Ad Hoc Networks 59 (2017) 71–85

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Optimizing privacy-preserving DSA for mobile clients☆

Erald Troja[a], Spiridon Bakiras[b,*]

[a] Department of Computer Science, The Graduate Center CUNY, 365 Fifth Avenue New York, NY 10016, USA
[b] College of Science and Engineering, Hamad bin Khalifa University, Doha, Qatar

ARTICLE INFO

Article history:
Received 1 February 2016
Revised 11 January 2017
Accepted 10 February 2017
Available online 12 February 2017

Keywords:
Location privacy
White-space database
Dynamic spectrum access
Private information retrieval
Variable order markov model
Trajectory prediction

ABSTRACT

The exponential growth of connected wireless devices has led to a depletion of the available wireless spectrum. To this end, dynamic spectrum access (DSA) has been proposed as a viable framework for maximizing the usability of the wireless spectrum by allowing some portions of it to be accessed and used in a dynamic manner. Contrary to the legacy fixed spectrum access policy, DSA enables license-exempt users to access licensed bands during their respective owner's idle times. Specifically, in the database-driven DSA model, mobile users issue location-based queries to a white-space database and request *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. Results from our experimentation on two real life datasets show that, compared to the current state-of-the-art protocol, our methods reduce the query processing cost at the mobile clients by a factor of 2 to 8.

1. Introduction

The allocation of radio spectrum for mobile wireless networking is governed by federal agencies via a fixed (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 [1]. Furthermore, the actual usage of pre-assigned spectrum bands has been measured to have a very low average utilization. For example, in the US, the federal communications commission (FCC) has reported that many spectrum bands allocated via static assignment policies have been used only in bounded geographical areas and over very limited periods of time. Such utilization has been measured to be between 15% and 85% [2].

Currently, there is wide consensus that the static method of spectrum allocation has major drawbacks. As a result, the need for opportunistic and dynamic spectrum access technologies has risen sharply. A flexible and dynamic spectrum access strategy is necessary, in order to eliminate the underutilization and spectrum depletion effects of the current static allocation scheme. The FCC has stated that no other technology "holds greater potential for literally transforming the use of spectrum in the years to come than the development of software-defined and cognitive/smart radios" [3].

Cognitive radio (CR), which is built on top of a software defined radio [4], is an intelligent wireless communications system that is aware of its spectral operational environment. A CR node must be able to dynamically adapt to the environmental spectral changes, in order to abide by the spectral etiquette set forth by the FCC. One of the most important functions that a CR node must perform, is the identification of unoccupied spectrum opportunities (SOPs). SOPs are space, time, and frequency dependent blocks, during which the license-exempt can utilize the registered owner's spectrum in a DSA manner. Prior to May 2012, SOP discovery was mainly done through distributed and cooperative sensing. In such an approach, CR nodes rely on sheer power detection methods, and coordinate in order to identify spectrum activity and locate available SOPs [5–9].

On the opposite end of this approach lies a database-driven spectral learning technique that allows CR nodes to understand their spectral surroundings in a three-step process. A node attempting to analyze the surrounding SOPs would first learn its geographical location through a GPS device. Subsequently, it would

☆ This research has been funded by the NSF CAREER Award IIS-0845262.
* Corresponding author.
E-mail addresses: etroja@gradcenter.cuny.edu (E. Troja), sbakiras@hbku.edu.qa (S. Bakiras).

http://dx.doi.org/10.1016/j.adhoc.2017.02.001
1570-8705/

Fig. 1. Mobile user geo-located near Tsinghua university (from Microsoft's GeoLife trajectory dataset).

contact a central repository (database) and issue its GPS coordinates as part of the query. Finally, it would download the centrally fused and compiled repository report containing the available SOPs [10] at that location. The compilation and fusion of the SOPs is assumed to be done by specialized entities called spectrum database operators (SDOs). The available SOPs are compiled by applying appropriate propagation modeling and interference avoidance algorithms for a given geographical location.

The FCC's May 2012 ruling [11] obsoletes the distributed and cooperative sensing method for the white-space TV bands. The ruling requires that all CR nodes operating in the white-space TV bands utilize the centralized white-space database (WSDB) spectrum lookup method. In order to allow mobile television band devices (TVBDs) to learn their spectral surroundings, the FCC has designated 10 WSDB providers, out of which only Google, Spectrum Bridge, and Telcordia Technologies have been approved for operation [12].

Nevertheless, the database-driven DSA approach is prone to severe location privacy leakage. According to FCC specifications [13], a mobile TVBD must issue a new query whenever it moves further than 100 m 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.). As an example, Fig. 1 shows a mobile TVBD's trajectory that is formed by latitude/longitude data points taken at consecutive time intervals, near Tsinghua university. Given the starting point of the trajectory, the WSDB server can identify (to a certain extent) the user associated with this trajectory (e.g., it may correspond to a home address). In addition, given the end point of the trajectory, the WSDB server can infer (with a certain probability) that the aforementioned TVBD user is affiliated with Tsinghua university.

To this end, Gao et al. [14] 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. partition 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[1] [16] whose communication cost (for a single query) is equal to a large percentage of the database size. 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 [16] 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 of [14] 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 [14], our methods leverage the PIR scheme of Gentry and Ramzan [17], 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) [18]. 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 cell IDs. Due to the properties of the underlying HSFC, these cells are 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 two trajectory prediction methods. The first method is based on the simple linear regression line (SLR) generated by 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. The second method is based on prediction using variable order Markov models (VMMs). Here, after some initial training, users utilize their knowledge of coordinates traveled in the past to assign, in real time, movement probabilities to surrounding coordinates. The movement probabilities are considered in the context of the most recent travel history. Furthermore, predicted coordinates are evaluated and verified against actual traveled coordinates and utilized for further, real time re-training of the VMMs.

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 [19] and Microsoft's GeoLife GPS dataset [20]. The experimental results show that, compared to the state-of-the-art computationally efficient PIR protocol [21], our methods reduce the query processing cost at the mobile clients by a factor of 2 to 8.

Note that, this article is an extension of a previously published conference paper [22]. Compared to the conference version, this article makes the following contributions.

- We removed Ref. [14] from the experimental evaluation, as its underlying PIR protocol has been proven insecure. We replaced it with the current state-of-the-art protocol [21] for computationally efficient PIR.
- We employed new trajectory prediction models (Section 4.6) that improve the performance over our previous work by up to 33%.

[1] Note that, the PIR protocol of Trostle and Parrish was recently broken by Lepoint and Tibouchi [15].

- We significantly revised the experimental evaluation section to account for the new PIR protocol and the new trajectory prediction algorithms.

The remainder of this paper is organized as follows. Section 2 presents a literature review on location privacy. Section 3 provides the necessary background on the various primitives utilized in our work. Section 4 describes our methods in detail, and Section 5 presents the results of the experimental evaluation. We conclude our work in Section 6.

2. Related work

Most existing approaches for location privacy rely on the notion of k-anonymity [23] or l-diversity [24]. 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 [25–27].

l-diversity based methods [28,29], 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. [30] attempts to provide location privacy using location semantics.

The k-anonymity and l-diversity based approaches, as well as collaborative location privacy protection methods [31,32], often rely on third party trusted anonymizers, which is not always a viable solution. On the other hand, Ghinita et al. [33] 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 [34].

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 [35–37].

Due to the recency of the FCC's ruling, 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. [14], which builds upon a modified version of Trostle and Parrish's PIR scheme [16]. 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). Nevertheless, their scheme incurs a high 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, which is approximately 2.6% of the whole database size. For highly mobile clients, the cost of this approach can exceed the cost of downloading the entire database.

