Achieving Efficient Query Privacy for Location Based Services

Femi Olumofin
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
fgolumof@cs.uwaterloo.ca

Piotr K. Tysowski
Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
pktysows@uwaterloo.ca

Ian Goldberg
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
iang@cs.uwaterloo.ca

Urs Hengartner
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
uhengart@cs.uwaterloo.ca

Abstract

Mobile smartphone users frequently need to search for nearby points of interest from a location based service, but in a way that preserves the privacy of the users' locations. We present a technique for private information retrieval that allows a user to retrieve information from a database server without revealing what is actually being retrieved from the server. We perform the retrieval operation in a computationally efficient manner to make it practical for resource-constrained hardware such as smartphones, which have limited processing power, memory, and wireless bandwidth. In particular, our algorithm makes use of a variable-sized cloaking region that increases the location privacy of the user at the cost of additional computation, but maintains the same traffic cost. Our proposal does not require the use of a trusted third-party component, and ensures that we find a good compromise between user privacy and computational efficiency. We evaluated our approach with a proof-of-concept implementation over a commercial-grade database of points of interest. Query response time on experimental hardware dropped from between 25 and 70 seconds to less than a second for state or provincial level granularity of location privacy. We also measured the performance of our query technique on a smartphone and wireless network.

Keywords: Location based service, private information retrieval, various-size grid Hilbert curve

1 Introduction

Users of mobile devices tend to frequently have a need to find Points Of Interest (POIs), such as restaurants, hotels, or gas stations, in close proximity to their current locations. Collections of these POIs are typically stored in databases administered by Location Based Service (LBS) providers such as Google, Yahoo!, and Microsoft, and are accessed by the company's own mobile client applications or are licensed to third party independent software vendors. A user first establishes his or her current position on a smartphone such as a RIM BlackBerry, Apple iPhone, or Google Android device through a positioning technology such as GPS (Global Positioning System) or cell tower triangulation, and uses it as the origin for the search. The problem is that if the user's actual location is provided as the origin to the LBS, which performs the lookup of the POIs, then the LBS will learn that location. In addition, a history of locations visited may be recorded

and could potentially be used to target the user with unexpected content such as local advertisements, or worse, used to track him or her. The user's identity may be divulged through the inclusion of the originating dynamic IP address, e-mail address, or phone number in requests to the LBS server so that the results of an LBS query can be routed back to the correct user via a TCP data connection, e-mail reply, or SMS reply, respectively. If a location can always be correlated to each request, then the user's current pattern of activity and even personal safety is being entrusted to a third party, potentially of unknown origin and intent. Although search engines routinely cache portions of previous queries in order to deliver more relevant results in the future, we are concerned when the user's exact location history is tracked, and not just the key words used in the search.

For many users, this constitutes an unacceptable violation of privacy, and efforts should be made to avoid it. As location technology becomes commonplace, users will become increasingly aware of and concerned about location privacy. Not only are privacy and personal safety important considerations, but recent advances in mobile advertising have even opened the possibility of location-based spam. In February 2010, the Energy and Commerce Joint Subcommittee of the U.S. House of Representatives held a joint hearing on the implications of location-based services on the privacy of consumers[1]. Our challenge has been to design a system whereby a user can retrieve useful POI information without having to disclose his or her exact location to a third party such as the LBS server. The user should also not have to reveal what particular POIs were searched for and found, as each POI record typically includes precise location coordinates. Thus, the server will be unable to infer the user's current location or likely destination, or accumulate a history of requests made for profiling purposes. Generally speaking, a user will typically be comfortable with a certain degree of privacy, meaning that the user could be expected to be anywhere within a certain geographic area, such as a city or neighbourhood without fear of discovery.

Today's smartphones have high-performing processors which are suitable for cryptographic operations that can enable location privacy. For instance, the Apple iPhone 3GS contains a Samsung ARM 833 MHz CPU, while the BlackBerry Storm 2 contains a Qualcomm 528 MHz CPU. However, these devices have limited memory and bandwidth. For instance, the iPhone and Storm are both limited to 256 MB of dynamic RAM, 32 GB of flash memory, and operate on 3G wireless networks no faster than the (theoretical) 7.2 Mbps HSDPA network. Consider these data limits with respect to a typical commercial POI database for the U.S. and Canada, which can contain 6 to 12 million entries and require 1 to 2 GB or more of flash data storage. Requiring that the smartphone download the entire database for each request so as not to provide information about its current location is clearly not practical [37]; nor is requiring that it periodically download just the updated data to ensure accuracy of results, given the practical bandwidth limits, data usage limits, and associated overage charges (penalties for exceeding the limits) of smartphone data plans. Thus, it is desirable to provide a cryptographic way for a mobile user to request local information while preserving location privacy. Although extra server-side processing demands must be anticipated on a privacy-enhanced LBS server, it may easily be scaled to multiple computers in a distributed fashion, which is a reasonable tradeoff.

1.1 Requirements and Assumptions

Our basic scenario entails a mobile device user who operates a smartphone with location technology and wireless data transfer capability. The user searches for nearby POIs (i.e., nearest neighbour) by first constructing and sending a query to a known LBS server over the wireless network. The LBS server retrieves the query, performs a search of its POI database, and returns a set of results to the user containing all POIs found in the specified region. Our protocol must meet the following requirements:

- The LBS server must not learn the user's exact location. It may only identify a general region that

[1] http://energycommerce.house.gov/

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.

- There must be no third parties, trusted or otherwise, in the protocol between the user and the server.
- 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.
- The approach cannot rely on a secure processor that is not typically found on a commercial smartphone.

Clearly, these requirements present the need for a mechanism to directly retrieve information in a secure and private way without revealing the contents of the query results, and without the need for an intermediary between the user and the database server to provide some kind of a masking function. Fortunately, there is a branch of cryptography that is associated with retrieving information from a database without revealing which item is being retrieved; it is known as Private Information Retrieval (PIR) [11]. Our proposed solution is sufficiently generic to allow an application to rely on any PIR scheme. We make the same assumptions as that of the underlying PIR scheme, where retrieval is either by object index or keyword [10]. We describe a server that can find the relevant POI entries based on the user's location of interest included in the request; this is possible because the entries in the POI database are indexed by their location.

Although PIR satisfies our baseline privacy constraints, current implementations of it fail to satisfy our third condition, which is usable performance on modern smartphone hardware. Our challenge has been to complement PIR with a new algorithmic approach that effectively reduces the amount of computations without significantly sacrificing the user's location privacy.

Note that we make no effort to hide the user's identity from the location-based service. We assume that it is acceptable to reveal the user's identity for the purpose of routing the response to a location-based request, and for offering a customized LBS experience. A user that also wishes to hide his or her identity to some extent may wish to make use of an onion router, such as Tor [15]. However, we note that there are application domains where the protection of a user's location using our proposed technique is superior to anonymizing the user's identity. For example, it is easy to try to identify a user who made a query with a particular geographical coordinate, simply by looking up the user who lives at the corresponding residential address and assuming the request did not originate elsewhere. On the other hand, our proposed technique hides query contents from the LBS, and leaves no useful clues for determining the user's current location.

When a typical mobile phone accesses a third-party LBS provider through a wireless 3G data connection, we assume that it reveals only its identity and the query itself to the provider. Unavoidably, a mobile communications carrier is always aware of the user's location based on the cell towers in contact, and so it must not collude with the LBS provider. Our assumption relies on the LBS provider not being integrated into the carrier's infrastructure, such as a traffic reporting service using cell tower data that discovers a user's location passively. Our assumption is valid for the vast majority of LBS applications, which are unaffiliated with the carrier; these include search portals, social applications, travel guides, and many other types. When communicating with such an application, the mobile user's IP address is of no help in determining the user's physical location, as it is dynamically assigned independent of location. Only a central gateway that is administered by the telecommunications carrier will be identified. We assume that no other information will be gleaned by the LBS provider. In the case where a mobile user utilizes Wi-Fi instead, the user will be assigned an address that points to the nearby access point, however, and may need to employ other techniques, such as Tor, to mask the address.

