behavior can be leveraged in future games. For example, the game can be designed to provide additional in-game incentives for known periods of low activity. As expected, the players also show a clear daily pattern (Figure 6.19), which can be leveraged in future games as well. For instance, more aliens should be placed around the players during periods of high activity (e.g., 2PM-6PM).

Finally, we evaluate the indoor area coverage, which could be very difficult outside labs and classrooms. Since some offices are closed to students, while some buildings contain administrative offices where students do not "dare" to go, we believe that the 35% coverage of the upper floors achieved in the study is a promising result. The players mostly covered the hallways and open spaces in each building. Our conjecture is that incentives designed specifically for these places would probably help to increase this coverage. Figure- 6.20 plots the correlation of active players and the number of squares covered at upper floor levels over time (we started to place aliens on upper floors on day 12). These results demonstrate that indoor coverage correlates well with the number of active players, and the pattern is similar to outdoor coverage. Overall, the game achieved a 35% coverage of the upper floors. Despite its apparent low value, this result is encouraging: The players covered many hallways and open spaces in each building, but could not go into offices and other spaces that are closed to students (where aliens could be placed as well).

At the end of the user study, we collected game feedback from the players to understand the effectiveness of the game design by asking them "What made you to continue playing the Alien vs. Mobile User game?". We received answers from 16 players. The responses show that the majority of the players were curious about the game, and they liked tracking the aliens hiding in the campus buildings. Other reasons for playing, based on their counts in the players' answers, were: moving to the next game levels and being on top of the leaderboard; competing with friends; winning game achievements; and checking the

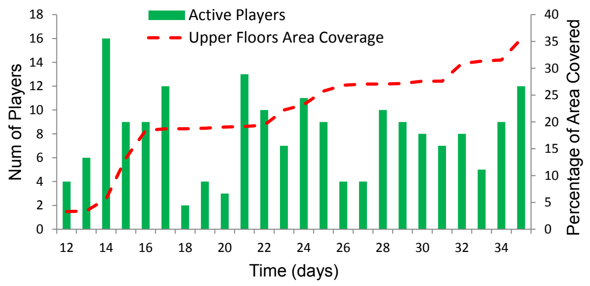


Figure 6.20 Correlation of active players and the number of squares covered at different floor levels over time in last two weeks of the user study.

game graphics. Lastly, very few players were interested to win the game prizes announced at the end.

6.6 Design Guidelines for Mobile Sensing Games

This section generalizes several design principles for mobile sensing games based on our observations in the overall game progress [125]. These principles represent a step toward transforming existing mobile games (e.g., first person shooter games) into sensing games. Several of these principles are related to maintaining the players' interest in the game at a high level over time. Note that we use aliens as characters, but they can be replaced with any other game character that will raise players' interest. In addition, these principles identify factors that contribute to maximizing the utility of the sensing game.

Game Difficulty Level: The movement patterns of the aliens should have a certain degree of unpredictability such that the players remain interested in searching for the aliens. The alien movement strategy must be balanced between a random pattern (which would make the game too difficult) and an optimal area coverage pattern (which would make the game too easy, e.g., aliens always move to adjacent squares).

Balanced Number of Aliens: When many aliens are introduced in the game, players will find them often, leading to a decrease in players' interest. If too few aliens appear in the game, players may also lose interest because the aliens are too hard to find. Hence, the strategy to maintain the number of aliens must strike a balance between these two extremes.

In-game Incentives: Players should receive interesting in-game incentives. These can include player rankings based on a point structure (which incentivizes players to be more active and collect more points), alien-finding hints, or awarding a higher number of points for destroying aliens in unpopular regions. It also includes leveraging the sensed data into the game when possible.

Mobile Resource Usage: When possible, the game should reduce the computational load on the phones in order to maximize their performance and battery lifetime. Computationally extensive tasks should be offloaded to the server side, which is not energy constrained.

Network Data Usage: The amount of data transferred between the players' phones and the game server should be balanced. An interesting option is to give players the ability to control the frequency of game status updates, thus choosing the desired trade-off between game accuracy and saving the phone's resources.