Troja and Bakiras [38] introduce a protocol that allows mobile DSA users to share their cached channel availability information in a privacy-preserving manner. The protocol leverages an anonymous veto protocol that anonymizes the exchange of information among a group of users. This method is orthogonal to our work and may be employed independently, in order to further reduce the number of PIR queries sent to the WSDB server.

3. Preliminaries

In this section we give a brief overview of the various primitives utilized in our work. Section 3.1 introduces the concept of private information retrieval and Section 3.2 presents the Hilbert space filling curve algorithm. Section 3.3 describes variable order Markov models.

3.1. 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 ith 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 [39] 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 [34] introduced the first CPIR protocol for a single database, whose security is based on the quadratic residuosity assumption. The communication complexity of [34] is $O(n^{\epsilon})$. Further work [40,41] demonstrates CPIR schemes with polylogarithmic communication complexity.

A recent trend in the PIR literature is for computationally efficient protocols with large communication cost (but still better than the trivial PIR case). The current state-of-the-art protocol is XPIR [21], which employs the latest results from lattice-based cryptography. Nevertheless, as we will show in the experimental evaluation, the communication cost of such schemes is not well-suited for mobile applications.

In this work, we leverage the protocol of Gentry and Ramzan [17], 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, and its functionality is summarized as follows.

Setup. We assume a database of N records, where each record is of size ℓ bits. During a preprocessing phase, the server associates every record j with a prime power $\pi_j = p_j^{c_j}$, where p_j is a small prime and c_j is the smallest integer, such that $\log \pi_j > \ell$. All these values are public knowledge. Next, using the Chinese remainder theorem (CRT), the server expresses the entire database as an integer e, which is the solution to the congruences $e \equiv B_j \bmod \pi_j$, for all $j \in \{1, 2, \ldots, N\}$. ($B_j$ is the binary representation of record j.) Client queries are processed on the transformed database e.

Query generation. Groth et al. [42] show that Gentry and Ramzan's protocol can be used to retrieve multiple records with a single query. Let $i_1, i_2, \ldots, i_k$ be the indexes of the records to be retrieved. The client computes $\pi = \prod_{j=1}^{k} \pi_{i_j}$ and chooses two large prime numbers p and q, such that $p = 2\pi r + 1$ and $q = 2st + 1$, where r, s, and t are large random integers. The client sets $m = pq$ and proceeds to select a random element $g \in \mathbb{Z}_m^*$ with order πv, where $\gcd(\pi, v) = 1$. Finally, the client sends (g, m) to the server. For security reasons, it should hold that $\log m > \max(1024, 4\log \pi)$.

Database response. The server computes $c = g^e \bmod m$ and sends the result back to the client.

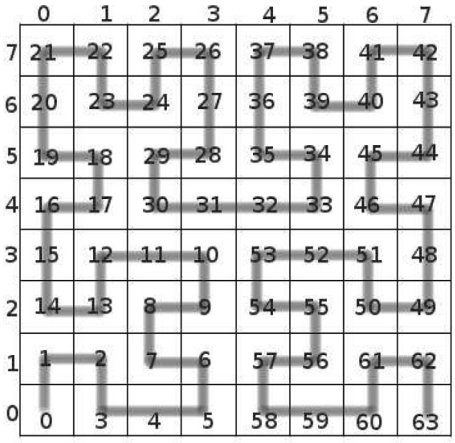

Fig. 2. A level 3 Hilbert space filling curve.

Result retrieval. To reconstruct the records, the client computes, for each i_j, $c^{\pi v/\pi_{i_j}} \bmod m$, which should be equal to $(g^{\pi v/\pi_{i_j}})^{B_{i_j}} \bmod m$. Therefore, the client can retrieve record B_{i_j}, using the Pohlig–Hellman algorithm for discrete logarithms [43].

3.2. Hilbert space filling curve

The Hilbert space filling curve [18] 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. 2 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. Note that, the Hilbert function could be initialized on any of the four corners (0, 0), (0, 7), (7, 7), and (7, 0), without affecting the 2-D to 1-D mapping. The mapping of points to their Hilbert IDs might change, but it would still preserve locality.

3.3. Variable order markov model

Variable order Markov models (VMMs) are stochastic processes that are used to predict discrete sequences over a finite alphabet. Their main usage is to learn probabilistic finite state automata, which are typically employed to model various real life problems. Formally, a VMM can be represented as follows. Let Σ be a finite alphabet. An initial training sequence $q_1^n = q_1 q_2 \dots q_n$ is presented to a trainee, such that $q_i \in \Sigma$ and $q_i q_{i+1}$ is the chaining of q_i and q_{i+1}. The goal of a trainee is to learn a model $\hat{P}$, which provides probabilities for future outcomes based on past training sequences, i.e., for a given context $s \in \Sigma^*$ and any given symbol $\sigma \in \Sigma$, the trainee should generate a conditional probability distribution $\hat{P}(\sigma|s)$ [44].

4. Efficient location privacy for mobile DSA clients

In this section, we present the details of our methods. Section 4.1 introduces the underlying threat model and Section 4.2 describes the system architecture. Section 4.3 introduces our basic approach and Section 4.4 presents an enhanced method that retrieves multiple cells from the area surrounding the query point. Section 4.5 introduces the trajectory prediction algorithm based on SLR and Section 4.6 presents trajectory prediction algorithms based on various VMMs. Lastly, Section 4.7 describes the case where there is full a priori trajectory knowledge.

Table 1
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
$\log m$	Bit-size of RSA modulus (Section 3.1)
R	Number of rings to explore in the surrounding area

4.1. 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.

4.2. System architecture

Similar to previous work [14], we assume a fixed grid of $n \times n$ cells. According to the FCC specifications [13], 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.

4.3. Single row retrieval

Papadopoulos et al. [45] conducted a in-depth study of the PIR protocol by Gentry and Ramzan [17] 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 1 summarizes the symbols used in the remainder of this paper.) In the toy example of Fig. 2, 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.

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, so as 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

Table 2
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

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 $(2+k)\log m$, where m is an RSA modulus. Table 2 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.

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 2, 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.

4.4. 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. 3. 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. 2, 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 on all space filling curves, and are more severe on higher level curves (which is typically the case in real life applications).

To address these shortcomings, we take advantage of Gentry and Ramzan's multi-record retrieval feature, as described by Groth et al. [42]. 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 (as explained in Section 3.1), 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 $(2+ku)\log m$.

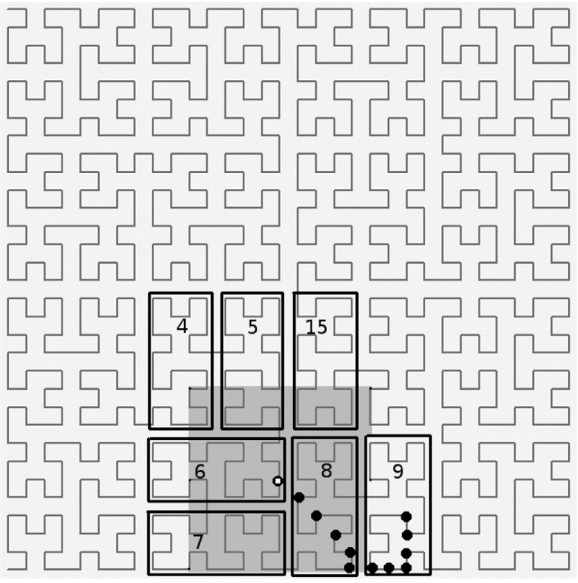

Fig. 3. Exploring the surrounding area with 2 and 4 sub-segments.

