Efficient Location Privacy for Moving Clients in Database-driven Dynamic Spectrum Access

Erald Troja
The Graduate Center
City University of New York
Email: etroja@gc.cuny.edu

Spiridon Bakiras
John Jay College
City University of New York
Email: sbakiras@jjay.cuny.edu

***Abstract*—Dynamic spectrum access (DSA) is envisioned as a promising framework for addressing the spectrum shortage caused by the rapid growth of connected wireless devices. In contrast to the legacy fixed spectrum allocation policies, DSA allows license-exempt users to access the licensed spectrum bands when not in use by their respective owners. More specifically, in the database-driven DSA model, mobile users issue location-based queries to a white-space database, in order to identify *idle* channels in their area. To preserve location privacy, existing solutions suggest the use of private information retrieval (PIR) protocols when querying the database. Nevertheless, these methods are not communication efficient and fail to take into account user mobility. In this paper, we address these shortcomings and propose an efficient privacy-preserving protocol based on the Hilbert space filling curve. We provide optimizations for mobile users that require privacy on-the-fly and users that have full a priori knowledge of their trajectory. Through experimentation with two real life datasets, we show that, compared to the current state-of-the-art protocol, our methods reduce the query response time at the mobile clients by a large factor.**

I. Introduction

The allocation of radio spectrum for mobile wireless networking is governed by federal agencies via a static spectrum sharing strategy. However, with the ever growing need for mobile wireless services and applications, the static sharing method has led to the depletion of the available spectrum. To this end, Dynamic Spectrum Access (DSA) allows users to access licensed spectrum bands when not in use by their respective owners. DSA is built on top of Cognitive Radio (CR), an intelligent wireless communications system that is aware of its spectral environment [16]. Initially, DSA relied mostly on distributed and cooperative sensing. In this approach, CR nodes employ sheer power detection methods and collectively measure spectrum activity in their surrounding area.

Alternatively, in database-driven dynamic spectrum access, a CR node understands its spectral surroundings in a three-step process. A node attempting to analyze its spectral surrounding would first learn its geographic location through a GPS device. Subsequently, it would contact a centralized white-space database (WSDB) and issue its GPS coordinates as part of the query. Finally, it would download the centrally fused repository report containing the available spectrum at that location. The compilation and fusion of the spectrum reports is done by specialized entities, called Spectrum Database Operators (SDOs), by applying appropriate propagation modeling and interference avoidance algorithms for a given geographic location.

Nevertheless, the database-driven DSA approach is prone to severe location privacy leakage. According to FCC specifications [3], a mobile Television Band Device (TVBD) must issue a new query whenever it moves further than 100m from its previous location. Since the GPS coordinates must be part of every query, a WSDB operator could easily build a detailed history of the mobile TVBD's trajectories, which could reveal sensitive information about the underlying user (such as health condition, habits, etc.).

To this end, Gao et al. [5] introduce a scheme that leverages a private information retrieval (PIR) protocol to query the WSDB in a privacy-preserving manner. A PIR protocol allows any user to retrieve a record from a database server, while maintaining the identity of the record secret from the server. Therefore, Gao et al. partitions the space with a fixed $n \times n$ grid and require users to download the location-dependent (based on the cell where they are located) channel information, through the PIR protocol. This is the only protocol so far in the literature dealing with location privacy in database-driven DSA but, unfortunately, it suffers from several drawbacks.

First, Gao et al. utilize the PIR scheme of Trostle and Parrish [20] whose communication cost (for a single query) is significant. More precisely the scheme incurs a communication cost of $(2n + 3) \cdot \log\ p$ bits, where p is a 2048-bit modulus. For instance, if $n = 5000$, the amount of data exchanged to retrieve the bitmap of a single cell is 2.5 MB. For highly mobile clients, the cost of this approach can exceed the cost of downloading the entire database. Second, most PIR protocols typically return multiple records per query that, in the case of mobile users, could be used to answer future queries. However, the authors modify [20] so that the PIR reply contains channel availability information for a *single* cell (as opposed to n in the original protocol). Finally, they view each query as an independent event, without taking into account user mobility. As a result, when a user is constantly moving, the communication cost can surpass the cost of downloading the entire database.

In this paper, we first argue that dynamic spectrum access will most likely be utilized in areas with poor/intermittent cellular connectivity. As such, the underlying query processing protocol should be communication efficient. Therefore, unlike [5], our methods leverage the PIR scheme of Gentry and Ramzan [6], which is the most communication efficient protocol to date. Furthermore, to address user mobility, we index the WSDB based on the Hilbert space filling curve (HSFC) [10]. In this way, neighboring cells are typically stored in consecutive locations on the white-space database. Finally, to allow for the retrieval of multiple cells with a single PIR query, we split the WSDB into multiple, disjoint segments. As such, a PIR query is processed independently on each segment, and the user retrieves channel availability information from a large number of consecutive cells IDs. Due to the properties of the underlying HSFC, these cells will be spatially close (with a very high probability), and could reduce the number of PIR queries in the near future.