6.7 Chapter Summary

This chapter presented a mobile sensing game designed to incentivize mobile users to collect sensing data across large areas. The "Alien vs. Mobile User" game incorporates game story-related incentives to convince participants to cover all the regions of a target area. After analyzing three strategies to attract users to unpopular regions in order to cover the entire target area, we concluded that Progressive Movement leads to the lowest coverage latency while incurring a reasonable user effort as measured by the entropy of the area coverage. The game was prototyped for Android-based mobile devices and deployed as part of a user study in our campus. Based on a comparison with results obtained from a study employing micro-payments as incentives for crowd sensing, we conclude that mobile gaming can be a successful alternative for efficient area coverage in crowd sensing. The user study results demonstrate that mobile gaming ensures high area coverage. Furthermore, the results show that our game is able to find a good balance between attempting to attract users to the uncovered regions quickly and maintaining player interest in the game.

CHAPTER 7

CONCLUSION

Encouraged by the ever expanding ecosystem of mobile devices such as smart phones, tablets and even automobiles, we envisioned in this proposal a novel approach to tackle the ever more vigorous problem of data reliability in mobile crowd sensing. Our approach is based on improving the location reliability as a step toward achieving data reliability in sensed data. In addition, a cost-effective gamification technique is proposed to increase user participation for achieving large scale region coverage without (or with minimal) cost. A mobile crowd sensing system is designed, developed and used to evaluate the proposed protocols.

The results of the location reliability protocols demonstrate that they successfully detects the false location claims associated with the sensed data and the low power consumption of these protocols on user phones alleviates the users' power concerns. Extensive simulation results and security analysis show that the proposed protocols work well at various node densities and quickly thwart attacks from individual malicious claimers or malicious verifiers. Over time, the proposed protocols also detect a number of more complex attacks involving groups of users colluding out-of-band.

Based on a comparison with results obtained from a study employing micro-payments as incentives for crowd sensing, we conclude that mobile gaming can be a successful alternative for efficient area coverage in crowd sensing. The user study results demonstrate that mobile gaming ensures high area coverage. Furthermore, the results show that our game is able to find a good balance between attempting to attract users to the uncovered regions quickly and maintaining player interest in the game. After analyzing three strategies to attract users to unpopular regions in order to cover the entire target area, we concluded that Progressive Movement leads to the lowest coverage latency while incurring a reasonable user effort as measured by the entropy of the area coverage.

We believe that data reliability and incentive mechanisms such as those proposed in this dissertation will pave the way toward large scale mobile crowd sensing, and implicitly toward many useful services for society in areas such as transportation, healthcare, and environment protection.

REFERENCES

[1] O. Riva and C. Borcea, "The Urbanet revolution: Sensor power to the people!" *Pervasive Computing, IEEE*, vol. 6, no. 2, pp. 41–49, 2007.

[2] Global smartphone shipments forecast. Retrieved Nov 12th, 2014. [Online]. Available: http://www.statista.com/statistics/263441/global-smartphone-shipments-forecast/.

[3] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich, "Mobiscopes for human spaces," *Pervasive Computing, IEEE*, vol. 6, no. 2, pp. 20–29, 2007.

[4] R. Honicky, E. A. Brewer, E. Paulos, and R. White, "N-smarts: networked suite of mobile atmospheric real-time sensors," in *Proceedings of the second ACM SIGCOMM workshop on Networked systems for developing regions*. ACM, 2008, pp. 25–30.

[5] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson, "People-centric urban sensing," in *Proceedings of the 2nd annual international workshop on Wireless internet*. ACM, 2006, p. 18.

[6] P. Mohan, V. N. Padmanabhan, and R. Ramjee, "Nericell: rich monitoring of road and traffic conditions using mobile smartphones," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 323–336.

[7] T. Choudhury, S. Consolvo, B. Harrison, J. Hightower, A. LaMarca, L. LeGrand, A. Rahimi, A. Rea, G. Bordello, B. Hemingway *et al.*, "The mobile sensing platform: An embedded activity recognition system," *Pervasive Computing, IEEE*, vol. 7, no. 2, pp. 32–41, 2008.