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.

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 4–5). Next, the algorithm iterates over all cells within the area defined by R, and counts how many times the underlying row numbers ap-

Algorithm 1 Single row retrieval.

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

Algorithm 2 Surrounding area.

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

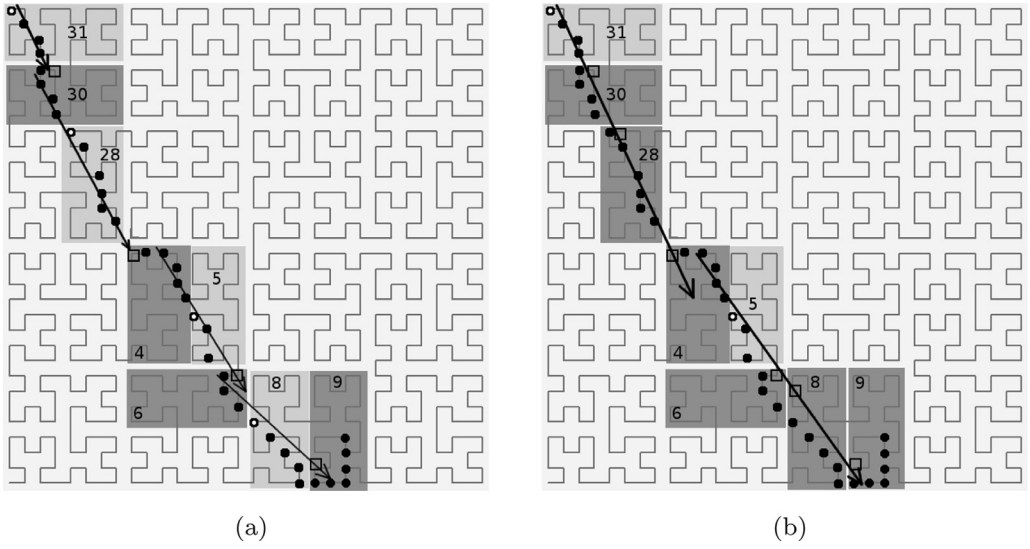

Fig. 4. Trajectory prediction example. (a) Using 2 sub-segments. (b) Using 4 sub-segments.

pear in the result (lines 6–11). Finally, it selects the $(u-1)$ most frequent row numbers and adds them into *rows* (lines 12–13). 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. 3, when $u=2$ we retrieve rows 6 and 8. On the other hand, when $u=4$ we retrieve rows 6, 8, 7, and 15.

4.5. Trajectory prediction using SLR

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 5). 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 6–10).

Algorithm 3 SLR trajectory prediction.

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

Fig. 4 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. 4a), 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. 4a.

On the other hand, when $u=4$ (Fig. 4b) 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 $40\log m$ bits, while the 4 sub-segment case requires $36\log m$ 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).

4.6. Trajectory prediction using variable order markov models

SLR-based trajectory prediction is only effective when the user travels on the same direction for a sufficiently large period of time. Therefore, our next step is to explore the applicability of more sophisticated prediction algorithms that utilize the user's entire trajectory history (i.e., through a training process). Our decision to employ VMMs as the prediction engine is based on the repetitive nature of a user's movement, which can be represented as a stochastic process. Similar to the SLR-based method, the predicted future location coordinates are evaluated in order to maximize the amount of useful information retrieved by a PIR query.

Algorithm 4 shows the detailed steps of this approach. The algorithm maintains a first in first out (FIFO) buffer c of the database rows that are retrieved by the most recent PIR queries. This buffer is used by the underlying VMM engine as the prediction context. Initially, the algorithm inserts into c the row that corresponds to the HID of the current coordinate points (line 5). Next, it identifies the geographically neighboring rows of the last entry in c (line 7). For each one of those rows, it evaluates the probability that the

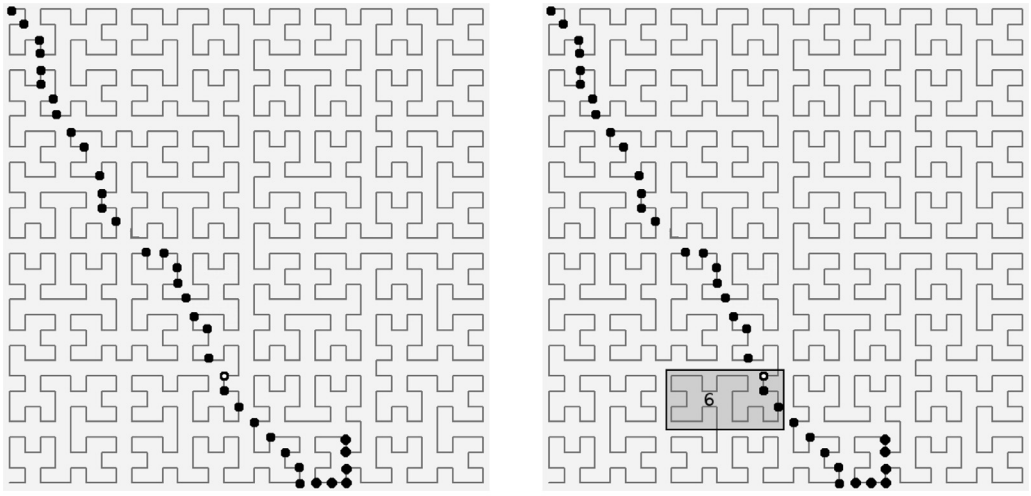

Fig. 5. Identification of the first row to be retrieved.

row will be retrieved in the future, given the past retrieval context in c (lines 9–14). The algorithm then chooses the row with the highest retrieval probability (line 12) to be part of the current PIR query. This process is repeated for $(u-1)$ additional cells, whose underlying rows are not present in the cache. The selected rows are eventually retrieved via the PIR query, and are consequently added to the cache (lines 18–19).

Figs. 5 and 6 illustrate an example of the VMM-based trajectory prediction method, for $u = 2$. The dots represent the user's trajectory, starting from the upper left corner. We assume that the user moves into the location coordinate identified by the hollow black dot, as shown in Fig. 5a. Here, the user executes Algorithm 4 and discovers that the HID of the coordinate is not included in its cache. Therefore, in Fig. 5b, the user identifies the row (row 6) containing the HID of the coordinate and adds it to the *rows* buffer that will be retrieved by the following PIR query.

Next, row 6 is added to buffer c and, since $u = 2$, the algorithm needs to identify one more row for retrieval. Fig. 6a illustrates the geographically neighboring rows (lightly-shaded regions) that are viable candidates. Then, the VMM prediction algorithm leverages the current context $c = \{30, 28, 4, 5, 6\}$, in order to identify the row with the highest predicted utility. In our toy example, based on the initial training that the VMM receives, the neighboring row for which the VMM predicts to have the highest probability of being selected (given context c), is 8. This step is reflected in Fig. 6b, i.e., the two dark-shaded regions (rows 6 and 8) are retrieved in the following PIR query.

