Private Information Retrieval Techniques for Enabling Location Privacy in Location-Based Services*

Ali Khoshgozaran and Cyrus Shahabi

University of Southern California
Department of Computer Science
Information Laboratory (InfoLab)
Los Angeles, CA 90089-0781
[jafkhosh,shahabi]@usc.edu

Abstract. The ubiquity of smartphones and other location-aware handheld devices has resulted in a dramatic increase in popularity of location-based services (LBS) tailored to user locations. The comfort of LBS comes with a privacy cost. Various distressing privacy violations caused by sharing sensitive location information with potentially malicious services have highlighted the importance of location privacy research aiming to protect user privacy while interacting with LBS.
The anonymity and cloaking-based approaches proposed to address this problem cannot provide stringent privacy guarantees without incurring costly computation and communication overhead. Furthermore, they mostly require a trusted intermediate anonymizer to protect a user's location information during query processing. In this chapter, we review a set of fundamental approaches based on *private information retrieval* to process range and k-nearest neighbor queries, the elemental queries used in many Location Based Services, with significantly stronger privacy guarantees as opposed to cloaking or anonymity approaches.

1 Introduction

The increasing availability of handheld computing devices and their ubiquity have resulted in an explosive growth of services tailored to a user's location. Users subscribe to these *location-aware* services and form spatial queries (such as range or k-nearest neighbor search) to enquire about the location of nearby points of interest (POI) such as gas stations, restaurants and hospitals. Processing these queries requires information about the location of the query point or a query

* This research has been funded in part by NSF grants IIS-0238560 (PECASE), IIS-0534761, IIS-0742811 and CNS-0831505 (CyberTrust), and in part from the METRANS Transportation Center, under grants from USDOT and Caltrans. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

region of interest. However, providing this information to a potentially untrusted location-based server has serious privacy implications as it can easily reveal the querying user's location information. Misusing this sensitive information as well as other malpractices in handling such data have resulted in a variety of distressing and increasingly more concerning privacy violations.

Similar to many other existing approaches in areas such as data mining and databases, various techniques based on the K-anonymity principle [1] have been extensively used to provide location privacy [2–7]. With these approaches, usually a trusted third party known as the *anonymizer* is used to ensure that the probability of identifying the querying user remains under $\frac{1}{K}$ where K is the size of the anonymity set received by the untrusted location server. Alternatively, users can generate the anonymity set in a decentralized fashion. With these approaches, the user's location is usually *cloaked* in a larger region which includes other users to make it harder for the untrusted server to locate the querying user. Aside from requiring users to trust a third party during query processing (or all other users for the decentralized case), recent studies [8–11] have shown that such approaches suffer from many drawbacks such as an insufficient guarantee of perfect privacy, vulnerability to correlation attacks and a huge performance hit for privacy paranoid users. To overcome such restrictions, a new class of transformation-based techniques have emerged that map a user's location to a space unknown to the untrusted server. Using the query transformation process, the untrusted server is blinded while processing spatial queries to ensure location privacy [8, 12]. Although these approaches mitigate some of the privacy implications of the anonymity and cloaking-based approaches, they cannot provide strong privacy guarantees against more sophisticated adversaries. Finally, several cryptographic-based approaches are proposed for location privacy which utilize two-party computation schemes to achieve privacy [13–15]. While these approaches can provide strong privacy guarantees, they suffer from yet another drawback. They cannot avoid a linear scan of the entire data and thus are not efficient for real-world scenarios.

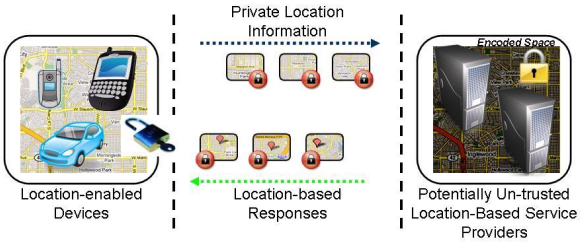


Fig. 1. Location Privacy in LBS

In this chapter, we review two fundamental approaches that go beyond the conventional approaches proposed for location privacy and devise frameworks to

eliminate the need for an anonymizer in location-based services and satisfy significantly more stringent privacy guarantees as compared to the anonymity/cloaking-based approaches. Both of these techniques are based on the theory of *Private Information Retrieval* (PIR) to protect sensitive information regarding user locations from malicious entities. Using a PIR protocol, a client can retrieve a database item hosted at an untrusted server without revealing which item is retrieved from the host (Figure 1). Although PIR can be used to privately generate a query result set, avoiding a linear private scan of the entire object space is challenging. This is due to the fact that the server owning the objects information cannot be trusted to perform the query processing and choose what to be returned as responses. Alternatively, moving this knowledge to the users will require the query processing to happen at the client side which is very costly. Utilizing spatial partitionings based on PIR, these approaches devise private algorithms that significantly reduce the amount of information that is privately transferred to the querying clients from the untrusted server. While the first approach relies on hardware-based PIR techniques [10, 16], the second one employs computational PIR protocols to provide location privacy [9]. We elaborate on each approach and detail the merits and shortcomings of both techniques.

The remainder of this chapter is organized as follows. In Section 2, we review some related work. Section 3 details various PIR schemes and in Section 4, we first provide our trust and threat model and then show how PIR can be used to design privacy-aware spatial query processing. Sections 5 and 6 elaborate on how hardware-based and computational PIR techniques are employed to enable private evaluation of spatial queries, respectively. In Section 7, we discuss some of the limitations of PIR-based approaches and finally, Section 8 concludes the book chapter with possible future research directions.

2 A Brief Survey

Protecting a user's private location information while interacting with location-based services has been the subject of many recent research efforts. These studies can be broken into four fundamentally different groups of anonymity/cloaking, transformation, cryptographic and PIR-based approaches.

Anonymity and Cloaking-Based Approaches: The earlier work on location privacy focused on protecting a user's private location information by disguising it among $K-1$ other user locations or extending it from a *point* location to an area (spatial extent). With the first approach, user u, as well as $K-1$ other user locations form an *anonymity set* which is sent to the server instead of u's precise location. Similarly, with cloaking techniques, the resulting *cloaked* region (which contains u and several other users) is sent to the server. These techniques try to ensure the user's location cannot be distinguished from the location of the other $K-1$ users or the exact user location within the cloaked region is not revealed to the untrusted server responding to location queries. Depending on the method used, the untrusted server executes the query for every object in the anonymity set or for the entire cloaked region. Several techniques

based on cloaking and K-anonymity have been proposed in the literature to reduce the probability of identifying a user's location [2–7].