We consider two distinct cases in our work: (i) the user's trajectory is known a priori and (ii) the user's trajectory is generated on-the-fly. For the latter case, we propose a trajectory prediction method, based on a simple linear regression model of the recently traveled coordinates. The predicted values are then used to retrieve the corresponding cells from the WSDB and, thus, reduce further the number of future PIR queries. In the case of the a priori trajectory knowledge, our approach enables mobile users to simulate their routes and invoke the optimal number of PIR queries. We tested our methods on two real life datasets, namely Microsoft's T-Drive dataset [21] and Microsoft's GeoLife GPS dataset [22]. The experimental results show that, compared to the protocol of Gao et al. [5], our methods reduce the query response time at the mobile clients by at least a factor of 30.

The remainder of this paper is organized as follows. Section II presents a literature review on location privacy. Section III provides the necessary background on PIR protocols and Hilbert space filling curves. Section IV-B describes our methods in detail, and Section V presents the results of the experimental evaluation. We conclude our work in Section VI.

II. Related Work

Most existing approaches for location privacy rely on the notion of k-anonymity [19] or l-diversity [15]. In location-based services, a spatial query is said to be k-anonymous, if it is indistinguishable from at least $k-1$ other queries originating from the same region. This region is called a spatial cloaking region (SCR), and encloses the querying user as well as at least $k-1$ other users. To compute the SCR, existing k-anonymity algorithms typically extend the SCR around the query point until it encloses $k-1$ other users [9].

l-diversity based methods [15], on the other hand, extend the SCR until $l-1$ different locations are included. Although k-anonymity and l-diversity provide some degree of location privacy, they may still leak semantic location information. For example, if the SCR only contains casinos, the server can infer that the mobile user is interested in gambling. To this end, the work of Lee et al. [12] attempts to provide location privacy using location semantics.

The k-anonymity and l-diversity based approaches, as well as collaborative location privacy protection methods [4], often rely on third party trusted anonymizers, which is not always a viable solution. On the other hand, Ghinita et al. [7] propose the first privacy-preserving protocol (for nearest neighbor queries) that does not require a trusted third party. Instead, their method achieves perfect location privacy via the cryptographic primitive of private information retrieval [11].

Location privacy work in the DSA realm has mainly focused on the collaborative spectrum sensing aspect. In particular, most of the existing algorithms aim towards securing the location privacy of secondary users that submit sensing reports to a malicious fusion center [13], [18].

Location privacy research in database-driven DSA networks is still in its early stages. The state-of-the-art protocol is due to Gao et al. [5], which builds upon a modified version of Trostle and Parrish's PIR scheme [20]. They assume a fixed grid of $n \times n$ cells, where each cell contains a bitmap that represents the channel availability information (typically 32 bits). The authors modify [20] so that the PIR reply contains channel availability only for a *single* cell. Furthermore, each query is seen independently, without any regard to user mobility. As a result, for highly mobile clients, the communication cost in [5] can surpass the cost of downloading the entire database.

III. Preliminaries

In this section we give a brief description of PIR protocols and Hilbert space filling curves. Section III-A introduces the concept of private information retrieval and Section III-B describes the threat model of our methods. Section III-C presents the Hilbert space filling curve algorithm.

A. Private Information Retrieval

PIR protocols allow a user to obtain information from a database server, in a manner that prevents the database from knowing which data was retrieved. Typically, the server holds a database of N records and the user wants to retrieve the i-th record, such that i remains unknown to the database. The trivial PIR case consists of downloading the entire database, which clearly preserves privacy but has an unrealistic communication cost. Therefore, the objective of a PIR protocol, as applied to mobile applications, is to reduce the communication cost.

Information theoretic PIR protocols [2] are secure against computationally unbounded adversaries. However, they require that the database be replicated into multiple non-colluding servers. This non-collusion assumption is not realistic in typical applications, so information theoretic protocols are not utilized in practice. On the other hand, *computational* PIR (CPIR) protocols base their security on well-known cryptographic problems that are hard to solve (such as discrete logarithm or factorization). As such, their security is established against computationally bounded adversaries. Kushilevitz and Ostrovsky [11] introduced the first CPIR protocol for a single database, whose security is based on the quadratic residuosity assumption. The communication complexity of [11] is $O(n^{\epsilon})$. Further work [1], [14] demonstrates CPIR schemes with polylogarithmic communication complexity.

In this work, we leverage the protocol of Gentry and Ramzan [6], because it is the most communication efficient PIR protocol to date. For a particular instantiation, it exhibits a *constant* communication cost that is independent of the database size (it typically involves the exchange of three 128-byte numbers). The security of the protocol is based on "ϕ-hiding" assumption.

B. Threat Model and Security

In this work we are concerned with privacy against the WSDB operator (adversary). We assume that the adversary's goal is to derive any relevant information regarding the location of any user that has sent a query to the database. We also assume that the adversary runs in polynomial time and follows the *honest-but-curious* adversarial model, i.e., it follows the protocol correctly but tries to gain an advantage by examining the communication transcript. Note that our methods inherit the security of the underlying PIR protocol, since the only interaction between the WSDB operator and the users is through a series of PIR invocations.