Algorithm 4 VMM trajectory prediction.

```
1:  procedure VMM-TRAJECTORY-PREDICTION(HID, k, u, c[ ])
2:      if (HID ∉ cache) then
3:          r ← HID/8k;
4:          insert r into rows[ ];
5:          insert r into c[ ];
6:          for i = 1 to (u − 1) do
7:              neighbors[ ] ← find-neighboring-rows(c[last]);
8:              maxp = 0, best = null;
9:              for j = 0 to (neighbors − 1) do
10:                 if (VMM-predict(j, c[ ]) > maxp) then
11:                     maxp ← VMM-predict(j, c[ ]);
12:                     best = j;
13:                 end if
14:             end for
15:             insert best into rows[ ];
16:             insert best into c[ ];
17:         end for
18:         cells[ ] ← PIR(rows[ ]);
19:         cache ← cells[ ];
20:     end if
21: end procedure
```

4.7. A Priori trajectory knowledge

Our last method is designed for 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 5 depicts that simulation. It simply initializes an empty hash table *HT*, and inserts therein the row numbers of all cells that intersect trajectory *T*. This method invokes the least number of PIR queries.

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).

Algorithm 5 A priori trajectory simulation.

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

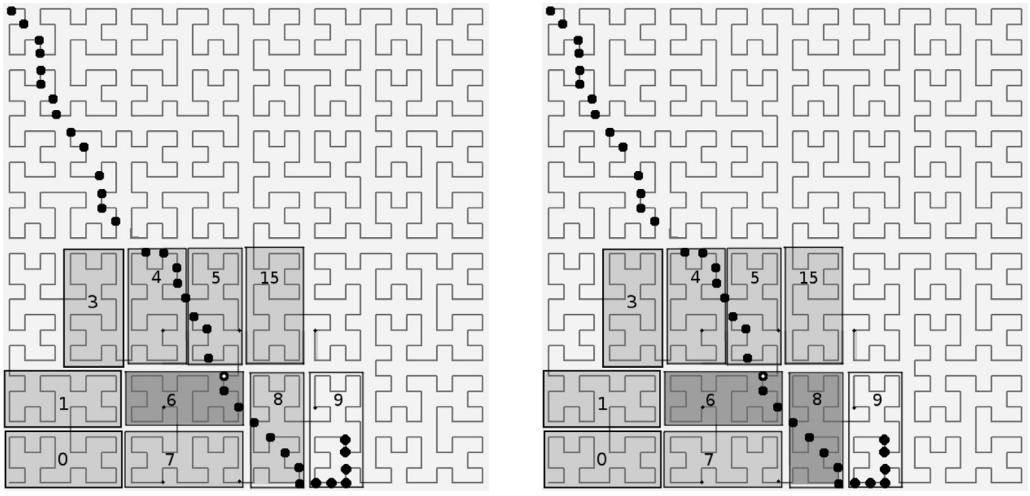

Fig. 6. Identification of the second row to be retrieved.

5. Experimental evaluation

In this section, we evaluate experimentally the performance of our proposed methods. Section 5.1 describes the setup of our experiments, and Section 5.2 provides the detailed results.

5.1. Experimental setup

We developed our experiments in Java SDK, running on Ubuntu 14.04 LTS. For the experimental tests, we utilized two real life datasets, namely Microsoft's GeoLife GPS Trajectories[2] and Microsoft's T-Drive GPS dataset[3]. Both are excellent datasets, containing real life trajectories from users traveling around Beijing, China. The GeoLife GPS trajectory dataset [20] was collected as part of the Microsoft Research Asia GeoLife project. It monitors 182 users for a period of over five years (from Apr. 2007 to Aug. 2012). A GPS trajectory from this dataset is represented by a sequence of time-stamped points, each containing information regarding the user's latitude, longitude, and altitude. The dataset contains 17,621 trajectories, with a total distance of 1,292,951 km, and a total duration of 50,176 hours. These trajectories were recorded by different GPS loggers and GPS-enabled phones, and have a variety of sampling rates. 91.5% of the trajectories are logged in a dense representation, e.g., every 1–5 s or every 5–10 m per point. A sample trajectory from the GeoLife GPS dataset is depicted in Fig. 7a. The T-Drive dataset [19] contains GPS trajectories from 10,357 taxis, during the period of Feb. 2 to Feb. 8, 2008. The total number of points in the dataset is about 15 million, and the total distance from all trajectories reaches up to 9 million kilometers. The average sampling interval is about 177 seconds, with a distance of about 623 m. A sample trajectory from the T-Drive dataset is shown in Fig. 7b.

In our experiments, we set a bounding box of 409.6 km × 409.6 km (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 $maxlong = 118.9$. We run our experiments on 17,621 trajectories belonging to 182 unique users from the GeoLife dataset and 55,712 trajectories belonging to 10,357 unique users from the T-Drive dataset. Specifically, the experiments were performed as follows. For each user, we randomly selected 50% of the trajectories and used them to train the VMM engine. Then, the remaining 50% of the trajectories were used to conduct the actual experiments (for all methods). Each experiment was repeated 10,000 times and the results were averaged.

Table 3
Cost of PIR operations .

Cost	GR [17]	XPIR [21]
Query generation (client)	450 ms	11 ms
Server processing (64 CPUs)	4560 ms	0.6 ms
Server processing (128 CPUs)	2280 ms	0.3 ms
Result extraction (client)	125 ms	3.2 ms
Communication cost	20,800 bytes	5,403,444 bytes

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 Gentry-Ramzan protocol using the GMP[4] multiple precision arithmetic library. We also downloaded and installed the XPIR[5] library, as implemented by the authors. Table 3 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 XPIR, we fixed the modulus size to 1024 bits. To reduce the communication cost, we divided the database into 128 rows, by forming groups of 32 rows from the original 4096 × 4096 database. As a result, each row contains channel availability information for $32 \times 4096 = 131,072$ cells. Note that, a single XPIR query retrieves one of the 128 rows, i.e., 0.78% of the whole database. For comparison, our methods retrieve at most 4096 cells with a single query, which is just 0.02% of the database. For Gentry and Ramzan's protocol we set the modulus size m equal to 1280 bits, in order to satisfy the security requirement outlined in Section 3.1. The values shown in the table above correspond to the single row retrieval method, where $k = 128$ and $u = 1$. While part of the query generation algorithm is precomputed offline (prime q

[2] http://research.microsoft.com/en-us/projects/GeoLife/
[3] http://research.microsoft.com/en-us/projects/tdrive/
[4] http://gmplib.org
[5] https://github.com/XPIR-team/XPIR

Fig. 7. (a) Sample dense data points from the GeoLife trajectories (b) Sample sparse data points from the T-drive trajectories.

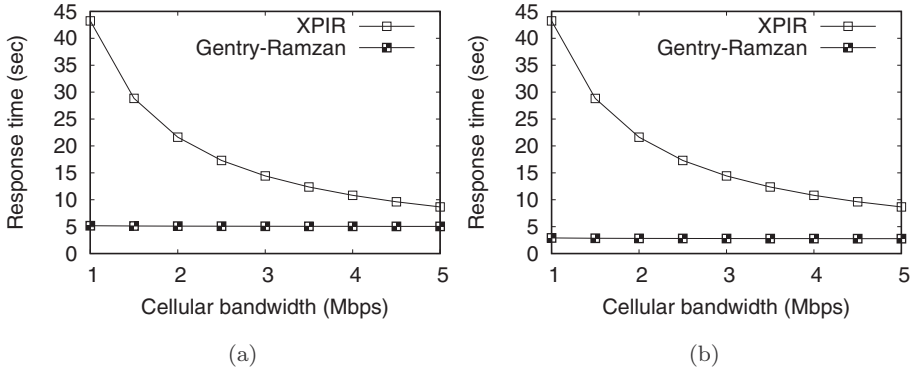

Fig. 8. Response time for a single PIR query. (a) 64 CPUs (b) 128 CPUs.

of the RSA modulus), prime p depends on the queried row(s) and should be computed online.