Cloaking and K-anonymity approaches have some important limitations. First, by design the majority of cloaking approaches rely on a trusted intermediary to "anonymize" user locations which means all queries should involve the *anonymizer* during the system's normal mode of operation. The anonymization can also be performed in a decentralized fashion among users which means each user has to trust all other users in the system with her location. In other words, while users do not trust the location server, they either have to trust another third party, as sophisticated as the server or all other users. Second, a limitation of cloaking techniques in general is that either the quality of service or overall system performance degrades significantly as users choose to have more strict privacy preferences. Third, many of the cloaking techniques are subject to attacks that exploit the information from the formation of the cloaked region or the history of user movement to *infer* precise user location [9]. Fourth, the concept of K-anonymity does not work in all scenarios. For example, in a less populated area, the size of the extended area can be prohibitively large in order to include $K-1$ other users. Even worse, not enough number of users may be subscribed to the service to construct the required cloaked region. Finally, these techniques assume that all users are trustworthy. However, if some of them are malicious, they can easily collude to compromise the privacy of a targeted user. The interested reader might refer to the "Anonymity and Historical-Anonymity in Location-Based Services" chapter for a more detailed description of the anonymity and cloaking techniques.

Transformation-Based Approaches: A second class of approaches emerging to mitigate some of the weaknesses of the techniques discussed above are based on query transformation to prevent the server from learning information about user locations. The first work to utilize spatial transformation techniques for location privacy is [8]. In this study, space filling curves are utilized as one-way transformations to encode the locations of both users and points of interest into an encrypted space and to evaluate a query in this transformed space. The transformed space maintains the distance properties of the original space which enables efficient evaluation of location queries in the transformed space. Subsequently, upon receiving transformed query results, users can reverse the transformation efficiently using the trapdoor information which is only provided to them and are protected from the server. Recently, Yiu et al. proposed a framework termed SpaceTwist to blind an untrusted location server by incrementally retrieving points of interest based on their ascending distance from a fake location near the query point termed the anchor point [12]. Note that with this approach, the query is still evaluated in the original space but the query point is transformed to an anchor point.

The key advantage of transformation-based approaches over the anonymity and cloaking-based techniques is the elimination of the need for a trusted third party during the query processing. Furthermore, [8] provides very efficient query processing without compromising privacy and [12] utilizes the existing query

processing index structures present in non-privacy aware servers which makes it readily applied to existing location servers. However, the nearest neighbor algorithm of [8] is approximate and [12] suffers from several privacy leaks and costly computation/communication if exact results and strict privacy are required simultaneously. Furthermore, it offers no lower bound for the size of the privacy region where it can become even smaller than a cloaked region.

Cryptographic-Based Approaches: This class of techniques blind the untrusted party (i.e., the server or another user) by utilizing secure multi-party computation schemes. The protocol proposed in [14] privately evaluates the distance between Alice's point and other n points that Bob owns. After executing the protocol, Bob knows nothing about Alice's point and Alice only learns the nearest neighbor from Bob's points. Although the solution proposed is mainly of theoretical interest and does not focus on spatial queries or LBS, it can be considered as a method for providing location privacy in LBS. Zhong, Goldberg and Hengartner propose three solutions to what they define as the "nearby-friend problem" [15]. The problem is defined as allowing users to learn information about their friends' locations if and only if their friends are actually nearby. The three protocols are all efficient in terms of the amount of computation and communication required by each party. Each protocol is an instance of a multi-party computation scheme with certain strengths and restrictions (in terms of number of messages transferred and the resilience to a malicious party). Finally, Zhong et al. provide two protocols aiming at protecting user locations in LBS. While the first protocol allows a user to share her location information with other users via an untrusted server, the second protocol enables a dating service where a user learns whether other users with similar profiles (found by the server) are located in the same region she is located [13]. This protocol, which is of more interest to location privacy in LBS, assumes the entire user profile is known to the server, and the server first finds any potential matches between a user and all other users. The server then sends all matched profiles to the requester so that she can blindly compare their locations with her own location. Similar to the other protocols discussed above, a multi-party computation protocol is proposed which involves the requester, the dating service and any other matched user.

The main advantage of the three methods discussed above is their strong privacy guarantees. Building their framework on well-known cryptographic primitives and widely used one-way functions, these protocols do not suffer from privacy leaks of cloaking/anonymity and transformation-based methods. Furthermore, their problem-specific designs allow very efficient implementations of the protocols mostly involving only a handful of computations and few message transfers. However, the fundamental disadvantage of the protocols discussed in this category is their high computation or communication complexity when being used for spatial query processing. For instance, in [14] the distance between query point and each and every point of interest must both be computed or transferred to the client, i.e., $O(n)$ computation or communication complexity where n is the size of the database. This is because the points of interest are treated as vectors with no exploitation of the fact that they are in fact points

in space. Therefore, the main limitation of cryptographic-based techniques discussed above is the loss of spatial information via encryption. This loss either results in a linear scan of the entire database if used to evaluate a spatial query (as in [14]), or makes the protocol unusable for spatial query processing (as in [15, 13]). Similarly, with the protocols proposed in [15], Alice will know whether a certain user Bob is nearby. However, verifying whether a certain friend is nearby Alice is a different problem than finding Alice's nearest friends. Finally, the work of Zhong et al. suffers from the same drawbacks since it only describes a matching protocol in an encrypted space between two users located in the same region. However, the real challenge is finding nearby matches which is not possible in an encrypted space.

PIR-Based Approaches: Several approaches discussed so far attempt to improve the efficiency or the privacy aspects of evaluating spatial queries privately in LBS. However, they mostly suffer from a privacy/quality of service trade-off. While on one extreme end the cryptographic-based techniques provide perfect privacy, they result in very costly spatial query processing schemes. Likewise, on the other side of the spectrum, efficient cloaking or spatial transformation approaches might result in severe privacy leaks under certain user, object or query distributions. The approaches studied in this category are based on the solutions proposed to the well-known problem of *Private Information Retrieval* (PIR) discussed in Section 1. These approaches construct private spatial indexes on top of PIR operations to provide efficient spatial query processing, while the underlying PIR scheme guarantees privacy.

In this chapter, we discuss two location privacy schemes based on hardware-based [10, 16] and computational PIR [9] protocols. The former approach superimposes a regular grid on the data and uses PIR to privately evaluate range and kNN queries. The latter technique supports approximate and exact nearest neighbor query evaluation by utilizing various 1-D and 2-D partitionings to index the data and then restructuring partitions into a matrix that can be privately queried using PIR. Sections 5 and 6 detail these two approaches and Section 7 compares the two techniques and highlights the strengths and weaknesses of each approach. To the best of our knowledge, the only other study to propose employing PIR to provide location privacy is [11] which presents an architecture that uses PIR and trusted computing to hide location information from an untrusted server. With this approach, PIR is used to prevent the untrusted location server from learning user locations and trusted computing is used to ensure users that the PIR algorithm and other services provided by the server are only performing the operations as intended. In fact, similar to hardware-based PIR, [11] places a trusted module as close as possible to the untrusted host to disguise the selection of records. However, the proposed techniques do not specifically focus on spatial query processing (such as range and kNN) and the proposed architecture is not yet implemented.