[8] D. P. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and F. L. Wong, "SenSay: A Context-Aware Mobile Phone." in *International Symposium on Wearable Computers*, vol. 3, 2003, p. 248.

[9] A. Kansal, M. Goraczko, and F. Zhao, "Building a sensor network of mobile phones," in *Proceedings of the 6th international conference on Information processing in sensor networks*. ACM, 2007, pp. 547–548.

[10] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: the design, implementation and evaluation of the CenceMe application," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 337–350.

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

[12] H. Lu, W. Pan, N. D. Lane, T. Choudhury, and A. T. Campbell, "SoundSense: scalable sound sensing for people-centric applications on mobile phones," in *Proceedings of the 7th international conference on Mobile systems, applications, and services*. ACM, 2009, pp. 165–178.

[13] Amazon Mechanical Turk. Retrieved Nov 12th, 2014. [Online]. Available: http://www.mturk.com.

[14] chacha: Your mobile bff. Retrieved Nov 12th, 2014. [Online]. Available: http://www.chacha.com.

[15] A. Gupta, W. Thies, E. Cutrell, and R. Balakrishnan, "mClerk: enabling mobile crowdsourcing in developing regions," in *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*. ACM, 2012, pp. 1843–1852.

[16] P. Narula, P. Gutheim, D. Rolnitzky, A. Kulkarni, and B. Hartmann, "MobileWorks: A Mobile Crowdsourcing Platform for Workers at the Bottom of the Pyramid." in *Human Computation*, 2011.

[17] Smart phone sensing research @ Dartmouth College. Retrieved Nov 12th, 2014. [Online]. Available: http://sensorlab.cs.dartmouth.edu/research.html.

[18] Urban sensing research. Retrieved Nov 12th, 2014. [Online]. Available: http://urban.cens.ucla.edu/.

[19] Waze. Way to go! Retrieved Nov 12th, 2014. [Online]. Available: http://waze.com.

[20] Reality mining project. Retrieved Nov 12th, 2014. [Online]. Available: http://reality.media.mit.edu/.

[21] S. Mardenfeld, D. Boston, S. J. Pan, Q. Jones, A. Iamntichi, and C. Borcea, "Gdc: Group discovery using co-location traces," in *Social computing (SocialCom), 2010 IEEE second international conference on*. IEEE, 2010, pp. 641–648.

[22] Garmin, Edge 305. Retrieved Nov 12th, 2014. [Online]. Available: www.garmin.com/products/edge305/.

[23] Intel Labs, The Mobile Phone that Breathes. Retrieved Nov 12th, 2014. [Online]. Available: http://scitech.blogs.cnn.com/2010/04/22/the-mobilephone-that-breathes/.

[24] MIT News. Retrieved Nov 12th, 2014. [Online]. Available: http://web.mit.edu/newsoffice/2009/blood-pressure-tt0408.html.

[25] Mobile Millennium Project. Retrieved Nov 12th, 2014. [Online]. Available: http://traffic.berkeley.edu/.

[26] S. Mathur, T. Jin, N. Kasturirangan, J. Chandrasekaran, W. Xue, M. Gruteser, and W. Trappe, "Parknet: drive-by sensing of road-side parking statistics," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 123–136.

[27] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan, "The pothole patrol: using a mobile sensor network for road surface monitoring," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM, 2008, pp. 29–39.

[28] Google Mobile Ads. Retrieved Nov 12th, 2014. [Online]. Available: http://www.google.com/ads/mobile/.

[29] MobAds. Retrieved Nov 12th, 2014. [Online]. Available: http://www.mobads.com/.

[30] J. Herrera, D. Work, R. Herring, X. Ban, Q. Jacobson, and A. Bayen, "Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 4, pp. 568–583, 2010.

[31] J. White, C. Thompson, H. Turner, B. Dougherty, and D. Schmidt, "WreckWatch: automatic traffic accident detection and notification with smartphones," *Mobile Networks and Applications*, vol. 16, no. 3, pp. 285–303, 2011.