Fig. 8 shows the query response time for the two PIR protocols (based on Table 3) as a function of the cellular bandwidth available at the mobile client. Clearly, the cost of the XPIR scheme is dominated by the network transfer time, since each PIR query necessitates the exchange of over 5 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.

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 s to download at 1 Mbps, and 108 s at 5 Mbps.

5.2. Experimental results

In the first experiment we investigate the performance of the single row retrieval method ($k = 128,\ u = 1$), as explained in Section 4.3. Figs. 9 and 10 depict the average cumulative query response time as a function of the available bandwidth.

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 (as shown in Fig. 7). 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. XPIR outperforms our basic method for low bandwidth situations, because it reduces the number of PIR queries sent to the server. This is due to the large number of cells retrieved by each XPIR query (0.78% of the whole database), which may be utilized throughout a clients's trajectory.

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 4.4 (we set $R = 50$ rings as the explored area). Figs. 11 and 12 illustrates the cumulative query response time as a function of the available bandwidth for the two datasets. It is evident that our method considerably outperforms XPIR in almost all settings. Even though XPIR retrieves a lot more information per query, our algorithm is more intelligent in retrieving cells that have the potential of becoming useful in the future. 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.

Next, we evaluate the performance of the SLR-based trajectory prediction method, as described in Section 4.5. For the rest of our experimental evaluation, including this experiment, we purposely omit the results for the trivial PIR and single row methods, because

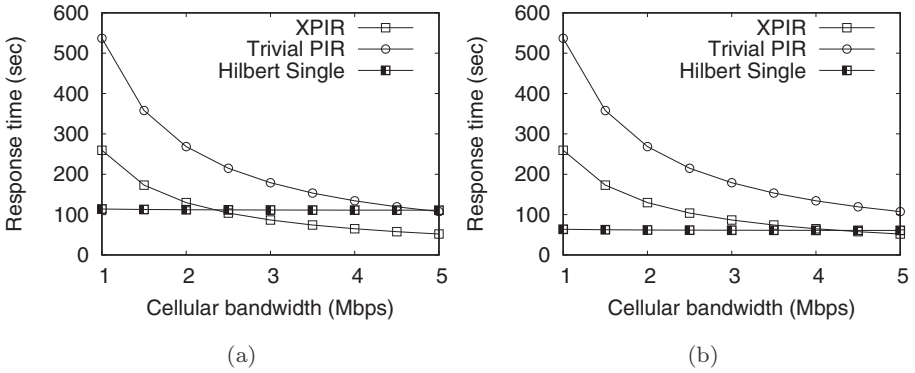

Fig. 9. Cumulative query response time for the single row retrieval method (GeoLife). (a) 64 CPUs (b) 128 CPUs.

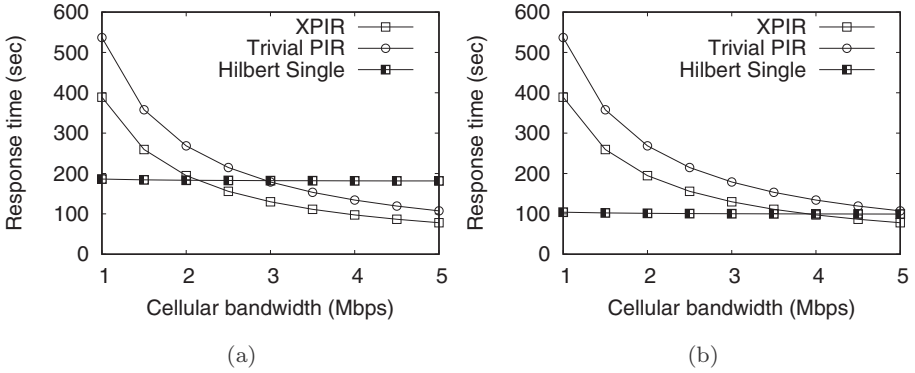

Fig. 10. Cumulative query response time for the single row retrieval method (T-Drive). (a) 64 CPUs (T-drive)(b) 128 CPUs (T-drive) .

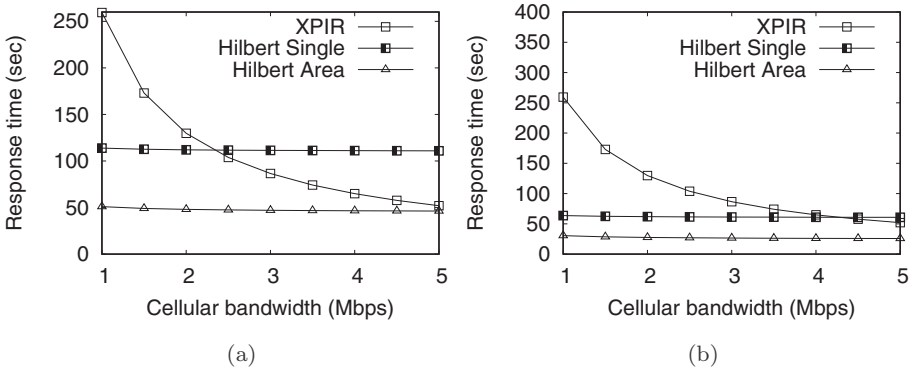

Fig. 11. Cumulative query response time for the surrounding area method (GeoLife). (a) 64 CPUs (b) 128 CPUs.

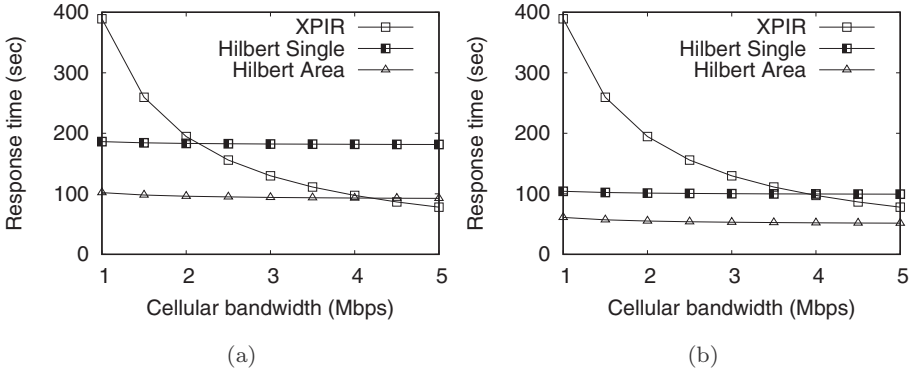

Fig. 12. Cumulative query response time for the surrounding area method (T-drive). (a) 64 CPUs (b) 128 CPUs .

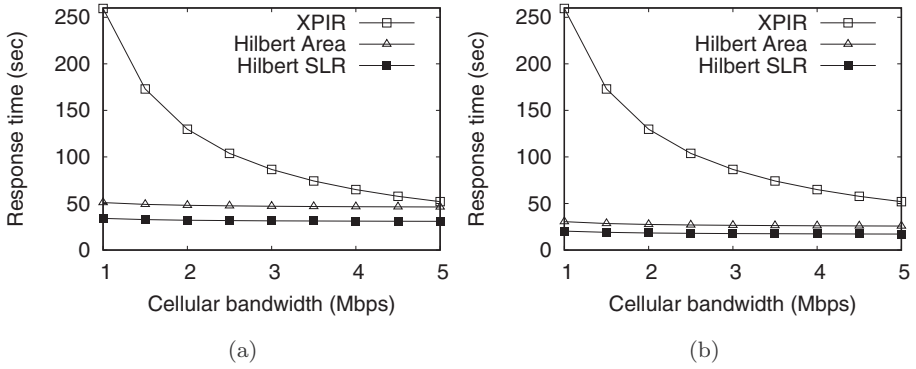

Fig. 13. Cumulative query response time for the surrounding area method vs. the SLR-based trajectory prediction method (GeoLife). (a) 64 CPUs (b) 128 CPUs.

