Achieving Efficient Query Privacy for Location Based Services[⋆]

Femi Olumofin[*], Piotr K. Tysowski[**], Ian Goldberg[*], and Urs Hengartner[*]

[*]Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
{fgolumof,iang,uhengart}@cs.uwaterloo.ca
[**]Department of Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
pktysows@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. 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

[⋆] An extended version of this paper is available [27].

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

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

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 [31]; 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 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 hardware, such as smartphones, which are resource constrained. 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) [7]. 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 [6]. 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 [10]. 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 [33]. Similarly, in a position paper in 2008 [11], 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 [18, 34], or other means such as decentralized trusted peers [9] or distributed k-anonymity [35].

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.

2.2 PIR-based techniques

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.

When the PIR problem was first introduced in 1995 [7], 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 [3, 15]. The first single-database PIR proposal was in 1997 [5]; 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) [5, 22] offers the benefit of fielding a single database. 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) [24], 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 [12]. 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.

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 [33] which does not utilize a third party is [13]. Our approach differs from the PIR approach in [13] in three important ways. First, the approach is specifically based on the 1997 computational PIR scheme by Kushilevitz et al. [22]. 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 [13]). 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.

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 [1, 19]. 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 [1, 20].

2.3 Hybrid techniques

Hybrid techniques [11] permit privacy-efficiency tradeoff decisions to be made by combining the benefits of cloaking- and PIR-based techniques. Chor et al. [8] 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 [11], which is to further reduce the performance overhead of PIR techniques. The authors' own optimization of PIR in [13] (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 [23, 29]. Ghinita [11] 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.

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 [31].

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.

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 [32]. 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

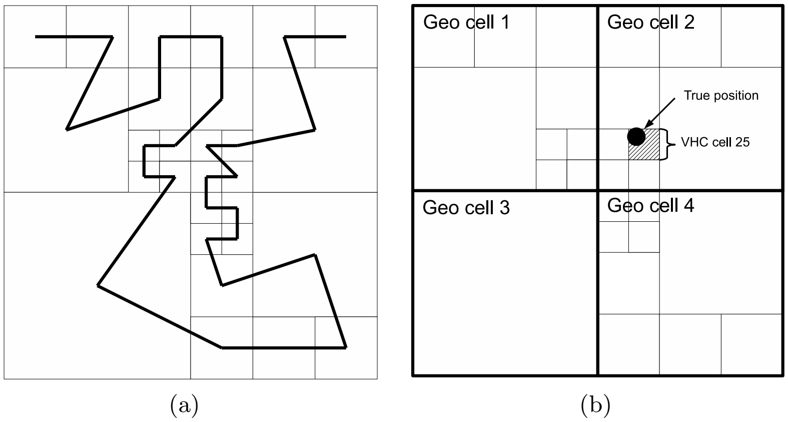


Fig. 1. (a) A Various-size-grid Hilbert Curve (VHC) mapping with uniform POI density. (b) A user's true position inside VHC cell 25 (shaded) and within a cloaking region bounded by the single geographical grid cell 2. The POI results for VHC cell 25 only will be returned in a query. If a larger cloaking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

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 [18], but it has the disadvantage (from its use in Casper [25]) of forming an unnecessarily large cloaking region which can impair performance [2]. Another technique is called VHC (Various-size-grid Hilbert Curve) mapping [28], 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.

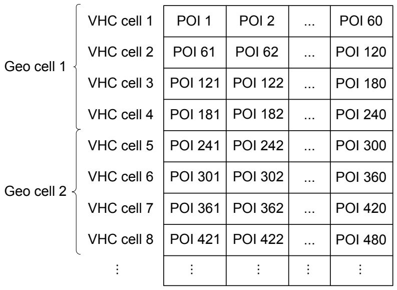
Now, we create a geographic grid overlaying the U.S. and Canada regions with fixed-size square cells that are 200 km in length (the area of each is 40,000 km^2). This corresponds to the maximum size of a single VHC cell as described above. Each geographic grid cell, however, may contain any number of smaller-sized 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) [21]. For instance, each grid cell may be two latitude degrees in length, which roughly



Geo cell 1	VHC cell 1	POI 1	POI 2	...	POI 60
	VHC cell 2	POI 61	POI 62	...	POI 120
	VHC cell 3	POI 121	POI 122	...	POI 180
	VHC cell 4	POI 181	POI 182	...	POI 240
Geo cell 2	VHC cell 5	POI 241	POI 242	...	POI 300
	VHC cell 6	POI 301	POI 302	...	POI 360
	VHC cell 7	POI 361	POI 362	...	POI 420
	VHC cell 8	POI 421	POI 422	...	POI 480
	⋮	⋮	⋮	⋮	⋮

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

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 [28], 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 extended version of this paper [27].