1.2 Our Results

We propose a novel hybrid LBS technique that integrates location cloaking and private information retrieval. We have also implemented and evaluated our proposal to determine its practicality on resource-constrained hardware. The results show that users can achieve a good compromise between privacy and computational efficiency with our technique unlike all other existing LBS proposals.

2 Related Work

We provide a brief overview of cloaking- and PIR-based approaches for location privacy. A survey and classification of methods for location privacy in LBS can be found in [39]. Similarly, in a position paper in 2008 [16], Ghinita introduced a taxonomy for LBS privacy techniques.

2.1 Location cloaking techniques

Location cloaking in general seeks to prevent an attacker from being able to match queries to particular users and to thus compromise their privacy. The attacker may be in a position to observe traffic flowing through the network or even be situated at the LBS provider endpoint.

One popular cloaking technique is based on the principle of *k*-anonymity, where a user is hidden among *k-1* other users. Queries from multiple users are typically aggregated at an anonymity server which forms an intermediary between the user and the LBS provider. This central anonymity server can provide spatial and temporal cloaking functions, so that an attacker will encounter difficulty matching multiple queries that are observed with users at particular locations and at particular points in time. Many cloaking solutions for location privacy suggest either a central anonymity server as described [23, 40], or other means such as decentralized trusted peers [13] or distributed *k*-anonymity [42].

The chief problem is that the anonymity server must normally be part of the trusted computing environment and represents a single point of vulnerability. If it is successfully attacked, or collusion with the LBS server occurs, then the locations of all users may be divulged. It is also observed that although a cloaking technique by itself is advantageous in that it does not result in increased computational cost on the server, it can carry with it a high communication cost from the LBS provider to the client. This can mean a large and unacceptable penalty for mobile phone users. Finally, if a reduced sample population results from the number of active users in a particular geographic area, it may not suffice to satisfy the desired degree of anonymity. If the anonymity server delays execution of a request until the *k*-anonymity condition is satisfied, then this delay may prove to be unacceptable to the user from a feature interaction point of view.

We describe some specific location cloaking techniques for the rest of this subsection.

2.1.1 Trusted Anonymity server

In [23], mobile nodes communicate with external services through an anonymity server. Each node sends its position and time information, and the server perturbs the position data according to a cloaking algorithm to reduce the risk of correlation of queries and identification of the user. It also randomly reorders messages to prevent linking of incoming and outgoing traffic. The problem is that devices must constantly report their position so that the density in a region can be ascertained, which is additional traffic that our solution does not require.

In [40], the problem of continuous location-based services is presented, which entails frequent location updates from users versus occasional and independent queries. An anonymity server is entrusted with periodically identifying a cloaking area for the user according to the latest positions of mobile nodes. A drawback with this kind of technique is that devices must constantly report their current position. In our

technique, a sufficiently large cloaking region can be chosen such that it will encompass an expected path of travel during which location updates will be issued.

In [31], a location anonymity server blurs the user's exact location information inside cloaked spatial regions. A privacy-aware query processer embedded inside the LBS is used to handle these regions. A novelty is the specification of anonymity requirements through a user privacy profile. Also, an adaptive location optimizer is presented that allows specification of a location region at one of various levels to satisfy the user's privacy requirements. The reliance on a central anonymity server is still a drawback.

2.1.2 Decentralized trusted peers

Some researchers suggest a decentralized approach to overcome the shortcomings and security threats inherent in an anonymity server that acts as an intermediary between the client and the server.

A decentralized approach overcomes the shortcomings and security threats inherent in an anonymity server. For instance, in [13], a peer-to-peer spatial cloaking algorithm is presented. Before requesting any location-based service, the user forms a group from peers discovered via single-hop communication and/or multi-hop routing. Basically, a user constructs a cloaking region around nearby peers, and nominates one of them to be an agent that will send the query to the database server. The original user then filters out the answer from all the other false positives generated for the other peers. The peer discovery relies on calculating hop distance and latency to other nodes, which is unsuitable for wireless networks. A drawback of this approach is the requirement that the user must also trust the peers with his or her location, and the LBS with the contents of the request.

2.1.3 Distributed k-anonymity

Some of the drawbacks of the above approaches are addressed with distributed k-anonymity [42], where homomorphic encryption is used to allow network operators to collaborate and inform a user whether k-anonymity holds for his or her current area without the operators learning any additional information. In this case, the user can ensure that k-anonymity is achieved.

2.2 PIR-based techniques

2.2.1 Overview

A PIR technique can be used to ensure that queries and their results are kept private. Specifically, PIR provides a user with a way to retrieve an *item* from a *database*, without the database (or the database administrator) learning any information about which particular item was retrieved. PIR satisfies our requirements for privacy and low communication cost. However, existing PIR techniques have drawbacks of high computational cost for applications that require low latency.

The PIR database is typically organized as an n-bit string, broken up into r blocks, each n/r bits long. The user's private input or query is typically an index $i \in \{1, ..., r\}$ representing the i^{th} block of bits. A trivial solution for PIR is for the database to send all r blocks to the user and have the user select the desired block at index i, but this carries a maximum cost of communication and is unsuitable in a resource-constrained environment such as a wireless network.

The three important requirements for PIR are correctness, privacy and non-triviality [14]. Correctness requires that the user obtain the correct database bit or block. Privacy requires the database to learn nothing about the user's private input i or the retrieved block of bits. Non-triviality expects a better solution than the trivial one; that is, the communication complexity between the user and the database must be sub-linear in n. Another requirement, which is not often addressed in the published literature, is implementation efficiency. In fact, the literature has dedicated the most attention to reducing communication complexity at the expense

of computational complexity [1, 37]. This neglect of computational overhead has led to PIR constructions that are impractical on resource-constrained hardware. The promise of PIR in several application domains like patent databases, pharmaceutical databases, online census, real-time stock quotes, and location based services, has been largely unrealized.

When the PIR problem was first introduced in 1995 [11], it was proven that a single-database solution with information theoretic privacy and a sub-linear communication complexity (between the user and the database) is impossible to achieve. Information theoretic privacy assures user privacy even for an adversary with unlimited computational capability. Using at least two replicated databases, and some form of restrictions on how the databases can communicate, PIR schemes with information theoretic privacy are possible, and sometimes hold attractive properties like byzantine robustness [6, 20]. The first single-database PIR proposal was in 1997 [9]; its PIR scheme only assures privacy against an adversary with limited computational capability (i.e., polynomially bounded attackers). The type of privacy protection known as computational privacy, where computational capability is expected to be limited, is a weaker notion of privacy compared to information theoretic privacy. Nonetheless, computational PIR (CPIR) [9, 28] offers the benefit of fielding a single database, unlike information theoretic PIR [6, 11, 20] that requires replication and some form of restrictions on how the databases can communicate. The possibility of adding robustness support to PIR compensated for the privacy risks of fielding multiple databases [6, 20]. Goldberg designed a PIR protocol that offers information theoretic privacy with built in robustness properties for non-responding database servers and for database servers that respond incorrectly (in error or by malice).

Basic PIR schemes place no restriction on information leaked about other items in the database that are not of interest to the user; however, an extension of PIR, known as *Symmetric* PIR (SPIR) [30], adds that restriction. The restriction is important in situations where the database privacy is equally of concern. The only work in an LBS context that attempts to address both user and database privacy is [17]. Although, not strictly an SPIR scheme, it adopts a cryptographic technique to determine if a location is enclosed inside a rectangular cloaking region. The goal of the paper was to reduce the amount of POIs returned to the user by a query. Unlike ours, the approach fails to guarantee a constant query result size which defeats correlation attacks, and it requires dynamic partitioning of the search space which may be computationally intensive. It also requires two queries to be executed, whereas a single query-response pair is sufficient in ours. Another cryptographic construction related to PIR is *oblivious transfer* (OT) [32, 33]. In OT, a database (or sender) transmits some of its items to a user (or chooser), in a manner that preserves their mutual privacy. The database has assurance that the user does not learn any information beyond what he or she is entitled to, and the user has assurance that the database is oblivious or unaware of which particular items it received. OT can thus be regarded as a form of generalization of SPIRs or the superset of protocols that deal with the private exchange of information between two parties in a manner that preserves privacy for both parties.