[32] K. Toyama, R. Logan, and A. Roseway, "Geographic location tags on digital images," in *Proceedings of the eleventh ACM international conference on Multimedia*. ACM, 2003, pp. 156–166.

[33] Photo journalism website. Retrieved Nov 12th, 2014. [Online]. Available: http://www.flickr.com/groups/photojournalism.

[34] Sensordrone: The 6th Sense of Your Smartphone. Retrieved Nov 12th, 2014. [Online]. Available: http://www.sensorcon.com/sensordrone.

[35] Trusted Platform Module. Retrieved Nov 12th, 2014. [Online]. Available: http://www.trustedcomputinggroup.org/developers/trusted_platform_module.

[36] ARM. Trustzone-enabled processor. Retrieved Nov 12th, 2014. [Online]. Available: http://www.arm.com/pdfs/DDI0301D_arm1176jzfs_r0p2_trm.pdf.

[37] J. Srage and J. Azema, "M-Shield mobile security technology," Texas Instruments White paper, Tech. Rep., 2005.

[38] N. Sastry, U. Shankar, and D. Wagner, "Secure verification of location claims," in *Proc. of the 2nd ACM Workshop on Wireless Security (Wise'03)*, Sep 2003, pp. 1–10.

[39] S. Capkun and J. Hubaux, "Secure positioning of wireless devices with application to sensor networks," in *INFOCOM'05. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 3. IEEE, 2005, pp. 1917–1928.

[40] S. Čapkun, M. Čagalj, and M. Srivastava, "Secure localization with hidden and mobile base stations," in *in Proceedings of IEEE INFOCOM*. Citeseer, 2006.

[41] T. Kindberg, L. Zhang, and N. Shankar, “Context Authentication Using Constrained Channels,” in *Proc. of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'02)*, Jun 2002, pp. 14–21.

[42] C. Wullems, O. Pozzobon, and K. Kubik, “Trust your receiver? Enhancing location security,” *GPS World*, vol. 1, Oct 2004.

[43] S.Brands and D. Chaum, “Distance-bounding protocols,” in *Proc. of Workshop on the Theory and Application of of Cryptographic Techniques (EUROCRYPT'93)*, May 1993, pp. 344–359.

[44] K. Rasmussen and S. Čapkun, “Location privacy of distance bounding protocols,” in *Proc. of the 15th ACM Conference on Computer and Communications Security (CCS'08)*, Oct 2008, pp. 149–160.

[45] A. Vora and M. Nesterenko, “Secure Location Verification Using Radio Broadcast,” *IEEE Trans. Dependable Secur. Comput.*, vol. 3, no. 4, pp. 377–385, 2006.

[46] T. Jiang, H. J. Wang, and Y.-C. Hu, “Preserving location privacy in wireless LANs,” in *Proceedings of the 5th international conference on Mobile systems, applications and services*. ACM, 2007, pp. 246–257.

[47] Y. Chon, N. D. Lane, Y. Kim, F. Zhao, and H. Cha, “Understanding the coverage and scalability of place-centric crowdsensing,” in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013, pp. 3–12.

[48] S. Reddy, D. Estrin, M. Hansen, and M. Srivastava, “Examining micro-payments for participatory sensing data collections,” in *Proceedings of the 12th ACM international conference on Ubiquitous computing*. ACM, 2010, pp. 33–36.

[49] S. Reddy, K. Shilton, G. Denisov, C. Cenizal, D. Estrin, and M. Srivastava, “Biketastic: sensing and mapping for better biking,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2010, pp. 1817–1820.

[50] L. Deng and L. P. Cox, “Livecompare: grocery bargain hunting through participatory sensing,” in *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*. ACM, 2009, p. 4.

[51] S. Mavandadi, S. Dimitrov, S. Feng, F. Yu, R. Yu, U. Sikora, and A. Ozcan, “Crowd-sourced BioGames: managing the big data problem for next-generation lab-on-a-chip platforms,” *Lab on a Chip*, vol. 12, no. 20, pp. 4102–4106, 2012.