C. Hilbert Space Filling Curve

The Hilbert space filling curve [10] is a continuous fractal that maps space from 2-D to 1-D. If (x, y) are the coordinates of a point within the unit square and d is the distance along the curve when it reaches that point, then points with nearby d values will also be spatially close. As an example, Fig. 1 shows a HSFC of level $l = 3$, containing $4^l = 64$ cells. Each of the cells is identified by its (x, y) coordinates, starting with $(0, 0)$ on the lower left hand corner and ending with $(x = 2^l - 1, y = 2^l - 1)$ on the right upper hand corner. The values shown in the individual cells correspond to their Hilbert IDs (HIDs), i.e., their specific order within that mapping.

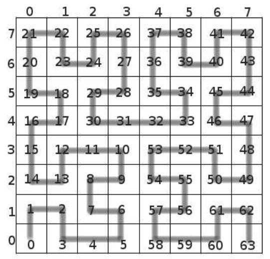


Fig. 1. A level 3 Hilbert space filling curve.

IV. Efficient Location Privacy for Moving DSA Clients

In this section, we present the details of our methods. Section IV-A describes the system architecture, and Section IV-B introduces our basic approach. Section IV-C presents an enhanced method that retrieves multiple cells from the area surrounding the query point. Section IV-D introduces our best algorithm that incorporates trajectory prediction, and Section IV-E describes the case where there is full a priori trajectory knowledge.

A. System Architecture

Similar to previous work [5], we assume a fixed grid of $n \times n$ cells. According to the FCC specifications [3], each cell is 100m$\times$100m in size, and users must query the WSDB whenever they move into a cell with no prior spectrum availability knowledge. The dimensions of the grid (i.e., n) can be made arbitrarily large, which has a direct effect on the database size. Mobile TVBDs are allowed to communicate only in the frequency ranges 512-608 MHz (TV channels 21-36) and 614-698 MHz (TV channels 38-51), i.e., there are a total of 31 possible white-space TV band channels that can be accessed in a DSA manner. Therefore, we represent the daily channel availability as 32 bits (per cell), where bit 0 represents a busy channel and bit 1 represents an idle channel.

B. Single Row Retrieval

Papadopoulos et al. [17] conducted a in-depth study of the PIR protocol by Gentry and Ramzan [6] that we employ in our methods. As they point out, due to the security constraints of the algorithm, the optimal strategy in terms of communication and computational cost is to set the size of each record to 32 bytes. Therefore, based on our system settings, each record can store channel availability information from 8 distinct cells. A straightforward implementation would then be to (i) sort the cells based on their unique Hilbert IDs, and (ii) create a single database (DB) with $N = \lceil n^2/8 \rceil$ records, such that record 0 stores cells 0–7, record 1 stores cells 8–15, etc. (Table I summarizes the symbols used in the remainder of this paper.) In the toy example of Fig. 1, we would have a database of $N = 8$ records, and a user located inside cell 30 would retrieve the record containing cells 24–31. Observe that, due to the properties of the HFSC, all the retrieved cells are spatially close and could be useful in subsequent queries.

TABLE I
Summary of Symbols

Symbol	Description
n	Number of rows/columns in the grid
k	Number of DB segments
N	Number of records in each DB segment ($N = \lceil n^2/8k \rceil$)
u	Number of records retrieved from each DB segment
$\log m$	Size of PIR request/reply (Gentry-Ramzan)
R	Number of rings to explore in the surrounding area

Nevertheless, the single DB approach would not work well in practice. First, it is beneficial for a client to retrieve a large number of cells that are in proximity to his current location, in order to reduce the number of future PIR queries. Second, Gentry and Ramzan's protocol is computationally expensive (at the server side), due to its heavy use of cryptographic operations. As such, we would like to parallelize its operation, to the extent possible, by utilizing large CPU clusters that are typical in most cloud computing platforms.

The obvious solution to both limitations is to partition the database into k distinct segments. By doing so, we can employ k CPUs to process each segment in parallel, thus reducing the computational time by a factor of k. The price we have to pay is an increase in the communication cost, since the client receives k PIR replies instead of one. Specifically, the communication cost is equal to $(2 + k) \cdot \log m$, where m is an RSA modulus. Table II shows a sample DB segmentation (for $k = 4$) for a level 4 HFSC, containing 256 cells. The segments are constructed by assigning the original records to each segment in a round-robin manner.

TABLE II
SAMPLE DB SEGMENTATION WITH 4 SEGMENTS

DB segment 0	DB segment 1	DB segment 2	DB segment 3
0–7	8–15	16–23	24–31
32–39	40–47	48–55	56–63
64–71	72–79	80–87	88–95
96–103	104–111	112–119	120–127
128–135	136–143	144–151	152–159
160–167	168–175	176–183	184–191
192–199	200–207	208–215	216–223
224–231	232–239	240–247	248–255

During query processing, the client first identifies the row r that contains his current cell's HID ($r = HID/8k$). He then constructs the corresponding PIR query that is processed on all k DB segments, in parallel. In the example of Table II, if the client is located in cell 180, he will retrieve all cells in row $r = 180/32 = 5$, i.e., all cells within the range 160–191. The results are stored in the client's cache and may be utilized when the client moves into a new cell. Algorithm 1 lists the detailed algorithm for the single row retrieval method.