2.2.2 PIR-based Location Privacy

PIR has been applied to solving the problem of keeping a user's location private when retrieving location-based content from a PIR database. This content typically consists of points of interest (POI's), with each entry consisting of a description of a place of interest as well as its geographical location. The only work cited for PIR in the survey from [39], which does not utilize a third party, is [18]. The key strengths of the solution in [18] are the nondisclosure of location information and the fact that it is resistant against correlation attacks for both stationary and highly mobile users. Our approach differs from the PIR approach in [18] in three important ways. First, the approach is specifically based on the 1997 computational PIR scheme by Kushilevitz et al. [28]. It would require considerable re-invention before it could be used with recent and more efficient PIR schemes. For instance, it re-organizes a POI database into a square matrix M despite the reduced communications costs attainable from using a rectangular matrix. On the other hand, our approach is flexible and supports any block-based PIR schemes. Secondly, the costs of computation and

communication with the approach are $O(n)$ and $O(\sqrt{n})$, respectively, where n is the number of items, or POIs, in the database. The user has no flexibility for dealing with this linear computational cost for large n and it reveals too many POIs to the user; it is too costly for low-bandwidth devices. Our hybrid technique departs from this one-size-fits-all approach and enables users to negotiate their desired level of privacy and efficiency with LBS providers. Thirdly, the scope of the approach did not consider a privacy-preserving partitioning approach for the data set. It considers partitioning with kd-tree and R-tree in the general sense, without specific privacy considerations (see Section 4.2 in [18]). On the other hand, we will show how to use a different method of partitioning of POI data that permits cloaking, and offers privacy protection when used in conjunction with PIR.

The common criticism against this PIR-based approach in the literature is that it is too costly to be practical [29], and that the computational overhead is unsuitable for resource-constrained hardware, such as smartphones [35].

The taxonomy for LBS privacy in [16] discusses how each technique realizes tradeoffs in privacy and efficiency. The taxonomy, in increasing order of privacy protection and decreasing order of performance, is: two-tier spatial transformations (e.g., SpaceTwist [41]), three-tier spatial transformations (e.g., Casper [31]) and cryptographic transformations (e.g., the PIR approach from [18]). The paper defines the taxonomy using the architecture of the various techniques and the transformation of the user's location (i.e., through perturbation or encryption). Techniques based on the two-tier spatial transformation do not utilize an anonymity server and are therefore vulnerable to background knowledge attacks. The three-tier spatial transformation techniques employ an anonymity server and resist background knowledge attacks, but query performance is not as good as in the two-tier transformational techniques.

Most of the PIR-based approaches for location privacy rely on hardware-based techniques, which typically utilize a secure coprocessor (SC) at the LBS server host [2, 24]. This hardware creates a computing space that is protected from the LBS, to realize query privacy. A major drawback of SC-based PIR is that it requires the acquisition of specialized tamperproof hardware and it usually requires periodic reshuffling of the POIs in the database, which is a computationally expensive operation [2, 26].

In [25], trusted computing and secure logging were used to preserve location privacy. The paper addressed the inefficiency of PIR by adopting an alternative solution that requires the Trusted Module to cloak a user's location before accessing the POI database. The service provider can learn a user's cloaking region, but not the exact location. The motivation for this paper was to avoid using PIR in such a way that it would have to process all of the POIs in the database in order to respond to a query, because the computational cost is linear to the size of the database.

2.3 Hybrid techniques

Hybrid techniques [16] permit privacy-efficiency tradeoff decisions to be made by combining the benefits of cloaking- and PIR-based techniques.

Chor et al. [12] conjectured a tradeoff between privacy and computational overhead as a means of reducing the high computational overhead for some application areas of PIR. Our work concretizes and validates their conjecture in the context of LBS, and also realizes the future work left open in [16], which is to further reduce the performance overhead of PIR techniques. The authors' own optimization of PIR in [18] (paper previously mentioned above) reuses partial computation results (i.e., multiplications of large numbers) and parallelizes the computations. This optimization reduces CPU cost by 40%, but the overall query response time is still impractical [29, 35]. Ghinita [16] suggests improving the performance of PIR-based techniques for LBS privacy through a hybrid method that includes a PIR phase on a restricted subset of the data space. Our work answers the open question of how to reduce the processing cost of PIR, without requiring the LBS to have multiple CPUs to take advantage of parallelization. Parallel processors are not typically found on smartphones, either.

Asonov et al. [3, 4] propose a relaxation of the strong privacy requirement of PIR to reduce preprocessing complexity; their PIR requires the support of a secure coprocessor (SC). This relaxation intends to replace the strong privacy requirement of *no* information about the user query being revealed to a weaker privacy notion of *not much* information about the user query being revealed. PIR with relaxed privacy is called Repudiative Information Retrieval (RIR). RIR introduced in [4] requires a PIR with SC support; however, the tradeoff solution presented in our work has no such requirements. The server running the PIR protocols does not need to have a SC. Furthermore, while RIR assumes that an observer cannot determine with certainty if the user queries any of the records in the database, our proposal does not necessarily have such requirements. Our approach reveals the identity of the portion of the database that the user is interested in, but retrieval of the particular item within the portion is realized with PIR. Database item retrieval with our approach thus preserves the stronger notion of privacy that PIR offers. For instance, if the underlying PIR scheme used with our approach offers information theoretic privacy or computational privacy, our approach will maintain that same level of privacy.

3 Our tradeoff solution

We have developed a hybrid solution that consists of PIR to achieve query privacy in the context of a location-based service, and a cloaking technique to reduce the computational cost of PIR to a feasible level. Our technique essentially describes how the user creates a cloaking region around his or her true location, and performs a PIR query on the contents of the cloaking region only. The benefits are numerous: the user's location is kept hidden from the server to an acceptable degree regardless of the number of other users in the area; there is no intermediary server that is responsible for cloaking and that would need to be trusted; and the computational cost of the cryptographic algorithms employed is still practical. We ensure that the user downloads only the POIs that are of interest to the smartphone, keeping wireless traffic to a minimum to reduce costs and conserve the battery. We describe our solution in this section.

The approach that we propose entails two phases. First, there is a pre-processing phase in which the system is set up for use. The pre-processing operation must be carried out whenever significant changes are made to the POI database on the server. In practice, it can occur every few months during a period of low usage on the server such as nighttime maintenance activities. Second, there is an execution phase, in which the LBS server responds to queries for POIs from users. At a high level, the pre-processing phase consists of the following steps:

1. A geographic region is projected onto a two-dimensional plane.
2. A suitable grid is formed on the plane.
3. A collection of POIs is saved in a database such that each row corresponds to one POI.
4. 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).
5. The grid structure is transmitted and saved on the client device in a local mapping database so that it can be referenced in a subsequent query.

The execution phase, in which a query is made for a set of nearby POIs, consists of the following steps:

1. The user determines the area of interest, either based on the current physical position as determined through GPS, or some other arbitrary area that the user may be traveling to in the future.
2. The user chooses a desirable 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. Also, 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 POIs returned to the user to choose amongst.

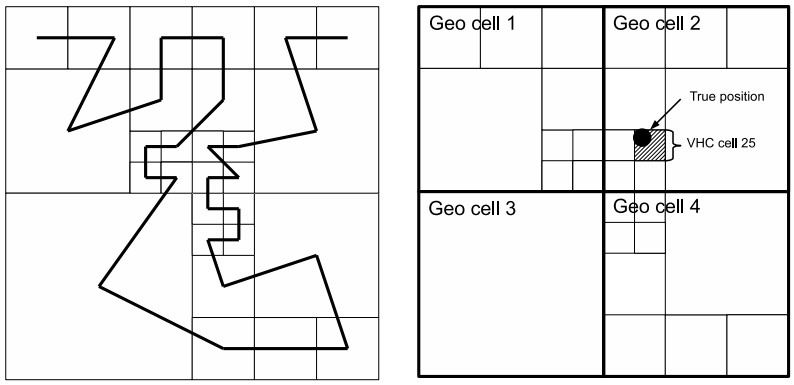
3.1 Level of privacy for the PIR query

To defeat a server's ability to narrow down the search space for the item of interest to the user, PIR protocols typically process every item, or POI, in the PIR database. This results in a computational complexity that is linear in n (where n is the number of items in the PIR database). This is the main hindrance to practical PIR deployment [37].