In summary, the novel approaches discussed in this chapter do not rely on anonymizers for query processing. Furthermore, they do not suffer from the privacy vulnerabilities of cloaking/anonymity and transformation-based approaches

or the prohibitive communication and computation costs of cryptographic-based techniques. In the next section, we formally define the PIR problem and detail some of its implementations.

3 Private Information Retrieval

Private Information Retrieval, in its most abstract setting, enables a user to query an item from a database without disclosing the queried item to the (potentially) untrusted server hosting the data. Suppose Alice owns a database D of n bits and Bob is interested to retrieve the i^{th} bit from the database. PIR protocols allow Bob to privately retrieve $D[i]$ without disclosing i to Alice. This definition of PIR offers a theoretical point of view. In a more practical scheme, users are interested in privately retrieving blocks of data (or records) [17].

Given the privacy requirements of users, PIR approaches can be divided based on whether they provide *Information Theoretic* or *Computational* privacy. While the former class of approaches guarantee privacy against an adversary with unbounded computational power, the latter class assumes a computationally bounded adversary. Therefore, the information theoretic approaches guarantee perfect privacy while the security of the computational approaches relies on the intractability of a computationally complex mathematical problem, such as Quadratic Residuosity Assumption [18]. However, the perfect privacy of the first group comes with a prohibitive cost. In fact, Chor et al. have proved the communication cost of such techniques to be $\Omega(n)$ for a database of n bits [17]. Furthermore, the server's computation cost is also linear since not processing any single database record r indicates to the server that r is not requested by the user and thus by definition violates the privacy requirement of PIR. Therefore, while being of theoretical interest, information theoretic PIR cannot efficiently be integrated into data-intensive and practical applications. The computational PIR approaches, on the other hand, achieve significantly lower complexity by assuming some limitations on the server's computational power.

While computational PIR incurs more reasonable costs for retrieving objects, the proposed PIR protocols are still expensive and require a significant amount of server resources. In other words, although they can improve the communication complexity, all database records still have to be processed at the server. In fact, Sion et al. argue in [19] that the cost of privately retrieving database items from the server is significantly higher than sending the entire database to the client. This argument has certain important restrictions [9] and we show that practical results can be achieved by avoiding some of the redundant PIR operation costs. However, the per item cost of computational PIR approaches are still high.

To obtain perfect privacy while avoiding the high cost of the approaches discussed above, a new class of *Hardware-based PIR* approaches has recently emerged which places the trust on a tamper-resistent hardware device. These techniques benefit from highly efficient computations at the cost of relying on a hardware device to provide privacy [20–23]. Placing a trusted module very close

to the untrusted host allows these techniques to achieve optimal computation and communication cost compared to the computational PIR approaches.

In this chapter, we show how recent studies have used these two classes of PIR approaches to enable location privacy for range and kNN queries. We stress that there are various other versions of the PIR problem that we do not consider in this chapter. For instance, the PIR techniques we have discussed so far are all instances of *single server* PIR schemes. A different set of studies focus on multi-server PIR protocols. Here, it is assumed that several servers exist to execute the PIR operations with the requirement that these servers should not be able to communicate or collaborate with each other. Under this assumption of non-communicating servers, it is possible to achieve sub-linear communication complexity [24, 17, 25]. However, the underlying non-collusion assumptions made by these studies are rather hard to achieve and makes it difficult to develop practical schemes based on these multi-server PIR protocols. In another setting, *symmetric PIR* strives to protect server privacy as well as user privacy by limiting user's knowledge to the physical value of the retrieved item [26]. Such schemes are not of particular interest in the context of location-based services as we assume server data is publicly available and thus server privacy is not as important as user privacy. We detail our trust and threat model assumptions in Section 4.1.

4 PIR in Location Privacy

With many location-based services, users carrying location-aware handheld devices are interested in finding the location of nearby points of interest (POI) such as restaurants and hotels. Users form various spatial queries such as range or kNN queries to request such information. Therefore, the location of the query point (or region), as well as the query result set usually reveal the location of the user. The key idea behind using PIR techniques for location privacy is to prevent the untrusted location server from learning any information about a query and its result set. Using PIR, users can request information about their locations of interest without revealing any information about their whereabouts. However, a major challenge in performing this task is that users are essentially unaware of the way records are indexed on the untrusted server and hence cannot directly request the records that might contain their desired information. Therefore, avoiding a linear scan or full transfer of the entire server database is challenging. This is due to the fact that the server owning the objects information cannot be trusted to perform the query processing and choose what to be queried. Alternatively, moving this knowledge to users will require the query processing to happen at the client side which is very costly. For the rest of this section, we discuss how private PIR-based spatial algorithms address this issue. We first define the trust and the threat models and then detail how spatial queries are translated to PIR requests.

4.1 Trust and Threat Model

We consider a model in which users query a central untrusted server for POI data. While users trust their client devices to run legitimate software, they do

not trust any other entity in the system including the location server (henceforth denoted by LS). Users might collude with LS against other users and thus from each user's point of view, all other users as well as LS can be adversarial. LS owns and maintains a database of POIs and responds to users queries as a service provider. Users subscribe to LS's services. As part of our threat model, we assume that the server's database is publicly accessible and available and thus an adversary can perform the so-called *known plaintext attack*.

As we discussed earlier, an adversary's goal is to find a user's location information. Therefore, the obvious objective of any location privacy scheme is to protect such private information from potentially malicious servers and other adversaries. In order to achieve location privacy, a user's location and identity information, as well as the identity of query results should be kept secret both on the server and during query evaluation [8].

We assume there is a secure communication channel between users and LS and thus the connection cannot be sniffed by adversaries. However, the server can gain valuable information from user queries as well as their result sets and therefore, these entities should not leak any information to an adversary. Based on our assumption of a secure client-server communication channel, no adversary can learn about a user's location without colluding with the server. Therefore, for the rest of this chapter, we only focus on the location server as the most powerful adversary and assume that adversaries are computationally bounded.

4.2 Converting Spatial Queries to PIR Requests

One important property of the PIR problem is the underlying assumption on the protocol usage; it is assumed that the bits (or records) are stored in an array and users know the index of the element they wish to retrieve. Therefore, the key problem is to enable users to privately map location queries into their corresponding record indexes of the database.

With this problem characteristic in mind, it is clear that the key step in utilizing PIR for spatial query processing is to devise efficient schemes which allow users to find objects relevant to their queries that should be privately retrieved from a remote database. In this section, we elaborate this argument and discuss several techniques to utilize PIR for location privacy. In particular, we study the recent work that addresses private evaluation of range, NN and kNN queries. In Section 5, we discuss how hardware-based PIR techniques are employed to enable location privacy. Similarly, Section 6 presents an approach to privately evaluate nearest neighbor queries using computational PIR.