Algorithm 1 Single Row Retrieval

```
1: procedure SINGLE-ROW-RETRIEVAL($HID, k$)
2:
3:     if ($HID \notin cache$) then
4:         $r \leftarrow HID/8k$;
5:         $cells[\ ] \leftarrow PIR(r)$;
6:         $cache \leftarrow cells[\ ]$;
7:     end if
8:
9: end procedure
```

C. Exploring the Surrounding Area

When a mobile user's trajectory is generated on-the-fly, i.e., without any prior planning, retrieving a single row per PIR query is not the optimal strategy. Consider, for example, a user that issues a PIR query from the cell marked with a white dot in Fig. 2. The numbered boxes in this figure indicate the cells that comprise the corresponding database rows. According to that figure, the user first retrieves row 6 and then moves to the next location that is part of row 8 (the black dots show the remaining trajectory points). He now has to send a new query to the WSDB and all the information contained in row 6 is rendered useless.

A second drawback of the single row retrieval approach, is the structure of the Hilbert curve itself. As evident in Fig. 1, a cell's nearest neighbors are not always mapped on consecutive Hilbert IDs. For instance, cells 5 and 58 are direct neighbors on the grid, but their Hilbert IDs are very far apart. These inconsistencies are common in all space filling curves, and are more severe on higher level curves (which is typically the case in real life applications).

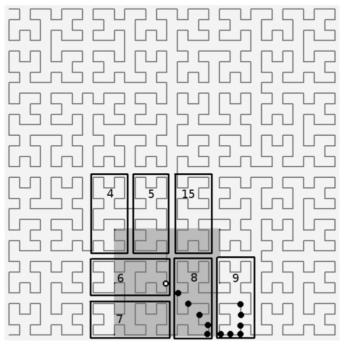


Fig. 2. Exploring the surrounding area with 2 and 4 sub-segments.

To address these shortcomings, we take advantage of Gentry and Ramzan's multi-record retrieval feature, as described by Groth et al. [8]. Specifically, given a database segment containing N 32-byte records, we partition the segment into u sub-segments, each storing N $(32/u)$-byte records. By doing so, it is possible to retrieve u records from each sub-segment, while keeping the computational cost unchanged. Therefore, by sacrificing some communication cost, we can retrieve more relevant results with a single query. Note that, the communication cost in the multi-record retrieval scheme is $(2 + k \cdot u) \cdot \log m$.

As a first step towards improving our basic scheme, we require the user to explore the area surrounding his current location, and retrieve the database rows that maximize the coverage of that area. The intuition is that, if the user has no prior knowledge of his trajectory, we should anticipate his movement towards any possible direction. Algorithm 2 illustrates the functionality of this approach. We define as R the number of rings surrounding the user's current cell that we want to explore.

Algorithm 2 Surrounding Area

```
1: procedure SURROUNDING-AREA($HID, k, u, R$)
2:
3:     $count[N] \leftarrow \{0\}$;
4:
5:     if ($HID \notin cache$) then
6:         $r \leftarrow HID/8k$;
7:         insert $r$ into $rows[\ ]$;
8:         for each cell $i$ in the area defined by $R$ do
9:             $r' \leftarrow hid(i)/8k$;
10:            if ($r' \neq r$ and $hid(i) \notin cache$) then
11:                $count[r']++$;
12:            end if
13:        end for
14:        find the top $(u-1)$ values in array $count$;
15:        insert their indexes into $rows[\ ]$;
16:        $cells[\ ] \leftarrow PIR(rows[\ ])$;
17:        $cache \leftarrow cells[\ ]$;
18:    end if
19:
20: end procedure
```

The algorithm maintains an array $rows$, which stores the row numbers that should be retrieved from the database. The first row is always the one containing the user's current cell (lines 6–7). Next, the algorithm iterates over all cells within the area defined by R, and counts how many times the underlying row numbers appear in the result (lines 8–13). Finally, it selects the $(u-1)$ most frequent row numbers and adds them into $rows$ (lines 14–15). The cells from all u rows are then retrieved via the PIR query and are eventually cached at the client. In the example of Fig. 2, when $u = 2$ we retrieve rows 6 and 8. On the other hand, when $u = 4$ we retrieve rows 6, 8, 7, and 15.

D. Trajectory Prediction

Even if a mobile user is unaware of his exact trajectory, he is very likely to occasionally follow a specific direction (e.g., south-east) for a sufficiently large period of time. Therefore, in our next method, we explore the feasibility of employing a trajectory prediction algorithm, in order to maximize the amount of useful information retrieved from a PIR query. To this end, we assume that the client maintains a cache v of his most recent GPS measurements that are taken at regular time intervals.

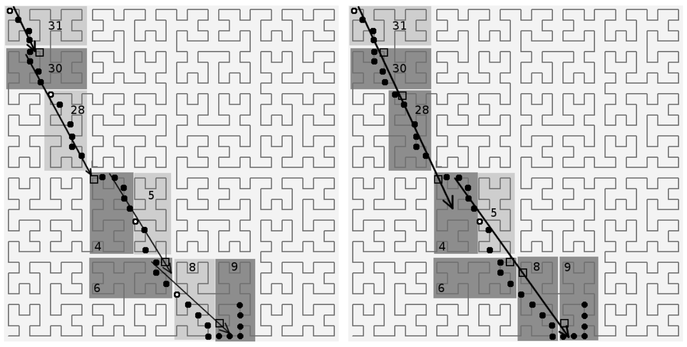
Algorithm 3 shows the detailed steps of this approach. As in our previous method, we retrieve a total of u rows, where the first row is always the one containing the user's current cell. Next, the client applies a simple linear regression (SLR) model on the vector v of GPS measurements, and computes a straight line l that predicts the following trajectory points (line 6). This line is then extended forward, until it encounters $(u-1)$ additional cells whose underlying rows are not present in the cache. The row numbers of these cells are also added to the PIR query (lines 7–11).