We propose a tradeoff, in the tradition of PIR development over the years, to make PIR-based solutions practical. For example, information theoretic privacy necessitates replacing a single database with at least two replicated databases; another option is to compromise information theoretic privacy for lower privacy (i.e., attain computational privacy). Our proposal is to offer users the choice of trading off privacy for better query performance, by specifying the levels of privacy that they want for their queries. A level of privacy for the query determines the number of items that the PIR server must process in order to provide a response. Setting levels of privacy is a common practice in several domains where privacy is important (e.g., web browsers). In the specific case of location privacy, we argue that resource-constrained device users are willing to trade off privacy to obtain reasonable performance. On the other hand, such users are equally willing to trade off some levels of performance to gain some levels of privacy support.

A user sets the desired privacy level by specifying the size of the cloaking region. The ratio of the number of POIs inside this region to the number of POIs in the entire POI database defines the level of privacy. The privacy level can be specified in terms of cities/towns (city level), states/provinces (provincial level), and so on, to enhance user-friendliness. Thus, a privacy level value of 1 indicates that the user desires query privacy at the same level as that offered by a typical PIR protocol. Similarly, if a user sets the query privacy level to 0.6, the PIR query will execute faster. Although the cost is still linear in the number of items in terms of computational complexity, the constant term is modified (i.e. in terms of Big-O notation), leading to significant performance gains. At the same time, it will be disclosed to the server that a particular amount of $0.4n$ items are not of interest to the user; this leakage of information does not necessarily constitute a significant breach of location privacy.

The cloaking region is thus identified as a subset of the entire world described by the database. If we imagine that the world is mapped as a grid of so-called geographic grid cells that are equally distributed, then one of these cells will be chosen to comprise the cloaking region. If a higher privacy level is desired, then the cloaking region may be expanded to include multiple geographic grid cells, and thus a larger portion of the database that describes the world. It is sufficient to identify each grid cell by its cell number if the mapping is static and published. The process of mapping the world to a geographic grid occurs during the pre-processing phase, described next.



(a) (b)

Figure 1: (a) A Various-size-grid Hilbert Curve (VHC) mapping with uniform POI density. (b) A user's true position inside VHC cell 25 (shaded) and within a cloaking region bounded by the single geographical grid cell 2. The POI results for VHC cell 25 only will be returned in a query. If a larger cloaking region consisting of geographic grid cells 1 to 4 was specified (for privacy), the same POI results would still be returned.

3.2 Pre-processing and location cloaking

The first step in the pre-processing phase is to represent a geographic area such as the United States and Canada on a two-dimensional plane using a map projection method such as the commonly used Miller cylindrical projection [38]. Once that is done, the user's location of interest may be found on this plane. It is necessary to obscure the user's location by creating a cloaking area around the user's true position or area of interest. POIs will be found anywhere by the LBS server within this cloaking region. The region must be sufficiently large in order to achieve sufficient privacy for the user, but at the same time it must be sufficiently small to minimize the amount of computation required on the user's mobile device to process the query results, as well as to constrain the amount of wireless data traffic required to transport them.

Several techniques allow POIs to be mapped to a cloaking region. One technique is quad-tree mapping [23], but it has the disadvantage (from its use in Casper [31]) of forming an unnecessarily large cloaking region which can impair performance [5]. Another technique is called VHC (Various-size-grid Hilbert Curve) mapping [34], which suits our purpose. In particular, it solves the problem of the density of POIs varying by geographic area. If the density of POIs is significantly higher for a given region (such as a city), then a higher data traffic cost will result if the size of the cloaking region remains constant, and the query will be much slower. If on the other hand, the density becomes significantly lower (such as in a sparsely populated region like the countryside), then the result size may be so minimal that the server may guess the user's likely destination with a high degree of confidence, leading to loss of privacy. VHC solves this problem by creating variable-sized regions that can be used for cloaking, based on the density of the POIs in the geographic area.

Essentially, in VHC, the two-dimensional geographic grid is mapped to a one-dimensional space such that it has equal POI density everywhere (see Fig. 1a). Assume that a typical POI database that covers the regions of Canada and the U.S. will have 6 million POIs. If each VHC cell must contain the same number of POIs, such as 60, then there will be a total of 100,000 VHC cells that will cover this geographic region. Suppose that the lowest POI density found in the database is 60 POIs per 40,000 km^2. Thus, the maximum size of a VHC cell will be 40,000 km^2.

Now, we create a geographic grid overlaying the U.S. and Canada regions with fixed-size square cells that are 200 km in length (the area of each is 40,000 km^2). This corresponds to the maximum size of a single VHC cell as described above. Each geographic grid cell, however, may contain any number of smaller-sized

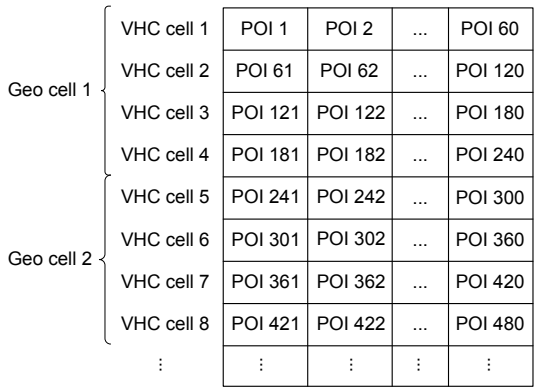


Figure 2: Illustration of the relationship between geographical grid cells, VHC cells, and POIs as stored in database rows.

VHC cells if the POI density of the region is greater (see Fig. 1b).

Finally, the client determines a cloaking region based on a particular privacy level which will dictate the number of geographic grid cells to include inside the cloaking region. Suppose that the client chooses a privacy level such that the cloaking region consists of four geographic grid cells. The user's true location is in one of these grid cells. Inside of the geographic grid cell, there is a set of variable-sized VHC cells according to the distribution of the POIs in the geographic grid cell. The user's area of interest, in which POIs will be searched, will be the single current VHC cell found inside the geographic grid cell. The number of POIs per VHC cell is known, and in our case, it is 60. Thus, the user will initiate a request that will reference the cloaking region, as well as the specific VHC cell in which the user is located or interested in. The user will receive a set of 60 POIs that are found in his or her current VHC cell only. The server will only know that the location of interest is somewhere within the cloaking region defined by the geographic grid cells.

The geographic grid is useful in specifying the size of the cloaking region and for identifying which VHC cells will comprise the cloaking region. The level of privacy, defined from 0 to 1, establishes the size of the cloaking region. The client then sends this cloaking region to the server, by identifying the bounding coordinates (i.e., the longitude and latitude of the top-left and bottom-right corners). The server will then be able to identify which VHC cells belong to this cloaking region, and therefore which portion of the database must be read. The client must also encode the VHC cell containing the area of interest inside a PIR query. (Each VHC cell in the system is uniquely identified by a numeric value.) Fig. 2 further illustrates the relationships among a geographical grid, VHC cells and POIs.

Thus, our cloaking technique provides a way of reducing the search space of the POI database by employing multiple levels of database segmentation. The cloaking region itself is described as a single, or multiple, geographic grid cell or cells. Inside each geographic grid cell are found one or multiple VHC cells, the number depending on the POI density. The user's true location is inside one of these VHC cells, and the user retrieves POI's corresponding to that VHC cell only. As far as the LBS server is concerned, though, the user could be located anywhere within the larger geographic grid cell.

The geographic grid is fixed. The initial grid cell dimensions are configured based on the maximum size of each VHC cell, but once established, will not need to change. Both the client and server must have the same knowledge of the geographic grid. It can be distributed offline (along with the software for the user's smartphone). A simple approach to determining grid cell dimensions is to use a geographic coordinate system such as Degrees-Minutes-Seconds (DMS) [27]. For instance, each grid cell may be two latitude degrees in length, which roughly equates to 200 km at the 30 degree latitude. A population of tens of thousands to millions of users may typically inhabit and stay within the bounds of a grid cell that is 200 km^2 in size, leading to excellent privacy. Cells of larger size will afford province- and state-level privacy if

desired.

