uOttawa

Privacy Preservation for Nearby-Friends and Nearby-Places Location-Based Services

by

Maryam Hezaveh

A thesis

Presented to University of Ottawa

in partial fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

in

Electrical and Computer Engineering

Supervisor: Dr. Carlisle Adams

Abstract

This thesis looks at the problem of discovering nearby friends and nearby places of interest in a privacy-preserving way using location-based services on mobile devices (e.g., smartphones). First, we propose a privacy-preserving protocol for the discovery of nearby friends. In this scenario, Alice wants to verify whether any of her friends are close to her or not. This should be done without disclosing any information about Alice to her friends and also any of the other parties' information to Alice. We also demonstrate that our approach can be efficiently applied to other similar problems; in particular, we use it to provide a solution to the socialist millionaires' problem.

Second, we propose a privacy-preserving protocol for discovering nearby places of interest. In this scenario, the proposed protocol allows Alice to learn whether there is any place that she is looking for near her. However, the location-based service (LBS) that tries to help Alice to find nearby places does not learn Alice's location. Alice can send a request to the LBS database to retrieve nearby places of interest (POIs) without the database learning what Alice fetched by using private information retrieval (PIR). Our approach reduces the client side computational overhead by applying the grid square system and the POI types ideas to block-based PIR schemes to make it suitable for LBS smartphone applications. We also show our second approach is flexible and can support all types of block-based PIR schemes.

As an item of independent interest, we also propose the idea of adding a machine learning algorithm to our nearby friends' Android application to estimate the validity of a user's claimed location to prevent users from sending a fake location to the LBS application.

Keywords: Privacy, Location-Based Services, Homomorphic Encryption, Private Information Retrieval, Smartphones Application.

Acknowledgements

This work would not have been possible without the meticulous guidance and encouragement of my supervisor, Dr. Carlisle Adams. First of all, I would like to thank him for his great support, guidance, friendship, and trust throughout my PhD's study. Dr. Adams not only advises me on how to do great research, but also teaches me how to be a rightful person. He has been and will be the role model for me in academic life and also in personal life. I would also like to thank my family for their love, support, encouragement, and sacrifices.

Dedication

I dedicate this thesis to my family; particularly, to my supportive parents who have given me all their love and kindness.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

A location-based service (LBS) is an information service that offers various types of applications based on the user's location, such as identifying the location of a person, object or place, weather service, parcel and vehicle tracking, etc. An LBS retrieves the user's geographical location from the user's mobile phone via the global positioning system (GPS), cell tower triangulation, or wireless local area network (WLAN). If the users have given that application appropriate permission at the time of application installation, the device automatically releases the user's location to the service provider applications. By granting this permission, the user's location is sent periodically, or whenever the service is required, to the service provider. However, revealing accurate location information in a manner that is completely invisible to the users can give rise to privacy concerns. While LBSs may be useful for mobile users for safety-related LBSs (for example, it is helpful if emergency services know a user's location details), users might not be aware of the loss of personal information to other third party services. Indeed, instead of using an LBS to improve life, it could easily be misused and turn into a tracking tool. Hence, an important aspect for the research community is protecting the user's real-time location while they are using LBSs.

1.1 Motivation and Threat Model

Enormous enthusiasm for geographical referencing of individual information is apparent on the Web these days. The majority of people currently use smartphones with many complex sensors closely related to their activities. Most of these smartphones are equipped with high-precision localization sensor such as a GPS receiver. GPS-enabled devices have allowed people

to store their mobility tracks, tag photographs and occasions. In addition, the number of sensors in our environment that interact with smartphones have been increased. Although most people like the convenience of using these personal communication devices, there is an inherent trade-off between convenience and privacy. Clients may not be completely aware of what information of their location is being gathered, how the data is utilized and by whom, and subsequently clients can disregard the potential risks that can happen by using their location information.

Location-aware capabilities allow the service providers to offer their users the ability to geo-reference their posts and to share their location with other users. In this way, clients can utilize the location identifier to search and browse for different resources. A wide range of LBS nearby applications have been released and become popular recently, such as Foursquare, Yelp, Facebook Nearby Places, AroundMe and Places to help users to identify their nearby locations; and Facebook Nearby Friends, Loopt, and WeChat to help users discover their nearby friends. An essential key for providing these services is to gather real-time location information on clients and additionally other logical data including client relationships and client provided content updates perhaps over long periods of time. Specifically, in nearby location applications, service providers are not only able to collect clients' location information, but also able to gather other personal information by allowing clients to write their opinions and experience in terms of reviews and tips on the visited place. Subsequently, clients' historical location data can be identified with relevant and semantic data freely accessible online and can be utilized to discover individual and sensitive data about clients and to develop comprehensive client profiles. The user activities, relationships, interests and mobility patterns could be extracted from these profiles. Although these location-based profiles may be considered helpful to improve and personalize the quality of applications for the clients, they can potentially be utilized for unwanted purposes and

can cause different levels of privacy threats. Users' mobility tracks are not only a collection of locations on a map. The content of these tracks includes the users' interests, activities, habits, and relationships. It may also disclose users' private information and secrets. It can expose the users to undesirable commercials and spam, or even threat of physical harm. All of these imply that the negative side effects of lacking location privacy have increased.

Location privacy can be defined as a special type of information privacy which concerns the ability of individuals to determine for themselves when, how, and to what extent location information about them is communicated to others. In short, control of location information is the central issue in location privacy [16]. The main aim of this thesis is to protect the users' location privacy against a passive adversary, active adversary and malicious service providers while they are using LBSs. We consider the following threats in our architecture:

Passive adversary. A malicious location based service provider (LBSP) or an external observer who has access to the data that passes between the user and the database on the communication channel but cannot change the data.

Active adversary. A malicious external observer who has access to data that passes between the user and the database on the communication channel and can modify, delete or insert data. The LBSP is considered to be trusted in Claim 4.2.

Malicious service provider. A malicious server refers to an LBSP that tries to modify, delete or insert new messages in response to the user.

Users should have the privilege of controlling the amount of information (about their location) that is revealed and shared with others. This can be achieved in different ways such as users have a right to choose not to share their location information to untrusted applications, legislating

privacy policies to force organizations and service providers to protect their users' location privacy; finally, designing a system in a privacy preserving manner, so it does not disclose users' location information to others.

1.2 Thesis Statement

During the last two decades, privacy-preserving protocols for LBSs have been proposed based on non-cryptographic and cryptographic approaches. Non-cryptographic approaches use trusted third parties to maintain the user's privacy, such as "dummy locations", "K-anonymity" and "cloaking" approaches [42][52][54][63][103]. The K-anonymity approaches protect a user's location by sending the user's query with the queries of at least K other users within that region. Therefore, the service provider cannot determine the user or the user's location. As an alternative to mixing the user's query with K-1 genuine queries, the user could generate K-1 dummy queries and send these to the service provider. The main goal of cloaking is to modify or blur the exact location of the user within a certain area (from perspective of the LBS provider).

The main drawback of these approaches is that they use a trusted third party as an anonymizer for K-anonymity approaches or cloaking approaches to prevent the service provider from guessing the user's location. Both of these approaches may protect the user's location information in many LBS applications, but neither of these is suitable for the nearby friends' application because the nearby-friends' application needs to know the identity of the person who is nearby (therefore, the K-anonymity approach is not an appropriate choice for this application). In addition, they are unsuitable because the nearby-friends' application should protect both friends in the proximity relationship. Furthermore, the approximate location or K-1 dummy query features can affect the returned information (making it less useful).

Moreover, the trusted third party-based construct has an inherent drawback: it could become the primary security vulnerability of the system since it stores sensitive information of all users. If an attacker breaches this third party, the users' private information will be compromised. Note, too, that it is difficult to find an entity that will be trusted fully by all users in practice. The main challenge is to tackle these problems to retrieve required information without disclosing any user's location information.

Our main focus in this thesis is to use cryptographic mechanisms to help the user search on her smartphone for nearby friends or particular places of interest (POIs) while keeping her location private from other parties and the location-based service provider (LBSP). The cryptographic mechanisms use encryption methods to protect user location. The main difference between our two protocols is that in the nearby friends' protocol, users interact with each other to find out if they are nearby or not, but in the nearby places protocol, the user should send her request to the service provide to find out the nearby places. We show our two proposed protocols are not only useful for location-based applications, but also suitable to apply to other kinds of applications that need to protect users' privacy while they are searching for similarity of their data with other data without revealing the value of their data, or when they are searching for specific data in a database.

As an item of independent interest, we propose the idea of adding a machine learning algorithm to our nearby friends' Android application to estimate the validity of a user's claimed location to prevent users from sending a fake location to the LBS application. Our approach provides anomaly detection by using a machine learning algorithm as an additional step to centralized and/or distributed location proof systems to enable more reliable location proofs. A location proof is a digital certificate attesting the position of a user at a specific time. Our

proposed protocol is able to distinguish between regular or expected users' locations and irregular or unexpected users' locations.

1.3 Thesis Overview

The rest of this thesis has the following structure: chapter 2 presents our first protocol "Privacy-Preserving Discovery of Nearby Friends". We also discuss the security analysis and prototype implementation of our first protocol. In chapter 3 we use the approach described in chapter 2 to construct an efficient solution to the socialist millionaires' problem. Chapter 4 constructs a PIR scheme to improve the computation cost on the client-side of smartphone applications and we present our second protocol "PIR-based Scheme for Discovering Nearby Places" based on that. We also discuss the security analysis and prototype implementation of our second protocol. In chapter 5 we propose an idea of adding a machine learning algorithm to our nearby friends' Android application of chapter 2 to estimate the validity of the user's claimed location. Finally, chapter 6 concludes this thesis.

Chapter 2

Privacy-Preserving Discovery of Nearby Friends

Location-Based Services (LBSs) use the global positioning systems and the mobile phone network to calculate the geographical position of mobile phones. If the users have given that application appropriate permission at the time of application installation, the device automatically releases the user's location to the service provider applications. By granting this permission, the user's location is sent periodically, or whenever the service is required, to the service provider. However, revealing accurate location information in a manner that is completely invisible to the users can give rise to privacy concerns.

Social networking applications have added nearby friends features by using the features of LBSs. Some of the more popular applications are Facebook nearby friends, Loopt, WeChat, etc. These applications have become popular and widely used. They need to access the exact (or obfuscated) user's location to calculate the distance between friends and return the result. However, the leakage of the user's precise location to friends and the service provider may be a concern.

Secure LBSs [42][52][54][63][103] have been proposed using cryptographic and non-cryptographic mechanisms. Most existing solutions are non-cryptographic approaches. These typically use a trusted third party as an anonymizer for K-anonymity approaches or cloaking approaches. The K-anonymity approaches protect a user's location by sending the user's query with the queries of at least K-1 other users within that region. Therefore, the service provider cannot guess the user's identity. As an alternative to mixing the user's query with K-1 genuine queries, the user could generate K-1 dummy queries and send these to the service provider. The

main goal of cloaking is to deviate or blur the exact location of the user within a certain area from the LBS provider.

As mentioned in Chapter 1, both of these approaches may protect the user's location information in many LBS applications, but neither of these is suitable for the nearby friends' application. The approximate location or K-1 dummy query features can affect the requested information. The main challenge is to tackle these problems to retrieve required information without disclosing any user's location information.

The cryptographic mechanisms use encryption methods to protect users' location. Homomorphic encryption makes it possible to perform addition and/or multiplication over encrypted data. The LBS service provider could, therefore, process the encrypted data and return required information to the user. The user who requested the data is the only one who can decrypt it. Therefore, the user location information can be kept private from the service provider. In nearby friends' applications, cryptographic mechanisms could remove all responsibility for proximity calculation from the server. Friends send their encrypted location directly to each other. On the receiver side, the encrypted data is processed and the result is returned to the sender without revealing location information.

2.1 Our Contributions

We propose a homomorphic cryptographic protocol, based on the Goldwasser-Micali probabilistic encryption scheme[45][46], for discovering nearby friends. Our protocol possesses a number of advantages over the existing ones. First, our protocol does not need a trusted third party for processing data. Second, the server simply registers users and forwards the encrypted

data between them – it cannot collude with friends to learn information about users' locations. Finally, our protocol is secure against an active adversary who has access to the ciphertext on the communication channel and a decryption oracle. The adversary cannot forge another ciphertext, which leads him to guess the user's location (IND-CCA2 security) [10]. Moreover, he cannot modify the ciphertext, which leads to forging the user's location and affects the outcome of the nearby friends' application (NM-CCA2 security) [10]. We have used the same approach to solve the socialist millionaires' problem; this is presented in Chapter 3.

The remainder of this chapter has the following structure. Section 2.2 presents a brief overview of previous work on privacy-preserving protocols for LBSs. Section 2.3 describes the details of our threat model and IND-CCA2 proposed protocol based on the Goldwasser-Micali cryptosystem. The security analysis of our proposed protocol is discussed in Section 2.4. Section 2.5 explains the efficiency of our proposed protocol. Section 2.6 gives an overview of our prototype implementation on Android. The limitations of our proposed protocol are discussed in Section 2.7. Finally, Section 2.8 summarizes the chapter.

2.2 Related Work

In this section, we give an overview of the existing location-based schemes for privacy preservation of a user's location as follows: dummy locations, K-anonymity, spatial obfuscation, encryption, and PIR-based approaches.

2.2.1 Dummy Locations

In this approach, the user generates some fake locations, which are called dummies, and sends them with her real location to the LBS. In this way, the LBS would not find out which of them is

the user's real location and the location of the user remains secure. The main advantage of this approach is that the user does not need to rely on a trusted third party to produce dummies. The difficult part of this approach is to generate the dummies in such a way that they are not recognizable from the real location. For example, if an adversary tracks the user for a period of time, he should not be able to find out which of the sent locations is the real one. Shankar et al. [92] presented an approach that assumed that the user generates a database from historical traffic and creates her dummy locations from it. In this way, her dummy locations are indistinguishable from her real location. Obviously, the security of this approach depends on the number of fake requests sent to the LBS. The main drawback of this approach is the fact that as the number of requests grows, the LBS may suspect that the user is an adversary and ignore the request. Furthermore, when the number of requests is increased, it slows down the server's response time.

Niu et al. [79] proposed a caching-based scheme, which presents two caching-aware dummy selection algorithms to preserve location privacy. However, this method is only suitable for the snapshot query in LBSs.

2.2.2 K-Anonymity

The concept of K-anonymity [32][44][77][34] in location privacy approaches tries to make sure that in a set of K users, a specific user's identity is indistinguishable from K – 1 other users. Here, K is the security parameter, which determines the level of anonymity. In K-anonymity, often a trusted location anonymizer is responsible for blurring users' locations into cloaked (i.e., obfuscation) areas, such that the probability of identifying the user is 1/K. The trusted anonymizer calculates an obfuscation area of a user that includes K-1 other users in that area [47]. By this approach, the trusted anonymizer can provide K-anonymity; that is, hiding the user's actual location by returning an area that has K-1 other people and protecting her identity

by a pseudonym. This approach assumes that all users have the same K-anonymity requirements and it suffers from scalability because the anonymizer calculates the set of K-1 other users and the obfuscation area for each user individually.

In Casper [77], the anonymizer maintains the locations of the user base in a pyramid data structure, like a Quad-tree. If the calculated obfuscation area contains k users, it will consider it as a cloaking region. Otherwise, the horizontal and vertical neighbors of the obfuscation area are added to the obfuscation area to provide enough users to apply k-anonymity. In the worst case, if the number of users is not enough, even with the neighbors, the anonymizer uses the parents of the obfuscation area and repeats this process until enough users are found. This approach guarantees k-anonymity, the minimum area and the maximum area within which the user wants to hide. Gedik et al. [34] proposed another K-anonymity scheme to support a personalized level of anonymity where each user is able to define the minimum and the maximum acceptable limits for the obfuscation area size and time periods. These approaches could present a problem due to the requirement of all the mobile users to periodically report their exact locations to the trusted location anonymizer. Storing the user's exact location at a server is not secure and the user's privacy is clearly impossible because the server could be a single point of attack.

To deal with these problems and remove the trusted anonymizer, Ghinita et al. [39] introduced a decentralized, cooperative, peer-to-peer model. The main idea is the dynamic formation of nearby peer groups that can perform location anonymization for each other to achieve reciprocity in an arbitrary spatial region, K-ASR. Using this approach, the individual in one K-ASR can cloak their location from all other K-ASRs. Consider a user u_q issuing a query and its associated K-ASR A_q. A_q satisfies the reciprocity property iff there exists a set of users AS lying in A_q such

that (i) $|AS| \geq K$, (ii) $u_q \in AS$ and (iii) every user $u \in AS$ lies in the K-ASRs of all other users in AS. Although a trusted anonymizer is not required anymore, all users must trust each other.

Similar to k-anonymity clustering, Mascetti et al. [74] presented an approach called historical k-anonymity that takes motion into consideration. They proved that an adversary could make the connection between each user and their location by observing continuous location updates. The ProvidentHider [74] algorithm uses historical information of each user to prevent this attack under certain assumptions. However, this approach did not consider query privacy. Query privacy is the ability to prevent other parties from learning the issuers of queries.

To hide the identity of the user who sends the request in a k-anonymity clustering approach, Chow et al. [22] used unstructured peer-to-peer networks. The user uses peer-to-peer communication to find a spatial region that contains k users. After clustering is finished, a random member of the cluster will send the query to the service provider server. Another fully distributed mobile peer-to-peer approach [40] called MobiHide presented Hilbert space-filling curves to anonymize a query by mapping it to a random group of k users. The mobile users should trust each other to exchange their location information and collaborate in computing cloaking regions. The main drawback of clustering k-anonymity is the overhead and latency of the group formation and entire communication.

Hu et al. [58] proposed another decentralized approach in which users collaborate to make a cloaking region without revealing their exact location to other peers. However, like previous ones, this approach considers just a snapshot of user locations and is not secure in continuous location updates.

Zhong and Hengartner [110][111] proposed two approaches to eliminate the requirement of fully trusting a single trusted third party, instead moving trust to suitable set of non-colluding entities. A number of location brokers track the location of different subsets of registered users. Each location broker learns the location and number of users that have registered with this broker, but not the total number of users in this cell (as some users may be registered with another broker). Moreover, a set of servers informs the user whether at least k users have registered in the user's current cell. The main drawbacks of their approaches are that entities (servers and users) need to trust each other, they are not secure against malicious entities, and they are vulnerable to collusion among the entities.

Ghaffari et al. [38] presented a distributed anonymizing protocol in which each mobile node collaborates with other peers to construct a specific anonymizing zone without accessing any information about other nodes. The main drawback of their protocol is that significant computation overhead on the user side may occur due to the processing of queries as well as the filtering of redundant results.

Ying and Nayak [108] proposed a Location Privacy-protection protocol based on sensor nodes which are scattered throughout the network to provide anonymized locations for users. Since the sensor nodes coverage must have a non-overlapping characteristic, it is difficult to deploy this scheme in the real world.

To enhance the anonymizing or cloaking technique, Sun et al. [97] introduced location labels to separate the locations of mobile users into sensitive and ordinary locations, and designed a location-label based (LLB) algorithm to protect the location privacy of users while minimizing the response time for LBS queries.

2.2.3 Location Obfuscation

"Location obfuscation" approaches try to preserve location privacy by changing, deviating or blurring the exact location of a user. Spatial cloaking approaches rely on a spatial area, not the number of users like K-anonymity. They may even accommodate the case that only one user is located at the spatial obfuscation region. The concept of spatial obfuscation was introduced by Ardagna et al. [4]. The user selects a circular region instead of her exact location and sends it to the LBS. This approach does not require a trusted third party to generate the obfuscation location. In addition, the user can choose their obfuscation area. However, the user should be careful about her choice because it affects the returned result. Cheng et al. [18] presented probabilistic cloaking, which preserves the privacy of the user's location by uniformly distributing it in a closed region around her. In Duckham and Kulik [28], the obfuscation regions are modeled as a set of vertices in graphs, which provides a more flexible model of geographic space than circles.

Hashem et al. [50] presented the first group-based approach that tries to find a location nearest to each of the group members. For example, a group of friends wants to meet in a restaurant that minimizes travel for all of the group members. For privacy preservation of the user's location, they offer a two-phase method. In the first phase, each user sends her imprecise location to the LBS instead of her exact location. In the second phase, the private filtering algorithm determines the exact result without disclosing the exact location of each user. Their approach preserves the location privacy for all users inside and outside of the group. However, it suffers from a high communication cost. All members have to send a distinct query to the LBS and the LBS has to send back a set of candidate answer points (where the exact result is one of them). Then, group members have to determine the exact result from the candidate set.

Damiani et al. [23] developed a semantic location cloaking framework called Probe (PRivacy-aware OBfuscation Environment). The idea here is that the user's location may have a different sensitivity depending on where the user has been. For example, being in a hospital is more sensitive than being in a crowded street. By considering this sensitivity, the user can expand their obfuscation area in such a way that the probability of being in a certain sensitive location is less than a defined threshold.

In all these schemes, the size of the obfuscation region can be reduced if the adversary has background knowledge about the users' location. In order to stop this attack, Ardagna et al. [5] introduced "Landscape-aware Location-Privacy Protection" which aims to provide a map-dependent obfuscation area by considering the probability that a user's location is actually contained in an area.

Social-aware Location-privacy Protection (SLP) [107] uses social ties to protect the privacy of the original requester and uses cloaking areas instead of exact coordinates to enhance security. In their proposed network, participants may be friends, strangers, or malicious persons. The original requester still must take the risk that strangers might learn everything about the query, as these strangers can collude to compromise the location privacy of the requester.

Multi-Hop location protection (MHLP) [59] provides location privacy by sending messages through trusted friends. This algorithm aims to provide location privacy for nodes by using an obfuscation path. In order to provide identity privacy, the last friend exchanges her information with the message sender's information to hide the sender's real identity. Messages are encrypted in this algorithm so that confidentiality is provided, and trusted friends are recognized according to a defined threshold value.

2.2.4 Cryptography-based Approaches

Cryptographic location privacy approaches use encryption to protect user location. Mascetti et al. [75] use symmetric encryption techniques to notify users when their buddies are within their proximity in order to prevent friends and a location-based service provider to find out the actual current user location. However, the location-based service provider could compare the encrypted values if the encrypted scheme is not semantically secure. Therefore, the LSB provider could collude with one of the parties to find out the location of other friends. Alternatively, it could misinform friends that they are far apart from each other even though they are actually nearby.

Zhong et al. [109] proposed three different protocols for location privacy to alert friends if and only if that friend is actually nearby. The protocols are called Louis, Lester, and Pierre. The service provider does not need to be aware of the users' locations. There are some drawbacks of each protocol such as one of the users will learn the exact location or distances of the other user in the Pierre protocol. The user can add a guessable location to check if her friend is near a special place or not in the Lester protocol. Finally, all of them require the user to calculate distance R, which is computationally expensive.