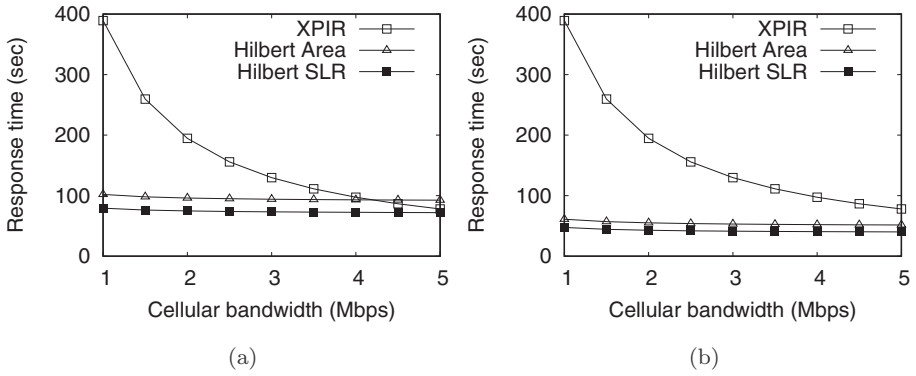

Fig. 14. Cumulative query response time for the surrounding area method vs. the SLR-based trajectory prediction method (T-Drive). (a) 64 CPUs (b) 128 CPUs.

they are clearly inferior to the surrounding area approach. Figs. 13 and 14 show the cumulative query response times for the surrounding area and SLR-based prediction methods. Our prediction algorithm is clearly superior. Utilizing 128 compute units at the server results in a response time of just 18 s (for the whole trajectory) in the GeoLife dataset and 42 s in the T-Drive dataset. Furthermore, the SLR-based 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 SLR-based trajectory prediction, we prefetch results according to a specific direction of movement instead of a generic rectangular area. Compared to XPIR, SLR-based trajectory prediction is significantly faster, even for a bandwidth of 5 Mbps. Furthermore, for low bandwidth situations, the performance is improved by a factor of 5 to 8.

In the next set of experiments we investigate the performance of the VMM-based trajectory prediction method, which was explained in Section 4.6. Since we already established the superiority of our SLR-based trajectory prediction method, we specifically omit direct comparison of our VMM-based methods against XPIR and exclude its performance from the rest of our figures. We employed four well-known VMM algorithms, namely the Lempel–Ziv 78 (LZ78) compression algorithm, an improved Lempel–Ziv (LZms) algorithm, the Context Tree Weighting (CTW) algorithm, and the Prediction by Partial Match (PPM) algorithm. These methods are described in great detail by Begleiter et al. [44], and the interested reader may find references for the corresponding Java source codes therein.

Figs. 15 and 16 compare the SLR-based trajectory prediction algorithm against the four VMM-based methods. For the T-drive dataset, VMM-based prediction improves the cumulative query cost by 29–33%. In the case of GeoLife dataset, the cost savings range between 16–24%. As expected, the performance improvement is more significant in the T-drive dataset, due to the sparseness of the underlying data points. Note that, in both datasets, the PPM prediction engine exhibits the best overall performance, so we will set it as the default engine in the following experiment.

In the following experiment, we investigate the performance of the a priori trajectory knowledge approach, which was explained in Section 4.7. Recall that, this method yields the lowest number of PIR requests, since the client avoids the retrieval of any unnecessary rows from the WSDB. In order to show the efficiency of the a priori method, we compare it against the PPM-based trajectory prediction approach. Figs. 17 and 18 illustrate the corresponding cumulative query response times. The a priori method entails a cost of just 12 s in the GeoLife dataset and 23 s in the T-drive dataset (for 128 compute units). Compared to the PPM-based trajectory prediction method, the a priori trajectory knowledge enables us to reduce the query processing cost by an additional 13% in the GeoLife dataset and 17% in the T-drive dataset.

In the last experiment we investigate the performance of our methods on large databases. All previous experiments hide the users' locations within an area of 168,772 km^2, which is approximately the size of the state of Florida. Here, we extend the bounding box of the dataset to fit a $32,768 \times 32,768$ grid, which corresponds to an area larger than the entire United States. As a result, the size of the WSDB is now equal to 4 GB, which adversely affects the server processing cost for the Gentry–Ramzan protocol (146 sec/query for 128 CPUs). For the XPIR protocol we formed groups of 32 rows, resulting in a database of 1024 rows, each containing information from over 1 million cells. The performance of XPIR is still dominated by the communication cost, as each query necessitates the exchange of over 41 MB of data.

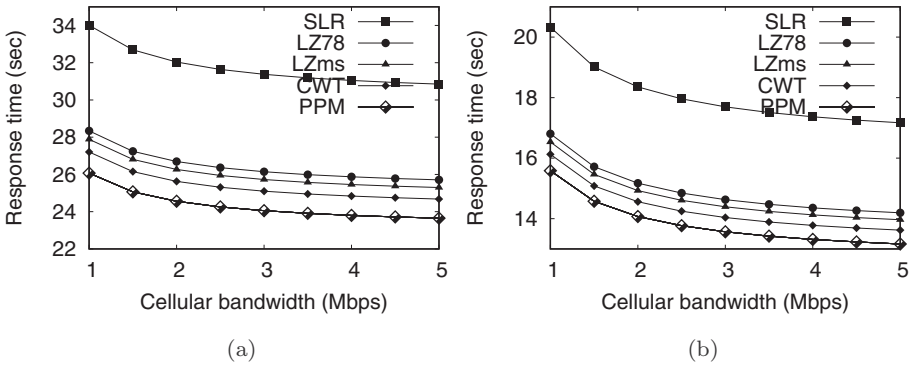

Fig. 15. Comparison of SLR-based and VMM-based trajectory prediction methods (GeoLife) (a) 64 CPUs (b) 128 CPUs.

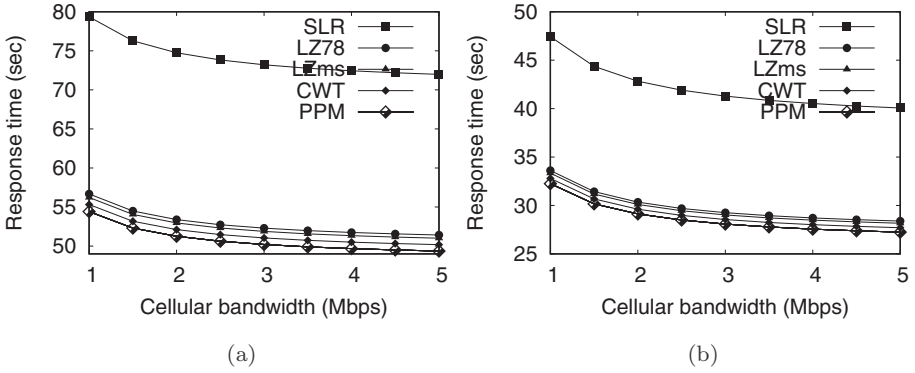

Fig. 16. Comparison of SLR-based and VMM-based trajectory prediction methods (T-drive) (a) 64 CPUs (b) 128 CPUs.