Algorithm 3 Trajectory Prediction

```
1: procedure TRAJECTORY-PREDICTION($HID, k, u, v$)
2:
3:     if ($HID \notin cache$) then
4:         $r \leftarrow HID/8k$;
5:         insert $r$ into $rows$[ ];
6:         $l \leftarrow SLR(v)$;
7:         for $i = 1$ to $(u-1)$ do
8:             extend $l$ until you find cell $j$: $hid(j) \notin cache$;
9:             $r \leftarrow hid(j)/8k$;
10:            insert $r$ into $rows$[ ];
11:        end for
12:        $cells$[ ] $\leftarrow PIR(rows$[ ]$)$;
13:        $cache \leftarrow cells$[ ];
14:    end if
15:
16: end procedure
```

Fig. 3 illustrates an example of the prediction algorithm for $u = 2$ and $u = 4$. The dots in these figures represent the user's trajectory (starting from the upper left corner), and the shaded boxes represent the rows retrieved from the WSDB. When $u = 2$ (Fig. 3a), the first PIR query is constructed by extending the predicted line, until it encounters the cell marked with the hollow square. As a result, the first PIR query retrieves rows 31 and 30. When the user enters row 28, a new query is issued for rows 28 and 4. This process repeats and the user issues a total of four PIR queries, represented by the white dots in Fig. 3a.

On the other hand, when $u = 4$ (Fig. 3b) the client is able to prefetch more results from the predicted trajectory, thus resulting in just two PIR queries for the entire trajectory. The first query retrieves rows 31, 30, 28, and 4, while the second one retrieves rows 5, 6, 8, and 9. Note that, reducing the number of PIR queries is very important, as they incur a high computational cost at the WSDB. Regarding the communication cost in our example, the 2 sub-segment case requires a total of $40 \cdot \log m$ bits, while the 4 sub-segment case requires $36 \cdot \log m$ bits. In other words, for approximately the same communication cost, we were able to reduce the computational cost at the WSDB by 50% (4 vs. 2 PIR queries).



(a) (b)

Fig. 3. Trajectory prediction example. (a) Using 2 sub-segments. (b) Using 4 sub-segments.

E. A Priori Trajectory Knowledge

Our last method deals with mobile users that have full a priori knowledge of their trajectories. This is not an unrealistic assumption, since that feature is common in GPS navigation systems. In this scenario, users are allowed to choose the trajectory starting and ending points, and then control the route connecting the two end points. Knowing the exact trajectory enables us to simulate the route on the underlying grid, and identify the cells that intersect with that route. Algorithm 4 depicts that simulation. It simply initializes an empty hash table HT, and inserts therein the row numbers of all cells that intersect trajectory T. Note that, this method is guaranteed to invoke the optimal number of PIR queries.

Algorithm 4 A Priori Trajectory Simulation

```
1: procedure A-PRIORI-TRAJECTORY-SIMULATION($T, k$)
2:
3:     $HT \leftarrow \emptyset$;
4:
5:     for each cell $i$ intersecting trajectory $T$ do
6:         $r \leftarrow hid(i)/8k$;
7:         insert $r$ into $HT$;
8:     end for
9:
10: end procedure
```

Once the algorithm computes the final hash table, the client has two options regarding query processing. The first one is to issue $|HT|/u$ PIR queries to the WSDB and retrieve all the necessary results beforehand. The second option is to issue the queries "on-demand." That is, when the client moves into a cell without any channel availability information, he retrieves the row of that cell as well as $(u-1)$ other rows from the hash table (it could be the ones that are spatially close to the query point).

V. Experimental Evaluation

In this section, we evaluate experimentally the performance of our proposed methods. Section V-A describes the setup of our experiments, and Section V-B provides the detailed results.

A. Experimental Setup

We developed our experiments in Java SDK, running on Ubuntu 14.04.1 LTS. For the experimental tests, we utilized two real life datasets, namely Microsoft's GeoLife GPS Trajectories[1] and Microsoft's T-Drive GPS Dataset[2]. Both are excellent datasets, containing real life trajectories from users traveling around Beijing, China.

The T-Drive dataset [21] contains GPS trajectories from 10,357 taxis, during the period of Feb. 2 to Feb. 8, 2008. The average sampling interval is about 177 seconds, with a distance of about 623 meters. The GeoLife GPS dataset monitors 182 users for a period of over five years (from Apr. 2007 to Aug. 2012), and 91.5 percent of the trajectories are logged in a dense representation, e.g., every 1–5 seconds or every 5–10 meters per point.

In our experiments, we set a bounding box of 409.6km $\times$ 409.6km (thus setting $n = 4096$) around Beijing's coordinates, which are 39.9139°N, 116.3917°E. The bounding box's coordinates are set as $minlat = 37.7$, $maxlat = 41.5$, $minlong = 114.1$, and $maxlong = 118.9$. From all the available trajectories, we compiled a list of the longest ones that are completely contained within the bounding box. In particular, we selected 9727 trajectories from the T-Drive dataset, and 2188 trajectories from the GeoLife dataset.