Narayanan et al. [82] presented three protocols to enable proximity testing without revealing the mobile users' real location information for privacy preserving distributed social discovery. It allows users to exchange location-based information with friends while protecting their privacy. These schemes allow two parties to exchange privately whether they are close or not, without disclosing any further information to each other, the server or any eavesdropper. However, with an oblivious server, the protocol cannot guarantee privacy if any two parties collude.

Siksnys et al. [94] presented a location-privacy aware friend-locator LBS called FriendLocator. Each user encrypts her location with a one-to-one pre-shared key. Later, the

server checks for finding a user's proximity by matching the encrypted location into four cells of a grid. This scheme has many flaws, such as lack of flexibility in user preference, dynamic-shape vicinities, and the server learns the information that was retrieved about the distance between users.

The buddy tracking application [3] presented an efficient algorithm for proximity detection called Strips. This approach focuses on the efficiency in terms of computation cost and communication complexity, such as reducing the messages exchanged between users, rather than location privacy issues. In general, cryptographic approaches try to determine whether location-based queries such as nearby friends' queries can be provided efficiently over the encrypted data.

In collaboration methods [48][67][96], each user collects her location data to generate the cloaking region. The communication channels between the users are secure: user data privacy is guaranteed with cryptographic solutions. Shokri et al. [93] presented a collaboration scheme to exchange context information among the interested users, which allows one user to answer LBS queries from another user. However, user interactions pose additional privacy risks in some cases. Generally, in a non-centralized architecture, these approaches incur a high processing overhead on the client. In addition, the redundant results returned from the LBS server also incur a high communication overhead between the user and the LBS server.

2.2.5 PIR-based Approaches

Private information retrieval (PIR) techniques have drawn a lot of attention for privacy-preserving LBS for nearby points of interest (POIs) queries [42]. PIR allows a user to query a specific record of the database, without revealing which record the user is interested in to the server. Most of the previous PIR-based approaches in the location-based services are based on

secure hardware-assistance, which makes use of a secure coprocessor at the LBS server [31][52]. The secure hardware-based PIR uses the idea of a tamper-resistant CPU, which is trusted by the user, connected to the server. The user sends his query to this CPU, while his query is unreadable by the server. The CPU extracts the requested information from the server's database and returns the results to the user. The main drawback of all secure hardware PIR schemes is that the proposed architectures are secure as long as the user can completely trust the hardware, which is often unrealistic. We review the details of previous work on PIR schemes and PIR-based LBSs in sections 4.2.1 and 4.2.2, respectively.

2.3 Proposed Privacy-Preserving Protocol for Nearby Friends

The main aim of cryptographic approaches in nearby friends is to make it possible to automatically detect nearby friends even when the user's location privacy is applied to the application. Our proposed protocol uses homomorphic encryption of the Goldwasser-Micali (GM) cryptosystem to achieve this purpose.

2.3.1 Review of the Goldwasser-Micali Scheme

An important family of homomorphic encryption schemes [90][106], using the first probabilistic public key encryption system proposed by Goldwasser and Micali in 1982 [45][46], is described in Figure 1. The GM cryptosystem uses probabilistic encryption which is based on randomness. By using randomness in the encryption algorithm, every time we encrypt the same message, the ciphertext is different. With this consideration, the computational complexity of this step is $O(l(p)^2)$, where $l(p)$ denotes the number of bits in p. Note that the computational cost of decrypting k bits to plaintext is $O(k.l(p)^2)$.

Unfortunately, by increasing the number of bits, this approach has a strong drawback. This is not very efficient for an application with long data input [30]. However, the input data of our proposed protocol does not exceed 100 bits per user in the worst case (which is an MGRS-0.1 km grid reference; see section 2.3.3). The description of how we calculate the input for our nearby friends' application is presented in the grid-based location section in more detail.

Prerequisite:	Alice computed a (public, private) key: she first choses $\boldsymbol{n} = \boldsymbol{pq}$, $\boldsymbol{p}$ and $\boldsymbol{q}$ being large prime numbers and $\boldsymbol{g}$ a quadratic nonresidue modulo n whose Jacobi symbol is 1; her public key is composed of $\boldsymbol{n}$ and $\boldsymbol{g}$, and her private key is the factorization of $\boldsymbol{n}$.
Goal:	Anyone can send an encrypted message to Alice.
Principle:	To encrypt a bit b , Bob picks at random an integer r $\in$ Z_n^*, and computes c $= g^b r^2 mod\ n$ (remark that c is a quadratic residue if and only if $b = 0$). To get back to the plaintext, Alice determines if c is a quadratic residue or not. To do so, she uses the property that the Jacobi symbol c/p is equal to $(-1)^b$. Please, note that the scheme encrypts 1 bit of information, while its output is at least 1024 bits long!
Security:	This scheme is the first one that was proved semantically secure against a passive adversary (under a computational assumption).

Figure 1 Goldwasser-Micali[45][46].

In the Goldwasser-Micali cryptosystem, as described in Figure 1, if the public key is the modulus n and quadratic non-residue g, then the encryption of a bit b is $\varepsilon(b) = g^b r^2 (mod\ n)$ for some random $r \in \{0, \dots, n-1\}$. The homomorphic property is then:

$$\varepsilon(b_1).\varepsilon(b_2) = g^{b_1} {r_1}^2 g^{b_2} {r_2}^2 (mod\ n) = g^{b_1+b_2}(r_1 r_2)^2 mod\ n = \varepsilon(b_1 \oplus b_2),$$

where "." denotes multiplication and "$\oplus$" denotes exclusive-or.

The Goldwasser-Micali encryption scheme is semantically secure. (This notion of security is also commonly referred to as Indistinguishability under a chosen-plaintext attack (IND-CPA) [46]). Semantic security means that no partial information about the plaintext can be learned from the ciphertext. However, this definition of secrecy is only valid if the adversary is passive. In the other attack scenario, the attacker who is impersonating the sender is allowed to additionally inject a ciphertext on the communication channel between the sender and receiver.

Thereby, it causes the receiver to decrypt the ciphertext and get some information back to the attacker. This attack is called “chosen-ciphertext-attack” (CCA1), where the attacker can submit any ciphertext of his choice to the decryption oracle, then he learns the entire corresponding plaintext. In this model, the attacker loses the decryption oracle after he receives the challenge ciphertext. There is also “adaptive chosen-ciphertext attack” (CCA2), where the attacker can continue querying the decryption oracle with one restriction that he cannot decrypt the challenged ciphertext. This is a quite strong attack; nevertheless, there are real scenarios in which this level of security is required (IND-CCA2).

The main goal to tackle this attack is to define valid ciphertexts in such a way that the adversary cannot forge another ciphertext that leads him to guess the plaintext (IND-CCA2); moreover, he cannot modify the ciphertext in a way that affects the outcome of the protocol (NM-CCA2). Mao [67] showed the vulnerability of the GM cryptosystem to CCA2. In section 3.5, we show our proposed protocol solves this attack by using an authenticated encryption scheme that is called “Encrypt-then-MAC” [10].

2.3.2 The Vulnerability of the GM Cryptosystem to CCA2

Mao [73] showed the vulnerability of the GM cryptosystem to CCA2 with an example as follows:

Alice plays the role of a GM decryption oracle, and Malice can send reasonable ciphertext queries to her. If the received ciphertext looks random to Alice, she returns the plaintext to Malice. Suppose that Malice eavesdropped a ciphertext $C = (c_1, c_2,\ldots, c_n)$ on a communication channel between Bob and Alice. The main goal of Malice is to find out the plaintext $B = (b_1, b_2,\ldots, b_n)$. He modifies the ciphertext as follows:

$C' = (zc_1, zc_2, \ldots, zc_n) (\text{mod N})$

Malice uses the following algebraic property:

$$ab(mod\ N) \in\ QR_N\ iff \begin{cases} a \in QR_N\ and\ b \in QR_N \\ a \in J_N(1)/QR_N\ and\ b \in J_N(1)/QR_N \end{cases},$$

He chooses z $\in J_N(1)/QR_N$. Therefore, C' looks random to Alice, and she returns the decryption $B' = (b_1', b_2', \ldots, b_n')$ to Malice. Then the "multiply-by-z" attack allows Malice to learn $B = (b_1, b_2, \ldots, b_n)$ by complementing each bit of the received B'.

Mao showed that an explicit oracle service is not necessary. Even if Alice stops decrypting Malice's random ciphertext, Malice can still find out the plaintext bit by bit with the following example:

Malice wants to know whether the decryption of c_1 is 0 or 1. He sends an encrypted question to Alice for her to answer (e.g., a question requiring a YES/NO answer). Malice encrypts the first half of the question using Alice's valid public key, but encrypts the second half of the question using $c_1$in place of g in Figure 1 above. If Alice decrypts the first part of the message correctly and the second part of the message is all zeroes, she will ask Malice why his message is incomplete. Malice will learn that c_1 is a quadratic residue and therefore the decryption of c_1 must be 0. Otherwise, Alice will answer the question correctly and Malice will learn that c_1 is a quadratic non-residue and therefore the decryption of c_1 must be 1.

With these two examples of active attack, Mao showed that the GM cryptosystem is not secure against an active attack. Therefore, the notion of IND-CPA security is not sufficiently strong against an active adversary.

2.3.3 Grid-based Location

Let us consider a client-server architecture: Alice has a list of friends, $\{U_1, U_2, \dots, U_n\} = Alice_{FriendList}$, and the service provider acts as a registration server. All users can send and receive data through the service provider. Alice would like to be notified when any of her friends are nearby.

Each user extracts her location via GPS, Wi-Fi or cell towers and then the application calculates her grid reference based on the Military Grid Reference System (MGRS) [51]. The MGRS is a system to specify point locations on the earth. It uses a grid of squares of lengths 10 km, 1 km, 100 m, 10 m and 1 m depending on the precision used. Our protocol assumes a grid system like MGRS where pairs of friends agree on the grid reference size and they consider that they are nearby if they are located in the same grid square. For example, if two friends agree on 100 m, the MGRS will be "18T VR 465 301", which includes three parts: "18T" is the grid zone designator: "VR" is the 100-meter square identifier; and "465 301" defines a numerical location where east is "465" and north is "301" [51]. Figure 2 shows the different levels of MGRS blocks for the Ottawa, Ontario, area [117].

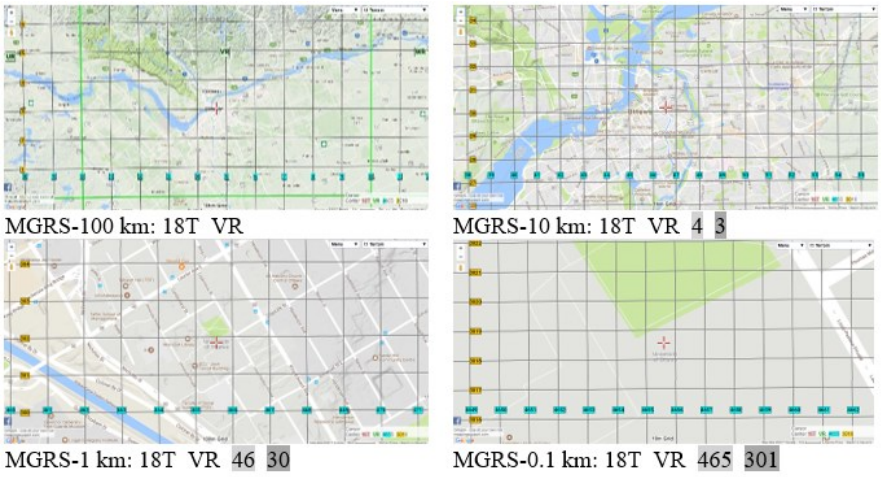


Figure 2 Different MGRS Levels [117]

2.3.4 Problem Statement

Alice and her friends have their location as their secret values. This protocol allows Alice and her friends to separately learn whether they are in a same pre-agreed grid reference or not. However, if they are not in the same grid reference, Alice does not learn the location of her friends and her friends do not learn the location of Alice. Note that our protocol does not recognize that friends are nearby if they are near each other but in different grid squares.

Our proposed protocol is based on the homomorphic feature of the Goldwasser-Micali cryptosystem. We require the proposed protocol to be secure against active and passive attacks. However, homomorphic encryption schemes are malleable by design [6][114]. To tackle this problem we use an authenticated encryption scheme called Encrypt-then-MAC [10]. Bellare and Namprempre showed that encrypting a message and subsequently applying a MAC to the ciphertext implies security against an adaptive chosen ciphertext attack [10]. We analyze the security of the proposed protocol, and we show an active adversary who has access to the ciphertext on the communication channel, and the decryption oracle, cannot forge another

ciphertext which allows him to learn Alice's location and her friends' location (IND-CCA2 security). Moreover, the active adversary cannot modify the ciphertext in a way that allows him to affect the result of the protocol (NM-CCA2 security).

The nearby friends' application notifies Alice if any of her friends are nearby her. The application could run in the background, and whenever it finds a nearby friend, a pop-up notification is displayed on Alice's phone.

2.3.5 Threat Model

For the security proofs of the proposed protocol, we consider two different types of adversary, passive and active. Note that there is a certain amount of trust required between friends. If Bob uses a fake location, instead of his real location, there is no way to detect it in the proposed protocol. However, there are some techniques to stop the user from spoofing his location [82][113][114] (see also chapter 5).

The passive adversary has access to two ciphertexts created by Alice and Bob (see section 2.3.6), C_A and C_{AB} on the communication channel, but he is not able to modify any ciphertext. Note that the GM encryption algorithm used in the proposed scheme is semantically secure. This notion of security is also commonly referred to as indistinguishability under chosen plaintext attack (IND-CPA secure). Also, a MAC computed over ciphertext cannot reveal any information about the plaintext since GM is IND-CPA secure. Therefore, the entire proposed scheme is secure against the passive attacks (IND-CPA secure).

The active adversary can manipulate the communication channel between users. Therefore, he has access to two ciphertexts C_A and C_{AB}, and the decryption oracle. He can send any modified ciphertext of his choice to the decryption oracle and receive the corresponding plaintext. The main

goal of the active adversary is to find out an individual's location, or to learn whether two individuals are in the same location.

If friends are located in the same grid reference, they learn each other's location. We desire that Alice does not learn her friend's exact location, if they are not in the same grid reference. Our proposed protocol uses the homomorphic property of the Goldwasser-Micali (GM) cryptosystem; therefore, if the result of the XOR is not zero, a malicious application on Alice's phone is able to detect her friend's location. This can be addressed by putting the application and Alice's private key in the secure element (SE) and using attestations. Note that the SE is not a trusted third party; it is a generic secure element that was built into her phone before she purchased the phone, and so it is not designed to attack Alice. The SE is simply a trusted component on her device that is assumed to execute cryptographic computations correctly. Note that the rest of her phone may or may not be trusted.

Given the assumption of having a trusted device raises the question of why we still need homomorphic encryption. Could not Alice's friend (Bob) encrypt his location with standard public-key encryption and send it to Alice's device? The SE would then decrypt the ciphertext, compare Bob's location (in plaintext) to Alice's location, and tell Alice only whether (or not) Bob is nearby.

We need homomorphic encryption to hide Bob's exact location from Alice. When Alice uses the nearby friends' application, her location is input to the application, and the application can validate the correctness of her location by using our proposed protocol in chapter 5. The application then homomorphically encrypts the (validated) location and sends the encrypted result to Bob. On the other hand, if only standard public key encryption is used, then when Bob's encrypted location comes to the SE, Alice inputs her location to SE. However, the SE is not able

to detect if Alice's location is real or fake (the SE cannot run a machine learning algorithm and is not able to access location proof services on the Internet). Therefore, Alice can learn Bob's location if the fake location that she inputs to the SE is the same as Bob's real location.

Note that in our proposed protocol, friends do not learn the exact location of their friends if they are in different grid squares. However, because we are using the gird reference system, if Alice's gird reference and Bob's grid reference have the same prefix, Alice will learn that Bob's prefix is the same as hers (see section 2.3.3).

2.3.6 Proposed Protocol

Our protocol contains two phases. The first phase is the Enrollment phase in which the whole protocol becomes ready to use, on the client side and also on the server side. This phase can be repeated in the future if any changes occur or if the client decides to change the level of her privacy. The second phase is the execution phase in which the user searches for nearby friends.

Enrollment Phase: Suppose Alice wants to use our nearby friends' application for the first time. Alice installs the application and logs on to the server by selecting a username and a password. The key generation algorithm generates a public and private key pair, (PU, PR), based on the Goldwasser-Micali cryptosystem, and a second public and private key pair, (PU_{RSA}, PR_{RSA}), based on the RSA cryptosystem, for Alice on the secure element (SE). Alice sends a friendship's request, her public key, and a preferred level of privacy to her friends. The level of privacy refers to the grid reference size (see section 2.3.3). Her friends who accept Alice's request send their public key back to her. Note that the server in our protocol just passes encrypted messages between the users and it does not learn information about user's location and whether they are close to each other or not.

Execution Phase: Suppose Alice wants to find out whether Bob is nearby her or not. The nearby process is as follows:

1. Alice first calculates the encryption of her location based on GM, $M_A = \varepsilon_{GM}(PU_A, L_A)$ where L_A is a corresponding grid reference of Alice's location.
2. Then, she applies Encrypt-then-MAC which is a composition of Encryption = (K_e, ε, D) where K_e is a key, ε is the encryption algorithm and D is the decryption algorithm for a symmetric encryption scheme, and MAC = $(K_m, T, \mathcal{V})$ where K_m is a key, T is tagging algorithm and $\mathcal{V}$ is tag-verifying algorithm in a message authentication scheme MAC. She sends "return" to Bob:

$$K_{AB-e} \leftarrow \varepsilon_{RSA}\ (PU_{RSA-B}, K_e)$$
$$K_{AB-m} \leftarrow \varepsilon_{RSA}\ (PU_{RSA-B}, K_m)$$
$$C'_A \leftarrow \varepsilon(K_e, M_A)$$
$$\tau'_A \leftarrow T(K_m, C'_A)$$
$$C_A \leftarrow C'_A || \tau'_A$$
$$return\ C_A and\ K_{AB-e}\ and\ K_{AB-m}$$

3. Bob calculates MAC to make sure the received message is from Alice, as follows:

$$Parse\ C_A\ as\ C'_A || \tau'_A$$
$$K_e \leftarrow D_{RSA}\ (PR_{RSA-B}, K_{AB-e})$$
$$K_m \leftarrow D_{RSA}\ (PR_{RSA-B}, K_{AB-m})$$
$$M_A \leftarrow D(K_e, C'_A\ )$$
$$v \leftarrow \mathcal{V}(K_m, C'_A, \tau'_A,)$$
$$if\ v = 1\ then\ return\ M_A\ else\ return\ \perp$$

4. Bob calculates $M_B = \varepsilon_{GM}\ (PU_A, L_B)$ and $M_{AB} = \varepsilon_{GM}\ (PU_A, L_B) * \varepsilon_{GM}(PU_A, L_A)$, and he sends "return" to Alice.

$$C'_{AB} \leftarrow \varepsilon\ (K_e, M_{AB})$$
$$\tau'_{AB} \leftarrow T(K_m, C'_{AB})$$

$$C_{AB} \leftarrow C'_{AB} || \tau'_{AB}$$

$$return\ C_{AB}$$

Note that $M_{AB} = \varepsilon_{GM}\ (PU_A, L_A \oplus L_B)$ by the properties of homomorphic encryption.

5. Alice calculates MAC to make sure the received message is from Bob, as follows:

$$Parse\ C_{AB}\ as\ C'_{AB} || \tau'_{AB}$$

$$M_{AB} \leftarrow D(K_e, C'_{AB})$$

$$v \leftarrow \mathcal{V}(K_m, C'_{AB}, \tau'_{AB}\ )$$

$$if\ v = 1\ then\ return\ M_{AB}\ else\ return\ \perp$$

6. Finally, to prevent Alice from learning Bob's location, our application on Alice's side sets E=0 and ORs it with decryption of the received $M_{AB} = \varepsilon_{GM}\ (PU_A, L_A \oplus L_B)$ bit by bit. Whenever the first "1" is detected, the decryption process stops. Otherwise, the result is "0" and Alice is notified that "Bob is nearby".

This protocol is asymmetric equality testing because Alice learns the answer, but Bob does not. Bob should repeat the above protocol with his own public key to learn whether Alice is nearby him or not. The protocol is based on the homomorphic feature of GM, in which the product of the encryptions of two bits is equal to the encryption of their XOR; therefore, if those two bits are the same, the result of XOR will be "zero".

Clearly, if Alice and Bob are in the same grid reference, Alice learns Bob's location. If they are not at the same grid location, we have to show that Alice does not learn Bob's location. Because of the stipulation that E=0, which is then ORed with the decryption of $\varepsilon_{GM}\ (PU_A, L_A \oplus L_B)$ bit by bit, and stops whenever the first "1" is detected, Alice does not learn Bob's location if the result of XOR is not zero. However, this stipulation does not prevent a malicious application. This can be addressed by putting the computation and Alice's private key

in the secure element (SE) and using attestations. If the risk is considered small, the application can be run by itself; if the risk seems too great, SE is used instead for the relevant computation.

2.4 Security Analysis of the Proposed Protocol

Lemma 2.1. The proposed scheme is IND-CCA2 secure.

Assumption. We assume Malice is an active adversary, which means that he can manipulate the communication channel between Alice and Bob. Therefore, Malice has access to two ciphertexts C_A and C_{AB}, and also the decryption oracle. He can send any modified ciphertext of his choice to the decryption oracle and receive the corresponding plaintext. The main goal of Malice is to find out Alice's location, or Bob's location, or whether Alice and Bob are in the same location.

Proof. Malice tries to modify C_{AB} , as follows, and sends C_M to the decryption oracle.

$$Parse\ C_{AB}\ as\ C'_{AB}||\tau'_{AB}$$
$$C_M = C'_M||\tau'_{AB}$$

The decryption oracle calculates:

$$Parse\ C_M\ as\ C'_M||\tau'_{AB}$$
$$M_M \leftarrow D_{RSA}(PR_A, C'_M)$$
$$\nu \leftarrow \mathcal{V}(K_m, C'_M, \tau'_{AB})$$
$$if\ \nu = 1\ then\ return\ M_M\ else\ return\ \perp$$

Since Malice modifies C'_{AB} to C'_M, the value of ν is not equal to 1, therefore, the decryption oracle returns $\perp$ to Malice. These chosen ciphertext queries did not help Malice to guess the plaintext. We also showed the proposed protocol is IND-CPA secure. Therefore, the proposed scheme is IND-CCA2 secure. Bellare et al. [10] proved that IND-CCA2 implies NM-CCA2. As a result,

attacker cannot modify the plaintext in a deliberately meaningful way by modifying the ciphertext.

2.5 Efficiency (Performance)

Computation time: Using the Goldwasser-Micali cryptosystem is not efficient, as a ciphertext may be several thousand times larger than the plaintext. However, our defined Android application runs automatically in the background and whenever it finds a nearby friend, it notifies the user. With this perspective, the user is not involved in the computation process even if it takes time.