5 Location Privacy with Hardware-based PIR Protocol

The class of hardware-based PIR techniques utilize a secure coprocessor to disguise the selection of records that are requested by the user from an untrusted server. The secure coprocessor performs certain operations that prevent the server from learning the requested record from database items read by the

coprocessor. In this section, we review how a secure coprocessor is used to implement a PIR protocol to privately and efficiently retrieve a selected record from a database.

5.1 Hardware-Based PIR

A Secure Coprocessor (SC) is a general purpose computer designed to meet rigorous security requirements that assure unobservable and unmolested running of the code residing on it even in the physical presence of an adversary [23]. These devices are equipped with hardware cryptographic accelerators that enable efficient implementation of cryptographic algorithms such as DES and RSA [19].

Secure coprocessors have been successfully used in various real-world applications such as data mining [27] and trusted co-servers for Apache web-server security [28] where the server hosting the data is not trusted. The idea behind using a secure coprocessor for performing the PIR operations is to place a trusted entity as close as possible to the untrusted host to disguise the selection of desired records within a black box.

Placing a secure coprocessor between user queries and the untrusted server raises the following simple yet important question. Why should one not trust a location server if the secure coprocessor is to be trusted? The response to this question is based on several fundamental differences between trusting a secure processor versus a location server. First, aside from being built as a tamper resistent device, the secure coprocessor is a hardware device specifically programmed to perform a given task while a location server consists of a variety of applications using a shared memory. Secondly, unlike the secure coprocessor in which the users only have to trust the designer, using a location server requires users to trust the server admin and all applications running on it as well as its designer. Last but not least, in our setting, the secure coprocessor is mainly a *computing* device that receives its necessary information, per session from the server, as opposed to a server that both *stores* location information and *processes* spatial queries.

We build our location privacy scheme based on the PIR protocols proposed in [20, 21] to achieve optimal (i.e., constant) query computation and communication complexity at the cost of performing as much offline precomputation as possible. We now provide an overview of our utilized PIR protocol.

Definition 1. Random Permutation: For a database DB of n items the random permutation π transforms DB into DB_π such that $DB[i] = DB_\pi[\pi[i]]$. For example for $DB = \{o_1, o_2, o_3\}$ and $DB_\pi = \{o_3, o_1, o_2\}$ the permutation π represents the mapping $\pi = \{2, 3, 1\}$. Therefore $DB[1] = DB_\pi[\pi[1]] = DB_\pi[2] = o_1$, $DB[3] = DB_\pi[\pi[3]] = DB_\pi[1] = o_3$, etc. It is easy to verify that the minimum space required to store a permutation π of n records is $n \log n$ bits.

In order to implement the PIR protocol, we first use the secure coprocessor to privately shuffle and encrypt the items of the entire dataset DB using a random permutation π. Once the shuffling is performed, DB_π is written back to the server while SC keeps π for itself. To process a query $q = DB[i]$, a user u encrypts q using SC's public key and sends it to SC through a secure channel.

SC decrypts q to find i and retrieves $DB_\pi[\pi[i]]$ from the server, decrypts and then re-encrypts it with u's public key and sends it back to u (hereinafter we distinguish between a *queried item* which is the item requested by the user and *retrieved/read record* which is the item SC reads from DB_π). However, the above scheme is not yet private as (not) retrieving the same object for the second query reveals to the server that the two queries are (different) identical. Therefore, SC has to maintain a list L of all items retrieved so far. SC also caches the records retrieved from the beginning of each session. In order to answer the kth query, SC first searches its cache. If the item does not exist in its cache, SC retrieves $DB_\pi[\pi[k]]$ and stores it in its cache. However, if the element is already cached, it randomly reads a record not present in its cache and caches it. With this approach, each record of the database might be read at most once regardless of what items are queried by users. This way, an adversary monitoring the database reads can obtain no information about the record being retrieved. The problem with this approach is that after $T_{threshold}$ retrievals, SC's cache becomes full. At this time a *reshuffling* is performed on DB_π to clear the cache and L. Note that since $T_{threshold}$ is a constant number independent of n, query computation and communication cost remain constant if several instances of reshuffled datasets are created offline [21], alternatively shuffling can be performed regularly on the fly which makes the query processing complexity equal to the complexity of the shuffling performed regularly.

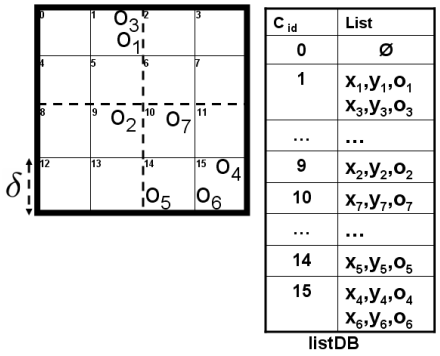
Algorithm 1 *read*(*permuted database* DB_π, *index* i)

Require: DB_π, T {Threshold}, L {Retrieved Items}
1: **if** $(|L| \geq T)$ **then**
2: $DB_\pi \leftarrow$ Reshuffle DB_π using a new random permutation π;
3: $L \leftarrow \emptyset$;
4: Clear SC's cache
5: **end if**
6: **if** $i \notin L$ **then**
7: $record \leftarrow DB_\pi[\pi[i]]$;
8: Add $record$ to SC's cache
9: $L = L \cup \{i\}$;
10: **else**
11: $r \leftarrow$ random index from $DB_\pi \backslash L$;
12: $temp \leftarrow DB_\pi[\pi[r]]$;
13: Add $temp$ to SC's cache
14: $L = L \cup \{r\}$;
15: **end if**
16: return $record$;

Algorithm 1 illustrates the details of the *read* operation which privately retrieves an element from the database. Note that it reads a different record per query and thus ensures each record is accessed at most once. All details regarding the shuffling, reshuffling, permutation etc. are hidden from the entity interacting with DB_π.

So far we have enabled private retrieval from an untrusted server. However, we have not focused on how spatial queries can be evaluated privately. Section 5.1 enables replacing a normal database in a conventional query processing with its privacy-aware variant. However, the query processing needs to be able to utilize

this new privacy-aware database as well. Note that what distinguishes our work from the use of encrypted databases is the impossibility of blindly evaluating a sophisticated spatial query on an encrypted database without a linear scan of all encrypted items. In this section, we propose private index structures that enable blind evaluation of spatial queries efficiently and privately. Using these index structures, we devise a sweeping algorithm to process range queries in Section 5.2. Similarly, we detail a Spiral and a Hilbert-based approach (or Hilbert for short) to privately evaluate kNN queries in Section 5.3.