As performance metric, we measure the average *cumulative* query response time from all PIR queries that are issued to the WSDB throughout the duration of a mobile user's trajectory. This cost includes (i) the query generation time at the client, (ii) the processing time at the server, (iii) the network transfer time, and (iv) the result extraction time at the client. To provide realistic results, we implemented the underlying PIR protocols ([6] and [20]) using the GMP[3] multiple precision arithmetic library. Table III shows the detailed costs. The client-side computations are performed on an iPhone 5 device running iOS 7.1, while the server-side computations are performed on a 3.5 GHz Intel Core i7 processor.

For Trostle and Parrish's scheme we set the modulus size equal to 2048 bits, as described in [5]. Note that, the query generation cost at the client can be avoided, since it involves the computation of n random values that are independent

[1]http://research.microsoft.com/en-us/projects/GeoLife/
[2]http://research.microsoft.com/en-us/projects/tdrive/
[3]http://gmplib.org

TABLE III
Cost of PIR operations

Cost	GR [6]	TP [20]
Query generation (client)	450ms	Pre-processing
Server processing (64 CPUs)	4560ms	18ms
Server processing (128 CPUs)	2280ms	9ms
Result extraction (client)	125ms	0.5ms
Communication cost	20800 bytes	2121728 bytes

of the queried row. As a result, these values can be pre-computed offline. For Gentry and Ramzan's protocol we set the modulus size m equal to 1280 bits, in order to satisfy the security requirement of the protocol. Also, the query generation algorithm for Gentry and Ramzan depends on the queried row(s) and should be computed online. The values shown in the table above correspond to the single row retrieval method, where $k = 128$ and $u = 1$.

Fig. 4 shows the query response time for the two PIR protocols (based on Table III) as a function of the cellular bandwidth available at the mobile client. Clearly, the cost of Trostle and Parrish's scheme is dominated by the network transfer time, since each PIR query necessitates the exchange of over 2 MB of data. On the other hand, Gentry and Ramzan's protocol is practically independent of the available bandwidth, and its cost is determined solely by the computing power at the WSDB (64 vs. 128 CPUs). Nevertheless, as we mentioned earlier, the primary deployment targets for database-driven DSA are areas with scarce cellular bandwidth, making Gentry and Ramzan's protocol a better choice as the underlying PIR mechanism.

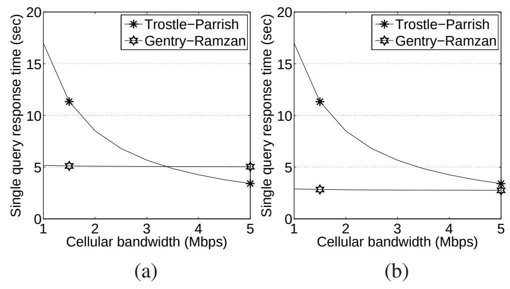
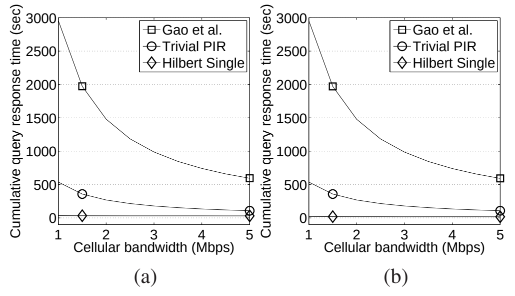


Fig. 4. Response time for a single PIR query. (a) 64 CPUs (b) 128 CPUs

Note that, another option for achieving location privacy is through the *trivial* PIR case, i.e., by downloading the entire spectrum WSDB with one query. However, this is only viable when the database size is small or when there is ample bandwidth to do so. In our experiments, the database size is over 67 MB, which takes around 536 seconds to download at 1 Mbps, and 108 seconds at 5 Mbps.

B. Experimental Results

In the first experiment we investigate the performance of the single row retrieval method ($k = 128$, $u = 1$), as explained in Section IV-B. Fig. 5 depicts the average cumulative query response time as a function of the available bandwidth. The curve labeled "Gao et al." corresponds to an improved version of the original protocol [5] that incorporates Trostle and Parrish's unmodified scheme, which retrieves one row with a single query. (Note that, the two variants have practically identical performance in terms of both computation and communication cost.) Clearly, Gao et al. is not designed for moving clients, and averages 174 PIR queries (per trajectory) for the GeoLife dataset, and 135 queries for the T-Drive dataset. As a result, the overall cost of Gao et al. exceeds the cost of the trivial PIR case by a wide margin and we will, thus, omit it from further comparisons in our experiments.