Energy consumption: The amount of data processed is not large, although any computationally intensive operation that happens continually in the background will significantly impact battery drainage rate. The user can control how much background processing takes place (e.g., she can check for nearby friends every 60 minutes, or whenever her location changes).

2.6 Implementation

We implemented a client-server component to show the performance of our proposed protocol. The registration server is a tomcat server, which forwards encrypted messages between users. The user application is an Android API level 15, IceCreamSandwich, which is installed on the user's phone and, after registration, the user can add her friends. Alice and each of her friends should agree on the grid reference size (e.g.,10 km, 1 km, 100 m or 10 m). As we mentioned before, the user has control over updating her location. For example, she can check for nearby friends every 60 minutes, or whenever her location changes. We set the application to send the

update location every 10 minutes. Alice's task finishes at this point and the application does the computation part in the background. Whenever the application finds a nearby friend, a notification pops up on the user's phone.

For showing the efficiency of the Goldwasser-Micali algorithm, we ran the desktop application, which encrypts and decrypts different grid reference sizes: 10 km, 1 km, 0.1 km and 0.01 km. We tried this for different numbers of friends and we ran our application one hundred times to calculate the overall computation time for Goldwasser-Micali. Figure 3 shows the linear increase of calculation time when the number of friends increases. As expected, 10 km has the least calculation time since MGRS has only 6 characters (e.g., 4QFJ16); therefore, it reduces the calculation time in the Goldwasser-Micali cryptosystem. In the worst case, we have 14 characters for presenting each user's grid reference. Therefore, the computation process does not take long even if the number of friends increases (i.e., less than 2 seconds in the worst case in Figure 3).

We implemented our proposed protocol application and we used two "Samsung S5 neo" phones and one "Sony Ericsson XPERIA TX lt29i" to calculate the computation time for one and two friends in a real scenario. We wanted to make sure the computation time of Goldwasser-Micali on the Android application is practical. We tried it in the Ottawa region where the MGRS is "18TVR 44 14" for 1000 m, "18TVR 442 148" for 100 m and "18TVR 4420 1489" for 10 m. We tried this application 10 times, and the average computation time is shown in Figure 4.

The worst case occurred when the MGRS grid size is 10 m, which is 12 characters. However, the total computation time is sub-second for two friends. This shows that it is computationally possible to use the Goldwasser-Micali cryptosystem for realistic ciphertext sizes. As mentioned

before, all these calculation processes take place in background activity, and the user is not stuck waiting for the application to complete processing.

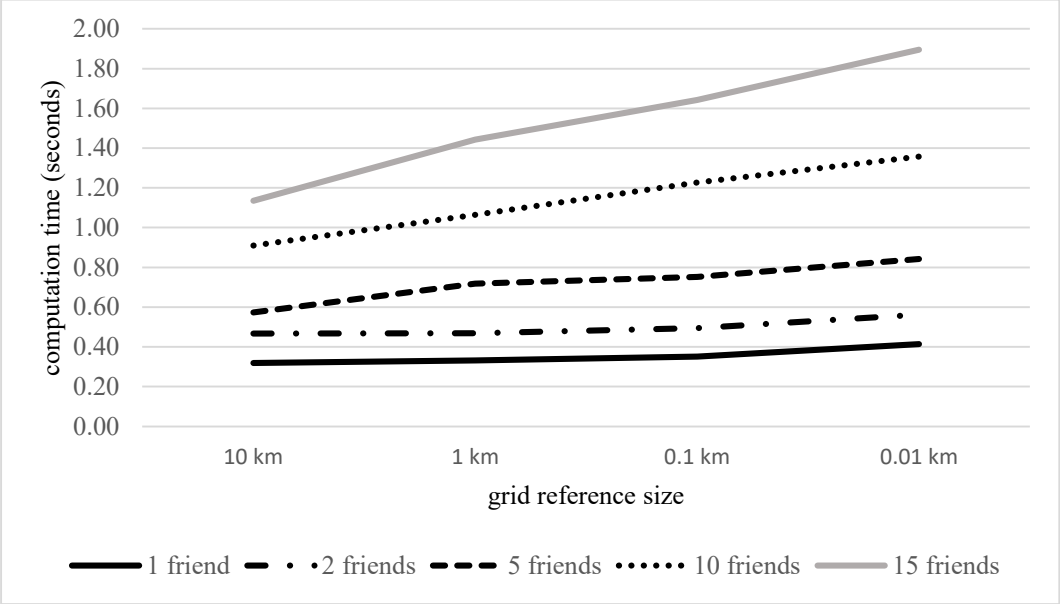


Figure 3 Encryption and decryption time for Goldwasser-Micali

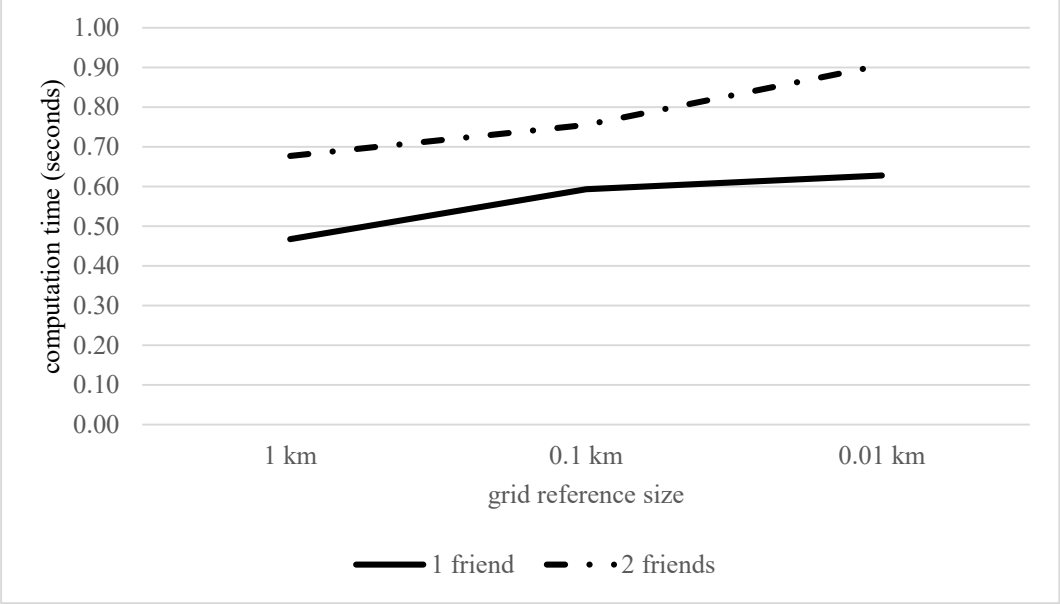
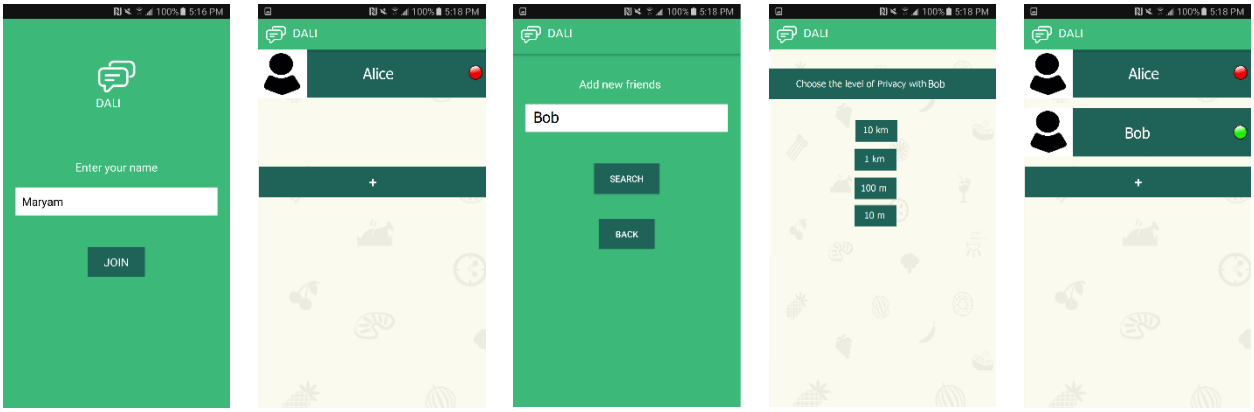


Figure 4 Computation time of the proposed protocol

Figure 5 shows the implementation of our Android application. The user should enter her name in order to login to the application (Figure 5a). When the user clicks on "join", the first screen shows the list of the friends which were added in the past. The first time that the user logs in, this list is empty. The user can click on "+" to add new friends. A new screen pops up and the user can search the name of the new friend. In the next step, the user should select the level of privacy with this new friend and the friends list will be updated. The application is running in the background, and whenever it finds a nearby friend, a pop-up notification is displayed on Alice's phone and Alice can see a green light beside the nearby friend on the friends list (Figure 5e).



a) Sign in b) Friends list c) Add new friend d) Choose the level of privacy e) Updated friends list

Figure 5 DALI application

2.7 Limitations of our Proposed Protocol

The main limitation is if one of the users uses a fake location, instead of his real location, there is no way to detect it in our proposed protocol. There are some techniques to stop the user from spoofing his location (see Chapter 5). However, none of these techniques can completely

prevent spoofing. We also require an SE on Alice's phone to do the cryptographic computation if Alice is malicious.

Our proposed protocol is asymmetric equality testing because at the end of the protocol Alice who asked for her nearby friend learns the answer that Bob is near her or not, but Bob does not. If Bob trusts Alice, she can forward the result to Bob, otherwise Bob should repeat the protocol to learn whether Alice is nearby him or not.

As we mentioned in section 2.3.3 our protocol assumes a grid system like MGRS where pairs of friends agree on the grid reference size and they consider that they are nearby if they are located in the same grid square. However, if they are located in two different adjacent MGRS squares, our protocol is not able to determine that they are close even though they may be located on the border of these MGRS squares. Moreover, as we mentioned in section 2.3.5, the Enrollment phase should be repeated every time, if the user decides to change the level of her privacy.

2.8 Summary

In this chapter, we proposed a protocol for a nearby friend privacy preserving LBS. Our proposed protocol is based on the Goldwasser-Micali probabilistic encryption scheme. The homomorphic property of the Goldwasser-Micali cryptosystem makes the corresponding plaintext equal to the XOR of the two grid references. If two grid references are equal, the result is "zero". However, the Goldwasser-Micali cryptosystem is vulnerable to CCA2 attacks. Our proposed protocol uses authenticated encryption, Encrypt-then-MAC, to tackle this problem. We analyzed the security of the proposed protocol, and we showed that an active adversary, who has

access to the ciphertext on the communication channel and the decryption oracle, cannot forge another ciphertext that allows him to guess the user's location (IND-CCA2). Moreover, the active adversary cannot modify the ciphertext in a way that allows him to affect the outcome of the protocol (NM-CCA2). We also showed the performance and practicality of our protocol in real world applications.

Chapter 3

An Efficient Solution to the Socialist Millionaires' Problem

In cryptography, the socialist millionaires' problem [60] is to determine whether the wealth of two millionaires is equal, in a way that no information about each party's riches is leaked to the other party. This concept is mostly used in cryptographic protocols to verify the identity of two remote parties, in order to find out if both of them have the same secret information. This secret information could be a password or a passphrase that is shared between them outside the communication channel [60]. The socialist millionaires' problem is a variant of the Millionaires' Problem [104][105], in which two millionaires want to know who has the most money, without disclosing the actual amounts they own. For example, in cryptographic protocols, Alice has the secret value x and Bob has the secret value y. They want to know whether x or y is greater than the other; however, they do not want to disclose any information about x and y to each other. Note that most of the existing solutions are presented to make the Millionaires' Problem more efficient [29][70].

In the socialist millionaires' problem, a passive attacker can eavesdrop on the communication channel between Alice and Bob, and he has access to the ciphertext. However, it should be infeasible for him to figure out any information about x, y and even whether x=y (IND-CPA) [11]. Moreover, an active attacker who can actively send a message or interfere with the message on a communication channel, should not learn more than a passive attacker (IND-CCA2). He should also not be able to modify the ciphertext in such a way that it leads to a desired modification of the plaintext (NM-CCA2). Bellare et al. [11] presented relations among notions of

security for popular public-key encryption schemes, such as IND-CCA, NM-CCA, IND-CCA2 and NM-CCA2.

Note that even "*perfect secrecy*" [104], which is a very strong notion of security that does not rely on any computational assumptions, is not sufficient to imply non-malleability and therefore achieve NM-CCA2. Semantic security, defined by Goldwasser and Micali [45][46], is the computational analogue to Shannon's concept of perfect secrecy. Semantic security means that no partial information about the plaintext can be efficiently learned from the ciphertext. However, this definition of secrecy is only valid if the adversary is passive. As a matter of fact, semantic security offers no guarantee of secrecy at all if the adversary is active. Mao [73] showed the vulnerability of the GM cryptosystem to a chosen ciphertext attack.

The first solution to the socialist millionaires' problem [60] is based on complexity of polynomial operations but is not fair. Later, Boudot et al. [13] presented a fair and more efficient protocol for the socialist millionaires' problem that has the same properties as the Jakobsson et al. [60] protocol, and it requires $O(k)$ exponentiations, where k is the security parameter [13][17]. Later, Garay et al. [17] showed that none of the above protocols remains secure when concurrently composed. They presented [17] a very simple protocol that remains secure when arbitrarily composed with any protocol. However, their proposed protocol still needs $O(k)$ exponentiations.

Secure computation of the socialist millionaires' problem requires convincing two parties that the result is correct and neither party learns about the secret input of the other party. Moreover, the computation needs to be fair, which means that one of the parties cannot stop the protocol from sending the result to the other one. The socialist millionaires' problem also

assumes that both parties are honest about their input and they do not cheat to send a false input X' instead of X.

The focus of this section is to solve the socialist millionaires' problem efficiently by using techniques different from the ones presented previously. In this solution, we take advantage of the XOR-homomorphic property of the GM scheme. However, the GM cryptosystem is not secure against an Adaptive chosen-ciphertext-attack (IND-CCA2) and homomorphic encryption schemes are malleable by design [6][115]. Therefore, we present a solution to these problems. The proposed protocol can be used in cryptographic problems that are solvable with the XOR homomorphic property of the GM. In order to achieve IND-CCA2, we present a scheme that uses the authenticated encryption scheme called Encrypt-then-MAC.

3.1 Problem Statement

Two parties, Alice and Bob, have secret values X and Y, respectively. They want to know whether X=Y without disclosing information about X and Y to each other, and if X≠Y they learn nothing. This problem is known as socialist millionaires' problem [60]. In the following section we describe the socialist millionaires' problem and previous work

3.2 Proposed Protocol

Our proposed protocol is secure against active and passive attacks. We analyze the security of the protocol, and we show that an active adversary who has access to the ciphertext on the communication channel and the decryption oracle cannot forge another ciphertext that leads him to guess the plaintext (IND-CCA2 security). Moreover, the active adversary cannot modify the

ciphertext in such a way that it leads to a desired modification of the plaintext that will affect the outcome of the protocol (NM-CCA2 security).

The key generation algorithm generates a public and private key pair, (PU, PR), based on the Goldwasser-Micali cryptosystem and (PU_{RSA}, PR_{RSA}), based on the RSA cryptosystem for Alice and Bob on the secure element (SE). The public keys are distributed between interested parties (here Alice and Bob). Suppose Alice wants to find out whether X=Y.

1. Alice first calculates the encryption of her secret value, X, based on GM, $M_A = \varepsilon_{GM}\ (PU_A, X)$.

   Then, she applies Encrypt-then-MAC which is a composition of Encryption = (K_e, ε, D) where K_e is a key, ε is encryption algorithm and D is decryption algorithm for a symmetric encryption scheme, and MAC = $(K_m, T, \mathcal{V})$ where K_m is a key, T is tagging algorithm and $\mathcal{V}$ is tag-verifying algorithm in message authentication scheme MAC. She sends "return" to Bob:

$$K_{AB-e} \leftarrow \varepsilon_{RSA}\ (PU_{RSA-B}, K_e)$$
$$K_{AB-m} \leftarrow \varepsilon_{RSA}\ (PU_{RSA-B}, K_m)$$
$$C_A' \leftarrow \varepsilon(K_e, M_A)$$
$$\tau_A' \leftarrow T(K_m, C_A')$$
$$C_A \leftarrow C_A' || \tau_A'$$
$$return\ C_A and\ K_{AB-e}\ and\ K_{AB-m}$$

2. Bob calculates MAC to make sure the received message is from Alice, as follows:

$$Parse\ C_A\ as\ C_A' || \tau_A'$$
$$K_e \leftarrow D_{RSA}\ (PR_{RSA-B}, K_{AB-e})$$
$$K_m \leftarrow D_{RSA}\ (PR_{RSA-B}, K_{AB-m})$$
$$M_A \leftarrow D(K_e, C_A')$$
$$v \leftarrow \mathcal{V}(K_m, C_A', \tau_A',)$$

$$if\ v = 1\ then\ return\ M_A\ else\ return\ \perp$$

Bob calculates $M_B = \varepsilon_{GM}\ (PU_A, Y)$ and $M_{AB} = \varepsilon_{GM}\ (PU_A, Y) * \varepsilon_{GM}\ (PU_A, X)$, and he sends "return" to Alice.

$$C'_{AB} \leftarrow \varepsilon\ (K_e, M_{AB})$$
$$\tau'_{AB} \leftarrow T(K_m, C'_{AB})$$
$$C_{AB} \leftarrow C'_{AB} || \tau'_{AB}$$
$$return\ C_{AB}$$

Note that $M_{AB} = \varepsilon_{GM}\ (PU_A, Y \oplus X)$ by the properties of homomorphic encryption.

3. Alice calculates MAC to make sure the received message is from Bob, as follows:

$$Parse\ C_{AB}\ as\ C'_{AB} || \tau'_{AB}$$
$$M_{AB} \leftarrow D(K_e, C'_{AB})$$
$$v \leftarrow \mathcal{V}(K_m, C'_{AB}, \tau'_{AB}\ )$$
$$if\ v = 1\ then\ return\ M_{AB}\ else\ return\ \perp$$

Finally, to prevent Alice from learning Bob's secret value, our protocol on Alice's side sets E=0, then ORs it with the decryption of the received $M_{AB} = \varepsilon_{GM}\ (PU_A, Y \oplus X)$ bit by bit. Whenever the first "1" is detected, the decryption process stops. Otherwise, the result is "0" and Alice learns that X=Y.

This solution to the socialist millionaire's problem is asymmetric equality testing, because Alice learns the answer, but Bob does not. If Alice and Bob trust each other, Alice can tell the result to Bob (several of the existing solutions consider that Alice and Bob trust each other). Otherwise, Bob should repeat the above processes with his own public key to learn whether X=Y. The protocol is based on the homomorphic feature of GM, in which the product of the encryptions of two bits is equal to the encryption of their XOR; therefore, if those two bits are the same, the result of XOR will be "zero".

Clearly if Alice and Bob have the same secret value, Alice learns Bob's secret value. Otherwise, we have to show Alice does not learn Bob's secret value. Because of the stipulation that E=0, and then is ORed with the decryption of E (PU_A, X $\oplus$ Y) bit by bit and stops whenever the first "1" is detected, Alice does not learn Y. However, this stipulation does not prevent a malicious application. This can be addressed by putting the computation and Alice's private key in the secure element (SE) and using attestations. If the risk is considered small, the application can be run by itself; if the risk seems too great, SE is used instead.

3.3 Summary

We presented a protocol to solve the socialist Millionaires' problem. The proposed protocol is based on the homomorphic feature of the Goldwasser-Micali cryptosystem. Homomorphic encryption schemes are malleable by design. Therefore, we applied Encrypt-then-MAC to make our proposed protocol secure against passive and active attacks. We analyzed the security of the protocol (see Chapter 2), and we showed that an active adversary, who has access to the ciphertext on the communication channel and the decryption oracle, cannot forge another ciphertext which leads him to guess the plaintext (IND-CCA2 security). Moreover, the active adversary cannot modify the ciphertext to produce a desired modification of the plaintext which will affect the outcome of the protocol (NM-CCA2 security).

Chapter 4

A Practical PIR-based Scheme for Discovering Nearby Places for Smartphone Applications

Our focus in this chapter is to help the user search on her smartphone for particular places of interest (POIs) while keeping her location private from the location-based service provider (LBSP). For example, the user sends a request to the application to find a nearby restaurant, gas station, or ATM. This location could be the exact location of the user or a location to which she wants to travel in the future. These types of applications are especially useful for people who travel or have moved to a new city. A wide range of LBS nearby applications have been released recently, such as Facebook Nearby Places, AroundMe, NearBy Places, Yelp, Foursquare and Places to help users to identify their nearby locations quickly.

Modern multi-core smartphones have high-performing processors, which are well suited for cryptographic operations that can allow location privacy to the LBS applications. Unfortunately, the processing units often consume a considerable amount of energy, which in turn shortens the battery life of mobile devices. In addition, these devices have limited memory and bandwidth [64]. Therefore, downloading the entire database for finding POIs around the U.S. and Canada, which can contain ten million entries with respect to a typical commercial POI database (see Appendix A) [116] and require 3 to 4 GB of data storage, is obviously not practical. Furthermore, updating results periodically to make sure they are accurate enough, bandwidth limitation and data usage limitation of smartphones are other important factors that we should keep in mind when we are offering a cryptographic solution for LBS applications with respect to the privacy of the mobile user's location.

4.1 Our Contributions and Assumptions

Our privacy-preserving protocol for LBS helps Alice to search for specific nearby POIs on her smartphone by sending a query to the LBS service provider (LBSP) over the wireless network. In this scenario, the proposed protocol allows Alice to learn whether any place that she is looking for is near her. However, the location-based service (LBS) that tries to help Alice to find nearby places does not learn Alice's location. Alice can send a request to the LBS database to retrieve nearby places of interest (POIs) on her smartphone without the database becoming aware of what Alice fetched by using our practical PIR scheme. The LBS server retrieves the query from the POI database, and returns the list of results to the user containing the specific requested POIs type found in the specified area. In order to achieve this, our protocol must fulfill all of the following requirements [81]:

1. The LBS server must not learn the user's exact location. It may only identify a general region that is large enough, in terms of area and the number of POIs it contains, to confer a sufficient level of privacy to the user's satisfaction.
2. There must be no third parties, trusted or otherwise, in the protocol between the user and the server.
3. The implementation must be computationally efficient on resource-constrained hardware such as a smartphone. A user may be expected to tolerate a delay of no more than several seconds for any kind of query.
4. The approach cannot rely on a special secure processor (i.e., one that is not typically found on a commercial smartphone).

Our cryptographic approach is based on private information retrieval (PIR) for secure LBS applications that identify nearby places. PIR allows the user to fetch her required data from the database without revealing which data are fetched [19]. The POI database is labeled by the location of POIs; therefore, the LBS server is able to retrieve the POIs based on the user's location of interest in the requested query. PIR solves most of the previous problems associated with non-cryptographic approaches in LBS. PIR approaches do not have the privacy vulnerabilities of k-anonymity or cloaking, such as single point of attack of their anonymizer or server, which tries to help them to apply k-anonymity or generate an obfuscation area. Therefore, the user location information remains private from the service provider by using PIR approaches.