Both the client and server must agree on the same VHC mapping, and this mapping must be done off-line in advance. Because it is dependent on population density, it will remain relatively static over time even as the population grows, and can be dynamically updated on the client if necessary. In order to contain knowledge of the mapping to define the cloaking region, the user may make use of a pre-computed map file that is stored locally on the device. This mapping technique is a replacement for a cloaking region that is simply based on cells of constant size, and ensures that a constant and predictable number of results are returned for the user's grid cell.

The idea of using VHC to address the general problem of location privacy was proposed in [34], but in a way that is very different from ours. Specifically, VHC was used to map the user's current location to a 1-dimensional space. Random perturbation was then applied on the 1-dimensional value, which was then mapped back to 2-dimensional space according to the VHC mapping, to represent the user's true location. In essence, the random perturbation was applied to create confusion for an attacker about the user's true location. Our technique differs in that VHC is used for a different purpose; it defines the storage of POI entries of interest within a geographic cell, which comprises the cloaking region, in a way that allows proximate POIs to be stored as adjacent database entries. We then utilize this cloaking region within the context of a privacy-preserving PIR protocol. We do not perform perturbation of the location, which we argue would result in decreased privacy. Indeed, a non-stationary user whose true location is randomly perturbed is still subject to correlation attack. In our approach, we will demonstrate that the cost of computational and communication overhead through our use of PIR is acceptable, as we provide a method for retrieving only a subset of entries of the entire POI database for each query. Our technique is also impervious to correlation attacks.

The device must store a copy of the VHC map in local non-volatile memory, but the storage requirements are very reasonable. The current geographic grid cell encapsulating the user can be derived from the user's current latitude and longitude coordinate, if the mapping convention is known. A single coordinate for the intersection point of each VHC cell inside (i.e. one of its corners) can then be recorded. Hence, a single coordinate would suffice to store each VHC cell in device memory. For quick lookup and to minimize storage requirements, the coordinates of all VHC cells only in the current geographic cell could be stored. Assuming that the smallest VHC cell size is 1 km^2 in size, then the worst case is that 40,000 coordinates will need to be stored to account for all VHC's. Two bytes will be sufficient to store each VHC coordinate, because the origin of the geographic grid cell is known, so that the total cost will be approximately 80,000 bytes to store all VHC cells. This is the worst theoretical case; in practice, small VHC cells will only be encountered in very dense metropolitan areas, and they will not occupy an entire geographic cell.

3.3 Variable level of privacy

The size of the cloaking region and the performance of a query depend on the user's specified level of privacy. If the user wishes to obtain a higher level of privacy, then the size of the cloaking region can be defined to be larger, and to encompass a larger number of geographic grid cells (and thus VHC cells), but the amount of computation on the server will increase accordingly, delaying the response. Nevertheless, the chief benefit is that the processing time of the query on the server is predictable, because each VHC cell in each request contains the same number of POIs. The key fact is that the amount of data transmitted will be roughly proportional to the number of POIs in a single VHC cell (depending on the details of the PIR scheme being employed), but the server will only learn the client's location to the resolution of the cloaking region. The amount of variation allowed in the size of the cloaking region should be kept to a minimum, as this variable may be used to form part of a fingerprint of a target in a correlation attack. Allowing a one-cell or two-by-two-cell region only may be a good compromise. The latter could be employed by the user on a permanent basis to avoid the threat of inter-cell movement being discovered.

Our proposed algorithms for privacy-preserving queries, which allow the user to specify a level of privacy, are explained in detail in the appendix.

4 Security analysis

The location of interest may be either the user's true, physical location, or some other location that must be kept private. For the purpose of this analysis, we will simplify the job of a malicious LBS server by assuming the location of interest is the user's actual location. Without this assumption, it will be more difficult for the server to infer a user's actual location because the user may issue a query for a geographical area without physically residing in that area.

4.1 Collusion prevention for PIR

The approach presented in this paper is sufficiently generic to allow an application to rely on any existing computational or information-theoretic PIR (single-server, multi-server or hardware-assisted). We present an approach for preventing collusion between servers in the case of information-theoretic PIR. This concept is important to our discussion because information-theoretic PIR makes the usual assumption that the replicated PIR servers that implement it do not collude to violate the privacy of the users.

The problem of colluding servers is mitigated by practical business concerns. Realistically, a single POI database would be maintained by an organization that is independent of the LBS providers that a user may query. For instance, LBS providers such as Google and Microsoft may contract the use of a POI source such as the Yellow Pages, an organization that is responsible for its own content that it updates regularly. However, Google and Microsoft would be responsible for the content's distribution to end users as well as integration of partners through banner ads and promotions. Since the LBS providers are operating in the same or similar line of business where they compete to win users and deliver their own advertising models to reap economic benefits, there is no real incentive to collude in order to break the privacy of any user. In this model, it it conceivable that a user would perform a location-based query and would invoke it on the multiple LBS providers concurrently, and combine the results, without fear of the queries divulging the user's precise location. Additionally, individual service agreements can foreclose any chance of collusion with a third party on legal grounds. Users then enjoy greater confidence in usage of the service, and the LBS providers in turn can capitalize on revenue generation opportunities such as pay-per-use subscriptions and revenue-sharing ad opportunities.

4.2 Subscription service and privacy

There is still an open question concerning the effect of a subscription service on the privacy of users. If the model is structured as one that is pay-per-use, then the LBS provider must be provided with a means of identifying the user making a request for billing purposes. Similarly, in order to deliver a customized experience to a user based on personal preferences or history of usage, identification of the user must occur. This practical and reasonable requirement breaks the anonymizing component in proposals such as that of [34]. In contrast, this limitation does not lead to the loss of privacy in our approach because it is focused on preserving the privacy of the location rather than the identity of the user. The LBS provider may have multiple ways of determining which user it was that issued a particular query. Nevertheless, that correlation will not constitute a loss of privacy. If a requirement exists to keep the user's identity secret for any reason, then there is a need to employ cryptographic schemes based on private credentials and anonymous e-cash as described in [8].

4.3 Privacy and size of the cloaking region

Our solution preserves the privacy of the user's location irrespective of the number of other users initiating queries for the same location. The server can infer only the user's location based on the cloaking region. The user may adjust the size of the cloaking region based on his or her personal preferences (i.e., desired level of privacy, query performance, and cost), because a larger region will entail more computation.

The size of the cloaking region is based on a particular size of geographic area and does not need to be adjusted based on the known distribution of POIs within the region. The user only establishes a reasonable level of privacy based on the number of geographic grid cells that define a geographic area. The boundary of the cloaking region utilized in a request is established by the user and is based on the geographic cell map contained on the user's end and the level of privacy parameter. The size of the cloaking region and its boundaries are not controlled by the server.

In the very unlikely scenario where the user is known to be within the area of the cloaking region in advance of the request, then the request itself may be correlated with the user's identity. This is trivial if there are known to be no other users in this cloaking region. The technique of k-anonymity can be used with our approach to prevent such attacks, since an adversary will be unable to isolate a query to any particular user or group of users. Nevertheless, in practice, it is assumed that the geographic grid cells that are used to define the cloaking region will be sufficiently large to encompass a large number of users, to mitigate this risk.

In an extreme case, the user may choose a maximum level of privacy, despite the computational costs entailed. In this case, the user will define a cloaking region that includes the entire geographic region that can be queried; e.g., all of North America. The server would execute its query on all the rows of its database, and so there will be a significant computation cost. However, because only the coordinates of the cloaking region are sent in our protocol, and because only the POIs for the user's current VHC cell will be returned, there will be no additional bandwidth costs in this scenario. The level of privacy will be absolute, in that no information about the user's location will be leaked to the server.

In the opposite extreme case, the user may choose a minimum level of privacy, for maximum performance benefit. In this case, the user will not utilize the geographic grid to establish the boundary of the cloaking region. Instead, the user will pick the current VHC cell that he or she occupies as the actual cloaking region. The server will then infer that the user's location must be within that individual VHC cell. All the POIs for that VHC cell will be returned, but the user's exact location within the VHC cell will still be unknown to the server.