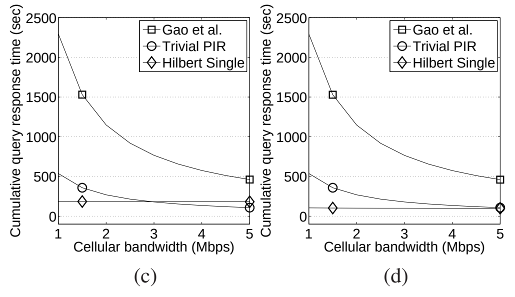
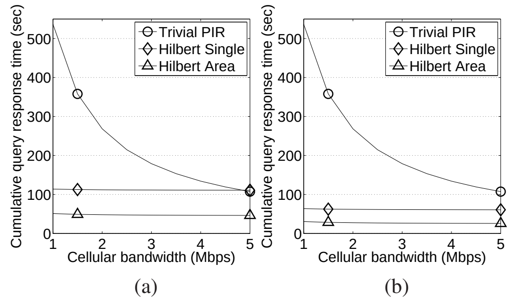


Fig. 5. Response time for the single row retrieval method. (a) 64 CPUs (GeoLife) (b) 128 CPUs (GeoLife) (c) 64 CPUs (T-Drive) (d) 128 CPUs (T-Drive)

Our single row retrieval method outperforms the trivial PIR case for the GeoLife dataset, and is marginally worse for the T-Drive dataset (for 64 CPUs) when there is adequate download bandwidth. The difference in performance across the two datasets is explained by the structure of the underlying trajectories. Recall that the data points in the T-Drive dataset are recorded at sparse distances (average 623m). The sparseness of the data points mimics well the requirements of a "paging" application, where ubiquitous connectivity is not a requirement. In this scenario, prefetching results from the surrounding area is not always beneficial, since the user may issue the next query from an entirely different area. On the other hand, the data points in the GeoLife dataset are very dense so, with a high probability, several consecutive queries may be issued within a small area.

In the remainder of this section, we investigate the performance of our multi-record retrieval protocols for the case of $k = 128$ and $u = 4$. We begin by evaluating the surrounding area method, which was explained in Section IV-C (we set $R = 50$ rings as the explored area). Fig. 6 illustrates the cumulative query response time as a function of the available bandwidth for the two datasets. It is evident that our method outperforms considerably the trivial PIR case in almost all settings. Compared to the single row retrieval method (which is also included in the figure for clarity), the surrounding area approach decreases the overall query cost by 57%, on average, for the GeoLife dataset, and 47% for the T-Drive dataset.



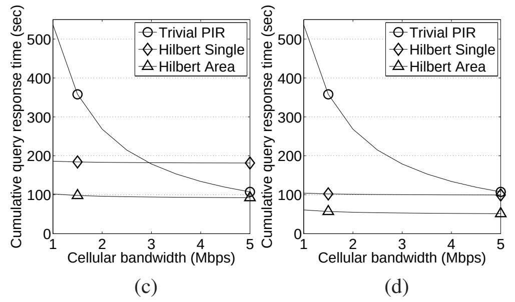


Fig. 6. Response time for the surrounding area method. (a) 64 CPUs (GeoLife) (b) 128 CPUs (GeoLife) (c) 64 CPUs (T-Drive) (d) 128 CPUs (T-Drive)

Next, we evaluate the performance of our trajectory prediction method, as described in Section IV-D. Fig. 7 shows the cumulative query response time for the three methods under various settings. The trajectory prediction approach is clearly superior to the trivial PIR case under all settings. Utilizing 128 compute units at the server results in a response time of just 18 sec (for the whole trajectory) in the GeoLife dataset and 42 sec in the T-Drive dataset. In addition, the trajectory prediction algorithm decreases the query processing cost even further compared to the surrounding area method. Specifically, it reduces the cost by an additional 34% in the GeoLife dataset, and 28% in the T-Drive dataset. This is due to the fact that, with trajectory prediction, we prefetch results according to a specific direction of movement instead of a generic rectangular area.

In our last experiment, we investigate the performance of the a priori trajectory knowledge approach, which was explained in Section IV-E. Recall that, this method is optimal in terms of PIR requests, since the client avoids the retrieval of any unnecessary rows from the WSDB. Fig. 8 illustrates the corresponding cumulative query response times. Our method outperforms the trivial PIR approach by a large factor, and entails a cost of just 12 sec in the GeoLife dataset and 23 sec in the T-Drive dataset (for 128 compute units). Compared to the trajectory prediction method, the a priori trajectory knowledge enables us to reduce the query processing cost by an additional 33% in the GeoLife dataset and 43% in the T-Drive dataset.

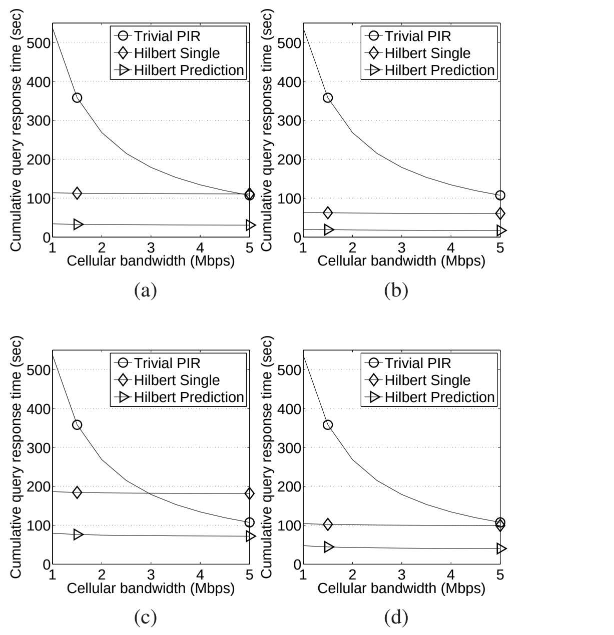


Fig. 7. Response time for the trajectory prediction method. (a) 64 CPUs (GeoLife) (b) 128 CPUs (GeoLife) (c) 64 CPUs (T-Drive) (d) 128 CPUs (T-Drive)