Different types of PIR-based approaches have been proposed during the last two decades. The common criticism of PIR approaches is that the computational overhead is not appropriate for smartphones with limited processing power, memory, and wireless bandwidth [88], and therefore, they are not practical [68]. We ensure that the proposed cryptographic PIR approach is practical for smartphone applications. Based on Devet et al. [25], there are five factors that affect the speed of the PIR query:

1. the time for the client to generate a private query;
2. the communication time required to send the query to the server(s);
3. the time for the server(s) to apply the query to the database;
4. the communication time required for the response from the server to the client;
5. the time for the client to decode the response(s) and retrieve the results of decoding.

Our approach expands the Olumofin et al. [81] idea of applying a cloaking region to reduce these five factors. Moreover, our approach reduces the time for the client to decode the response(s) and retrieve the results of decoding on the smartphone by approximately 50%

compared to Olumofin et al. [81] by applying the POI types idea to block-based PIR. Reducing the decode time is valuable in our application to satisfy the fifth requirement, so that it can be used on modern smartphones' hardware. The server(s) processing cost is similar to Olumofin et al. [81], in order to preserve the privacy of the user's location. Our proposed protocol is able to support all types of block-based PIR schemes.

In our proposed approach, the identity of the user is not hidden from the LBS, in order to return the results to the user. However, if the user wants to keep her identity hidden from LBS, she can use an onion routing technique, such as Tor [27]. Note that keeping the user's location private has priority in an LBS application over keeping the user's identity hidden from the LBS, because if the LBS knows the user's location, it is easy to identify the user. We should mention that a mobile communications operator is always aware of the user's location based on the cell tower. Therefore, we assume that this operator does not collude with the LBS service provider.

The remainder of this chapter has the following structure: Section 4.2 presents a brief overview of previous work regarding PIR and LBS approaches. Section 4.3 describes the details of our PIR scheme. Section 4.4 explains the details of our threat model and privacy-preserving protocol for LBS. The security analysis of our proposed protocol is discussed in Section 4.5. Section 4.6 gives an overview of our implementation and compares it with previous work. The limitations of our proposed protocol are discussed in Section 4.7. Finally, Section 4.8 summarizes the chapter.

4.2 Related Work

For greater understanding, we first review the definition of PIR and give a brief overview of different types of PIR schemes. Then, we provide a review of PIR-based approaches for the users' location privacy in LBS applications.

4.2.1 Review of Private Information Retrieval (PIR)

Nowadays, users are more aware of the privacy requirements of their information in their online activities. But is it actually possible to keep the user's query contents private while she issues a request to online applications? This appears to be an easy problem to solve. The user can connect to the application over the Tor network and send her request via Tor [27]. In this scenario, the server has no idea who searched for the data, but the server has access to the content of query in order to fetch the requested data from its database. Therefore, Tor is not a good option to solve our problem. The problem that we want to solve is to allow a user to query the database without sharing what the user looked for. Here, we are not trying to protect the identity of the user, but rather the content of the query. Private information retrieval (PIR) is a cryptographic database technique that solves the problem of allowing the user to query a database while the content of the user's query is hidden from the database. The need for PIR has been demonstrated in real applications, such as location-based services, online research, social networks, etc. [81].

The PIR problem was first introduced by Chor et al. [19], and it attracted the attention of cryptography and privacy research groups. The trivial PIR scheme is to transfer the entire database to the user and ask her to retrieve what she is seeking locally. This gives the user perfect privacy because the query does not contain any information about what the user is

looking for, but it creates a large communication overhead. Three important requirements for PIR are correctness, privacy, and non-triviality [43]. Correctness requires that the received data from the database should satisfy the user's query. To have privacy, the database should not be able to learn the user's input or the block of the database that she retrieved. Triviality is defined as the communication cost between the client and server, which is $o(n)$, where n is the number of bits in the database. We want the cost of the PIR approach to be non-trivial; that is, to be much less than the cost of downloading the whole database. Another requirement for PIR is implementation efficiency, which is not considered in most of the published literature. Most researchers attempt to reduce the communication complexity rather than the computational complexity [1][95]. This lack of attention to the computational overhead has resulted in PIR schemes that are impractical for resource-constrained hardware, such as smartphones.

The first non-trivial PIR scheme was defined by Chor et al. [19]. Gasarch [33] presents it informally as follows:

Definition 2.1. [19][33] A 1-round k-DB Information Retrieval Scheme with x$\in \{0,1\}^n$ and k databases has the following form.

1. Alice wants to know x_i. There are k copies of the database which all have $x = x_1 \dots x_n$. The DBs do not communicate with each other.
2. Alice flips coins and, based on the coin flips and i, computes (query) strings $q_1 \dots q_k$. Alice sends q_j to database DB_j.
3. For all j, $1 \leq j \leq k$, DB_j sends back a (answer) string $ANS_j(q_j)$.
4. Using the value of i, the coin flips, and the $ANS_j(q_j)$, Alice computes x_i.

The complexity of the above PIR scheme is $\sum_{j=1}^{k} |q_j| + |ANS_j(q_j)|$.

Computational PIR (CPIR): One class of PIR protocols assumes that the server(s) and the adversary have limited computational capability to guarantee the user's privacy. In order to break the security of CPIR protocols, the adversary has to solve a problem that is believed to be hard. This kind of assumption is usual for cryptography and privacy schemes.

In 1995, Chor et al. [19] showed that single-database PIR does not exist in the information-theoretic security sense. In 1997, Chor et al. [20][21] proposed the first CPIR to prove that the communication complexity of PIR can be reduced if one is willing to achieve computational privacy, rather than information-theoretic privacy. In the same year, Kushilevitz and Ostrovsky [66] presented a CPIR protocol that has the same assumption for the computational capability for the adversary, but it uses a single server. Their protocol was the first single-server CPIR. It is based on the Quadratic Residuosity problem, which is considered difficult to solve. The main advantage of single-server CPIR protocols is that the communication cost of PIR can be improved by using the CPIR recursively. Later, different types of single-server CPIR were proposed, which tried to reduce the communication complexity of PIR, such as φ-hiding problem [15][35], the existence of one-way trapdoor permutations [65], the Pailler homomorphic encryption [69], and the Hidden Lattice problem [1].

Sion and Carbunar [86] showed that CPIR schemes were not practical, given certain realistic assumptions at the time. However, in 2016, Aguilar-Melchor et al. [2] introduced XPIR. They showed that by using lattice-based cryptography, CPIR is of practical value and the conclusion of Sion and Carbunar is not valid anymore.

Information Theoretic PIR (IT-PIR): Information-theoretic privacy means the privacy of the user cannot be compromised even if an adversary has access to unlimited computational capability. In 1995, Chor et al. [19] showed that any single-server IT-PIR scheme must have

communication cost at least that of the trivial protocol. Therefore, IT-PIR protocols assume that if you have $k \geq 2$ non-cooperating servers, each having a copy of the database, then there are PIR schemes that achieve complete information theoretic security. Later, different types of IT-PIR were proposed [12][36][43][81] that tried to improve Chor et al.'s protocols [19].

By using multiple databases, we add robustness to the PIR, but this can affect privacy in the case of malicious servers or non-responsive servers [12][43]. To tackle this problem, Goldberg [43] defined the privacy threshold that must be lower than the total number of the servers. Therefore, we need extra responding servers in order to set a privacy threshold.

Trusted Hardware PIR: In 2006, Wang et al. [99] proposed a new type of PIR called the trusted hardware-based PIR. The trusted hardware-based PIR uses the idea of a tamper-resistant CPU, which is trusted by the user and connected to the server. The user sends her query to this CPU, where her query is unreadable by the server. The CPU extracts the requested information from the server's database and returns the results to the user. These types of PIR achieve the lower bound for both communication and computation costs. However, the proposed architecture is secure only if the user can trust the hardware.

Hybrid PIR: In 2014, Devet et al. [25] proposed a hybrid PIR that was a combination of CPIR and IT-PIR to reduce communication costs. Their goal was to combine the positive features of CPIR and IT-PIR to reduce the negative features of each. They considered the low communication and computation cost of multiple-server IT-PIR schemes and the recursion of single-server CPIR schemes to improve the communication cost of PIR queries.

4.2.2 Review of the PIR-based Scheme for Nearby Places

One of the motivations for developing useful and practical PIR schemes is to protect the users' private information while they are using mobile devices with positioning capabilities. In a stationary desktop scenario, when a user tries to query the database or the remote server, the primary concern is leaking information about the query's content. However, in an LBS scenario, when a user sends a query to the LBS server, the user's location is also released to the LBS server. The problem with location privacy is preserving the privacy of the user's actual location when she is using the LBS while providing the most precise and acceptable response.

Many of the previous problems of privacy preserving protocols for LBS that we encountered were solved by introducing PIR-based LBS protocols. The idea is to use the PIR scheme to let the user send a query to the LBS server without disclosing her actual location. This query typically consists of POIs, which includes a description of the POI and its geographic location.

Most of the previous works that attempted to apply PIR to location-based services were based on secure hardware, with a secure coprocessor at the LBS server [31][52][63][86]. The idea of using the secure hardware-based PIR in LBS was first proposed by Hengartner [52]. This hardware performs the trusted computing in order to hide the user's location from the LBSP. Recent work regarding secure hardware PIR was proposed by Fung et al. [31]. Their PIR technique was similar to Papadopoulos et al. [86], however it offered better efficiency, and it was more practical for large datasets. Their framework applied the concepts of ϵ-differential privacy and private information retrieval (PIR) to use the query statistics to increase efficiency without compromising privacy. All proposed solutions for secure-hardware PIR claim that the secure hardware method is the only practical PIR mechanism [31][86]. The major drawback of all secure

hardware PIR schemes is that the proposed architectures are secure only if the user can trust the hardware.

The common criticism of other PIR approaches for location privacy is that the computational overhead is not suitable and practical for resource-constrained hardware such as a smartphone [68][88]. In 2007, Ghinita et al. [42] proposed the first PIR-based approach for location privacy, without using a third party. Their proposed protocol used the idea of the trade-off between privacy and efficiency defined by Ghinita [41]. The approach of Ghinita et al. [42] used a single PIR request for each query. In their approach, all queries were indistinguishable, and it was able to achieve strong location privacy. Their proposed protocol included two stages to protect information about the user's location and her query. In the first stage, the user and the server engaged in a protocol, which is based on Paillier encryption [84], to determine the index of the user's location cell, without releasing the location to the server. In the second stage, the user uses PIR to retrieve the query results for the target cell. The advantages of the Ghinita protocol are the nondisclosure of location information and its security against correlation attacks for both stationary and mobile users.

Khoshgazaran et al. [63] described three drawbacks to Ghinita et al.'s [42] protocol. First, it focuses on the nearest neighbor queries. Second, it scans the entire database linearly for each query. Third, it has a high communication complexity. Additionally, the protocol is secure if the privacy of the user is a concern and the LBS is not able to learn the user's query, but it is not symmetric for LBS's database privacy since the user can infer the data that are in the same column as her query.

Later, Olumofin et al. [81] proposed a hybrid solution combining PIR and cloaking to protect the user's privacy without using trusted computing. Their idea of using cloaking reduces the

computational cost of PIR and makes it more practical. The user's location privacy depends on the size of the cloaking region. Their PIR approach is flexible and supports all types of block-based PIR schemes. Our proposed PIR protocol expands Olumofin et al. [81] idea. However, we focus on reducing the computational cost on the client side. We explain the details of our protocol in the following section.

4.3 A PIR Scheme to Improve the Computation Cost on the Client-Side of Smartphone Applications

The focus of this section is to propose a block-based PIR scheme for smartphone applications. In this solution, we take advantage of the partial query in order to reduce the computation and communication costs [19]. In addition, we categorize the database based on types of data, in order to minimize the computational cost on the client-side, which is considered to be a smartphone. The user retrieves the exact category of the data that she is looking for and she does not need to filter the received data to extract the subset that matches her request. This reduces the decode computation cost on the smartphone with limited resources. The proposed protocol is suitable for all applications that need to protect users' privacy while they are searching for data in a database (it is not restricted to just LBS applications).

4.3.1 Preliminaries

We explain our PIR scheme based on the 2007 Information-Theoretic PIR (IT-PIR) scheme by Goldberg [43]. The main advantage of the multiple-server IT-PIR solution is that it reduces the computation and computation costs in general. Goldberg's protocol uses Shamir's secret sharing [91] to split the user's query into l shares and transfers l shares to multiple servers. The

protocol is also robust for byzantine servers that may not respond to the query or may respond incorrectly. Note that our proposed protocol is flexible to support all types of block-based PIR schemes.

To reduce the computation cost on the client side, we use the idea of trading off privacy for better performance [81]. In this idea, the level of privacy refers to the number of data items in the database that the PIR server should process in order to respond to the client. Our proposed protocol has three improvements in comparison to Olumofin et al. [81].

First, our approach divides the database into classes. Each class is categorized based on the sub-types of data that the user is interested in. In this way, when the user asks for the specific data of her interest, the application can convert it to the proper portion of the database. Therefore, the result that comes back to the user is exactly the type she was looking for. In this way, the client side does not need to filter the received response to find data that the user was looking for. This reduces the computation cost on the client-side for decoding the received response(s).

Second, if the amount of data is higher for a sub-type, then a higher data traffic cost will result and the query will be much slower. If on the other hand, the amount becomes significantly lower in another sub-type, then the result size may be so minimal that the server may guess the user's sub-type of interest with a high degree of confidence and leading to loss of privacy. In order to tackle this problem, we equalize the amount of data in each row of the sub-type in a specific class by adding "null" to all the sub-types that are not equal to the maximum sub-type size in that class. The main advantage of our "null" solution is that if the user searches for a sub-type with less data, the PIR computation cost on the client-side is reduced (when receiving the first null in the decoding process, the computation process stops), and if the user searches for a sub-type with more data, the PIR computation cost on the client-side increases, without losing privacy.

Finally, as shown in Figure 6, the amount of data is different in different classes, unlike the proposed approach in Olumofin et al. [81]. Their approach has no concept of class and all rows in their defined database have an equal amount of data. Therefore, our PIR computation cost depends on the specific class that the user requests. If the user searches for a class with less data, the PIR computation cost is reduced. If the user searches for a class with more data, the PIR computation cost increases.

By considering these three improvements, if the user sends a query to the database for the sub-type of data in each class, the response that is returned to the user not only needs less computation time for decoding, but also does not need to be filtered on the client side to remove non-requested data.

4.3.2 Proposed PIR Scheme

Our protocol contains two phases. The first phase is the pre-processing phase in which the whole protocol becomes ready to use, on the client side and also on the server side. This phase can be repeated in the future if any changes occur on the server side, or the client decides to change the level of her privacy. The second phase is the execution phase in that the user sends her request to the server. Her request contains the class of data which she is looking for concatenated with the sub-type category.

Pre-processing Phase contains the following steps:

1. Given a chosen level of the user's privacy, "Class", "sub-type" and "data" category are applied to the database.
2. The "class" and "sub-types" are defined to have a number based on their specific categories. As shown in Figure 6, for example, Class-0 is considered 00 and Sub-type-4 is considered 0100. Note that in this Figure we just showed "10" different "sub-types"

for each “class”. This depends on the number of different sub-types of data in the database and also on the level of the user’s privacy which is applied in step 1.

3. Each database index will be the “class” concatenated with the “sub-type”. For example, in Figure 6, the database index for the Class-0||sub-type-4 is considered as 000100.

Execution Phase contains the following steps:

1. The user chooses the class of interest.
2. The user chooses the sub-type of her interest from the suggested list by the client application. For example, she is looking for Class-0||sub-type-4.
3. The client application sends the class to the server. Also, the client application identifies which portion of the class contains the sub-type, in a way that is hidden from the server. In this example, the request is 000100 which refers to Class-0||sub-type-4.
4. The server receives the request, and finds the database portion corresponding to the class. A block of rows is retrieved from this portion based on the user’s sub-type by using PIR. The sub-types present in these rows are transmitted back to the client application.
5. The client application decodes the results and the results are shown on the user’s smartphone.

Class	Category Types					
Type-0 00	Sub-type 0	0000	data1	data2		
	Sub-type 1	0001	data 1	null		
	Sub-type 2	0010	data1	data2		
	Sub-type 3	0011	null	null		
	Sub-type 4	0100	null	null		
	Sub-type 5	0101	data1	data2		
	Sub-type 6	0110	data1	data2		
	Sub-type 7	0111	null	null		
	Sub-type 8	1000	null	null		
	Sub-type 9	1001	data1	null		
Type-1 01	Sub-type 0	0000	null			
	Sub-type 1	0001	null			
	Sub-type 2	0010	null			
	Sub-type 3	0011	null			
	Sub-type 4	0100	null			
	Sub-type 5	0101	null			
	Sub-type 6	0110	null			
	Sub-type 7	0111	null			
	Sub-type 8	1000	null			
	Sub-type 9	1001	null			
type-2 10	Sub-type 0	0000	null	null	null	null	null
	Sub-type 1	0001	data1	data2	data3	data4	data5
	Sub-type 2	0010	null	null	null	null	null
	Sub-type 3	0011	data1	data2	null	null	null
	Sub-type 4	0100	data1	Null	null	null	null
	Sub-type 5	0101	data1	data2	data3	data4	data5
	Sub-type 6	0110	data1	null	null	null	null
	Sub-type 7	0111	data1	data2	data3	data4	null
	Sub-type 8	1000	null	null	null	null	null
	Sub-type 9	1001	null	null	null	null	null
…	…						

Figure 6 Illustration of the relationship between Class types, Sub-types, and data, as stored in database rows.

4.4 The Proposed Privacy-Preserving Protocol for LBS

The main aim of cryptographic approaches in nearby places is to make it possible to automatically detect nearby places even when the user's location privacy is applied to the application. Our proposed protocol uses private information retrieval (PIR) to achieve this purpose.

4.4.1 Problem Statement

Alice has her location as her secret. Alice wants to use a location-based service (LBS) application to search and find nearby places of interest. We propose a protocol that allows Alice to find nearby places for which she is looking. However, the LBS that helps Alice to find her nearby place does not learn Alice's location. Alice can send a request to the LBS's database to

fetch her nearby places of interest without the LBS being aware of what Alice fetched by using private information retrieval (PIR). Most of the existing PIR schemes are not acceptable in LBS applications because of their use of secure hardware. The focus of this section is to solve the PIR-based LBS issues by offering a practical PIR without using secure hardware or a trusted third party and reduce the computational cost on the client side application on the smartphone. At the end of this protocol, the proposed application should list the POIs that meet Alice's search criteria or show her that there is no POI in the selected area.

4.4.2 Threat Model

Our proposed PIR protocol applies to all existing block-based PIR schemes (computational PIR and information-theoretic PIR). In this section, we use the information-theoretic PIR (multi-server) to explain our threat model (as was done by Olumofin et al. [81]). The primary assumption in IT-PIR schemes is that servers should not collude to break users' query privacy. Under this assumption, the IT-PIR protocol itself has been proven secure [12][19][43]. Given the cryptographic security of the IT-PIR scheme, we analyze the security of our proposed protocol, and we demonstrate that it is secure against passive and active attacks.

4.4.3 Review of the Olumofin, et al., Protocol

Our protocol follows Olumofin et al.'s [81] hybrid solution that uses PIR to protect the privacy of the user's query and a cloaking scheme to reduce the computational cost of PIR to a practical level. The benefits of the hybrid solution are as follows: the user's location is kept secret from the LBSP to an acceptable privacy level chosen by the user without depending on the other users in the selected area; there is no need to have a trusted third party to calculate the cloaking region or cryptographic algorithms; and the computational cost of the PIR algorithms is practical.

The approach that Olumofin et al. [81] proposed entails two phases. First, there is a pre-processing phase in which the system is set up for use. Second, there is an execution phase, in which the LBS server responds to queries for POIs from users.

The pre-processing phase consists of the following steps:

1. An appropriate static grid system is applied on the geolocation plane.
2. A collection of POIs is saved in a database.
3. Each cell of the grid is mapped to a portion of the database, i.e., a particular set of database rows (each containing a POI).

The execution phase consists of the following steps:

1. The user determines the area of interest.
2. The user chooses a desired level of privacy.
3. The client creates a cloaking region corresponding to this level of privacy, which will enclose the area of interest.
4. The client sends the cloaking region to the server. In addition, the client identifies which portion of the cloaking region contains the area of interest, in a way that is hidden from the server.
5. The server receives the request and finds the database portion corresponding to the cloaking region. A block of rows is retrieved from this portion based on the user's specified location of interest. The POIs present in these rows are transmitted back to the client.
6. The client decodes the result and automatically finds the nearest neighbour POI, or presents the full list of returned POIs to the user (for the user to choose amongst).

4.4.4 Proposed Privacy-Preserving Protocol for LBS

We first informally describe our proposed protocol via an example. Suppose Alice is located in Ottawa and she wants to look for a specific type of POI, for example a restaurant, near Bank Street. Since she is privacy conscious, she sets her cloaking region to be a 10 km MGRS grid square (see section 4.4.3). The client application sends the cloaking region to the server. Also, the PIR allows the client application to identify which portion of the cloaking region contains the restaurant, in a way that is hidden from the server. The entries in the POI database are indexed by their MGRS block concatenated with the POI type. The row that contains the restaurant(s) is retrieved from the selected MGRS grid square on the database and the results are transmitted back to Alice. The client application decodes the results and sorts the results, and the nearest restaurants are shown on her phone's local map.

Our proposed protocol has two improvements compared to Olumofin et al. [81]. First, due to the user's request for a specific POI type, our protocol categorizes the cloaking region in the database into the POI type. Thus, when the user asks for her POI in her selected cloaking region, the results that are returned to her are of the type that she is looking for. Therefore, our protocol on the client side does not need to filter the block of different types of POIs to identify the POI that the user requested. This reduces the computational costs, and saves the battery and data usage on the smartphone. For example, if there are no restaurants near the user, she does not need to wait to decode all POIs in that cloaking region and then filter the restaurant to find that actually the answer is "null".

Second, we propose a new technique based on a static grid-based approach for defining our cloaking region and mapping our POIs to a cloaking region. Therefore, the number of POIs is different in different MGRS blocks, unlike the Olumofin et al. [81] approach which uses the VHC

(Various-size-grid Hilbert Curve) technique [87]. Their approach has no concept of class and all rows in their defined database have an equal amount of data. Therefore, our PIR computation cost depends on the specific MGRS block that the user requests. If the user searches for an MGRS block with fewer POIs, the PIR computation cost is reduced. If the user searches for a block with more POIs, the PIR computation cost increases. Our client-side computation is therefore less on average than Olumofin et al. because their computation cost corresponds to the maximum number of POIs across all blocks.