In typical usage, however, the user is expected to define a cloaking region that bounds one or more geographic grid cells, depending on the desired level of privacy. For example, a matrix of 2-by-2 geographic grid cells, or 3-by-3 cells, etc. This user can always adjust the level of privacy for each request. This is useful in the case where the user is traveling between VHC cells. For example, if the user is traveling by car on a highway, and may cross the current geographic grid cell before the next query is issued, then a larger cloaking region formed from the neighbouring grid cells may be appropriate to request.

4.4 Passive attacks

The cloaking region is selected based on the user's location. Let us suppose that the user's preferred level of privacy will be satisfied with a cloaking region that is defined by a matrix of 2-by-2 geographic grid cells. The user will be located in one of these geographic grid cells. Three additional geographic cells must be chosen to form the cloaking region. If the user proceeds to pick these geographic cells randomly from the surrounding set of geographic cells for each request, then each request may send a different cloaking region. After a number of these requests, and if the user remains stationary inside one of these geographic grid cells, then the server will be able to correlate these requests, and determine the overlapping grid cell which likely

contains the user.

In our design, we have elected to specify fixed groups of geographic grid cells at various levels of privacy. In other words, if the user remains stationary, then for each request, the user, depending on the privacy level, will select the same 2-by-2 matrix, or 3-by-3 matrix, etc. Therefore, the correlation attack described above is impossible because the client will send the same cloaking region for all queries for a given privacy level.

Next, consider the case of a mobile user who is physically moving between VHC cells. As long as the user does not move outside of the cloaking region, then the same cloaking region will be transmitted for all queries, and the user will not be subject to a correlation attack, even if the user moves between VHC cells. The same is true if the user moves between geographic grid cells, but still within the same cloaking region defined by the user's level of privacy.

If the user moves to a VHC cell that is outside of the original cloaking region, then a new cloaking region must be formed. The user will now occupy a new geographic grid cell that will define a new cloaking region. The server could observe that requests are being made for neighbouring cloaking regions, and could infer that the user is somewhere close to the edge of these regions, or is traveling between them. This is a consequence of the user having to create cloaking regions such that there is never any overlap between them. The assignment of geographic grid cells to cloaking regions must satisfy this requirement, and is done based on their consecutive ordering on the grid. To avoid the possibility of the server detecting movement between cloaking regions, it is recommended that the user instead increase the privacy level so that a greater number of geographic grid cells will define the cloaking region. This enlarged cloaking region should contain the original cloaking region. Now, if the user moves between grid cells, the same cloaking region will still cover them, and a correlation attack will be unsuccessful. Thus, the user should be aware of his or her likely movement in the future and choose a privacy level such that the cloaking region will contain all current and likely future locations. If the size of the cloaking region is constantly adjusted up and down, based on movement and stationary states, then the server may be able to re-construct a history of the user's movement patterns. Note that in the general proposed scheme, it is not strictly required for the user to be able to predict the path of future travel for all requests. It simply confers an optional improvement to privacy if the user can encapsulate an intended path of travel through a more informed choice of the cloaking region boundaries.

In addition, if an attacker observes the communication between the client and the server, then only the client's identity will be disclosed. Neither the contents of the query nor the results can be decoded if end-to-end encryption, such as Transport Layer Security (TLS), is utilized for the communications link. The server must be able to identify the address of a user's mobile device in order to route its response. The address itself will not give away the user's position, however. This is the case with IP addresses dynamically assigned to mobile devices. Only a central gateway that is administered by the telecommunications carrier will be identified, which will not be useful in tracing the physical location of the user. The gateway typically assigns IP addresses dynamically to all mobile users, in a manner that does not depend on the actual location. Recall that our proposal does not seek to protect the identity of the user from the LBS, but rather to keep his or her location private.

4.5 Active attacks

4.5.1 Replay attack

If an attacker observes a request from the client and launches a replay attack against the server, then the server will respond to the request, but neither the attacker nor the server will receive any additional information on the user's location. TLS can of course also mitigate this attack if necessary.

4.5.2 Result tampering attack

A malicious server can return POIs that are not found in the cloaking region, but the user can filter the results out based on their coordinates. If the server returns false POIs, the user will not be able to verify and detect them. This is unlikely, as the server would need to replace all the rows in the portion of the database (corresponding to the cloaking region) with false information. If the server returns an empty result instead of the actual content in its database portion, then the user may expand the search by increasing the level of privacy and increasing the size of the cloaking region or by exploring neighbouring VHC cells within the cloaking region, but the server will fail to learn anything additional about the user's location.

4.5.3 Timing attack

An attacker could delay a request or a result, such that the results for multiple queries would be received by the client out-of-order. The client will not be confused in this case, as each POI result will include location coordinates, and the user can filter the POIs based on the current VHC cell that he or she is occupying.

5 Experimental evaluation

5.1 Implementations

We developed a C++ prototype and a Java prototype for our proposal using two available implementations of the PIR protocol. The evaluation of our approach in terms of feasibility and scalability is based on the C++ prototype. The point of the Java prototype is to demonstrate the successful porting of our implementation to a smartphone platform. We did not intend to compare these implementations or run them with the same set of parameters. The runtime parameters of the Java prototype are not intended to evaluate our approach, but to show that PIR-enabled applications can be ported to resource-constrained smartphone devices.

The C++ prototype is based on Percy++, an open source PIR protocol written in C++ [19, 20]. The Percy implementation offers computational, information theoretic and hybrid (a mix of both) PIR. We modified Percy++ to support our proposal for allowing PIR queries to be based on a database portion defined by the cloaking region and added code for instrumentation. We measured the computational performance of the PIR algorithm when it does take into account the query level of privacy, and when it does not take it into account. We ran the PIR implementation against a database of 6 million synthetic POIs, the typical number of POIs in a commercial POI database for the U.S. and Canada [21, 22]. We note that a similar experiment in [18] considers a much smaller database; only 10,000 and 100,000 POIs. A head-to-head comparison with [18] is infeasible because we used different PIR implementations and test data. Each POI consists of 256 bytes that we generated randomly. Again, this size is a conservative representation of practical POI sizes. In comparison, the POIs from [18] are only 64 bits in length. The (x, y) location coordinates are stored with each POI.

The Java prototype is based on a computational SPIR protocol implementation [36]. This SPIR protocol was derived from the oblivious transfer protocol by Naor and Pinkas [32] and is the only publicly available Java implementation to our knowledge. This second prototype development consists of both a server component and a client component that we deployed on a smartphone platform.

We initially attempted to port the Java prototype from [36] to the BlackBerry Java Development Environment 4.7 from Research In Motion, but we encountered numerous challenges. The PIR implementation was written using the Java 2 Standard Edition (J2SE) class framework. However, the BlackBerry environment only supports J2ME (Java 2 Mobile Edition) and the Connected Limited Device Configuration (CLDC), which is a strict subset of the class libraries present in J2SE. For example, it does not include an implementation of *BigInteger* required for cryptographic routines, nor does it support floating-point math operations.

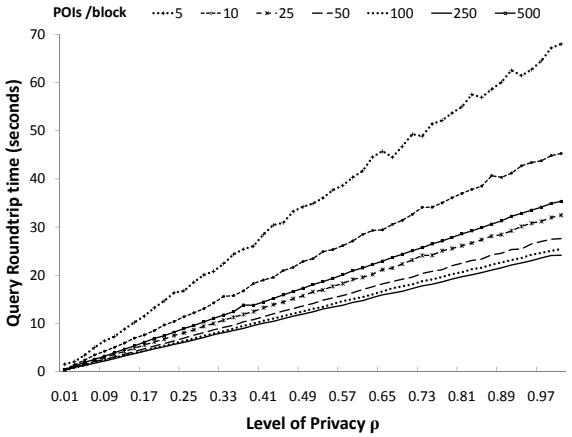


Figure 3: Query roundtrip time and level of privacy for various numbers of POIs returned per query. A single measurement was taken per data point.