[52] K. Han, E. A. Graham, D. Vassallo, and D. Estrin, “Enhancing Motivation in a Mobile Participatory Sensing Project through Gaming,” in *Proceedings of 2011 IEEE 3rd international conference on Social Computing (SocialCom'11)*, 2011, pp. 1443–1448.

[53] M. Talasila, R. Curtmola, and C. Borcea, “Mobile Crowd Sensing Chapter,” in *Handbook of Sensor Networking: Advanced Technologies and Applications*. CRC Press, 2014.

[54] A. Campbell, S. Eisenman, N. Lane, E. Miluzzo, R. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G. Ahn, "The rise of people-centric sensing," *Internet Computing, IEEE*, vol. 12, no. 4, pp. 12–21, 2008.

[55] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava, "Participatory sensing," in *In: Workshop on World-Sensor-Web (WSW'06): Mobile Device Centric Sensor Networks and Applications*, 2006, pp. 117–134.

[56] J. Froehlich, M. Y. Chen, S. Consolvo, B. Harrison, and J. A. Landay, "MyExperience: a system for in situ tracing and capturing of user feedback on mobile phones," in *Proceedings of the 5th international conference on Mobile systems, applications and services*. ACM, 2007, pp. 57–70.

[57] SenseWeb Project. Retrieved Nov 12th, 2014. [Online]. Available: http://research.microsoft.com/nec/senseweb.

[58] S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and A. Schmidt, "Micro-blog: sharing and querying content through mobile phones and social participation," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*. ACM, 2008, pp. 174–186.

[59] E. Miluzzo, C. T. Cornelius, A. Ramaswamy, T. Choudhury, Z. Liu, and A. T. Campbell, "Darwin phones: the evolution of sensing and inference on mobile phones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 5–20.

[60] M. Ra, B. Liu, T. La Porta, and R. Govindan, "Medusa: A programming framework for crowd-sensing applications," in *Proceedings of the 10th international conference on Mobile systems, applications, and services (MobiSys'12)*. ACM, 2012, pp. 337–350.

[61] T. Yan, M. Marzilli, R. Holmes, D. Ganesan, and M. Corner, "mCrowd: a platform for mobile crowdsourcing," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys'09)*. ACM, 2009, pp. 347–348.

[62] Twitter. Retrieved Nov 12th, 2014. [Online]. Available: http://twitter.com/.

[63] M. Demirbas, M. A. Bayir, C. G. Akcora, Y. S. Yilmaz, and H. Ferhatosmanoglu, "Crowd-sourced sensing and collaboration using twitter," in *World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a*. IEEE, 2010, pp. 1–9.

[64] T. Yan, V. Kumar, and D. Ganesan, "Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones," in *Proceedings of the 8th international conference on Mobile systems, applications, and services*. ACM, 2010, pp. 77–90.

[65] G. Xu, C. Borcea, and L. Iftode, "A Policy Enforcing Mechanism for Trusted Ad Hoc Networks," *Dependable and Secure Computing, IEEE Transactions*, vol. 8, no. 3, pp. 321–336, 2011.

[66] P. Gilbert, J. Jung, K. Lee, H. Qin, D. Sharkey, A. Sheth, and L. Cox, "YouProve: authenticity and fidelity in mobile sensing," in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys'11)*. ACM, 2011, pp. 176–189.

[67] A. Dua, N. Bulusu, W. Feng, and W. Hu, "Towards trustworthy participatory sensing," in *HotSec'09: Proceedings of the Usenix Workshop on Hot Topics in Security*, 2009.

[68] J. McCune, B. Parno, A. Perrig, M. Reiter, and H. Isozaki, "Flicker: An execution infrastructure for TCB minimization," *SIGOPS Operating Systems Review*, vol. 42, no. 4, pp. 315–328, 2008.

[69] M. Nauman, S. Khan, X. Zhang, and J. Seifert, "Beyond kernel-level integrity measurement: enabling remote attestation for the android platform," *Trust and Trustworthy Computing*, pp. 1–15, 2010.