Our proposed protocol describes how the POI database is initialized and how the protocol creates a cloaking region around the user's exact location, and performs a PIR query on the contents of the cloaking region. We name our phases similarly to Olumofin et al. [81] to highlight the similarities and differences between our phases. Note that each POI consists of 300 bytes that includes longitude and latitude coordinates, name, exact address, the phone number, website address, etc.

The pre-processing phase contains the following steps:

1. An appropriate static grid system is applied on the geolocation plane.
2. POIs are categorized based on their type and saved in a database.
3. Each row of the database refers to a cloaking region concatenated with the POI type.

The execution phase contains the following steps:

1. The user chooses the area of interest; it could be her current location as determined through GPS, or some other location that the user may be traveling to in the future.
2. The user selects a preferred level of privacy based on the grid reference size (see section 4.4.5).

3. The user's corresponding cloaking region is calculated based on the level of her chosen privacy.
4. The user chooses the POI type(s) from the suggested list provided by the client application.
5. The client application sends the cloaking region to the server. Also, the client application identifies which portion of the cloaking region contains the POI type(s), in a way that is hidden from the server.
6. The server receives the request, and finds the database portion corresponding to the cloaking region. A block of rows is retrieved from this portion based on the user's specified POI type using PIR. The POIs present in these rows are transmitted back to the client application.
7. The client application decodes and sorts the results, and the nearest POIs are shown on her phone's local map.

4.4.5 Grid-based Cloaking

In our proposed protocol, the client application extracts the user's location via GPS, Wi-Fi, or cell towers, and then it calculates the user's cloaking region based on the military grid reference system (MGRS). The MGRS is a geo-coordinate standard for locating points on the Earth [51]. The Earth is divided into grid squares with sizes of 100 km, 10 km, 1 km, 0.1 km, etc., based on the level of accuracy and degree of precision. Our proposed protocol uses MGRS to help ensure the user's location privacy. Each MGRS block is considered as a block in the database that is categorized based on POI type.

POIs are considered to be nearby if they are located in the same MGRS block as the user. Note that our protocol does not recognize that POIs are nearby if they are near the user but in a

different MGRS block. If the user could not find the POI that she searched for, she could search in a larger MGRS block. The user's location privacy level increases if she chooses a larger MGRS block. However, a larger MGRS block includes more POIs, and it affects the computational cost of our proposed protocol. We will discuss this issue in more detail in the following section. Figure 2 shows the different levels of MGRS blocks for the Ottawa, Ontario area [117].

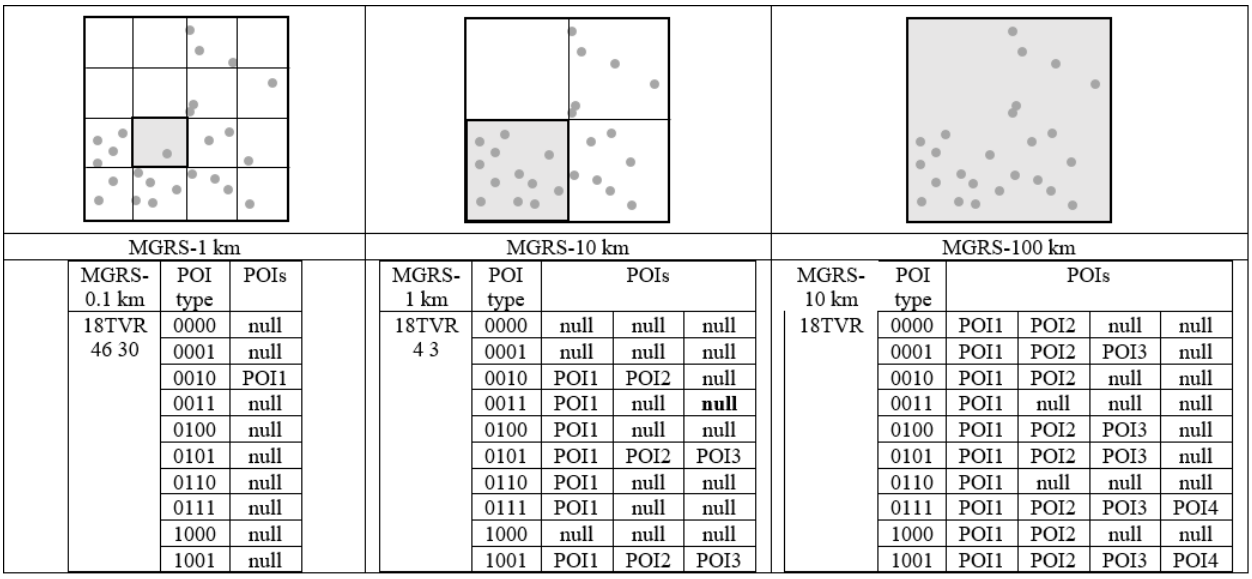
4.4.6 Pre-Processing and Location Cloaking

The first step in the pre-processing phase is to apply different MGRS levels on the geographic area, such as the United States and Canada. Then, the user's cloaking region is calculated based on the user's selected MGRS level and her current location or the location of interest. POIs are considered to be nearby if they are located in the same cloaking region.

The selected MGRS level should be large enough to achieve the user's location privacy within the requested cloaking region, but at the same time it must be small enough to reduce the computational cost on the smartphone client side application to process the results, as well as communication cost to transfer the result via the wireless data traffic.

In order to map POIs to a cloaking region, Olumofin et al. [81] used the VHC (Various-size-grid Hilbert Curve) technique. Olumofin et al. [81] chose VHC because it solves the problem of density of POIs varying by geographic area. If the selected area has a high density of POIs (such as a city), then the data traffic cost will increase. On the other hand, if the selected area has a lower density of POIs (such as countryside), then the result size decreases and the server is able to guess the user's location, which leads to a loss of privacy. VHC can solve this problem by creating a variable-sized cloaking region based on the density of POIs. However, this solution has

the disadvantage of receiving a list of POIs, which may or may not be useful for the user. If the selected area has no POI that the user is looking for, she still has to wait for the client application to calculate the result, which is based on all POIs in a selected VCH region, and then show the result, which is actually "null". This can cause a high computational cost on the client side application.



MGRS-1 km

MGRS-0.1 km	POI type	POIs
18TVR 46 30	0000	null
	0001	null
	0010	POI1
	0011	null
	0100	null
	0101	null
	0110	null
	0111	null
	1000	null
	1001	null

MGRS-10 km

MGRS-1 km	POI type	POIs		
18TVR 4 3	0000	null	null	null
	0001	null	null	null
	0010	POI1	POI2	null
	0011	POI1	null	**null**
	0100	POI1	null	null
	0101	POI1	POI2	POI3
	0110	POI1	null	null
	0111	POI1	null	null
	1000	null	null	null
	1001	POI1	POI2	POI3

MGRS-100 km

MGRS-10 km	POI type	POIs			
18TVR	0000	POI1	POI2	null	null
	0001	POI1	POI2	POI3	null
	0010	POI1	POI2	null	null
	0011	POI1	null	null	null
	0100	POI1	POI2	POI3	null
	0101	POI1	POI2	POI3	null
	0110	POI1	null	null	null
	0111	POI1	POI2	POI3	POI4
	1000	POI1	POI2	null	null
	1001	POI1	POI2	POI3	POI4

Figure 7 Illustration of the relationship between the MGRS block, POI types, and POIs as stored in the database rows.

In order to manage the computational cost based on the density of the POIs on the smartphone application, and prevent high computational cost in the lower density POIs area, we propose a new technique to map POIs to a cloaking region based on the MGRS fix-sized blocks. First of all, we categorize POIs based on their types in each MGRS block. In Figure 7 we consider ten POIs types per MGRS block (see Appendix A) [116]. Then, each row of the database refers to the MGRS block concatenated with the POI type. As Olumofin et al. mentioned [81], the density of POIs varies by geographic area. Therefore, each row of the defined database has a variable size.

To preserve the user's location privacy and prevent the server to guess which POI type is fetched by the user, we need to equalize the number of POIs in each selected cloaking region.

Therefore, if the number of POIs in one POI type is not equal to the maximum POI size in the selected cloaking region, the rest of the row must be set to "null" (note that computation stops on the client side if a "null" is encountered). By this technique, our PIR client side computational time depends on the location of the user and the level of selected MGRS. If the user's location has low POI density, the PIR client side computation time will decrease. If the user's location has high POI density, the PIR computational cost on client side will increase. It is important to note that the server cannot observe the differences between computational costs for queries in different locations, because we equalized the number of POIs in the selected cloaking region. Otherwise, the server, which is able to observe the difference between computational cost based on different user's queries, would be able to guess the user's location. Figure 7 shows an example of the POI density in the selected area based on different levels of MGRS and illustrates the relationships among the MGRS block, POI types, and POIs as stored in the database rows.

4.5 Security Analysis of the Proposed Privacy-Preserving Protocol for LBS

The user can use her current location or a location that she wishes to visit in the future. This feature adds one more level of privacy in our protocol because the observer or the location-based service provider (LBSP) is not aware of whether the requested MGRS block corresponds to the user's current location. Therefore, we have two types of privacy: first, protection of the user's location privacy within the requested MGRS block, and second, the LBSP or an observer does not know whether the request is the user's real location at the time of the request. In both cases, our goal is to keep the user's location private from the LBSP and any other observer.

4.5.1 Security Analysis

Claim 4.1. If T is the type of POI that Alice is looking for, and B is an MGRS block of level L chosen by Alice, our proposed protocol is secure against a *passive adversary* within B.

Passive adversary. A malicious LBSP or an external observer who has access to the data that passes between the user and the database on the communication channel but cannot change the data.

Justification. Our proposed protocol calculates the portion of the database in which Alice wants to find a nearby place of interest. It is based on the level of privacy she selected, such as MGRS-100 m. Therefore, Alice's query and privacy will be limited to that portion of the database. In our proposed database, the number of POIs in each type of an MGRS block is set to be equal to the maximum number of POIs in that MGRS block by adding "null" to the ends of the other types. Thus, the passive adversary cannot guess which type of POI was fetched by Alice. In other words, if Alice's type of POI changes while she is still in the same MGRS block, then for a new request, Alice will send the same query for the same MGRS block of the database.

Because Alice selected her level of privacy to be, for example, a block of MGRS-100 m, it is impossible for the passive adversary to detect her movement as long as she is in the same MGRS block. Therefore, a correlation attack is impossible because Alice will send the same request for all queries for a given privacy level. Additionally, if Alice knows that she is going to move, she should choose a larger MGRS block that includes both her current location and her movement. Thus, her movement will not be detectable.

The PIR scheme provides the security for our proposed protocol against a passive adversary. The only information that the passive adversary can access is the user's identity. However, we can

use Tor to protect the user's identity if required. Additionally, the contents of the user's query and the database response can be protected against a malicious observer by using end-to-end encryption, such as transport layer security (TLS) along the communication channel. Both schemes (Tor and TLS) are optional for the user, as they slow down the protocol and cause an additional computational cost.

Claim 4.2. If T is the type of POI that Alice is looking for, and B is an MGRS block of level L, chosen by Alice, our proposed protocol is secure against an *active adversary* within B.

Active adversary. A malicious external observer who has access to data that passes between the user and the database on the communication channel and can modify, delete or insert data. The LBSP is considered to be trusted in Claim 4.2.

Justification. Message reordering attack. In this attack, the active adversary tries to delay or reorder requests or responses to confuse the communication and results. However, in our case, if the results of multiple queries are received out-of-order, it has no effect on Alice or the server because each result holds the requested data. The adversary also gains no information about Alice's query or location.

Justification. Message tampering attack. If the adversary starts to send false responses to Alice, she will not be able to verify them. Therefore, a denial-of-service attack (DoS attack) is possible. However, in using this attack, the active adversary is not able to learn any information about Alice's query or location, which is the main focus here. Note that this attack can be prevented by using TLS over the communication channel.

Justification. Message insertion or deletion attack. If an active adversary tries to delete or insert data from the server's response or Alice's request, it can cause a DoS attack. The adversary

does not receive any information about Alice's request or location. Again, TLS can be used on the communication channel to prevent this attack.

Justification. Message replay attack. If an active adversary launches a replay attack against the server or Alice, it does not harm either of them. The server responds to the queries, and Alice can discard multiple responses with the same content. The adversary will not receive any information about Alice's query or location. Similar to the previous attacks, using TLS can help prevent this attack as well.

Claim 4.3. If T is the type of POI that Alice is looking for, and B is an MGRS block of privacy level L chosen by Alice, our proposed protocol is secure against a malicious server within B.

Malicious Server. A malicious server refers to an LBSP that tries to modify, delete or insert new messages in response to the user.

Justification. If the malicious server sends a false response, does not return a response, or sends additional messages with the response, it can cause a DoS attack against the user. However, this attack does not enable the malicious server to learn any information about the user's query or location. The only thing that could help the malicious server find information about the user's location is the content of the query, which is protected by using the PIR scheme within the MGRS block.

4.6 Experimental Evaluation and Results

We implemented a prototype of our proposed protocol based on Percy++, an open source PIR protocol written in C++ [1] [21][43][44][53][26][71]. We ran our C++ prototype on a virtual

machine with an Intel® Core™ i7-8550U CPU @ 1.80 GHz, 4GB RAM, and Ubuntu Linux. We followed all assumptions of Olumofin et al. [81] in our implementation to compare our results with their approach. We randomly generated and distributed ten million POIs within Canada and the U.S. [116]. Each POI consisted of 300 bytes that included the longitude and latitude coordinates, name, exact address, phone number, website address, etc. of the POI [81]. We set the number of the databases to two to use the Percy++ PIR [81]. For each of these, we applied MGRS to our generated map to create four databases for four levels of user privacy: MGRS-0.1 km, MGRS-1 km, MGRS-10 km, MGRS-100 km.

Recall that the main focus of this section is to reduce the decode time on a smartphone to make the PIR scheme practical for resource-constrained hardware. The decoding time has a direct correlation with the number of POIs in each MGRS block. As we defined the fixed number of POI types for different levels of MGRS blocks (ten types), the PIR decoding time on the smartphone depends on the number of POIs in each type. Therefore, if the number of POIs in a type increases, the decoding time increases. We equalized the number of POIs in all types of MGRS blocks by adding "null" entries, as explained above. Therefore, there can be types that have no data or less than the maximum POI type. This improves the decoding time compared to a type that has the maximum number of POI. As observed in Figure 8, when the user's level of privacy is increased (larger MGRS block), the probability of obtaining more POIs per type increases. Therefore, an MGRS block with a maximum of five POIs per type has faster data retrieval compared to 500 POIs per type. However, there are some exceptions. For example, MGRS-0.1 km required less computational time with 50 POIs per type than with five POIs per type. These exceptions can happen if the type that was requested has fewer POIs than 50, and the rest of the data in that type is "null" (to make the total number of POIs equal to the maximum

number of POIs in the requested MGRS block). During the decode time, whenever the first “null” block is detected, the decode operation stops and results are returned to the user.

We ran our prototype using different levels of MGRS blocks, and we generated 100 random requests to calculate the average time to decode and retrieve the results. As observed in Figure 8, the probability that we had 500 POIs from one type in an MGRS-0.1 km was equal to zero. This means, for example, that in an area of 0.1 km2, there cannot be 500 banks. The opposite scenario may also occur. The probability that we have five POIs of one type in an MGRS-100 km block is rare. Considering this, the MGRS-10 km is the best choice if we want to show the results for an MGRS block with various numbers of POIs/types.

In our proposed protocol, the number of rows in each MGRS block is the same as the number of POI types, which was set to ten types per MGRS block, regardless of the size of the MGRS block. Thus, if our MGRS block becomes larger, the number of types (rows) does not change, but the number of POIs per type increases. We also have different numbers of POIs per type. Thus, in an MGRS block, we have some types that have no data or less than the maximum number of POIs in that MGRS block. This is important because we want to compare our results with Olumofin et al. [81], and due to the different definition of privacy and the reasons we just mentioned, it is difficult to give an exact comparison.

However, in order to demonstrate the performance of our proposed protocol compared to Olumofin et al. [81],we followed their implementation by setting the privacy level equal to one. Thus, we considered one large MGRS block that covered the U.S. and Canada, and the number of POI types was set equal to the number of rows in Olumofin et al. [81]. We ran our PIR implementation against a database of ten million POIs. Figure 9 shows the time for decoding and retrieving the results for various numbers of rows and POIs.

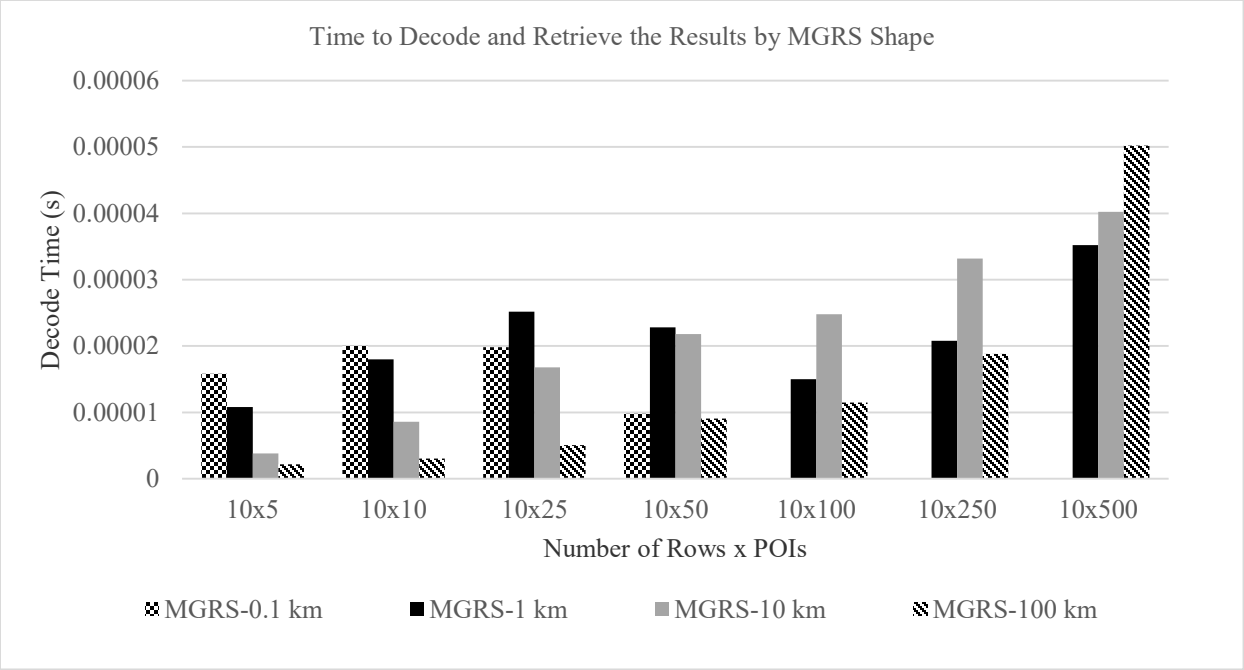


Figure 8 Comparison of time to decode and retrieve the results, by MGRS size at the client side. The results show the computation time for queries on one MGRS block (ten POI types) for different number of POIs (each POI consists of 300 bytes).

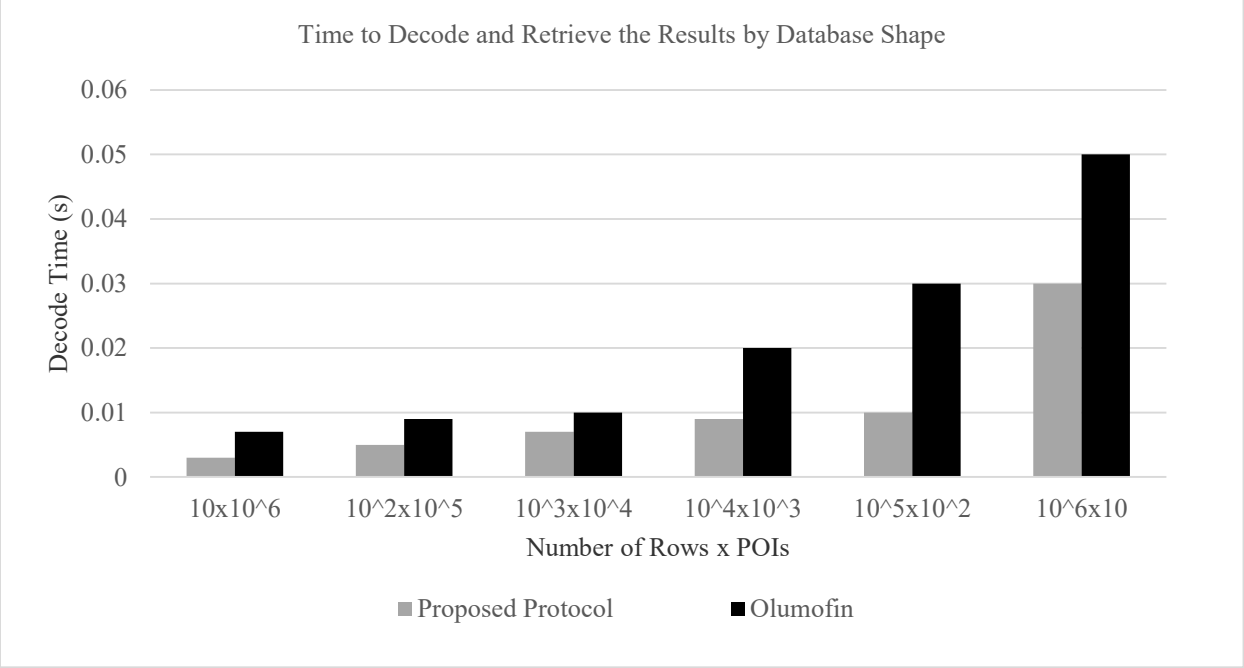


Figure 9 Comparison of time to decode and retrieve the results, by database shape at client side, in our proposed protocol and in Olumofin. The results show the computation time for queries on a 3 GB database for different database shapes (each POI consisted of 300 bytes.)

As observed in Figure 9, our performance is approximately 50% better than that of Olumofin et al. [81] because our method considers the POI type and using the MGRS that applies a fix-sized cloaking region and a variable-sized block to the database. In our proposed protocol, the user receives exactly the type of POI that she was looking for. By increasing the number of POIs per row, the decode and retrieve time increases in the Olumofin et al. [81] protocol because after decoding the rows of the database, the results must be filtered to show the POI that the user was looking for.

4.7 Limitations of our Proposed Protocol

In general, there may be a case in which the user will not find a suitable POI within the requested cloaking region. Therefore, she may wish to search further in a larger MGRS block (i.e., in a broader geographical area). When this happens, the user's privacy does not decrease in our proposed protocol; it is still guaranteed to the level of the original cloaking region.

As mentioned in section 4.3.3, due to the four levels of MGRS (100 km, 10 km, 1 km and 0.1 km), we required four different configurations for our database. In order to use our protocol in a chained hierarchical architecture, servers must use the same database configuration to be able to work together. This is a disadvantage of our proposed protocol compared to Olumofin et al. [81]. This could increase processing on the server when adding or removing POIs from the databases (for example, when a restaurant closes or a new one opens in a specific MGRS block).