C_{id}	List
0	Ø
1	x_1,y_1,o_1 x_3,y_3,o_3
...	...
9	x_2,y_2,o_2
10	x_7,y_7,o_7
...	...
14	x_5,y_5,o_5
15	x_4,y_4,o_4 x_6,y_6,o_6

listDB

Fig. 2. The Object Space (left) and listDB Index (right)

5.2 Private Range Queries

As we discussed in Section 1, the key idea behind employing spatial index structures is to avoid the private retrieval of database objects not relevant to user queries. However, a challenge in designing these indexes raises from the fact that while they are stored at the untrusted host, query processing cannot be performed by the server. This requirement forces us to use efficient indexes that can quickly identify the subset of database records that should be privately retrieved. For processing range (and kNN) queries, we utilize a regular $\delta \times \delta$ grid to index objects within each cell. The key reason behind using a grid structure in our framework is while being efficient, grids simplify the query processing. Several studies have shown the significant efficiency of using the grid structure for evaluating range, kNN and other types of spatial queries [29–31].

Without loss of generality, we assume the entire area enclosing all objects is represented by a unit square. The grid index uniformly partitions the unit square into cells with side length δ ($0 < \delta < 1$). Each cell is identified by its cell ID (c_{id}) and may contain several objects each being represented by the triplet $< x_i, y_i, obj_{id} >$. These cells are then used to construct the *listDB* index which stores the objects and their location information for each cell. The *listDB*

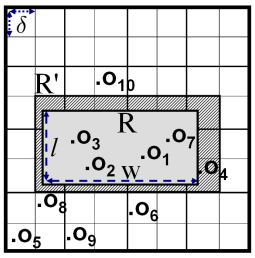


Fig. 3. Range Query Processing

schema represents a flat grid and looks like $< c_{id}, list >$ where $list$ is a sequence of triplets representing objects falling in each grid cell. Figure 2 illustrates the original object space and the *listDB* index. There is an obvious trade-off between two competing factors in choosing the right value of δ. As δ grows, a coarser grid (having less cells) decreases the total number of cell retrievals during query processings. This is desirable given the relatively high cost of each private read from the database. However, large cells result in retrieving more excessive (unneeded) objects which coexist in the cell being retrieved. These excessive objects result in higher computation and communication complexity which increase the overall response time. These trade-offs and the discussion on choosing the right grid granularity are studied in more detail in [10, 16].

Using the *listDB* index, processing range queries is straightforward. During the offline process, database records are created each containing a *listDB* entry which corresponds to all objects within a cell. To ensure that the server cannot distinguish between the records based on differences in record sizes (which is affected by object distributions), each record is padded to reach the maximum record length. Next, SC generates a random permutation π and privately shuffles the and encrypts *listDB*. The encrypted shuffled version of *listDB* is then written back to the server. During the range query processing, SC uses a sweeping algorithm to privately query the server for all cells which overlap with the specified range. A range query $range(R)$ is defined as a rectangle of size $l \times w$ $(0 < l, w \leq 1)$. Therefore, to answer each range query using *listDB*, we must first find the set of cells R' that encloses R. R' forms a $L \times W$ rectangular grid where $L \leq \lceil \frac{l}{\delta} \rceil + 1$ and $W \leq \lceil \frac{w}{\delta} \rceil + 1$. The function $read(listDB, c_{id})$ privately queries $listDB$ and performs the necessary processing to return a list of all objects enclosed in a c_{id}. We use the sweeping algorithm to query the cells in R' privately.

5.3 Private kNN Queries

The main challenge in evaluating kNN queries originates from the fact that the distribution of points can affect the size of the region R that contains the result set (and hence the cells that should be retrieved). In other words, no region is

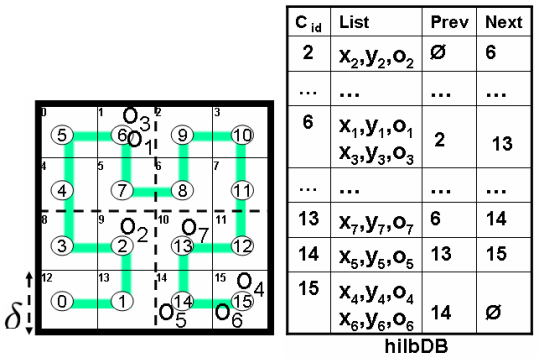
guaranteed to contain the k nearest objects to a query point (except in a uniform distribution) which implies that R has to be progressively computed based on object distributions. Therefore, it is important to minimize the total number of cells that should be privately queried. In this Section, we propose two variants of evaluating kNN queries and discuss how our index structures allow us to query only a small subset of the entire object space. Note that due to the strong similarity of these algorithms with their *first nearest neighbor* counterparts (i.e., $k = 1$), we directly consider the more general case of kNN.

Similar to range query processing, we can use regular grids and the *listDB* index to perform kNN queries. However, despite its simplicity, *listDB* might not be an efficient index for skewed object distributions as the kNN algorithm utilizing *listDB* might experience a performance degradation caused by numerous empty cells in the grid. Increasing δ does not solve this problem as it results in coarse-grained cells containing many excessive objects that have to be queried/processed and even a linear decrease in δ incurs at least a quadratic increase in the number of empty cells. Therefore, we also introduce a *hilbDB* index which uses Hilbert space filling curves [32] to avoid the stated shortcomings of processing kNN queries using regular grids. The main intuition behind using Hilbert curves is to use their locality preserving properties to efficiently approximate the nearest objects to a query point by only indexing and querying the non-empty cells. This property significantly reduces the query response time for skewed datasets [16].

We define H_2^N ($N \geq 1$), the N^{th} order Hilbert curve in a 2-dimensional space, as a linear ordering which maps an integer set $[0, 2^{2N} - 1]$ into a 2-dimensional integer space $[0, 2^N - 1]^2$ defined as $H = \nu(P)$ for $H \in [0, 2^{2N} - 1]$, where P is the coordinate of each point. The output of this function is denoted by *H-value*.

To create the *hilbDB* index, an H_2^N Hilbert curve is constructed traversing the entire space. After visiting each cell C, its $c_{id} = \nu(C)$ is computed. We use an efficient bitwise interleaving algorithm from [33] to compute the H-values (the cost of performing this operation is $O(n)$ where n is the number of bits required to represent a Hilbert value). Next, similar to the *listDB* index, the c_{id} values are used to store object information for each cell. Finally, in order to guide the next retrieval, each record also keeps the index of its non-empty c_{id} neighbors in *hilbDB*, stored in the *Prev* and *Next* columns, respectively. These two values allow us to find out which cell to query next from *hilbDB* hosted at the server. Figure 4 illustrates the original object space and the *hilbDB* index. The circled numbers denote each cell's c_{id} constructed by H_2^2 for the *hilbDB* index. For clarity, we have not shown that all records are in fact encrypted.