[70] F. B. Schneider, K. Walsh, and E. G. Sirer, "Nexus authorization logic (NAL): Design rationale and applications," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 8:1–8:28, Jun. 2011.

[71] P. Gilbert, L. Cox, J. Jung, and D. Wetherall, "Toward trustworthy mobile sensing," in *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications (HotMobile'10)*. ACM, 2010, pp. 31–36.

[72] S. Saroiu and A. Wolman, "I am a sensor, and I approve this message," in *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications (HotMobile'10)*. ACM, 2010, pp. 37–42.

[73] T. Humphreys, B. Ledvina, M. Psiaki, B. OHanlon, and P. Kintner Jr, "Assessing the spoofing threat: Development of a portable GPS civilian spoofer," in *Proceedings of the ION GNSS International Technical Meeting of the Satellite Division*, 2008.

[74] N. Tippenhauer and S. Čapkun, "Id-based secure distance bounding and localization," in *Proc. of the 14th European conference on Research in computer security (ESORICS'09)*, Sep 2009, pp. 621–636.

[75] J. T. Chiang, J. Haas, and Y.-C. Hu, "Secure and Precise Location Verification using Distance Bounding and Simultaneous Multilateration," in *Proc. of the 2nd ACM Conference on Wireless Network Security (WiSec'09)*, Mar. 2009, pp. 181–192.

[76] N. Chandran, V. Goyal, R. Moriarty, and R. Ostrovsky, "Position Based Cryptography," in *Proc. of the the 29th International Cryptology Conference (CRYPTO'09)*, Aug. 2009, pp. 391–407.

[77] V. Shmatikov and M.-H. Wang, "Secure Verification of Location Claims with Simultaneous Distance Modification," in *Proc. of the 12th Annual Asian Computing Science Conference (Asian'07)*, Dec. 2007, pp. 181–195.

[78] S. Capkun, K. El Defrawy, and G. Tsudik, "Group distance bounding protocols," in *TRUST'11. Proceedings of the 4th international conference on Trust and trustworthy computing*. Springer, 2011, pp. 302–312.

[79] S. Čapkun, K. Rasmussen, M. Čagalj, and M. Srivastava, "Secure location verification with hidden and mobile base stations," *IEEE Transactions on Mobile Computing*, pp. 470–483, 2008.

[80] Z. Ren, W. Li, and Q. Yang, "Location verification for VANETs routing," in *Wireless and Mobile Computing, Networking and Communications, 2009. WIMOB 2009. IEEE International Conference on*. IEEE, 2009, pp. 141–146.

[81] J. Manweiler, R. Scudellari, and L. Cox, "SMILE: Encounter-based trust for mobile social services," in *Proceedings of the 16th ACM conference on Computer and communications security (CCS'09)*. ACM, 2009, pp. 246–255.

[82] A. Kalamandeen, A. Scannell, E. de Lara, A. Sheth, and A. LaMarca, "Ensemble: cooperative proximity-based authentication," in *Proc. of MobiSys '10*. ACM, 2010, pp. 331–344.

[83] Z. Zhu and G. Cao, "Applaus: A privacy-preserving location proof updating system for location-based services," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 1889–1897.

[84] S. Buchegger and J.-Y. L. Boudec, "Performance Analysis of the CONFIDANT Protocol," in *Proc. of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'02)*, Jun. 2002, pp. 226–236.

[85] P. Michiardi and R. Molva, "CORE: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks," in *Proc. of the IFIP TC6/TC11 6th Joint Working Conference on Communications and Multimedia Security*, Feb. 2002, pp. 107–121.

[86] X. Chu, X. Chen, K. Zhao, and J. Liu, "Reputation and trust management in heterogeneous peer-to-peer networks," *Springer Telecommunication Systems*, vol. 44, no. 3-4, pp. 191–203, Aug 2010.

[87] B. Yang and H. Garcia-Molina, "PPay: micropayments for peer-to-peer systems," in *Proceedings of the 10th ACM conference on Computer and communications security*. ACM, 2003, pp. 300–310.