Modern smartphones with multi-core processors may be able to handle the 1.5 GB database for the U.S. and Canada, which is used in the Olumofin et. al. [81]evaluation section, as well as the most recent 3 GB location database [116] that we used in our implementation. However, we

should mention that not all people have the most recent smartphones and so our proposal, which reduces computational cost on the client by almost 50%, may be of particular interest for such environments.

4.8 Summary

In this chapter, we proposed a protocol to help the user searches for nearby places of interest while protecting her location's privacy by using private information retrieval (PIR). For this purpose, we first proposed a block-based PIR scheme to reduce computational cost on smartphone applications [56]. We demonstrated that by applying our PIR approach to the LBS, the computational cost on the client side was reduced by approximately 50% compared to that reported in a previous work [81]. This reduction is valuable for the implementation of PIR in smartphone applications with limited resources. We demonstrated that our proposed protocol is secure against active and passive adversaries, as well as against a malicious server that tries to identify information about the user's query and location.

Our approach of retrieving the specific POIs within a cloaking region is much less expensive than the naive approach of requiring the user to download the entire contents of the cloaking region and extract the POI locally. At the same time, privacy is not compromised because the user requests the same cloaking region as if she was requesting the entire contents of the cloaking region. This is a great benefit that reduces the cost of the wireless communication and also the memory usage on the smartphone.

Chapter 5

How to Prevent GPS/Geo-Location Spoofing in Android Applications

Recently, many different types of location-based service (LBS) applications have been released and have attracted a considerable amount of interest from the users' side and the service providers' side. The main factor in these applications is a user's location. LBSs employ the user's location to offer different services, such as discovering nearby friends, locating nearby social events, providing information about traffic jams, etc. Most of these services rely on users' current locations, and the claimed locations of users must be accurate. Most of the time, users can profit from being at a particular location, and if a motivation exists, users may lie about their locations. For example, consider an LBS application that allows users to receive a discount coupon if they frequently visit a specific store: this application requires that users do not fake their location to prevent the application from sending coupons to users who are not eligible to receive them. Alternatively, consider a doctor who pretends to be in a specific ward of a hospital to gain access to a patient's records. Social network applications that enable users to locate their nearby friends are only meaningful if the users use their actual locations.

Unfortunately, some applications attempt to help users fake their location. The motivation of these applications is to protect users' location privacy, while they use online applications. However, as we previously mentioned, LBS applications have relied on users' current locations. Note that some schemes, such as the schemes that we proposed in chapters 2 and 4, allow users to use their real locations on LBS applications without compromising their location privacy. To prevent users from sending a fake location to the LBS application, different centralized and

distributed location proof schemes have been proposed to enable the service provider to validate users' locations [32][61][72][89][98][100][102]. Formally, a location proof is an electronic certificate that proves the existence of someone at a certain geographic location at a certain time. The main challenge is to ensure that a location proof scheme does not violate users' privacy while gathering location proofs. In addition, the location proof scheme should have the ability to recognize users who attempt to trick the architecture by providing location proofs for locations where they are not located. Finally, the architecture must be suitable for smartphone applications with limited processing power, memory, and wireless bandwidth.

As we mentioned in chapter 2, section 2.4.1, if Bob uses a fake location instead of his real location, our proposed protocol is unable to detect this occurrence. Since we consider the Android platform for our nearby friend application implementation, in this chapter, we propose a protocol to detect and reject a request if a user is sending a fake location by using her Android application. Unfortunately, the Android platform allows users to set a "mock location" in the settings menu, which enables users to send a fake location using some applications. Although blocking advertisements or having fun with friends seem to be a reasonable feature, nearby friends' applications are only meaningful if users use their real locations. Sending a fake location may affect the results of the location-based applications, which rely on the authenticity of location information. In the following section, we first review some existing techniques that help Android providers to stop users from sending fake locations. Then, we provide an overview of the previous work on location proof schemes.

5.1 Our Contributions

In this chapter, we propose the idea of adding a machine learning algorithm to our nearby friends' Android application (see chapter 2) to estimate the validity of a user's claimed location. In this idea, the machine learning algorithm can decide if the submitted location is real or not, based on the user's location history, and send the result to the LBS service provider without compromising the user's location privacy. Simultaneously, the LBS service provider is able to verify if the submitted location is real or not by using one of the previous centralized and distributed location proof systems. The LBS service provider's final decision on the validation of the submitted location will be based on the results received from our machine learning algorithm and the location proof system.

Our approach provides anomaly detection by using a machine learning algorithm as an additional step to centralized and/or distributed location proof systems to enable more reliable location proofs. Our proposed protocol is able to distinguish between regular or expected users' locations and irregular or unexpected users' locations. Our final goal in this chapter is to identify a model that represents the regular location behavior of a user over time. Note that our proposed machine learning algorithm is a proof-of-concept and an overall direction for future work. It is necessary to test different types of machine learning algorithms and more data in order to confirm that our proposed protocol is suitable for real life applications.

The remainder of this chapter has the following structure. Section 5.2 presents a brief overview of location proof schemes and unsupervised machine learning algorithms. Section 5.3 describes the details of our threat model and proposed protocol. Section 5.4 gives an overview of

our implementation. The limitations of our proposed protocol are discussed in Section 5.5. Finally, Section 5.6 summarizes the chapter.

5.2 Related work

We first review some existing schemes that help Android providers to stop users from sending fake locations. Then, we provide an overview of previous work on location proof schemes and unsupervised machine learning algorithms.

5.2.1 Android Developer Schemes

Many methods exist to prevent a user from spoofing a location on mobile devices. Developers can simultaneously use different location trackers to determine if an adversary is trying to spoof the location. Some common technologies in geographic tracking are described as follows:

GPS Reporting: A global positioning system (GPS) [85] provides the location and time of any device that contains a GPS receiver. A GPS uses several satellites to provide this feature; therefore, it requires a large amount of power to read these satellites. All satellites broadcast at the same frequency and use code division multiple access (CDMA) to encode signals. GPS devices can be spoofed by broadcasting incorrect GPS signals that resemble a normal GPS signal or by reusing the generated signals of another place at another time.

A secure-military-encrypted GPS [101] provides selective availability (SA) and antispoofing (AS) schemes by changing the GPS infrastructure to make it temporally and spatially unpredictable. Secure-military-encrypted GPS provides assured position,

navigation, and timing (PNT) and has high levels of antijamming or cyber resiliency. Other sensors inside the phone are used to protect PNT and find the real location beyond the GPS, such as inertial measurement units (IMUs). These sensors contain precise accelerometers and gyroscopes that can measure the movement of a phone without any input from outside references. The main goal of a secure GPS is the capability of working despite vulnerabilities such as spoofing and jamming attacks. Jamming employs interference signals that attempt to stop the reception of a GPS signal; simple approaches are available to stop jamming. The encryption technology "M-code" [9] offers a few measures to be more jam-resistant. The most challenging part is preventing spoofing, which requires that a person tracks the device to determine the exact location of the device before it starts spoofing. This step easily stops the device from spoofing its location. By encrypting GPS signals, a person can ensure that she is receiving authentic signals and that the signals are secure against spoofing. Depending on the developer and the reason for using the GPS, different levels of protection, sensor support, and enhanced timing capabilities can be added to future secure GPSs.

Another idea for ensuring that an application receives the real location is the use of the National Marine Electronics Association (NMEA) [114], whose GPS receiver communication is defined within the complete PVT (position, velocity, time) solution. The idea of NMEA is to send a line of data that are totally self-contained and independent, referred to as a sentence. Different types of sentences exist for various applications. For location data, the sentence needed is the "Recommended minimum specific GPS/Transit data", which is referred to as $GPRMC. For example, [114]

$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68

• 225446 (Time of fix 22:54:46 UTC)
• A (Navigation receiver warning A = OK, V = warning)
• 4916.45, N (Latitude 49 deg. 16.45 min North)
• 12311.12, W (Longitude 123 deg. 11.12 min West)
• 000.5 (Speed over ground, Knots)
• 054.7 (Course Made Good, True)
• 191194 (Date of fix 19 November 1994)
• 020.3, E (Magnetic variation 20.3 deg East)
• *68 (Mandatory checksum)

Note that mock location providers do not send or receive NMEA data. NMEA is a standard for communicating or receiving GPS data. Therefore, a person can create a NMEA listener, and if an application does not receive any NMEA updates while the location has changed in onLocationChanged(), then the location has definitely been mocked.

GSM Reporting: The Global System for Mobile (GSM) [78] communication standard for cellular networks regulates air-encrypted wireless communication among mobile phones via base transceiver stations (BTS) or cell towers. The nearby cell towers can triangulate a user's location at a particular time and track the user's movement. Spoofing the location in this method is extremely difficult and is not possible without external hardware. Additionally, the cell traffic is encrypted using a pre-shared key for user authentication.

LAN Reporting: A local area network (LAN) [62] is a computer network that connects devices, primarily via Wi-Fi. Since users need to access the Internet via Wi-Fi access points, they are a suitable choice for determining a user's location with high accuracy. Although this method is vulnerable to spoofing, it can be prevented by applying a strong encryption algorithm as suggested in [82].

WAN Reporting: A wide area network (WAN) [118] is a telecommunication network that is employed for long-distance network connections. This method is the easiest method to spoof; however, it has been employed extensively in mobile communications.

Bluetooth: Similar to Wi-Fi, real-time location systems use "Bluetooth tags" wireless signals [76] to identify the location of a user. However, the range of data transmission is too small to be very useful for LBS.

LocationAssistant [113] introduced a “stop mocking location” application, which contains four steps:

1. Request location updates with a certain accuracy and at fixed update intervals.
2. Ask the user to allow access to their device’s location.
3. If the requested location service, such as GPS, is turned off, direct the user to the page at which they can turn it on.
4. Detect and reject mock locations. If your application detects a mock location, ask the user to turn it off.

These steps are necessary to obtain reliable location updates for the Android platform. The gray boxes depend on user decisions. Step 4 aims to determine whether the location information is real or fake. Detecting mock locations relies on the Android API; if it is less than level 18, it is detectable by the Settings.Secure flag. The application can easily reject any location, while the mock location is enabled. For API level 18 and higher, the Location.isFromMockProvider() should flag mock locations.

Developers can leverage any of these technologies to make an application more difficult to spoof. For example, if an application unlocks features only if a user is in a specific location, a

person could check the GPS and the cell towers. Currently, no GPS spoofing applications can also spoof cell towers. Note that if the user allows for executing as a root, she can bypass these techniques.

5.2.2 Location Proof Schemes

Generally, previous location proof systems could be categorized into centralized systems and distributed systems according to the system architecture. Centralized systems are servers that are relied on for storing proofs of location. In centralized location proof systems [61][72][89], a trusted wireless infrastructure is deployed at different locations to check the physical presence of mobile users and produce location proofs for them. The users submit location proofs to the service provider. This process can be too expensive for a service provider since such a provider should deploy a large number of access points in different locations or locations without fixed wireless infrastructure may exist. Users have to trust the servers, either explicitly or implicitly, to share their locations, which may raise serious concerns about users' privacy. For distributed systems scenarios, mobile users collaborate with a system and produce location proofs for each other. In the literature, the user who wants to make a location claim is referred to as the prover, and other users who produce a location proof for the prover are referred to as witnesses.

Some privacy and security challenges are common between centralized location proof systems and distributed location proof systems, for example, Terrorist Fraud or a Prover–Prover collusion [8][14]. In this attack, a dishonest prover might collude with an adversary who is close to an access point (AP) in the centralized systems or close to an honest witness in the distributed systems to submit a location proof request on behalf of the dishonest prover.

Some of the attacks only target the distributed location proof systems. As we previously mentioned, the location proofs in these systems are generated by witnesses who might not always be honest or trusted. Consider a prover–witness collusion [100] for example. In this attack, a dishonest witness might collude with a remote malicious prover to generate a fake location proof for the prover. To protect users' location privacy in the distributed systems, provers and witnesses need to remain anonymous during the location proof generation phase. In the next two sections, we provide an overview of the existing centralized and distributed location proof schemes as follows.

Centralized System: In 2003, Waters and Felten [102] presented the notion of unforgeable location proofs. They introduced a secure approach that enables a user to receive her location proof from a location manager. The main drawback of their proposed approach is that the user needs to know the verifier to send him a location proof request. Later, Saroiu and Wolman [89] introduced another type of secure location proof approach, in which the user and wireless APs communicate with each other to exchange their public keys to produce location proofs based on a timestamp. In these types of approaches, the user and wireless APs can collude to generate fake proofs. Therefore, they are vulnerable to collusion attacks.

To address the problem of collusion attacks, in 2010, Luo and Hergartner [72] proposed a collusion resilience and privacy protection location proof architecture called VeriPlace. To provide privacy and security protection, VeriPlace requires three different trusted units: A Cheating Detection Authority (CDA), a Trusted Third Party for managing User information (TTPU) and a Trusted Third Party for managing Location information (TTPL). Each of these trusted units knows a user's location or her identity but does not simultaneously know both her location and her identity. Their collusion detection approach assumes that the user constantly

requests her location proof. In this way, if the user sends two location proofs that are far from each other, but their timestamps are shorter than the required time, these location proofs can be considered as anomalies. However, this assumption is not realistic since the user should be able to control the frequency of their queries.

To make it more difficult for a user to forge location proofs, Hasan and Burns [49] presented an approach in which the user needs to provide the location proof from two different entries: a location proof from wireless APs and a location proof from Bluetooth-enabled smartphone witnesses. Therefore, if the user wants to forge a location proof, she has to simultaneously collude with both entities. In this solution the verifier needs to trust a random WiFi AP and a smartphone [72]. All approaches presented in this section are centralized, that is, they all need central infrastructures, such as wireless APs to provide location proofs for the user. This requirement is the main disadvantage of centralized systems since a location may exist that has no central infrastructures.

Distributed Systems: Zhu and Cao [112] introduced a distributed location proof system called APPLAUS. APPLAUS is based on the nearby mobile devices that exchange information about their location and generate a location proof for each other using a short-range wireless interface, such as Bluetooth. For each mobile device, a set of M public/private key pairs are generated and registered with a trusted certificate authority (CA). M public keys are the pseudonyms of a user. To protect the privacy of users, mobile devices distribute their private information among a location proof server, a trusted CA, and the verifier. To protect users' real identities from each other and the location proof server, their pseudonyms periodically change. The main drawback of APPLAUS is the high communication, operation, and storage overhead due to the requirement for periodically changing pseudonyms and generating dummy location proofs.

Alibi is another distributed location proof system proposed by Davis et al. [24], which relies on nearby mobile devices to generate alibis (location proof) for each other. A user's identity is not revealed during alibi generation, and Alibi only reveals it when the user decides to prove the alibi to the judge. In their proposed designed system, however, they did not consider any collusion attacks.

Talasila et al. [98] introduced LINK for secure location verification, in which neighbor mobile devices are connected via a Bluetooth wireless signal and collaborate with the system to verify the location of each other. Verification messages are sent to a trusted location certification authority (LCA). The LCA makes a decision about the validity based on the spatiotemporal correlation among the users, each user's trust scores and historical patterns of the trust scores. LINK has certain drawbacks: first, privacy issues regarding the prover's identity, which broadcasts his ID to the neighbor verifiers; second, LINK is time- and battery-consuming because it is executed in real time to verify a user's location; and finally, because witness nodes are not always trusted in these kinds of systems, LINK is vulnerable to prover–witness collusions. A witness can create a location proof for a user even if the user or both the user and the prover are not at the claimed location.

PROPS [32] and STAMP [100] are other examples of systems that allow nearby users to be witnesses and generate location proof for each other in a private manner without considering a reliable solution for prover–witness collusions. Both of these methods are not protected against terrorist fraud (prover–prover collusions). A terrorist fraud attack refers to the remote malicious prover who colludes with an adversary who is close to an honest witness to convince the witness that he/she is in the claimed location.

In a recent study, SPARSE [80] has introduced a distributed system for mobile devices, in which mobile users generate location proofs for each other. SPARSE has numerous privacy protection and security properties for users, such as integrity, unforgeability, and non-transferability of the location proofs. In addition, SPARSE achieves a highly reliable performance against prover–prover and prover–witness collusions.

5.2.3 Unsupervised Machine Learning

In unsupervised learning, a training set contains a set of samples without labeling. The purpose of unsupervised learning is to attempt to find natural partitions in the training set, where the data do not have a target attribute [119]. In unsupervised learning techniques, we attempt to analyze the data to determine the connections among the data.

Clustering is a technique for identifying similarity groups in the data, which are referred to as clusters. In this technique, the algorithm attempts to find and classify data points that are nearest to each other into one group and classify the data points that are furthest from this set into another group. The quality of a clustering result depends on the algorithm, the distance function, and the application [120].

K-Means Clustering: The k-means algorithm divides data into k clusters. Each cluster has a cluster center, which is referred to as a centroid, and k is specified by the user. Given k, the k-means algorithm works as follows [121]:

1. Randomly choose k data points (seeds) to be the initial centroids/cluster centers.
2. Assign each data point to the closest centroid.
3. Recompute the centroids using the current cluster memberships.
4. If a convergence criterion is not satisfied, return to step 2.

Strengths of k-means [121]:

1. Simple: easy to understand and implement.
2. Efficient: Time complexity: $O(tkn)$, where n is the number of data points, k is the number of clusters, and t is the number of iterations. Since both k and t are small, k-means is considered a linear algorithm. The k-means algorithm is the most popular clustering algorithm.

Weaknesses of k-means [121]:

1. The algorithm is only applicable if the mean is defined.
2. The user needs to specify k.
3. The algorithm is sensitive to outliers.
    a. Outliers are data points that are located very far from other data points.
    b. Outliers can be errors in the data recording or special data points with very different values.

The k-means algorithm is the most popular algorithm due to its simplicity and efficiency [121]. However, the algorithm has some weaknesses. Note that other clustering algorithms also have weaknesses. There is no concrete evidence that one clustering algorithm performs better than another one. The performance depends on the data or applications. The best way to measure the performance of clustering algorithms is to test them with the data and compare the results. We choose k-means for this paper because we discovered the open source implementation of it [122] with scikit-learn, and it is a well-known and extensively employed algorithm for unsupervised learning.

5.3 Proposed Machine Learning Algorithm to Detect Location Spoofing in Android Applications

We explain our machine learning (ML) algorithm by using our proposed nearby friends' protocol (see chapter 2). The main aim of our nearby friends' protocol was to enable the automatic detection of nearby friends, while the user's location privacy is applied to the LBS application. Here, we ensure that the submitted location that is sent to our application is the user's real location.

5.3.1 Problem statement

Alice and her friends have their locations as their secret values. Our nearby friends' protocol enables Alice and her friends to separately learn whether they are in the same pre-agreed grid reference. However, if they are not in the same grid reference, Alice does not learn the location of her friends, and her friends do not learn the location of Alice (see chapter 2). Our nearby friends' Android application notifies Alice if any of her friends are nearby. Unfortunately, the Android platform allows users to set a "mock location" in the settings menu, which enables users to send a fake location using some applications. Sending a fake location can affect the results of our nearby friends' applications, which rely on the authenticity of the location information. The focus of this chapter is to propose a protocol to help the LBS application detect whether the submitted location is the real user's location without learning Alice's real location.

5.3.2 Threat Model

In our threat model, Alice can turn off the machine learning app or use the fake location application whenever she wishes, but we assume that she will not do this indefinitely – there is

no value to Alice in constantly using fake locations, particularly if she desires to use location based services. Therefore, we assume that she will typically report her real location, which allows our machine learning algorithm to learn her normal behavior. Note that if Alice never uses the machine learning application or never sends her real location to our machine learning application, our proposed protocol will still be able to determine the validity of Alice's location in most cases by using the third phase (the location proof scheme).

5.3.3 Proposed Protocol

Our protocol contains three phases. The first phase is the "Android mock location" phase, in which our application checks whether the mock location is on the Android platform. The second phase is the "machine learning" phase, in which we feed the user's location history to the ML algorithm to detect if the new submitted location is real or fake and send the result to the LBS service provider. Our machine learning algorithm decision enables our nearby friends' application to be more confident that it receives a valid location before it provides services. However, the algorithm does not allow the user to prove that her submitted location is real to the LBS service provider. To provide this feature, we can add the third phase to our protocol—the "location proof" phase—in which we use any type of previous location proof scheme to send a proof of submitted location to the LBS service provider. The LBS service provider's final decision on the validation of the submitted location will be based on the results received from the "machine learning" phase and the "location proof" phase.

Android Mock Location Phase: In this phase, we check whether the user enabled the "mock location app" on her Android smartphone [113][123] (see section 5.2.1). However, because the user might only turn on the "mock location app" after turning off our app, we need the second phase.

Machine Learning Phase: Because we are not sure if the visited places in our dataset are fake, we are not able to use supervised machine learning algorithms. Therefore, we apply our training set, which contains a set of samples without labeling, to the unsupervised machine learning algorithm. As we mentioned in section 5.2.4, the purpose of unsupervised learning is to attempt to find natural partitions in the training set, where the data do not have a target attribute [119]. In unsupervised learning techniques, we attempt to analyze the data to determine the connections between the data. Figure 10 shows the lifecycle of our predictive analytics protocol.

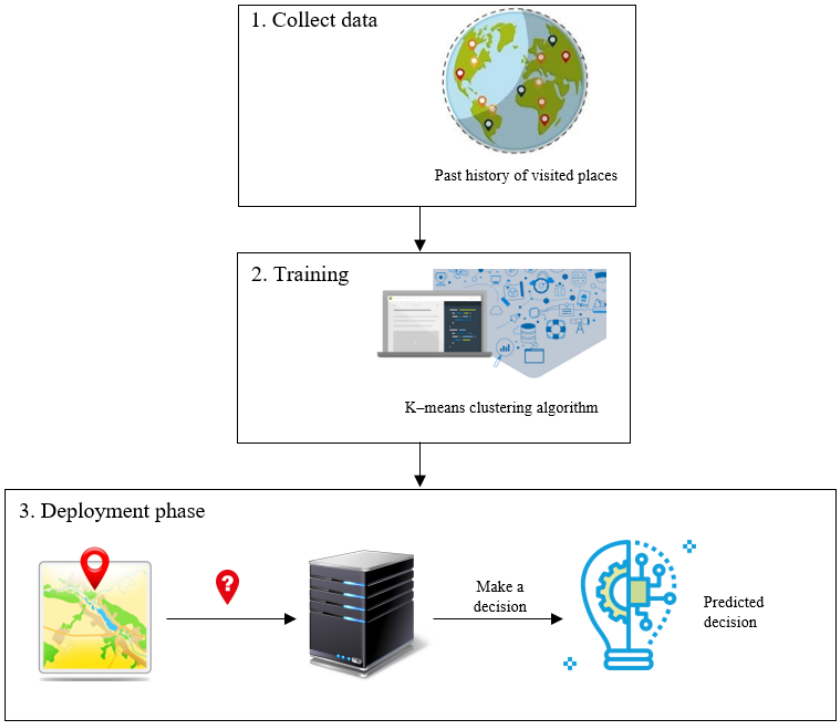


Figure 10 Lifecycle of Predictive Analytics

Because we needed to collect a sufficient amount of location data for our dataset to apply to our machine learning phase (one-year location history), we employed the "Google Map Timeline" and "Google Map Contributions" for one user (the author of this thesis). They provide information about places that the user visited when her location history was turned on for the period of one year from 01/01/2018 to 01/01/2019. In order to add some fake locations, the

author used a “fake GPS” application on her phone during the month of November and December. Table 1 lists the attributes that we applied.