Using the two private indexes discussed above, the following general approach is employed by both of our kNN algorithms: *(i)* create a region R and set it to the cell containing the query point q *(ii)* expand R until it encloses at least k objects *(iii)* compute the *safe region* R' as the region guaranteed to enclose the result set and *(iv)* find the actual k nearest objects in R' using $range(R')$ defined in Section 5.2. The main difference among the two algorithms is related to how they perform the step *(ii)* mentioned above. Note that, as Figure 5 illustrates,



C_{id}	List	Prev	Next
2	x_2,y_2,o_2	Ø	6
…	…	…	…
6	x_1,y_1,o_1 x_3,y_3,o_3	2	13
…	…	…	…
13	x_7,y_7,o_7	6	14
14	x_5,y_5,o_5	13	15
15	x_4,y_4,o_4 x_6,y_6,o_6	14	Ø

Fig. 4. The Hilbert Curve Ordering of Objects (left) and the hilbDB Index (right)

regardless of the approach, R is not guaranteed to contain the actual k nearest neighbors of q and therefore has to be expanded to the safe region R' (e.g., $O_7 \in 2NN(q)$ but $O_7 \notin R$ and $O_2 \in R$ although $O_2 \notin 2NN(q)$). As shown, if relative location of q and its furthest neighbor in R is known, a safe region can be constructed. However, this comes at the cost of querying *listDB* to retrieve the location of objects in R. This is in fact useful since $R \subset R'$ which means R has to be retrieved sometime during the query processing. Therefore, as an optimization, by querying every cell in R during the safe region computation, we can avoid querying them again for $range(R')$. It is easy to verify that R' is a square with sides $2 \times \lceil ||c_q - far_q(k)|| \rceil$ where c_q is the cell containing q and $far_q(k)$ is the cell containing q's k^{th} nearest object in R and $||.||$ is the Euclidean norm [31]. We now elaborate on how different expansion strategies for step *(ii)* mentioned above generate different results and discuss the pros and cons of each strategy.

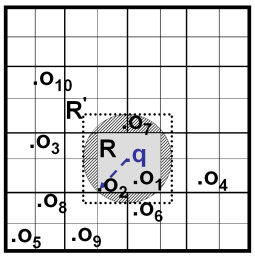


Fig. 5. Computing the Safe Region R'

Spiral Expansion With this approach, if k objects are not found in the cell containing the query point, we expand the region R in a spiral pattern until it

encloses k objects. The most important advantage of this method is its simple and conservative expansion strategy. This property guarantees that the spiral expansion minimizes R. However, the very same conservative strategy might also become its drawback. This is because for non-uniform datasets, it takes more time until the algorithm reaches a valid R. Note that the Spiral Expansion only uses the *listDB* index to evaluate a kNN query. Figure 6 (left) illustrates the order in which grid cells are examined.

Hilbert Expansion The spiral expansion strategy, and in general similar linear expansion strategies such as the one proposed in [31], are usually very efficient in finding the safe region for the query q due to their simplicity. However, the time it takes to find the region that includes at least k objects can be prohibitively large for certain datasets. The Hilbert expansion overcomes this drawback by navigating in the *hilbDB* index which only stores cells that include at least one object. The main advantage of using *hilbDB* is that once the cell with closest H-value to $\nu(q)$ is found, expanding the search in either direction requires only $k-1$ more private reads to generate a safe region. This 1-dimensional search gives a huge performance gain at the cost of generating a larger search region due to the asymmetric and 1-dimensional Hilbert expansion. These trade-offs are extensively studied in [16]. Figure 6 (right) shows how cells containing objects are examined based on their Hilbert ordering.

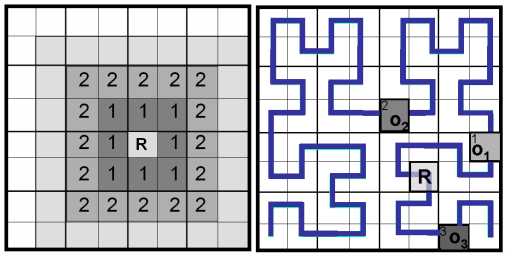


Fig. 6. kNN Query Processing

6 Location Privacy with Computational PIR Protocol

To avoid the impractical linear lower bound on communication complexity of information theoretic PIR protocols, the privacy requirements can be relaxed from information theoretic to computational secrecy. This latter class of approaches known as computational PIR enforces computational intractability for any adversary to find the item being queried, from the client server communications. We now briefly describe the protocol and its underlying *Quadratic Residuacity Assumption (QRA)* [18] which acts as the basis for the secrecy of the computational PIR protocol.

6.1 Computational PIR

In its theoretical setting, a computational PIR protocol allows a client to query for the value of the i^{th} bit from a server's database D of n bits that enables the client to derive the value of $D[i]$ while preventing the server to learn i. According to the QRA assumption, it is computationally hard to determine the quadratic and non-quadratic residues in modulo arithmetic for a large number N without knowing its two large prime factors q_1 and q_2. However, knowing q_1, q_2, one can efficiently determine the quadratic residues and quadratic non-residues modulo N. More details about the QRA assumption and its PIR usage can be found in [9]. Suppose the user wants to retrieve $D[i]$. Using this property, the server first converts D into a square $t \times t$ matrix M (which can be padded for non-square matrixes). Suppose that $M_{a,b}$ represents the requested value. The user first generates the modulus N using q_1, q_2 and then creates a row vector query message $y = [y_1, \ldots, y_t]$ such that only y_b is a quadratic non-residue and the rest of the vector are quadratic residues. The server computes a response z_r for each matrix row r according to equations 1,2 and sends z back to the user.

$$z = [z_1, \ldots, z_t] \ s.t. \ z_r = \prod_{j=1}^{t} w_{r,j} \tag{1}$$

$$w_{r,j} = y_j^2 \ if \ M_{r,j} = 0 \ or \ y_j \ otherwise \tag{2}$$

Using the Euler criterion and the Legendre symbol [34], the user can efficiently determine whether $M_{a,b} = 0$ or not [9]. Figure 7 illustrates these steps. Note that the above procedure allows the user to privately retrieve a single bit from the server. For objects represented as $m-$bit binary strings, the server can conceptually maintain m matrixes each storing one bit of each object and applying user's vector query message to all m matrixes. Similar to the PIR protocol described in Section 5.1, we assume $read(D_i)$ privately retrieves the i^{th} object from D using the above procedure. Equations 1 and 2 as well as the above discussion can be used to compute the PIR protocol complexity. For each PIR read, server's computation is linear in n and the client sever communication is $O(\sqrt{n})$. This is because the entire matrix is multiplied by the user's query vector and the server response is a column vector of size $O(\sqrt{n})$. Similarly, for objects of size m bits, computation and communication complexities increase to $O(m.n)$ and $O(m\sqrt{n})$, respectively.