[88] P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge, "Incentives for sharing in peer-to-peer networks," in *Electronic Commerce*. Springer, 2001, pp. 75–87.

[89] R. L. Rivest and A. Shamir, "PayWord and MicroMint: Two simple micropayment schemes," in *Security Protocols*. Springer, 1997, pp. 69–87.

[90] S. Reddy, D. Estrin, and M. Srivastava, "Recruitment framework for participatory sensing data collections," in *Pervasive Computing*. Springer, 2010, pp. 138–155.

[91] M. Musthag, A. Raij, D. Ganesan, S. Kumar, and S. Shiffman, "Exploring micro-incentive strategies for participant compensation in high-burden studies," in *Proceedings of the 13th international conference on Ubiquitous computing*. ACM, 2011, pp. 435–444.

[92] J.-S. Lee and B. Hoh, "Dynamic pricing incentive for participatory sensing," *Pervasive and Mobile Computing*, vol. 6, no. 6, pp. 693–708, 2010.

[93] C. E. Palazzi, G. Marfia, and M. Roccetti, "Combining web squared and serious games for crossroad accessibility," in *Serious Games and Applications for Health (SeGAH), 2011 IEEE 1st International Conference on*. IEEE, 2011, pp. 1–4.

[94] F. Alt, A. S. Shirazi, A. Schmidt, U. Kramer, and Z. Nawaz, "Location-based Crowdsourcing: Extending Crowdsourcing to the Real World," in *Proceedings of the 6th Nordic Conference on Human-Computer Interaction: Extending Boundaries (NordiCHI'10)*. ACM, 2010, pp. 13–22.

[95] I. Celino, D. Cerizza, S. Contessa, M. Corubolo, D. Dell'Aglio, E. D. Valle, and S. Fumeo, "Urbanopoly - a Social and Location-based Game with a Purpose to Crowdsource your Urban Data," in *Proceedings of the 2012 IEEE International Conference on Social Computing (SocialCom'12)*, 2012, pp. 910–913.

[96] I. Guy, "Crowdsourcing in the enterprise," in *Proceedings of the 1st international workshop on Multimodal crowd sensing*. ACM, 2012, pp. 1–2.

[97] L. Barkhuus, M. Chalmers, P. Tennent, M. Hall, M. Bell, S. Sherwood, and B. Brown, "Picking pockets on the lawn: the development of tactics and strategies in a mobile game," in *UbiComp 2005: Ubiquitous Computing*. Springer, 2005, pp. 358–374.

[98] N. D. Lane, M. Mohammod, M. Lin, X. Yang, H. Lu, S. Ali, A. Doryab, E. Berke, T. Choudhury, and A. Campbell, "BeWell: A smartphone application to monitor, model and promote wellbeing," in *5th International ICST Conference on Pervasive Computing Technologies for Healthcare*, 2011, pp. 23–26.

[99] M. Talasila, R. Curtmola, and C. Borcea, "Improving Location Reliability in Crowd Sensed Data with Minimal Efforts," in *WMNC'13: Proceedings of the 6 th Joint IFIP/IEEE Wireless and Mobile Networking Conference*. IEEE, 2013.

[100] McSense Android Smartphone Application. Retrieved Nov 12th, 2014. [Online]. Available: https://play.google.com/store/apps/details?id=com.mcsense.app.

[101] Google Play Android App Store. Retrieved Nov 12th, 2014. [Online]. Available: https://play.google.com/.

[102] M. Talasila, R. Curtmola, and C. Borcea, "ILR: Improving Location Reliability in Mobile Crowd Sensing," *International Journal of Business Data Communications and Networking*, vol. 9, no. 4, pp. 65–85, 2013.

[103] G. Cardone, L. Foschini, C. Borcea, P. Bellavista, A. Corradi, M. Talasila, and R. Curtmola, "Fostering ParticipAction in Smart Cities: a Geo-Social CrowdSensing Platform," *IEEE Communications Magazine*, vol. 51, no. 6, 2013.

[104] M. Talasila, R. Curtmola, and C. Borcea, "LINK: Location verification through Immediate Neighbors Knowledge," in *Proceedings of the 7th International ICST Conference on Mobile and Ubiquitous Systems, (MobiQuitous'10).* Springer, 2010, pp. 210–223.