For our training step, we generate clusters of visited places from the training dataset by using the k–means clustering algorithm. To generate our clusters, we select a place as a point and a radius. All places in this area are selected, and the mean of these places is chosen as the new center point. This process is repeated until the mean stops changing. Then, all places in this radius are considered one cluster and are removed from the dataset.

Table 1 Attributes of the place dataset

Datetime	A timestamp when the GPS is on.
Category	Type of places. Eighteen types of places are listed in the dataset.
Day	Day of week
X	Signifies the latitude of the location.
Y	Signifies the longitude of the location.

Once we have created clusters from all datasets, we allocate a unique ID to each cluster. Since we are interested in detecting the locations that the user attempts to spoof, we also consider the category of visited places and the time gaps in the dataset, which demonstrates the amount of time required for the user to move between two different places. For the prediction step, our algorithm checks if a visited place from the testing dataset has belonged to one of the generated real location clusters and categories and if a certain time “t” exists between the visited place and the previous place to conclude that the visited place labels as a real user’s location. Otherwise, the visited place is considered a fake location and it is discarded from the dataset.

Location Proof Phase: In this phase, we can utilize any of the previous location proof schemes, such as centralized or distributed systems (see section 5.2.2), to send the location proof to the LBS service provider.

5.4 Experimental Evaluation and Results of Proposed Machine Learning Algorithm

5.4.1 Attributes

We extract the attributes as follows.

5.4.1.1 Location of Visited Places

'Latitude' (X) and 'Longitude' (Y) have 163 unique entries of the total 552 entries. Among the 163 unique entries, one of the entries has a higher frequency (174 visits) than the remaining entries; we are confident that this place is the user's home. Eighty of the entries have been visited only one time. This result indicates that the user used her Google map when she visited for the first time, and her smartphone's GPS was on to help her find the new place.

5.4.1.2 Category of Visited Places

The user frequently visits a few places, and some places are rare. We categorize the places based on the types of property in the Google Maps Places API [124]. The Google Maps Places API has 90 types of places. However, the user that we chose only visited 18 types of these places.

Table 2 lists the types of visited places by the user, and Figure 11 shows the frequency of visiting these places for the period of one year from 01/01/2018 to 01/01/2019.

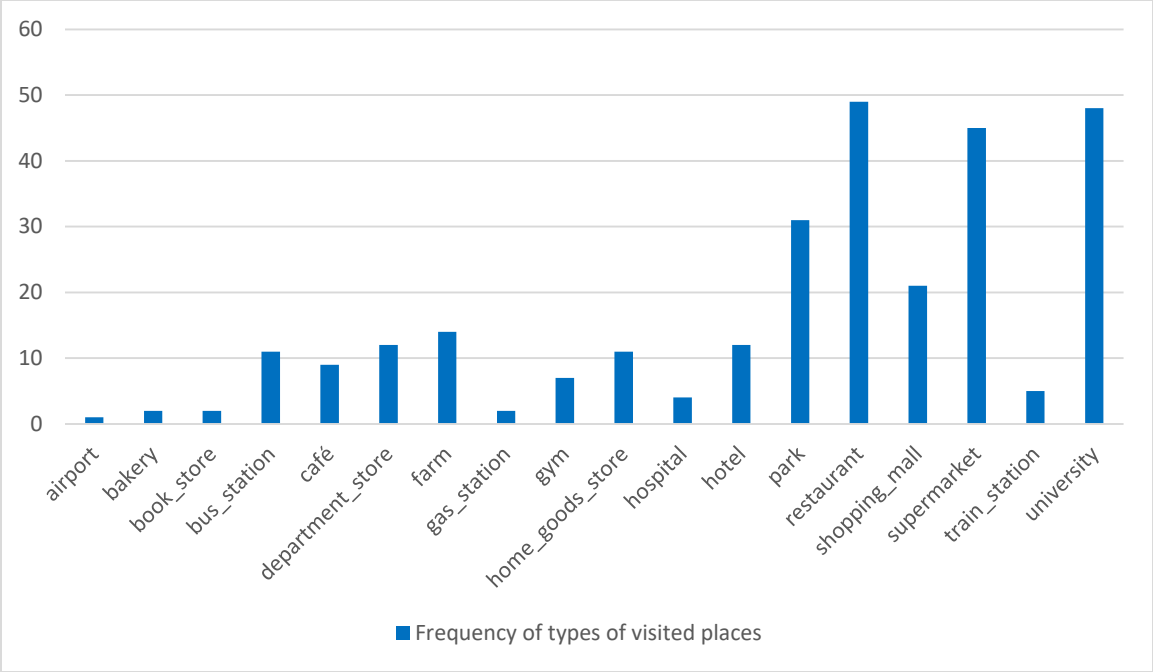


Figure 11 Frequency of types of visited places for the period of one year from 01/01/2018 to 01/01/2019.

Table 2 Types of places visited by the user for the period of one year from 01/01/2018 to 01/01/2019.

airport	bakery	book_store	bus_station	café	department_store
farm	gas_station	gym	home_goods_store	hospital	hotel
park	restaurant	shopping_mall	supermarket	train_station	university

5.4.1.3 Time of Visited Places

From the Google timeline stamp, four main features are extracted: "Hour", "Day", "Month" and "Year". Figure 12 shows the plots of the user's visited places during one year (2018); the user has more activities during summer (Jun to Aug) than during other months.

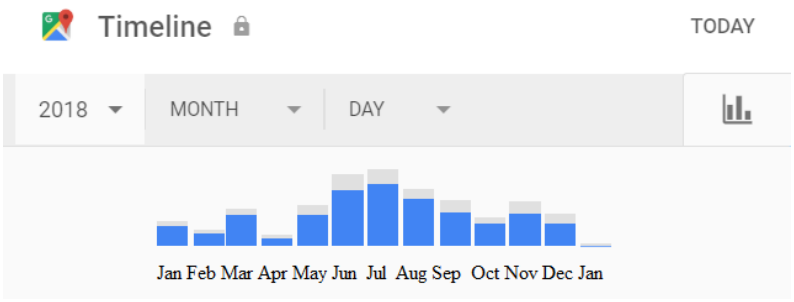


Figure 12 Frequency of visited places in different months of 2018

Considering the day of the week and the time of day provides information about the user's regular location behavior over time. As an example, Figure 13 shows the user's travel timeline on Monday, November 26, 2018.

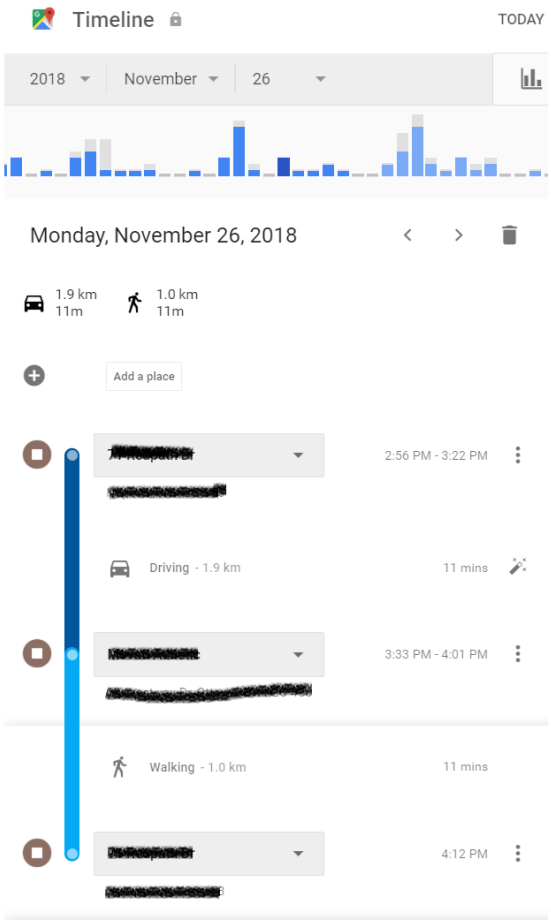


Figure 13 Visited places in different hour of the day November 26, 2018.

5.4.2 Preprocessing

We used the scikit-learn library [125] in Python to preprocess the dataset. The data from the "Day" and "Category" attributes have string values. To use these data in the machine learning models, we need to convert them into numeric values. Scikit-learn has a preprocessing package that is able to convert string data into numeric data by assigning an integer value to each unique item. Note that the datetime attribute is a string data type. However, we are able to convert it into three different datetime numerics: "Hour", "Month" and "Year".

5.4.3 Implementation and Results

The k-means unsupervised classification model is applied to our dataset to predict whether the location submitted by the user is real and fake. To validate our proposed protocol, we chose to examine recall, precision and the Matthews correlation coefficient (definitions and formulas of these terms are available in the Appendix B) in addition to accuracy. Accuracy alone is not a reliable metric for the real performance of a classifier because it will yield misleading results if the dataset is unbalanced.

We identified and tested the open source implementations of the k-means clustering model using the scikit-learn open source python library [122]. We divided our datasets into training datasets and testing datasets to obtain greater accuracy. Typically, most of the data is used for training, and the smaller portion is used for testing. Table 3 lists the results of the implemented methods for the user when the value of the test dataset size considered 15%, 25%, 35 and 45% of the original dataset.

While we were testing the k-means algorithm, we encountered an issue with mapping the results of the k-means clustering with our ground-truth state [83][126]. The issue is that the k-means algorithm is a clustering algorithm rather than a classification algorithm and clustering

works on unlabeled data. An unknown correlation exists between the clusters derived and the ground-truth state of the unlabeled data. However, our dataset is known because the user is the author of this thesis, and she is able to label our clusters as a “real location” and a “fake location”. We can apply a k-means clustering on the element attributes without the labels and then compare the clusters to the retained labels to determine a) a semantic mapping between our cluster centroids and our classes and b) a sense of "confidence" on how our model performs against the known classes of the training data. Given a) and b) above, we can evaluate a test set against the centroids. The closest centroid to a test row and our semantic mapping (from a) provides the predicted class. Since the test set is labeled, we obtain our ground truth. We can then derive “actual” and “predicted” of the two vectors that are required to create our confusion matrix. The "sense of confidence" obtained in b) is a function of the error observed in a cluster centroid (given the semantic mapping from a) and the ground truth (given by the label on the training rows). Note that in future testing (i.e., with more users and without ground truth), n-fold cross-validation will be used to help validate the test results.

As shown in Table 3, the results show that the algorithm is correct most of the time in guessing the true positives because we have more data in our “real location” state. The machine makes some mistakes when attempting to guess the true negatives in some situations for two reasons. First, the amount of data in our “fake location” state is insufficient. Second, some of our “real location” states are in the “fake location” state because the user may have submitted a fake location from the locations that she usually visits and the time gaps are reasonable with the previous place. Therefore, the machine considered the fake location to be a real location.

Table 3 Confusion matrix and results for K-means clustering

Dataset	True State							
Total 552 Entries	85% for Training and 15% for testing		75% for Training and 25% for testing		65% for Training and 35% for testing		55% for Training and 45% for testing	
Test Result	Real Location	Fake Location	Real Location	Fake Location	Real Location	Fake Location	Real Location	Fake Location
Real Location	TP = 70	FP = 1	TP = 91	FP = 7	TP = 167	FP = 2	TP = 160	FP = 0
Fake Location	FN = 5	TN = 7	FN = 8	TN = 32	FN = 5	TN = 19	FN = 11	TN = 77
	Recall 0.9333 Precision 0.9859 MCC 0.6783 Accuracy 0.9277		Recall 0.9191 Precision 0.9286 MCC 0.7341 Accuracy 0.8913		Recall 0.9709 Precision 0.9882 MCC 0.8264 Accuracy 0.9637		Recall 0.9356 Precision 1.0000 MCC 0.9048 Accuracy 0.9556	

5.5 Limitations of our Proposed Protocol

The main limitation is that our machine learning algorithm is not able to detect the real location from the fake location if there exists an application that allows a user to mock all movement of the user to the other location. Note that some time gaps exist when the GPS receiver cannot find any GPS satellites, such as when the user enters a building or is traveling.

We are aware that additional datasets and testing are required to evaluate our proposed protocol and the results. However, finding a real dataset can be difficult because we may need to convince people to share their location history or we may need to buy datasets. Therefore, for this thesis we employed one real dataset, which is the location history of the thesis author. For future testing, we will explore the applicability of some publicly available datasets used in other location privacy research, such as Foursquare check-ins.

5.6 Summary

In this chapter, we proposed a protocol to help an LBS service provider make a better decision about detecting and rejecting a request if a user is sending a fake location to the LBS

applications, such as nearby friends' applications. Unfortunately, the Android platform enables users to set a "mock location" in the settings menu, which enables users to send a fake location using some applications. Although it appears to be a suitable feature to stop most of the advertisements or to have fun with friends, nearby friends' applications are only meaningful if users use their real locations. Sending a fake location can affect the results of location-based applications, which rely on the authenticity of the location information.

Our approach suggests an idea of anomaly detection using a machine learning algorithm as an additional step for centralized and/or distributed location proof systems to enable more reliable location proofs. Our implementation results showed that our proposed protocol is able to distinguish between regular users' locations or expected users' locations and irregular users' locations or unexpected users' locations with approximately 95% accuracy.

Chapter 6
Conclusion

In this thesis, we proposed two cryptographic protocols, one for nearby friends and one for nearby places privacy preserving LBS. Our nearby friend proposed protocol is based on the Goldwasser-Micali probabilistic encryption scheme. The homomorphic property of the Goldwasser-Micali cryptosystem makes the corresponding plaintext equal to the XOR of the two grid references. If two grid references are equal, the result is "zero". However, the Goldwasser-Micali cryptosystem is vulnerable to CCA2 attacks. Our proposed protocol used authenticated encryption, Encrypt-then-MAC, to tackle this problem. We analyzed the security of the proposed protocol, and we showed an active adversary who has access to the ciphertext on the communication channel and the decryption oracle, cannot forge another ciphertext that leads him to guess the user's location (IND-CCA2 security). Moreover, the active adversary cannot modify the ciphertext to affect the outcome of the protocol in a deliberate way (NM-CCA2 security). We also showed the performance and practicality of our protocol in real-time Android applications. We proved that the presented protocol is able to solve the socialist Millionaires' problem, as well. Note that our solution can also be applied to other problems that are solvable with an exclusive-or homomorphic property.

Our nearby friend proposed protocol is based on the PIR approach. We demonstrated that by applying our approach to PIR, the computational cost on the client side was reduced by approximately 50% compared to that reported in a previous work. This reduction is valuable for the implementation of PIR in smartphone applications with limited resources. We demonstrated that our proposed protocol is secure against active and passive adversaries, as well as against a

malicious server that tries to identify information about the user's query and location. Our proposed PIR scheme is also applicable to all kinds of smartphone applications that want to send a request to database without revealing any information about their request to the database.

As an item of independent interest, we proposed the idea of adding a machine learning algorithm to our nearby friends' Android application to estimate the validity of a user's claimed location. In this idea, the machine learning algorithm can decide if the submitted location is real or not, based on the user's location history, and send the result to the LBS service provider without compromising the user's location privacy. Simultaneously, the LBS service provider is able to verify if the submitted location is real or not by using one of the previous centralized and distributed location proof systems. The LBS service provider's final decision on the validation of the submitted location will be based on the results received from our machine learning algorithm and the location proof system. Our implementation results showed that our proposed protocol is able to distinguish between regular users' locations or expected users' locations and irregular users' locations or unexpected users' locations by approximately 95% accuracy.

There exist a number of interesting directions for our privacy preserving protocol LBS. First, in order to show our proposed nearby friend protocol is better than the previous work, we need to add more detailed comparison results, such as battery consumption, number of operations, and confidence intervals for all performance experiments. Second, our implementation results are based on Goldberg's PIR [43]; in order to improve the computational cost of our proposed protocol, we could develop it based on the higher performance block-based PIR such as the hybrid PIR [25]. Third, our proposed protocol could be extended by supporting more complex types of queries. Finally, our proposed protocol could be combined with the Vehicle-to-Infrastructure (V2I) and the Vehicle-to-Vehicle (V2V) communication to help the user find her

nearby places while she is driving her car. In this scenario, the user receives the latest update about the nearby places for her response from other vehicles or street infrastructure such as traffic lights and signs instead of a solid database. To update information about places and their availability constantly, we could use a Blockchain infrastructure in which other vehicles or street infrastructure are able to update their recent observations about the places. For example, all the infrastructure components on the street are connected to the Blockchain and every change that occurs appears in a block. As a result, if the user is looking for nearby parking, the traffic light could let her know the nearest parking lot and if there is any spot available by checking the updated list on the Blockchain.

References

[1] C. Aguilar-Melchor and P. Gaborit, "A Lattice-based Computationally-Efficient Private Information Retrieval Protocol," In *Western European Workshop on Research in Cryptology*. 2007.

[2] C. Aguilar-Melchor, J. Barrier, L. Fousse, and M.O. Killijian, "XPIR: Private Information Retrieval for Everyone," In *Proceedings on Privacy Enhancing Technologies*, 2016, pp. 155–174.

[3] A. Amir, A. Efrat, J. Myllymaki, L. Palaniappan and K. Wampler, "Buddy Tracking–Efficient Proximity Detection among Mobile Friends," In *Pervasive and Mobile Computing,* 3(5), 2007, pp. 489–511.

[4] C. Ardagna, M. Cremonini, E. Damiani, S. De Capitani di Vimercati, P. Samarati, "Location Privacy Protection through Obfuscation-based Techniques," In *IFIP Annual Conference on Data and Applications Security and Privacy*, 2007, pp 47–60.

[5] C. A. Ardagna, M. Cremonini, and G. Gianini, "Landscape Aware Location-Privacy Protection in Location-Based Services," *Journal of System Architecture,* 55(4), 2009, pp. 243–254.

[6] F. Armknecht, S. Katzenbeisser, and A. Peter, "Group Homomorphic Encryption: Characterizations, Impossibility Results, and Applications," *Journal of Designs, codes and cryptography*, 67(2), 2013, pp. 209–232.

[7] D. Asonov, "Querying Databases Privately: A New Approach To Private Information Retrieval," SpringerVerlag, 2004.

[8] G. Avoine, C. Lauradoux, and B. Martin, "How Secret–sharing can Defeat Terrorist Fraud," In *Proceedings of the 4th ACM Conference on Wireless Network Security*, 2011, pp. 145–156.

[9] B. C. Barker, J. W. Betz, J. E. Clark, J. T. Correia, J. T. Gillis, S. Lazar, K. A. Rehborn, and J. R. Straton, "Overview of the GPS M Code Signal," In *Proceedings of the 2000 National Technical Meeting of the Institute of Navigation*, 2000, pp. 542–549.

[10] M. Bellare and C. Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," *Journal of Cryptology*, 21(4), 2008, pp. 469–491.

[11] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among Notions of Security for Public-key Encryption Schemes," In *Annual International Cryptology Conference*, 1998, pp 26–45.

[12] A. Beimel and Y. Stahl, "Robust Information-Theoretic Private Information Retrieval," *Journal of Cryptology*, 20(3), 2007, pp.295–321.

[13] F. Boudot, B.Schoenmakers, J.Traore, "A Fair and Efficient Solution to the Socialist Millionaires' Problem," In *Discrete Applied Mathematics, Special Issue on Coding and Cryptography,* 111(1-2), 2001, pp.23–36.

[14] L. Bussard and W. Bagga, "Distance–Bounding Proof of Knowledge to Avoid Real–time Attacks," In *Security and Privacy in the Age of Ubiquitous Computing*, 2005, pp. 223–238.

[15] C. Cachin, S. Micali, and M. Stadler. "Computationally Private Information Retrieval with Polylog Communication," In *International Conference on the Theory and Applications of Cryptographic Techniques,* 1999, pp. 402–414.

[16] M. Duckham and L. Kulik, "Location privacy and location-aware computing," In *Dynamic and Mobile GIS*, 2006, pp. 63–80.

[17] J. A. Garay, P. MacKenzie, and K. Yang, "Efficient and Secure Multi-party Computation with Faulty Majority and Complete Fairness," *IACR Cryptology ePrint Archive,* 2004.

[18] R. Cheng, Y. Zhang, E. Bertino, S. Prabhakar, "Preserving User Location Privacy in Mobile Data Management Infrastructures," In *Proceedings of the 6th international conference on privacy enhancing technologies*, 2006, pp. 393–412.

[19] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private Information Retrieval," In *Proceedings of the 36th Annual Symposium on the Foundations of Computer Science,* 1995, pp. 41–50.

[20] B. Chor and N. Gilboa, "Computationally Private Information Retrieval," In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing,* 1997, pp. 304–313.

[21] B. Chor, N. Gilboa, M. Naor, "Private Information Retrieval by Keywords," Technion-IIT, Department of Computer Science, 1997.

[22] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A Peer-to-peer Spatial Cloaking Algorithm for Anonymous Location-Based Services," In *Proceedings of the ACM Symposium on Advances in Geographic Information Systems,* 2006, pp. 351–380.

[23] M.L. Damiani, E. Bertino, C. Silvestri, "The Probe Framework for the Personalized Cloaking of Private Locations," *Transactions on Data Privacy*, 3(2), 2010, pp. 123–148.

[24] B. Davis, H. Chen, and M. Franklin, "Privacy Preserving Alibi Systems," In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, 2012, pp. 34–35.

[25] C. Devet and I. Goldberg, "The Best of Both Worlds: Combining Information-Theoretic and Computational PIR for Communication Efficiency," In *Privacy Enhancing Technologies Symposium*, 2014, pp. 63–82.

[26] C. Devet, I. Goldberg, and N. Heninger, "Optimally Robust Private Information Retrieval," In *USENIX Security Symposium*, 2012, pp. 269–283.

[27] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-generation Onion Router," In *Proceedings of the 13th conference on USENIX Security Symposium*, 2004.

[28] M. Duckham, L. Kulik, "Formal Model of Obfuscation and Negotiation for Location Privacy," In *Proceedings of the third international conference on pervasive computing,* 2005, pp. 152–170.

[29] M. Fischlin, "A Cost-effective Pay-per-multiplication Comparison Method for Millionaires," In *Cryptographers' Track at the RSA Conference*, 2001, pp. 457–471.

[30] C. Fontaine and F. Galand, "A Survey of Homomorphic Encryption for Nonspecialists," *EURASIP Journal on Information Security*, 2007 pp. 15.