From the description of the protocol, it is clear that in order to successfully employ the computational PIR for a database D of n bits, data has to be first transformed into its matrix representation. In other words, D is first translated into a $\lceil\sqrt{n}\rceil \times \lceil\sqrt{n}\rceil$ matrix. Therefore, any private request from the server retrieves $O(\sqrt{n})$ bits to the user. The challenge is then how to organize data into buckets of size $O(\sqrt{n})$ in order to maximize query result accuracy. Note that this requirement is imposed by the employed PIR scheme and is independent of the underlying indexing scheme used. Sections 6.2 and 6.3, detail several underlying spatial indexing techniques proposed in [9] that enable private evaluation of approximate and exact nearest neighbor queries, respectively.

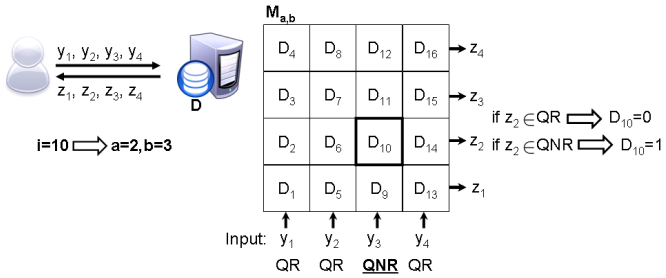


Fig. 7. Computational PIR

6.2 Approximate Private Nearest Neighbor Queries

Querying server's data privately with the computational PIR scheme requires some preprocessing to convert the original data into its PIR matrix representation. However, the challenge is to devise efficient index structures on top of the underlying PIR model to efficiently and privately respond to nearest neighbor queries. Once the data is bucketized into $\sqrt{n}$ pieces offline, PIR can be used to retrieve the section most relevant to the user's query. In this section, we review several 1-D and 2-D data partitioning schemes originally proposed in [9] that use various spatial index structures such as Hilbert curves, kd-trees and R-trees to privately evaluate NN queries.

Hilbert Curves: As we mentioned in Section 5.3, the major property of Hilbert curves is the proximity of objects in the Hilbert space. Using these curves, during the offline preprocessing step, the corresponding Hilbert values are computed for all POIs in the database D. Given a sorted list of these n Hilbert values, binary search can be used to find the nearest POI in terms of its Hilbert distance to the Hilbert value of the query point q in $O(\lg n)$ steps (Figure 8). This point is most probably the closest point to q. However, the logarithmic cost of each PIR request makes it impractical to use the above technique during the query processing. More importantly, each PIR request includes not one but $O(\lg n)$ points in its response. These factors lead to a very costly communication cost of $O(\sqrt{n}lgn)$ POIs for each approximate nearest neighbor search. For instance, finding the nearest neighbor to a point in a database of one million POIs would require the server to transfer approximately $20K$ points to the client while the result might still not be exact.

To mitigate the prohibitive communication cost of a single PIR operation, POIs can be indexed using a B^+-tree of height 2 where each node contains less than $\lceil\sqrt{n}\rceil$ points. The nodes of the B^+-tree represent the columns of the PIR matrix M. Given a query point q, the user u first computes $r = \nu(q)$ which denotes the Hilbert value associated to the query point. All values in a tree leaf are less or equal to their corresponding key stored at the tree root. Therefore, given the root, r is used to determine the tree node (corresponding to a matrix column) that should be retrieved. The user u then computes his nearest neighbor

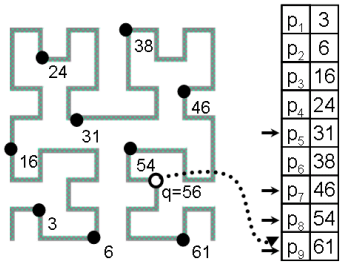


Fig. 8. Hilbert Values of POI Data

from the $\sqrt{n}$ POIs returned in the result set (Figure 9). Although retrieving several POIs at the vicinity of q can greatly improve accuracy, the method remains an approximation technique. Note that any information required by all queries does not need to be retrieved privately. In our case, the tree root is required for processing any query and thus its plain value is first transferred to u. Since the query is evaluated with one PIR request, the communication complexity is $O(\sqrt{n})$. Figure 9 illustrates how the objects in Figure 8 are organized into the 3-way B^+-tree. For $r = 56 = \nu(q)$, u requests the third column of M since $56 > 38$. Next, u computes $NN(q)$ from the result set $\{p_7, p_8, p_9\}$.

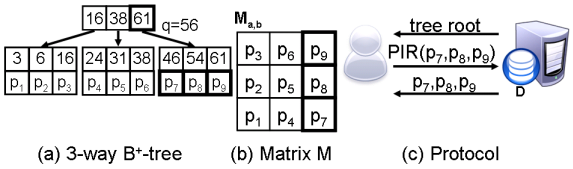


Fig. 9. 1-D Nearest Neighbor Approximation

kd-Trees and r-Trees: In the previous approach, dimension reduction techniques are used to retrieve POIs in the neighborhood of the querying point. However, any other spatial indexing scheme can employ the underlying PIR protocol as long as the data is effectively partitioned into buckets of size $\sqrt{n}$. Here, we briefly explain how the previous approach can be extended with 2-D partitionings such as kd-trees and r-trees [9]. Similar to the above case, the idea is to partition nearby POIs into buckets of size at most $\sqrt{n}$. This partitioning guarantees that we end up with at most $\sqrt{n}$ buckets hence conforming to the PIR matrix requirements.

For the case of kd-trees, the original algorithm partitions data horizontally or vertically until each partition holds a single point. The tree construction is modified such that the space is recursively partitioned into most balanced eligible

splits. In other words, at each step, the algorithm recursively chooses the most balanced split among the remaining partitions provided that the total number of partitions does not exceed $\sqrt{n}$. This partitioning is illustrated in Figure 10a.

Similar to kd-trees, the r-tree construction algorithm has to be modified in order to guarantee the PIR matrix requirements as follows. The new construction scheme does not allow a root node of more than $\sqrt{n}$ MBRs. Therefore, the recursive r-tree partitioning algorithm is modified to ensure that all MBRs contain equal number of POIs. The query processing for r-trees is similar to the kd-tree case. Figure 10b illustrates the r-tree partitioning. Figure 10 also illustrates how a block of POI data is privately retrieved from the server under each partitioning to find the nearest neighbor of the query point.

From the discussion above, it is obvious that all three algorithms retrieve $O(\sqrt{n})$ POIs for each nearest neighbor query (which can be used to return the approximate k^{th} nearest neighbor where $1 \leq k \leq \sqrt{n}$). Furthermore, it provides a general strategy for utilizing any spatial partitioning for private evaluation of nearest neighbor queries. Therefore, the choice of partitioning strategy only determines the approximation error and does not increase the communication complexity. More details regarding the empirical comparison of these strategies can be found in [9].