[105] Hadoop Website. Retrieved Nov 12th, 2014. [Online]. Available: http://hadoop.apache.org/.

[106] Memcached Website. Retrieved Nov 12th, 2014. [Online]. Available: http://memcached.org/.

[107] Y. Desmedt, "Major security problems with the unforgeable(feige)-fiat-shamir proofs of identity and how to overcome them," in *SecuriCom (1988)*, vol. 88, 1988, pp. 15–17.

[108] B. Cohen, "Incentives build robustness in BitTorrent," in *Workshop on Economics of Peer-to-Peer systems*, vol. 6, 2003, pp. 68–72.

[109] J. Douceur, "The Sybil Attack," in *Proc. of IPTPS '01*, 2002, pp. 251–260.

[110] J. Newsome, E. Shi, D. Song, and A. Perrig, "The Sybil attack in sensor networks: analysis & defenses," in *Proc. of IPSN '04*, 2004, pp. 259–268.

[111] C. Piro, C. Shields, and B. N. Levine, "Detecting the Sybil attack in mobile ad hoc networks," in *Proc. of SecureComm'06*, 2006.

[112] D. Singelee and B. Preneel, "Location verification using secure distance bounding protocols," in *Proc. of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS'05)*, Nov 2005, pp. 834–840.

[113] S. Capkun and J. Hubaux, "Secure positioning in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 221–232, 2006.

[114] A. Ranganathan, N. Tippenhauer, B. Škorić, D. Singelée, and S. Čapkun, "Design and Implementation of a Terrorist Fraud Resilient Distance Bounding System," *Computer Security–ESORICS*, pp. 415–432, 2012.

[115] W.-j. Hsu, T. Spyropoulos, K. Psounis, and A. Helmy, "Modeling time-variant user mobility in wireless mobile networks," in *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE.* IEEE, 2007, pp. 758–766.

[116] B. Peterson, R. Baldwin, and J. Kharoufeh, "Bluetooth inquiry time characterization and selection," *IEEE Transactions on Mobile Computing*, pp. 1173–1187, 2006.

[117] M. Talasila, R. Curtmola, and C. Borcea, "Collaborative Bluetooth-based Location Authentication on Smart Phones," *in Elsevier Pervasive and Mobile Computing Journal*, 2014.

[118] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. Dick, Z. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis (CODES/ISSS'10)*. ACM, 2010, pp. 105–114.

[119] A. Anand, C. Manikopoulos, Q. Jones, and C. Borcea, "A quantitative analysis of power consumption for location-aware applications on smart phones," in *IEEE International Symposium on Industrial Electronics (ISIE'07)*. IEEE, 2007, pp. 1986–1991.

[120] N. Ravi, P. Stern, N. Desai, and L. Iftode, "Accessing Ubiquitous Services Using Smart Phones," in *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (PERCOM'05)*. Los Alamitos, CA, USA: IEEE Computer Society, 2005, pp. 383–393.

[121] M. Talasila, R. Curtmola, and C. Borcea, "Alien vs. Mobile User Game: Fast and Efficient Area Coverage in Crowdsensing," in *Proceedings of the Sixth International Conference on Mobile Computing, Applications and Services (MobiCASE '14)*. ICST/IEEE, 2014.

[122] W. Zheng-jie and L. Wei, "A solution to cooperative area coverage surveillance for a swarm of mavs," *INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS*, vol. 10, 2013.

[123] Monsters vs NJIT. Retrieved Nov 12th, 2014. [Online]. Available: https://play.google.com/store/apps/details?id=com.mtlabs.games.avn.

[124] Alien vs. Mobile User Game website. Retrieved Nov 12th, 2014. [Online]. Available: http://web.njit.edu/~mt57/avmgame.

[125] M. Talasila, R. Curtmola, and C. Borcea, "Crowdsensing in the Wild with Aliens and Micro-payments," *under submission in IEEE Pervasive Computing Magazine*, 2014.