4 Experimental evaluation

4.1 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 C++ prototype is based on Percy++, an open source PIR protocol written in C++ [14, 15]. 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 [16, 17]. We note that a similar experiment in [13] considers a much smaller database; only 10,000 and 100,000 POIs. A head-to-head comparison with [13] 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 [13] 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 [30]. This SPIR protocol was derived from the oblivious transfer protocol by Naor and Pinkas [26] 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. Specifically, we ported the implementation from [30] to Google's Android smartphone platform, which supports the Java programming language. 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.

4.2 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 [15]. 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

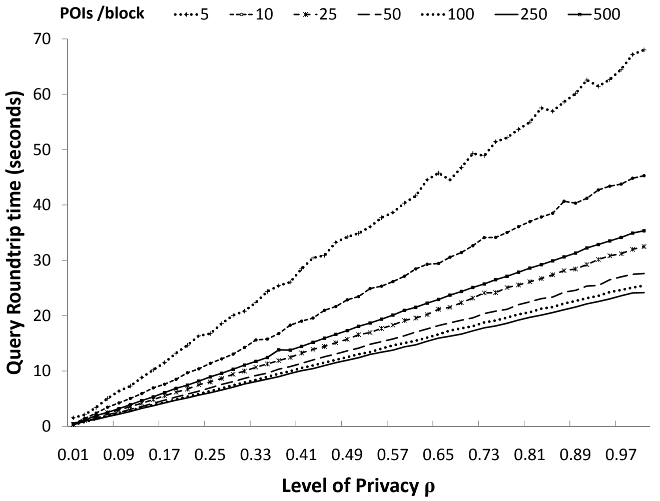


Fig. 3. Query roundtrip time and level of privacy for various numbers of POIs returned per query. A single measurement was taken per data point.

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 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. We observed that when the cloaking region was reduced to a quarter of its original size (i.e. a quarter of the POIs were returned), the query generation became 2.15 times slower, but the roundtrip time became 3.32 times quicker. 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. Further details of our implementation are available in [27].

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., the 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.

5 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.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments for improving this paper. We also gratefully acknowledge NSERC and MITACS for funding this research.

References

1. 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.
2. 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.
3. A. Beimel and Y. Stahl. Robust information-theoretic private information retrieval. *J. Cryptol.*, 20(3):295–321, 2007.
4. 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.
5. 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.
6. B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. Technical Report TR CS0917, Dept. of Computer Science, Technion, Israel, 1997.
7. 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.
8. B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. I. Goldberg. Percy++ project on SourceForge. http://percy.sourceforge.net/.
15. 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.
16. GPSmagazine. Garmin nuvi 780 GPS Review. http://gpsmagazine.com.
17. GPSreview.net. POI– Points of Interest. http://www.gpsreview.net/pois/.
18. 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.

19. U. Hengartner. Hiding location information from location-based services. In *Mobile Data Management, 2007 International Conference on*, pages 268–272, May 2007.
20. A. Iliev and S. W. Smith. Protecting Client Privacy with Trusted Computing at the Server. *IEEE Security and Privacy*, 3(2):20–28, 2005.
21. M. Kennedy and S. Kopp. *Understanding Map Projections*. ESRI (Environmental Systems Research Institute) press, 2000.
22. 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.
23. 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.
24. 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.
25. 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.
26. 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.
27. F. Olumofin, P. K. Tysowski, I. Goldberg, and U. Hengartner. Achieving Efficient Query Privacy for Location Based Services. Technical report, CACR 2009-22, University of Waterloo, 2009.
28. 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.
29. D. Riboni, L. Pareschi, and C. Bettini. Privacy in georeferenced context-aware services: A survey. In Bettini et al. [4].
30. 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.
31. R. Sion and B. Carbunar. On the computational practicality of private information retrieval. In *Proceedings of the Network and Distributed Systems Security Symposium*, 2007.
32. J. P. Snyder. *Flattening the Earth, two thousand years of map projections*. University of Chicago Press, 1993.
33. A. Solanas, J. Domingo-Ferrer, and A. Martínez-Ballesté. Location privacy in location-based services: Beyond TTP-based schemes. In Bettini et al. [4].
34. 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.
35. 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.