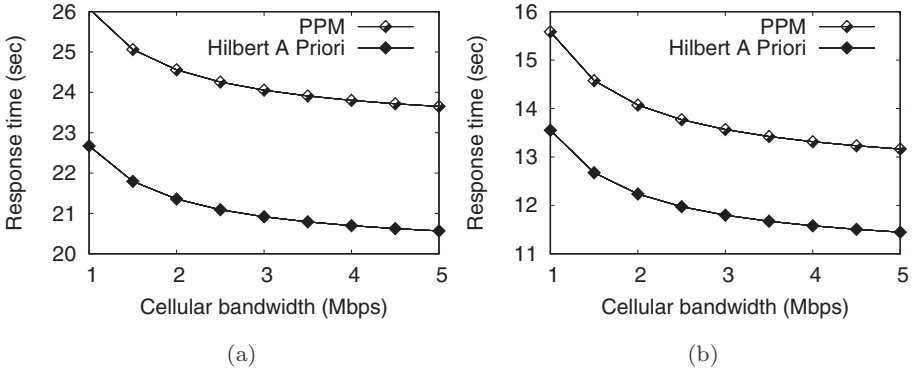

Fig. 17. Cumulative query response time for the PPM-based trajectory prediction method vs. the Hilbert a priori method (GeoLife). (a) 64 CPUs (b) 128 CPUs .

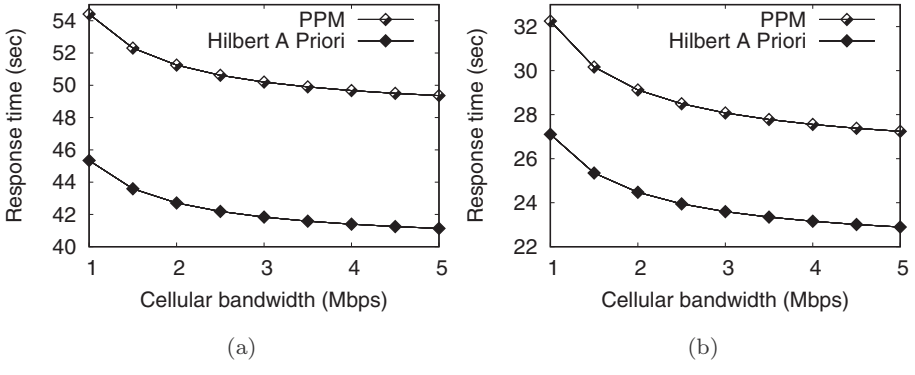

Fig. 18. Cumulative query response time for the PPM-based trajectory prediction method vs. the Hilbert a priori method (T-Drive). (a) 64 CPUs (b) 128 CPUs.

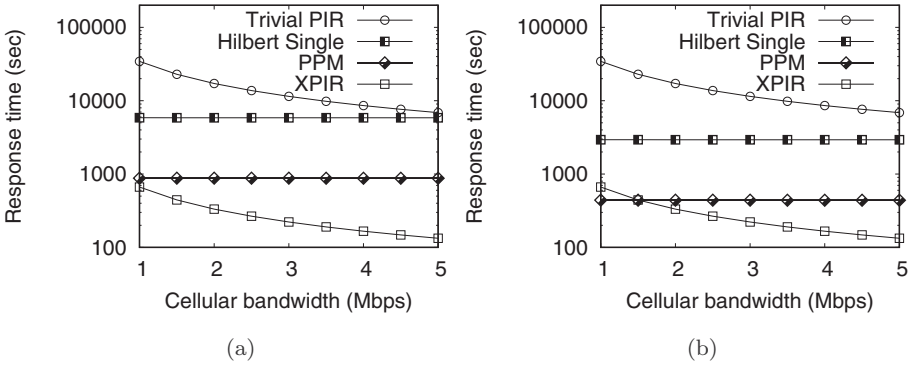

Fig. 19. Cumulative query response time for large WSDB (GeoLife) (a) 64 CPUs (b) 128 CPUs .

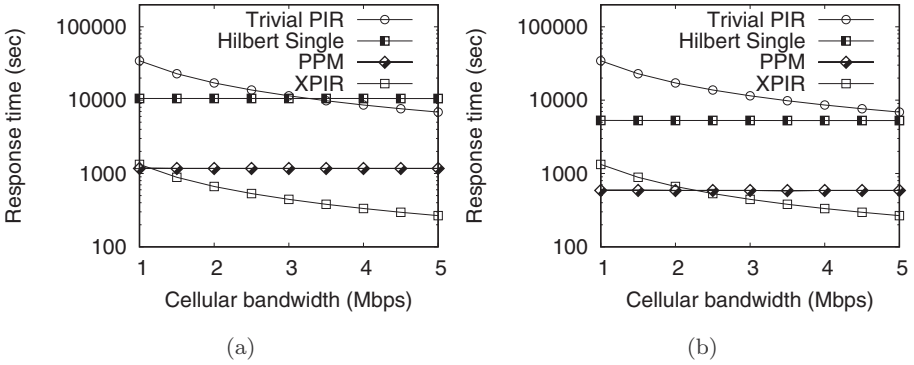

Fig. 20. Cumulative query response time for large WSDB (Tdrive) (a) 64 CPUs (b) 128 CPUs.

Figs. 19 and 20 compare the performance of our methods against XPIR and trivial PIR. Clearly, the computational efficiency of the XPIR protocol makes it a better choice for large databases. Our methods outperform XPIR only for low bandwidth scenarios (up to 2 Mbps) and 128 compute units. We thus argue that our techniques are mostly applicable to state-level WSDBs, while XPIR is more suitable for a nationwide WSDB. It is also worth noting that our optimizations still hold for very fine grids. In particular, the trajectory prediction method (PPM) outperforms the single row retrieval approach by a factor of 6 to 10 in all settings.

6. Conclusions

Database-driven dynamic spectrum access networks allow mobile users to query a white-space database, in order to identify idle channels in their area. Nevertheless, such location-dependent queries pose a serious privacy threat, as they may reveal sensitive information about the individual. 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 by prefetching results in the direction of the user's movement. Through extensive experimentation with real life datasets, we show that, compared to the current state-of-the-art protocol, our methods greatly reduce the query processing cost at the mobile clients.