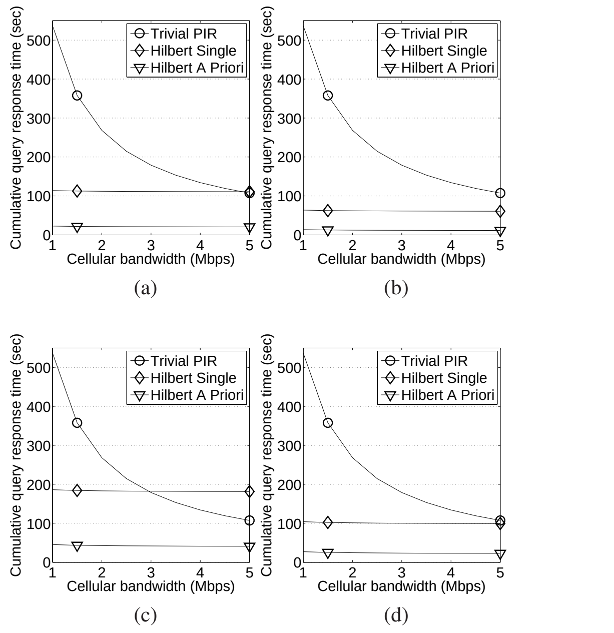


Fig. 8. Response time for the a priori trajectory knowledge method. (a) 64 CPUs (GeoLife) (b) 128 CPUs (GeoLife) (c) 64 CPUs (T-Drive) (d) 128 CPUs (T-Drive)

VI. Conclusions

Existing methods for location privacy in the database-driven DSA model are very inefficient, because they are not optimized for mobile clients. To this end, our work introduces an efficient solution, based on a Hilbert space filling curve indexing of the white-space database. Our methods leverage a communication-efficient PIR protocol, and employ trajectory prediction algorithms to minimize the number of PIR queries. Through extensive experimentation with real life datasets, we show that, compared to the current state-of-the-art protocol, our methods reduce the query response time at the mobile clients by a large factor.

Acknowledgments

This research has been funded by the NSF CAREER Award IIS-0845262.

References

[1] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, pages 402–414, 1999.

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

[3] FCC. Television band devices. page 11, 2013.

[4] J. Freudiger, M. H. Manshaei, J.-P. Hubaux, and D. C. Parkes. On non-cooperative location privacy: A game-theoretic analysis. In *ACM CCS*, pages 324–337, 2009.

[5] Z. Gao, H. Zhu, Y. Liu, M. Li, and Z. Cao. Location privacy in database-driven cognitive radio networks: Attacks and countermeasures. In *IEEE INFOCOM*, pages 2751–2759, 2013.

[6] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *ICALP*, pages 803–815. 2005.

[7] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: Anonymizers are not necessary. In *ACM SIGMOD*, pages 121–132, 2008.

[8] J. Groth, A. Kiayias, and H. Lipmaa. Multi-query computationally-private information retrieval with constant communication rate. In *PKC*, pages 107–123. 2010.

[9] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *ACM MobiSys*, pages 31–42, 2003.

[10] I. Kamel and C. Faloutsos. On packing R-trees. In *ACM CIKM*, pages 490–499, 1993.

[11] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *IEEE FOCS*, pages 364–373, 1997.

[12] B. Lee, J. Oh, H. Yu, and J. Kim. Protecting location privacy using location semantics. In *ACM SIGKDD*, pages 1289–1297, 2011.

[13] S. Li, H. Zhu, Z. Gao, X. Guan, K. Xing, and X. Shen. Location privacy preservation in collaborative spectrum sensing. In *IEEE INFOCOM*, pages 729–737, 2012.

[14] H. Lipmaa. An oblivious transfer protocol with log-squared communication. In *Information Security*, pages 314–328. 2005.

[15] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1), 2007.

[16] J. Mitola III. Cognitive radio: An integrated agent architecture for software defined radio. *Doctoral Dissertation, KTH, Stockholm, Sweden*, May 2000.

[17] S. Papadopoulos, S. Bakiras, and D. Papadias. pCloud: A distributed system for practical PIR. *IEEE Transactions on Dependable and Secure Computing*, 9(1):115–127, 2012.

[18] J. Shi, R. Zhang, Y. Liu, and Y. Zhang. Prisense: privacy-preserving data aggregation in people-centric urban sensing systems. In *IEEE INFOCOM*, pages 1–9, 2010.

[19] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[20] J. Trostle and A. Parrish. Efficient computationally private information retrieval from anonymity or trapdoor groups. In *Information Security*, pages 114–128. 2011.

[21] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *ACM GIS*, pages 99–108, 2010.

[22] Y. Zheng, L. Wang, R. Zhang, X. Xie, and W.-Y. Ma. GeoLife: Managing and understanding your past life over maps. In *IEEE MDM*, pages 211–212, 2008.