[31] E. Fung, G. Kellaris, and D. Papadias, "Combining Differential Privacy and PIR for Efficient Strong Location Privacy," *International Symposium on Spatial and Temporal Databases*, 2015, pp. 295–312.

[32] S. Gambs, M. O. Killijian, M. Roy, and M. Traore, "PROPS: A Privacy–preserving Location Proof System," *IEEE 33rd International Symposium on Reliable Distributed Systems*, 2014, pp. 1–10.

[33] W. Gasarch, "A Survey on Private Information Retrieval," *The Bulletin of the EATCS,* 82(72–107), 2004.

[34] B. Gedik and L. Liu, "MobiEyes: Distributed Processing of Continuously Moving Queries on Moving Objects in a Mobile System," In *Proceedings of the International Conference on Extending Database Technology*, 2004, pp. 67–87.

[35] C. Gentry, and Z. Ramzan, "Single-database Private Information Retrieval with Constant Communication Rate," *International Colloquium on Automata, Languages, and Programming,* 2005, pp. 803–815.

[36] Y. Gertner, S. Goldwasser, T. Malkin, "A Random Server Model for Private Information Retrieval," In *2nd International Workshop on Randomization and Approximation Techniques in Computer Science,* 1998, pp. 200–217.

[37] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, "Protecting Data Privacy in Private Information Retrieval Schemes," In *Journal of Computer and System Sciences*, *60*(3), 2000, pp. 592–629.

[38] M. Ghaffari, N. Ghadiri, M. H. Manshaei, and M. S. Lahijani, "P^4QS: A Peer-to-Peer Privacy Preserving Query Service for Location-Based Mobile Applications," In *IEEE Transactions on Vehicular Technology,* 66(10), 2017, pp. 9458–9469.

[39] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "PRIVE: Anonymous Location-Based Queries in Distributed Mobile Systems," In *Proceedings of World Wide Web Conferenece,* 2007, pp. 371–380.

[40] G. Ghinita, P. Kalnis, S. Skiadopoulos, "Mobihide: a Mobile Peer-to-peer System for Anonymous Location-Based Queries," In *Proceedings of the 10th international conference on advances in spatial and temporal databases,* 2007, pp. 221–238.

[41] G. Ghinita, "Understanding the Privacy-efficiency Trade-off in Location-Based Queries," In *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS,* 2008, pp. 1–5.

[42] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private Queries in Location-Based Services: Anonymizers are not necessary," In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data,* 2008, pp. 121–132.

[43] I. Goldberg, "Improving the Robustness of Private Information Retrieval," In *Proceedings of the IEEE Symposium on Security and Privacy,* 2007, pp. 131–148.

[44] I. Goldberg, Percy++ project on SourceForge, http://percy.sourceforge.net/. last accessed 2018/11/05.

[45] S. Goldwasser and S. Micali. "Probabilistic Encryption and How to Play Mental Poker Keeping Secret all Partial Information," In *Proceedings of the 14th ACM Symposium on Theory of Computing*, 1982, pp. 365–377.

[46] S. Goldwasser, S. Micali, "Probabilistic Encryption," *Journal of Computer and System Sciences*, 28(2), 1984, pp. 270–299.

[47] M. Gruteser and D. Grunwald. "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," In *Proceedings of the 1st international conference on Mobile systems, applications and services*, 2003, pp. 31–42.

[48] B.B. Gupta, D.P. Agrawal, S. Yamaguchi, "Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security," *IGI Global*, 2016.

[49] R. Hasan and R. Burns, "Where Have You Been? Secure Location Provenance for Mobile Devices," *arXiv preprint arXiv:1107.1821*, 2011.

[50] T. Hashem, L. Kulik, R. Zhang, "Privacy Preserving Group Nearest Neighbor Queries," In *Proceedings of the 13th international conference on extending database technology,* 2010, pp. 489–500.

[51] S. Hemisphere. “Northern Hemisphere,” Ann Arbor 1001, 2006. (see also: https://en.wikipedia.org/wiki/Military_Grid_Reference_System)

[52] U. Hengartner, “Hiding Location Information from Location-Based Services,” In *International Conference onMobile Data Management,* 2007, pp. 268–272.

[53] R. Henry, F. Olumofin, I. Goldberg, “Practical PIR for Electronic Commerce,” In *Proceedings of the 18th ACM conference on Computer and communications security*, 2011, pp. 677–690.

[54] M. Hezaveh and C. Adams, “Privacy Preserving Discovery of Nearby-Friends,” In *E-Technologies: Embracing the Internet of Things*, *Lecture Notes in Business Information Processing,* (289), 2017.

[55] M. Hezaveh and C. Adams, “An Efficient Solution to the Socialist Millionaires’ Problem,” In *IEEE 30th Canadian Conference on Electrical and Computer Engineering,* 2017, pp. 1–4.

[56] M. Hezaveh and C. Adams, “A PIR Scheme to Improve the Computation Cost on the Client-Side of Smartphone Application,” In *IEEE 31th Canadian Conference on Electrical and Computer* Engineering, 2018, pp. 1–4.

[57] M. Hezaveh and C. Adams, “A Practical PIR-based Scheme for Discovering Nearby Places for Smartphone Applications,” Advances in Science, Technology and Engineering Systems Journal, vol. 4, no. 1, 2019, pp. 27–39.

[58] H. Hu, J. Xu, “Non-exposure Location Anonymity,” In *Proceedings of the 25th IEEE international conference on data engineering,* 2009, pp. 1120–1131.

[59] R. Huang, B. Ying, and A. Nayak, “Protecting location privacy in opportunistic mobile social networks,” In *IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–8.

[60] M. Jakobsson, M.Yung, “Proving Without Knowing: On Oblivious, Agnostic and Blindfolded Provers,” In *Annual International Cryptology Conference*, 1996, pp. 186–200.

[61] C. Javali, G. Revadigar, K. B. Rasmussen, W. Hu, and S. Jha, “I Am Alice, I Was in Wonderland: Secure Location Proof Generation and Verification Protocol,” *IEEE 41st Conference on Local Computer Networks*, 2016, pp. 477–485.

[62] A. E. Karbowiak, and G. J. Anido, “Local Area Network,” U.S. Patent No. 4,663,748. 5 May 1987.

[63] A. Khoshgozaran, C. Shahabi, and H. Shirani-Mehr, “Location Privacy: Going Beyond K-anonymity, Cloaking and Anonymizers,” *Knowledge and Information Systems,* 26(3), 2011, pp. 435–465.

[64] Y, G. Kim, J. Kong, and S. W. Chung. “A Survey on Recent OS-level Energy Management Techniques for Mobile Processing Units,” *IEEE Transactions on Parallel and Distributed Systems,* 2018.

[65] E. Kushilevitz and R. Ostrovsky, “One-way Trapdoor Permutations are Sufficient for Non-trivial Singleserver Private Information Retrieval,” In *International Conference on the Theory and Applications of Cryptographic Techniques,* 2000, pp. 104–121.

[66] E. Kushilevitz and R. Ostrovsky, “Replication is not Needed: Single Database, Computationally-Private Information Retrieval,” In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science,* 1997, pp. 364.

[67] J. Li, Z. Liu, X. Chen, F. Xhafa, X. Tan, D.S. Wong, L-EncDB, “A lightweight framework for privacy-preserving data queries in cloud computing,” *Knowledge-Based Systems* (79), 2015, pp. 18–26.

[68] D. Lin, E. Bertino, R. Cheng, and S. Prabhakar, "Position Transformation: a Location Privacy Protection Method for Moving Objects," In *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS,* 2008, pp. 62–71.

[69] H. Lipmaa, "An Oblivious Transfer Protocol with Log-Squared Communication," In *International Conference on Information Security,* 2005, pp. 314–328.

[70] H.-Y. Lin and W.-G. Tzeng, "An Efficient Solution to the Millionaires' Problem Based on Homomorphic Encryption," In *International Conference on Applied Cryptography and Network Security* , 2005, pp. 456–466.

[71] W. Lueks, I. Goldberg, "Sublinear Scaling for Multi-Client Private Information Retrieval," In *International Conference on Financial Cryptography and Data Security,* 2015, pp. 168–186.

[72] W. Luo and U. Hengartner, "VeriPlace: A Privacy–Aware Location Proof Architecture," In *Proceedings of the 18th International Conference on Advances in Geographic Information Systems*, 2010, pp. 23–32.

[73] W. Mao. "Modern Cryptography: Theory and Practice," Upper Saddle River, NJ: Prentice-Hall PTR, 2003.

[74] S. Mascetti, C. Bettini, X. S. Wang, D. Freni, and S. Jajodia, "ProvidentHider: An Algorithm to Preserve Historical K-anonymity in LBS," In *Proceedings of the 10th IEEE international conference on mobile data management,* 2009, pp. 172–181.

[75] S. Mascetti, D. Freni, C. Bettini, X. Wang, and S. Jajodia, "Privacy in Geo-social Networks: Proximity Notification with Untrusted Service Providers and Curious Buddies," *The International Journal on Very Large Data Bases*, 20(4), 2010, pp. 541–566.

[76] B. A. Miller, and C. Bisdikian. "Bluetooth Revealed: the Insider's Guide to an Open Specification for Global Wireless Communication," Prentice Hall PTR, 2001.

[77] M.F. Mokbel, C.-Y. Chow, and W.G. Aref, "The New Casper: Query Processing for Location Services without Compromising Privacy," In *International Conference on Very Large Data Bases,* 2006, pp. 763–774.

[78] M. Mouly, M. B. Pautet, and T. Foreword By-Haug, "The GSM System for Mobile Communications," Telecom publishing, 1992.

[79] B. Niu, Q.H. Li, X.Y. Zhu, G.H. Cao, H. Li, "Enhancing privacy through caching in location-based services," In *IEEE Conference on Computer Communications*, , 2015, pp. 1017–1025.

[80] M. R. Nosouhi, S. Yu, M. Grobler, Y. Xiang, and Z. Zhu, "SPARSE: Privacy-Aware and Collusion Resistant Location Proof Generation and Verification," In *IEEE Global Communications Conference*, 2018, pp. 1–6.

[81] F. Olumofin, P. K. Tysowski, I. Goldberg, U. Hengartner, "Achieving Efficient Query Privacy for Location-Based Services," *International Symposium on Privacy Enhancing Technologies Symposium*, 2010, pp. 93–110.

[82] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg and D. Boneh, "Location Privacy via Private Proximity Testing," In *Proceedings of the Network and Distributed System Security Symposium*, 2011.

[83] A. A. Olaode, G. Naghdy, C. A. Todd, "Unsupervised Image Classification by Probabilistic Latent Semantic Analysis for the Annotation of Images," *IEEE International Conference on Digital lmage Computing: Techniques and Applications*, 2014, pp. 1–8.

[84] P. Paillier, "Public Key Cryptosystems Based on Composite Degree Residue Classes," In *International Conference on the Theory and Applications of Cryptographic Techniques,* 1999, pp. 223–238.

[85] B. W. Parkinson, P. Enge, P. Axelrad, and J. J. Spilker, J. eds. "Global Positioning System: Theory and Applications, Volume II," American Institute of Aeronautics and Astronautics, 1996.

[86] S. Papadopoulos, S. Bakiras, and D. Papadias, "Nearest Neighbor Search with Strong Location Privacy," In *Proceedings of the VLDB Endowment*, 3(1-2), 2010, pp. 619–629.

[87] A. Pingley, W. Yu, N. Zhang, X. Fu, and W. Zhao. "CAP: A Context-Aware Privacy Protection System For Location-Based Services," In *29th IEEE International Conference on Distributed Computin Systems,* 2009, pp. 49–57.

[88] D. Riboni, L. Pareschi, and C. Bettini, "Privacy in Georeferenced Context-Aware Services: A Survey," *Privacy in Location-Based Applications*, 2009, pp.151–172.

[89] S. Saroiu and A. Wolman, "Enabling New Mobile Applications with Location Proofs," In *Proceedings of the 10th workshop on Mobile Computing Systems and Applications*, 2009, p. 3.

[90] J. Sen, "Homomorphic Encryption: Theory & Application," In *Theory and Practice of Cryptography and Network Security Protocols and Technologies,* 2013, pp. 1–21.

[91] A. Shamir, "How to Share a Secret," *Communications of the ACM*, 22(11), 1979, pp. 612–613.

[92] P. Shankar, V. Ganapathy, and L. Iftode, "Privately Querying Location-Based Services with SybilQuery," In *Proceedings of the 11th international conference on ubiquitous computing,* 2009, pp 31–40.

[93] R. Shokri, G. Theodorakopoulos, P. Papadimitratos, E. Kazemi, J.P. Hubaux, "Hiding in the mobile crowd: Location privacy through collaboration," IEEE Transactions on Dependable Secure Computing, 11 (3), 2014, pp. 266–279.

[94] L. Siksnys, J. R. Thomsen, S. Saltenis, M. L. Yiu, and O. Andersen, "A Location Privacy Aware Friend Locator," In *Proceedings of the 11th Int. Symp. Adv. Spatial-Temporal Databases,* 2009, pp. 405–410.

[95] R. Sion and B. Carbunar, "On the Computational Practicality of Private Information Retrieval," In *Proceedings of the Network and Distributed Systems Security Symposium,* 2007.

[96] C. Stergiou, K.E. Psannis, B.G. Kim, B.B. Gupta, "Secure integration of IoT and cloud computing," *Future generation computer systems*, (78), 2016, pp. 964-975.

[97] G. Sun, D. Liao, H. Li, H. Yu, and V. Chang, "L2P2: A location-label based approach for privacy preserving in LBS," In Journal of *Future Generation Computer Systems* 74(1), 2017, pp. 375–384.

[98] M. Talasila, R. Curtmola, and C. Borcea, "Link: Location Verification through Immediate Neighbors Knowledge," In *International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, 2012, pp. 210–223.

[99] S. Wang, X. Ding, R. H. Deng, and F. Bao, F, "Private Information Retrieval Using Trusted Hardware," *European Symposium on Research in Computer Security*, 2006, pp.49–64.

[100] X. Wang, A. Pande, J. Zhu, P. Mohapatra, "STAMP: Enabling Privacy–Preserving Location Proofs for Mobile Users", *IEEE/ACM Transactions on Networking*, vol. 24, no. 6, 2016, pp. 3276–3289.

[101] J. S. Warner and R. G. Johnston, "GPS Spoofing Countermeasures," *Journal of Homeland Security* 25(2), 2003, pp. 19–27.

[102]B. Waters and E. Felten, "Secure, Private Proofs of Location," *technical report, Department of Computer Science, Princeton University*, 2003.

[103]H. Wu, and Y-C. Hu, "Location Privacy with Randomness Consistency," In *Proceedings on Privacy Enhancing Technologies,* 2016, pp. 62–82.

[104]A.Yao, "Protocols for Secure Computation," In *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, 1982, pp. 160–164.

[105]A.Yao, "How to Generate and Exchange Secrets," In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, 1986, pp. 162–167.

[106]X. Yi, R. Paulet, and E. Bertino, "Homomorphic Encryption and Applications," Springer Briefs in Computer Science. Springer, 2014.

[107] B. Ying, and A. Nayak, "Social location privacy protection method in vehicular social networks," In *IEEE International Conference on Communications Workshops*, 2017, pp. 1288–1292.

[108] B. Ying, and A. Nayak, "Location Privacy-protection based on p-destination in mobile social networks: A game theory analysis," In *IEEE Conference on Dependable and Secure Computing*, 2017, pp. 243–250.

[109]G. Zhong, I. Goldberg, and U. Hengartner, "Louis, Lester and Pierre: Three Protocols for Location Privacy," In *7th Workshop on Privacy Enhancing Technologies,* 2007, pp. 62–76.

[110]G. Zhong, and U. Hengartner. "Toward a Distributed K-anonymity Protocol for Location Privacy," In *Proceedings of the 7th ACM workshop on Privacy in the electronic society,* 2008, pp. 33–38.

[111]G. Zhong, and U. Hengartner. "A Distributed K-anonymity Protocol for Location Privacy," In *Pervasive Computing and Communications,* 2009, pp. 1–10.

[112]Z. Zhu and G. Cao, "Towards Privacy–Preserving and Colluding–Resistance in a Location Proof Updating System," *IEEE Transactions on Mobile Computing*, vol. 12, no. 1, 2011, pp. 51–64.

[113]Location on Android: Stop Mocking Me!, http://www.klaasnotfound.com/2016/05/27/location-on-android-stop-mocking-me/, last accessed 2018/11/05.

[114]NMEA data, http://www.gpsinformation.org/dale/nmea.htm, last accessed 2018/11/05.

[115]Homomorphic Encryption, https://en.wikipedia.org/wiki/Homomorphic_encryption, last accessed 2018/11/05.

[116]Factual, Global Places-Schema, https://my.factual.com/data/t/places, last accessed 2018/11/05.

[117]Mapping Support, https://mappingsupport.com/p/gmap4.php?tilt=off&mgrs=14SPG34308382&z=5&t=t1, last accessed 2018/11/05.

[118]What Is a Wide Area Network (WAN)?, https://www.lifewire.com/wide-area-network-816383, last accessed 2018/11/05.

[119]Unsupervised Learning, https://en.wikipedia.org/wiki/Unsupervised_learning, last accessed 2018/09/10.

[120]Unsupervised Learning, https://www.coursera.org/lecture/machine-learning/unsupervised-learning-olRZo, last accessed 2018/09/10.

[121]Unsupervised Learning Clustering, www.mit.edu/~9.54/fall14/slides/Class13.pdf, last accessed 2018/09/10.

[122]A Demo of K-Means Clustering on the Handwritten Digits Data, https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html, last accessed 2018/09/10.

[123]Location Assistant, https://github.com/klaasnotfound/LocationAssistant#locationassistant, last accessed 2018/01/12.

[124]Google Maps Platform, https://developers.google.com/places/supported_types, last accessed 2018/09/10.

[125]Scikit-learn Machine Learning in Python, https://scikit-learn.org/stable/index.html#, last accessed 2018/09/10.

[126]How can I Test the Performance of a Clustering Algorithm?, https://www.researchgate.net/post/How_can_I_test_the_performance_of_a_clustering_algorithm, last accessed 2018/09/10.

Appendix A

A global, open, collaborative, standardized points of interest database provided by Factual [116].

POI Type		Canada	US
1	Automotive	41913	617664
	Automotive, maintenance and repair	27267	285077
	Automotive, maintenance and repair, tires	0	73587
	Automotive, car parts and accessories	6917	79292
	Automotive, car dealers and leasing, car dealers	7729	105690
	Automotive, car dealers and leasing, used car	0	74018
2	Businesses and services, financial	35914	348814
	Businesses and services, financial, bank and finance, bank and credit union	14886	128993
	Businesses and services, financial, financial planning and investments	7919	91282
	Businesses and services, financial, access and bookkeeping	13109	128539
3	Businesses and services	93421	1181684
	Businesses and services, personal care, beauty salons and barbers	26148	325314
	Businesses and services, shipping freight, and material transportation	6232	51966
	Businesses and services, insurance	13687	224441
	Businesses and services, legal, attorney and law offices	12438	226907
	Businesses and services, real estate, real estate agents	9869	140201
	Businesses and services, Telecommunication services	6667	47031
	Businesses and services, computers	9684	92043
	Businesses and services, printing, copying and signage	8696	73781
4	Businesses and services, home improvement	138297	1403076
	Businesses and services, home improvement	93762	962116
	Businesses and services, home improvement, contractors	25612	260638
	Businesses and services, home improvement, heating, ventilating and air conditioning	5970	74885
	Businesses and services, home improvement, plumbing	6279	58550
	Businesses and services, home improvement, electrician	6674	46887
5	Community and government	57185	839554
	Community and government, organization and associations	14688	136389
	Community and government, education and secondary schools Primary and secondary school	14907	192245
	Community and government, day care and preschools	6859	80547
	Community and government, religious, churches	14531	362889
	Community and government, public and social services	6200	67484
6	Healthcare	51359	969517
	Healthcare, dentists	18683	230012
	Healthcare, pharmacies	11840	62652
	Healthcare, hospitals, clinics and medical centers	8221	155416
	Healthcare, physicians	12615	521437
7	Retail	81534	758818
	Retail, furniture and decor	8656	96082
	Retail, fashion, shoes	5946	61335
	Retail, fashion, clothing and accessories	22886	193354
	Retail, fashion, jewelry and watches	6115	67001
	Retail, construction supplies	5776	42307
	Retail, supermarkets and groceries	10862	98231
	Retail, food and beverage, beer, wine and spirits	6014	54942
	Retail, convenience stores	9610	100149
	Retail, glasses	5669	45417
8	Social	179478	1911585

	Social, food and dining, restaurants	101714	1017499
	Social, food and dinning, restaurants, fast food	17207	236356
	Social, food and dinning, restaurants, dining	16033	0
	Social, food and dinning, cafes, coffee and tea houses	15083	117367
	Social, food and dinning, restaurants, pizza	12888	136018
	Social, food and dinning, restaurants, American	0	225245
	Social, food and dinning, restaurants, Chinese	5724	0
	Social, Bars	10829	179100
9	Transportation	17906	206107
	Transportation, gas stations	11783	161466
	Transportation, taxi and car services, car and truck rentals	6123	44641
10	Travel	27937	177682
	Travel, travel agents and tour operators	6505	35613
	Travel, lodging, hotel and motels	21432	142069

Appendix B

Definitions and Formulas Used in The Results Tables:

True positives (TP): Correctly identified.

True Negatives (TN): Correctly rejected.

False Positives (FP): Incorrectly identified (Also known as a "Type I error.")

False Negatives (FN): Incorrectly rejected (Also known as a "Type II error.")

Accuracy: The proximity of measurement results to the true value.

$$\text{Accuracy} = \frac{\Sigma\,\text{True positive} + \Sigma\,\text{True negative}}{\Sigma\,\text{True positive} + \Sigma\,\text{True negative} + \Sigma\,\text{False positive} + \Sigma\,\text{False negative}}$$

Precision: The repeatability or reproducibility of the measurement.

$$\text{Precision} = \frac{\Sigma\,\text{True positive}}{\Sigma\,\text{True positive} + \Sigma\,\text{False positive}}$$

Recall: The fraction of relevant instances that have been retrieved over the total amount of relevant instances.

$$\text{Recall} = \frac{\Sigma\,\text{True positive}}{\Sigma\,\text{True positive} + \Sigma\,\text{False negative}}$$

MCC (Matthews correlation coefficient): Used in machine learning as a measure of the quality of binary (two-class) classifications.

$$\text{MCC} = \frac{(\Sigma\,\text{True positive} \times \Sigma\,\text{True negative}) - (\Sigma\,\text{False positive} \times \Sigma\,\text{False negative})}{\sqrt{(\Sigma\,\text{True positive} + \Sigma\,\text{False positive}) \times (\Sigma\,\text{True positive} + \Sigma\,\text{False negative}) \times (\Sigma\,\text{True negative} + \Sigma\,\text{False positive}) \times (\Sigma\,\text{True negative} + \Sigma\,\text{False negative})}}$$