References

[1] NTIA, United states frequency allocations – the radio spectrum (2003). URL http://www:ntia:doc:gov/files/ntia/publications/2003-allochrt:pdf.
[2] M. Cesana, F. Cuomo, E. Ekici, Routing in cognitive radio networks: challenges and solutions, Ad Hoc Netw. 9 (2011) 228–248.
[3] FCC, Facilitating opportunities for flexible, efficient, and reliable spectrum use employing cognitive radio technologies, FCC Rep. Order (2005).
[4] J. Mitola III, Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio, Doctoral Dissertation, KTH, Stockholm, Sweden, 2000.
[5] I.F. Akyildiz, W.-Y. Lee, M.C. Vuran, S. Mohanty, Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey, Comput. Netw. 50 (13) (2006) 2127–2159.
[6] I.F. Akyildiz, B.F. Lo, R. Balakrishnan, Cooperative spectrum sensing in cognitive radio networks: a survey, Phys. Commun. 4 (1) (2011) 40–62.
[7] Q. Zhao, S. Geirhofer, L. Tong, B.M. Sadler, Optimal dynamic spectrum access via periodic channel sensing, in: IEEE WCNC, 2007, pp. 33–37.
[8] S. Haykin, D. Thomson, J. Reed, Spectrum sensing for cognitive radio, Proc. IEEE 97 (5) (2009) 849–877.
[9] W. Zhang, R. Mallik, K. Letaief, Optimization of cooperative spectrum sensing with energy detection in cognitive radio networks, IEEE Trans. Wireless Commun. 8 (12) (2009) 5761–5766.
[10] T.X. Brown, A. Sethi, Potential cognitive radio denial-of-service vulnerabilities and protection countermeasures: a multi-dimensional analysis and assessment, Mob. Netw. Appl. 13 (2008) 516–532.
[11] FCC, Third memorandum opinion and order(2012) 12–36. URL http://transition:fcc:gov/DailyReleases/DailyBusiness/2012/db0405/FCC-12-36A1:pdf.
[12] FCC, White space database administrators guide(2013a) 3–10. URL http://www:fcc:gov/encyclopedia/white-space-database-administrators-guide.
[13] FCC, Television band devices(2013b) 11. URL http://www:ecfr:gov/cgi-bin/text-idx?c=ecfr&rgn=div6&view=text&node=47:1:0:1:1:16:8&idno=47#47:1:0:1:1:16:8:237:7.
[14] Z. Gao, H. Zhu, Y. Liu, M. Li, Z. Cao, Location privacy in database-driven cognitive radio networks: attacks and countermeasures, in: IEEE INFOCOM, 2013, pp. 2751–2759.
[15] T. Lepoint, M. Tibouchi, Cryptanalysis of a (somewhat) additively homomorphic encryption scheme used in PIR, in: Financial Cryptography Workshops, 2015, pp. 184–193.
[16] J. Trostle, A. Parrish, Efficient computationally private information retrieval from anonymity or trapdoor groups, in: Information Security, 2011, pp. 114–128.
[17] C. Gentry, Z. Ramzan, Single-database private information retrieval with constant communication rate, in: ICALP, 2005, pp. 803–815.
[18] I. Kamel, C. Faloutsos, On packing R-trees, in: ACM CIKM, 1993, pp. 490–499.
[19] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, Y. Huang, T-drive: driving directions based on taxi trajectories, in: ACM GIS, 2010, pp. 99–108.
[20] Y. Zheng, L. Wang, R. Zhang, X. Xie, W.-Y. Ma, GeoLife: managing and understanding your past life over maps, in: IEEE MDM, 2008, pp. 211–212.
[21] C. Cguilar-Melchor, J. Barrier, L. Fousse, M.-O. Killijian, XPIR : private information retrieval for everyone, PoPETs 2016 (2) (2016) 155–174.

[22] E. Troja, S. Bakiras, Efficient location privacy for moving clients in database–driven dynamic spectrum access, IEEE ICCCN, 2015.
[23] L. Sweeney, K-anonymity: a model for protecting privacy, Int. J. Uncertain. Fuzz. Knowl. Based Syst. 10 (05) (2002) 557–570.
[24] A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam, L-diversity: privacy beyond k-anonymity, ACM Trans. Knowl. Discov. Data (TKDD) 1 (1) (2007).
[25] M. Gruteser, D. Grunwald, Anonymous usage of location-based services through spatial and temporal cloaking, in: ACM MobiSys, 2003, pp. 31–42.
[26] M.F. Mokbel, C.-Y. Chow, W.G. Aref, The new Casper: query processing for location services without compromising privacy, in: VLDB, 2006, pp. 763–774.
[27] T. Xu, Y. Cai, Feeling-based location privacy protection for location-based services, in: ACM CCS, 2009, pp. 348–357.
[28] T. Xu, Y. Cai, Exploring historical location data for anonymity preservation in location-based services, in: IEEE INFOCOM, 2008, pp. 547–555.
[29] T. Wang, L. Liu, Privacy-aware mobile services over road networks, PVLDB 2 (1) (2009) 1042–1053.
[30] B. Lee, J. Oh, H. Yu, J. Kim, Protecting location privacy using location semantics, in: ACM SIGKDD, 2011, pp. 1289–1297.
[31] K. Vu, R. Zheng, J. Gao, Efficient algorithms for k-anonymous location privacy in participatory sensing, in: IEEE INFOCOM, 2012, pp. 2399–2407.
[32] J. Freudiger, M.H. Manshaei, J.-P. Hubaux, D.C. Parkes, On non-cooperative location privacy: a game-theoretic analysis, in: ACM CCS, 2009, pp. 324–337.
[33] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, K.-L. Tan, Private queries in location based services: anonymizers are not necessary, in: ACM SIGMOD, 2008, pp. 121–132.
[34] E. Kushilevitz, R. Ostrovsky, Replication is not needed: single database, computationally-private information retrieval, in: IEEE FOCS, 1997, pp. 364–373.
[35] W. He, X. Liu, H. Nguyen, K. Nahrstedt, T. Abdelzaher, PDA: privacy-preserving data aggregation in wireless sensor networks, in: IEEE INFOCOM, 2007, pp. 2045–2053.
[36] J. Shi, R. Zhang, Y. Liu, Y. Zhang, Prisense: privacy-preserving data aggregation in people-centric urban sensing systems, in: IEEE INFOCOM, 2010, pp. 1–9.
[37] S. Li, H. Zhu, Z. Gao, X. Guan, K. Xing, X. Shen, Location privacy preservation in collaborative spectrum sensing, in: IEEE INFOCOM, 2012, pp. 729–737.
[38] E. Troja, S. Bakiras, Leveraging P2P interactions for efficient location privacy in database-driven dynamic spectrum access, in: ACM GIS, 2014, pp. 453–456.
[39] B. Chor, E. Kushilevitz, O. Goldreich, M. Sudan, Private information retrieval, J. ACM 45 (6) (1998) 965–981.
[40] C. Cachin, S. Micali, M. Stadler, Computationally private information retrieval with polylogarithmic communication, in: EUROCRYPT, 1999, pp. 402–414.
[41] H. Lipmaa, An oblivious transfer protocol with log-squared communication, in: Information Security, 2005, pp. 314–328.
[42] J. Groth, A. Kiayias, H. Lipmaa, Multi-query computationally-private information retrieval with constant communication rate, in: PKC, 2010, pp. 107–123.
[43] S.C. Pohlig, M.E. Hellman, An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (corresp.), IEEE Trans. Inf. Theory 24 (1) (1978) 106–110.
[44] R. Begleiter, R. El-Yaniv, G. Yona, On prediction using variable order markov models, J. Artif. Intel. Res. (2004) 385–421.
[45] S. Papadopoulos, S. Bakiras, D. Papadias, Pcloud: a distributed system for practical PIR, IEEE Trans. Depend. Secure Comput. 9 (1) (2012) 115–127.

Erald Troja received his BS degree in Computer Science from Brooklyn College, NY in 1999, his M.S in Information Systems and Business Administration from Brooklyn College, NY in 2009, M.S in Computer Science from the Graduate Center, CUNY in 2012, and Ph.D from the Graduate Center, CUNY in 2015. His major research areas include dynamic spectrum access, cognitive radio networks and security and privacy in location based services.

Spiridon Bakiras received the BS degree in Electrical and Computer Engineering from the National Technical University of Athens in 1993, the MS degree in Telematics from the University of Surrey in 1994, and the PhD degree in Electrical Engineering from the University of Southern California in 2000. Currently, he is an associate professor in the College of Science and Engineering at Hamad bin Khalifa University, Qatar. Before that, he held teaching and research positions at Michigan Technological University, the City University of New York, the University of Hong Kong, and the Hong Kong University of Science and Technology. His current research interests include database security and privacy, mobile computing, and spatiotemporal databases. He is a member of the ACM and a recipient of the US National Science Foundation (NSF) CAREER award.