After we explored some other alternatives, it became evident that the implementation would require a significant rewrite to optimize it for the J2ME environment. An additional difficulty was that J2ME supports Java language features equivalent to a version of Java that is no higher than 1.3, whereas the PIR library required a higher version.

We found Google's Android platform to be accommodating for porting purposes, as it supported all required classes in the implementation from [36]. While Android supports the Java programming language, it does not run on a standard stack-based Sun Java Virtual Machine (JVM). It uses a proprietary one called Dalvik, which is a register-based architecture, but provides tools to convert Java bytecode into the instruction set used by the Dalvik VM. The Android platform is based on the Linux kernel and was developed by Google and later the Open Handset Alliance. The only aspect of the implementation that could not be adapted without light modification was the RMI mechanism, which we replaced with HTTP socket communication between the Android client process and a server process running on a desktop computer. We ported the PIR implementation to Android using the Android 1.1 SDK and the Android plug-in for Eclipse Ganymede 3.4.1.

5.2 Results and Discussion

We measured query roundtrip times for the C++ prototype on a machine with a 2.91 GHz dual-core AMD CPU, 3GB RAM, and running Ubuntu Linux. Since the Percy++ PIR uses replicated databases, we set the number of databases to 2 [20]. Fig. 3 shows query roundtrip times and levels of privacy for queries returning various numbers of POIs. The number of POIs returned for each query is equivalent to the number of POIs in a VHC cell. Similarly, the number of POIs returned by a query is equivalent of the number of blocks (in bytes), that a traditional PIR query returns. A block of 10 POIs is equivalent to 2560 bytes of data (each POI consists of 256 bytes).

The query roundtrip or response times for block sizes 5, 10, 25, 50, 100, 250, and 500, at query level of privacy 1, are between 25 and 70 seconds. This is because each PIR request runs against the entire database of 6 million synthetic POIs. However, the query roundtrip time improves with lower levels of privacy. For example, the query response times for the above block sizes at a privacy level of 0.17 are between 4 and 12 seconds. One must observe that setting the query level of privacy to 0.17 is equivalent to privately querying a block of POIs from a portion of the database consisting of 1.02 million POIs. If we assume there are equal number of POIs in all the provinces and states of Canada and US, a level of privacy set to 0.17 implies a cloaking region that covers approximately 10 provinces and/or states. Under a similar assumption, a user

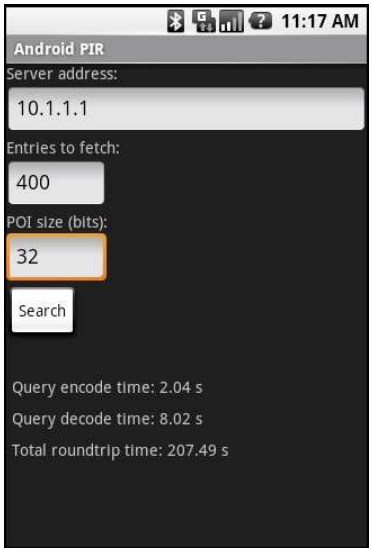


Figure 4: User interface of the Java prototype loaded on an Android emulator.

who intends to hide his or her query in a cloaking region consisting of one province or state will simply set his or her query level of privacy to a much lower value of 0.02. The query response time for this level of privacy is approximately 0.3 seconds for an optimal block size, which in our testing configuration consists of 256 POIs. It is easy to observe from the graph that the block that consists of 250 POIs gives the best performance. Furthermore, the worst performing block size is the one consisting of 5 POIs, the reason being that smaller block sizes require more rounds of computations to process the individual blocks, compared to larger block sizes. On the other hand, large block sizes, such as 500, carry performance penalties and overheads which depend on the characteristics of the underlying PIR scheme, and also on the resource constraints of the runtime hardware (e.g., RAM, disk and memory cache sizes, and network bandwidth). The network cost in the C++ implementation was negligible since the measurements were taken on a LAN.

We also installed the client for the Java prototype on a G1 Android smartphone from T-Mobile, which features a Qualcomm ARM processor running at 528 MHz, and includes 192 MB DDR SDRAM, and 256 MB flash memory. Although our locked smartphone was capable of running on T-Mobile's 3G network in the U.S., it did not support the 3G frequency bands in operation in Canada. We ran our tests using the Rogers EDGE network, which is slower by up to a factor of ten. We created an Android application with a user interface that allows the user to specify the server address and query parameters such as the size of the cloaking region and the size of the portion of the cloaking region to fetch, all in bits (see Fig 4).

For a cloaking region consisting of 400 cells (i.e., database rows), and a POI result size of 32 bits (i.e., size of a database row), the following performance was measured on the actual Android smartphone: query generation required 2.04 s, result processing required 8.02 s, and the total roundtrip time (including transmission over the wireless EDGE packet data network) required 207.49 s. In comparison, when the cloaking region was reduced to 100 cells, with the same POI result size, the performance measurements were: query generation of 4.40 s, result processing of 3.42 s, and a roundtrip time of 62.55 s. We observed little variance in the query encoding time, but the decoding time varied nearly proportionally with the size of the cloaking region, as one would expect. Overall, the implementation was usable even though it had not been originally designed and optimized for the Android platform, and we were restricted to a non-3G network.

6 Conclusions

In this paper, we have proposed an algorithm for private information retrieval that achieves a good compromise between user location privacy and computational efficiency. We have implemented and evaluated our algorithm and shown that it is practical on resource-constrained hardware.

Our approach of using a variable-sized cloaking region divided into VHC cells results in greater location privacy than the traditional approach of a single cloaking region, while at the same time decreasing wireless data traffic usage from an amount proportional to the size of the cloaking region to an amount proportional to the size of a VHC cell. It also allows the user to dynamically choose various levels of privacy. Although increasing the size of the cloaking region does result in higher computation in processing the query, we believe that this tradeoff is very reasonable, given that the processing power of today's smartphones is still less of a concern than the speed and cost of wireless network connectivity.

Our approach of retrieving the points of interest within a single cell, contained 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. At the same time, privacy is not compromised. Because either the user or the network provider must ultimately pay for the cost of the wireless data that is sent, the delays associated with the transfer of wireless traffic are significant, and memory on the device is constrained, it is a great benefit that traffic is kept to a minimum in our scheme.

On the server side, the increased computational demands of queries may be handled through employment of multiple, distributed servers. However, it would also be atypical for more than one query to be made at a time, as a smartphone user typically runs only a single application as a foreground task due to processor, memory, operating system, and screen size limitations. Overall, we have shown that the delay experienced in a POI lookup operation on a typical location based service is minimal. We have done so by testing on actual smartphone hardware available to users today.

Our work could be extended by improving on our general scenario where the user retrieves all of the POIs that belong to the VHC cell of interest. It is possible that the user will not find a suitable POI within this set, and will wish to search further in neighbouring VHC cells. This may be the case when the result set does not contain a desired POI. Therefore, the user may wish to expand the search by searching in a broader geographical area.

One way to accomplish this would be to perform an additional query for each of the other VHC cells in the current geographic grid cell. However, there is overhead associated with each such request. Another approach would be to have the user request POIs for all of the VHC cells within a geographic grid cell. Note that the size of the cloaking region, which consists of one or more geographic grid cells, remains constant. This is important in terms of making it difficult for the server to correlate the expanded search (for all VHC cells) with the original search (for a single VHC cell). An optimization of the first approach would be to reduce the number of queries for individual VHC cells, by querying for multiple VHC cells at the same time. The selection of multiple VHC cells within a geographic grid cell, while mitigating the effectiveness of a correlation attack, remains an open problem.

Furthermore, there is an optimal block size for any PIR protocol that will allow our algorithm to give the best query response time. However, the bandwidth of the communication channel may also play some role in determining which block size is optimal. An extension would be to study what impact network bandwidth will have on the performance of our algorithm, using a variety of PIR implementations.

References

[1] C. Aguilar-Melchor and P. Gaborit. A lattice-based computationally-efficient private information retrieval protocol. Cryptology ePrint Archive, Report 2007/446, 2007.

[2] H. S.-M. Ali Khoshgozaran and C. Shahabi. SPIRAL, a scalable private information retrieval approach to location privacy. In *Proceedings of the 2nd International Workshop on Privacy-Aware Location-based Mobile Services (PALMS)*, 2008.

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

[4] D. Asonov and J.-C. Freytag. Repudiative information retrieval. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, pages 32–40, New York, NY, USA, 2002.

[5] B. Bamba, L. Liu, P. Pesti, and T. Wang. Supporting anonymous location queries in mobile environments with privacygrid. In *Proceeding of the 17th international conference on World Wide Web*, pages 237–246, New York, NY, USA, 2008.

[6] A. Beimel and Y. Stahl. Robust information-theoretic private information retrieval. *J. Cryptol.*, 20(3):295–321, 2007.

[7] C. Bettini, S. Jajodia, P. Samarati, and X. S. Wang, editors. *Proceedings of the 1st International Workshop on Privacy in Location-Based Applications, Malaga, Spain, October 9, 2008*, volume 397 of *CEUR Workshop Proceedings*, 2008.

[8] J. Camenisch, A. Lysyanskaya, and M. Meyerovich. Endorsed e-cash. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 101–115, Washington, DC, USA, 2007.

[9] B. Chor and N. Gilboa. Computationally private information retrieval (extended abstract). In *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 304–313, New York, NY, USA, 1997.

[10] B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. Technical Report TR CS0917, Dept. of Computer Science, Technion, Israel, 1997.

[11] 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*, pages 41–50, Oct 1995.

[12] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[13] C. Chow, M. F. Mokbel, and X. Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *Proceedings of the 14th Annual ACM international Symposium on Advances in Geographic information Systems*, pages 171–178, New York, NY, USA, 2006.

[14] G. D. Crescenzo. Towards practical private information retrieval. Achieving Practical Private Information Retrieval (Panel @ Securecomm 2006), Aug. 2006.

[15] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 21–21, Berkeley, CA, USA, 2004.

[16] G. Ghinita. Understanding the privacy-efficiency trade-off in location based queries. In *SPRINGL '08: Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*, pages 1–5, New York, NY, USA, 2008.

[17] G. Ghinita, P. Kalnis, M. Kantarcioglu, and E. Bertino. A hybrid technique for private location-based queries with database protection. In *SSTD '09: Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases*, pages 98–116, Berlin, Heidelberg, 2009. Springer-Verlag.

[18] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 121–132, New York, NY, USA, 2008.

[19] I. Goldberg. Percy++ project on SourceForge. http://percy.sourceforge.net/.

[20] I. Goldberg. Improving the robustness of private information retrieval. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 131–148, Washington, DC, USA, 2007.

[21] GPSmagazine. Garmin nuvi 780 GPS Review. http://gpsmagazine.com.

[22] GPSreview.net. POI– Points of Interest. http://www.gpsreview.net/pois/.

[23] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys '03: Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42, New York, NY, USA, 2003.

[24] U. Hengartner. Hiding location information from location-based services. In *Mobile Data Management, 2007 International Conference on*, pages 268–272, May 2007.

[25] U. Hengartner. Location privacy based on trusted computing and secure logging. In *SecureComm '08: Proceedings of the 4th international conference on Security and privacy in communication netowrks*, pages 1–8, New York, NY, USA, 2008.

[26] A. Iliev and S. W. Smith. Protecting Client Privacy with Trusted Computing at the Server. *IEEE Security and Privacy*, 3(2):20–28, 2005.

[27] M. Kennedy and S. Kopp. *Understanding Map Projections*. ESRI (Environmental Systems Research Institute) press, 2000.

[28] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computationally-private information retrieval. In *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, page 364, Washington, DC, USA, 1997.

[29] D. Lin, E. Bertino, R. Cheng, and S. Prabhakar. Position transformation: a location privacy protection method for moving objects. In *SPRINGL '08: Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*, pages 62–71, New York, NY, USA, 2008.

[30] S. K. Mishra and P. Sarkar. Symmetrically private information retrieval. In *INDOCRYPT '00: Proceedings of the First International Conference on Progress in Cryptology*, pages 225–236, London, UK, 2000.

[31] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new Casper: query processing for location services without compromising privacy. In *VLDB '06: Proceedings of the 32nd international conference on Very large data bases*, pages 763–774, 2006.

[32] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 245–254, New York, NY, USA, 1999.

[33] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457, Philadelphia, PA, USA, 2001.

[34] 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 Computing Systems*, Jun 2009.

[35] D. Riboni, L. Pareschi, and C. Bettini. Privacy in georeferenced context-aware services: A survey. In Bettini et al. [7].

[36] F. Saint-Jean. Java implementation of a single-database computationally symmetric private information retrieval (CSPIR) protocol. Technical Report YALEU/DCS/TR-1333A, Yale University, New Haven, CT, USA, 2005.

[37] R. Sion and B. Carbunar. On the computational practicality of private information retrieval. In *Proceedings of the Network and Distributed Systems Security Symposium*, 2007.

[38] J. P. Snyder. *Flattening the Earth, two thousand years of map projections*. University of Chicago Press, 1993.

[39] A. Solanas, J. Domingo-Ferrer, and A. Martínez-Ballesté. Location privacy in location-based services: Beyond TTP-based schemes. In Bettini et al. [7].

[40] T. Xu and Y. Cai. Location anonymity in continuous location-based services. In *Proceedings of the 15th Annual ACM international Symposium on Advances in Geographic information Systems*, pages 1–8, New York, NY, USA, 2007.

[41] M. L. Yiu, C. Jensen, X. Huang, and H. Lu. Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 366–375, April 2008.

[42] G. Zhong and U. Hengartner. A distributed k-anonymity protocol for location privacy. In *Proceedings of Seventh IEEE International Conference on Pervasive Computing and Communication (PerCom 2009). Galveston, TX*, pages 253–262, 2009.

A Algorithms

In this appendix, we present the algorithms that implement our proposal to allow a user to set his or her level of query privacy (equivalent to the size of the cloaking region) in the PIR query. Let $PIR = \{PIREncode, PIRProcess, PIRDecode\}$ be some PIR protocol where $PIREncode$, $PIRProcess$, and $PIRDecode$ are the query encoding, response processing, and response decoding protocols, respectively.

A.1 Query generation (by PIR client)

Let n be the total number of items (or POIs) in the PIR database or databases (in the case of a PIR protocol with replicated databases), σ be the number of VHC grid cells in the map where each grid cell has n/σ items. Let i be the index of the database block that the user is after, and $\rho \in [0, 1]$ be the level of query privacy preset by the user. In models where users pay for computational resources used by PIR servers, this privacy level must have been negotiated by the user and LBS provider. The user selects this level by using an application interface on the smartphone. The level is specified in terms of cities/towns (city level), state/provinces (provincial level), and so on.

i. Compute $\ell = \lceil \rho n \rceil$ and set $R = \{r_1, r_2, r_3, \ldots, r_\ell\}$ to be the set of indices for the items corresponding to the cloaking region.

ii. Compute $(q, \tau) = PIREncode_R(i)$ as the PIR query encoding for i, using only the item indices in R (i.e., not all the indices in all geographical grid locations in the entire map are required). Here, q is the query to be sent to the database server, and τ is some state information that will be used to decode the server's response.

iii. Send $\{q, R\}$ to the database (or PIR server). Instead of sending R, it may be more efficient to send only the top left and bottom right coordinates of the bounding rectangle that covers the cloaking region, or the range of identifiers (or numbers) for the VHC grid cells that are within the cloaking region, or for the geographic cells that contain them. In any of these cases, the PIR server can use the provided information to determine R.

A.2 Response encoding (by PIR server)

i. Retrieve a database portion $D = \{d_1, d_2, d_3, \ldots, d_\ell\}$, where d_j is the database item with index r_j. Each item is a POI data structure with attributes longitude, latitude, name, address, phone, category, website address, and so on.

ii. Execute $PIRProcess_D(q)$ to obtain response r, which should be the block of POIs in the user's VHC grid cell.

iii. Return r back to the client.

A.3 Response decoding (by PIR client)

i Execute $PIRDecode_R(\tau, r)$ to obtain a database response to the query. The response should be the set of POIs that the query requested.

ii The client can locally compute the nearest neighbour using the set of POIs that was returned.