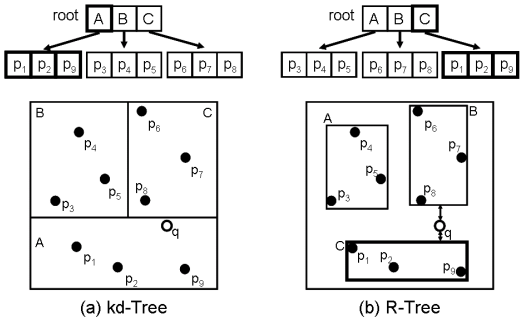


Fig. 10. 2-D Nearest Neighbor Approximations

6.3 Exact Private Nearest Neighbor Queries

Providing exact answers to nearest neighbor queries, require spatial index structures and algorithms that guarantee the exact answer is always returned to the user as part of the result set. Voronoi diagrams [35] have an important property which makes them very suitable for this purpose. If each POI p_i is used as the generator of a Voronoi cell, by definition p_i is the nearest neighbor of any point within that cell. Using this property, the Voronoi tessellation of all POIs is first computed. Next a regular $\delta \times \delta$ grid is superimposed on top of the Voronoi

diagram where each grid cell C holds the generator for each Voronoi cell with which it intersects [1]. This assignment guarantees that by returning all Voronoi cell generators corresponding to a grid cell C, the exact nearest neighbor to any query point q within C is included in the result set.

During the query processing, the user u first learns the grid granularity much in the same way he learns the tree roots in Section 6.2. Next, u finds the grid cell C which includes the query point q and privately requests for the content of C. As we discussed above, the result is guaranteed to include u's nearest neighbor. Note that the PIR matrix is employed in a different fashion compared to the approximate methods. First, while the entire result set is used to improve the approximation error in the case of approximate nearest neighbor search, the redundant POI data associated with a matrix column returned to the user has no use as the algorithm by construction guarantees the exact result to be included in the content of the grid cell enclosing q. Second, instead of individual POIs, all POIs within grid cells form the elements of the PIR matrix. Therefore, different grid granularities and object distributions might result in records of different sizes which in turn leads to an information leak while employing the PIR protocol. This is because the uneven object size distributions allow the attacker to distinguish between different PIR records. To avoid this vulnerability, all cells are padded with dummy data based on the density of the most crowded grid cell. Figure 11 illustrates the padding and the exact nearest neighbor search process. Since u's query q is located in $D3$, it privately computes p_4 as its nearest neighbor. The empirical comparison between the exact and approximate nearest neighbor search strategies discussed above as well as the efficiency of each partitioning algorithm are presented in [9].

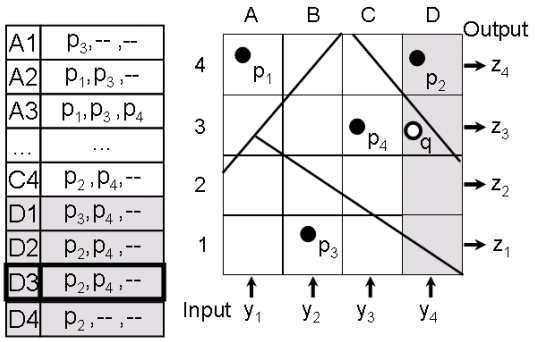


Fig. 11. Exact Nearest Neighbor Search

[1] This approach can be extended to support range queries by storing the set of POIs each cell encloses.

7 Open Problems

As we discussed in Sections 1 and 2, PIR-based approaches to location privacy are the first to provide perfect secrecy as well as provable security against correlation attacks while evaluating spatial queries. Furthermore, these approaches are not vulnerable to an adversary's prior knowledge about object distributions as the object retrieval patterns reveal no information to a malicious entity. These strong measures of privacy give a clear advantage to PIR-based techniques compared to other approaches for location privacy. However, the stringent privacy guarantees does not come without a cost.

With computational-based PIR, the server computation time as well as the communication complexity are rather high. Several techniques can be used to improve these complexities. For instance, compression techniques can utilize significant amount of redundancy in server's response to a PIR request and result in a 90% improvement in communication cost. As another optimization, we note that the PIR matrix M does not have to be square and changing its shape can improve the communication cost without exacerbating server's computation cost. Finally, data mining techniques can be used to reduce the redundancy in the multiplications of the PIR matrix to a user's request vector. This can also save up to 40% of server's computation cost[2]. However, none of these optimizations can break the linear server computation restriction which is inevitable by the definition of computational PIR. While empirical evaluations show that the implementation of the PIR operations and the query processing algorithms result in practical privacy-aware location-based services, the server's response time is still relatively high (several seconds) [9].

The class of hardware-based PIR approaches, on the other hand, avoid several inefficiencies of the computational PIR schemes by providing optimal server computation and communication costs. However, these optimizations are due to reliance on a tamper-resistent secure coprocessor. The major drawback of secure coprocessors are their limited storage and computation resources. Therefore, even very simple algorithms take significantly longer to run on a secure coprocessor compared to their execution time on non-secure processing platforms. The security measures used in securely packaging the processing unit imposes significant thermal and space restrictions. These factors highly limit the potential of embedding faster processors inside the secure package. Furthermore, from a financial standpoint, the relative high cost of secure coprocessors makes it harder to justify their use for some real-world scenarios.

From a practical point of view, PIR problem characteristics limit the efficient processing of spatial queries. As discussed in Section 4.2, computational PIR-based approaches to location privacy require a certain formulation of the problem which converts spatial query processing to private retrievals from a matrix representation of server's data. Similarly, hardware-based PIR approaches require frequent shuffling of data performed by the relatively slow secure coprocessor. These factors make it challenging to efficiently process spatial queries

[2] These optimizations are detailed in [9].

where the main objective is retrieving as few database items as possible for a given query.

Finally, another important restriction in the context of privacy-aware spatial query processing of PIR-based techniques is their inability to efficiently cope with location updates. The PIR-based techniques we discussed so far assume the location data is static and hence the data can be prepared for PIR operations during an offline process. However, with many emerging location-based and social networking scenarios, users are not merely requesting information about the POI around them but would also like to share their presence with their buddies and query their buddies' current locations as well. This requirement (i.e., querying dynamic location data) poses a fundamental challenge to PIR approaches. One opportunity for computational PIR schemes is to employ private information storage schemes [36] to solve this problem. As for hardware-based approaches, the secure coprocessor can receive location updates from the users subscribed to the service and update user locations during each database reshuffling [16]. However, both of these approaches demand additional computing resources from the already computationally constrained techniques.

8 Conclusion and Future Directions

The PIR-based approaches to location privacy presented in this chapter open door to a novel way of protecting user privacy. However, as mentioned in Section 7, full privacy guarantees of these approaches come at the cost of computationally intensive query processing. Therefore, reducing the costs associated with PIR operations can greatly increase the popularity of these approaches. For the approximate nearest neighbor queries discussed in Section 6.2, utilizing the excessive object information returned to a user to guarantee exact results is one promising research direction. As for the hardware-based approaches, employing more efficient shuffling techniques and moving as much non-secure processing as possible away from the SC can result in significant improvements.

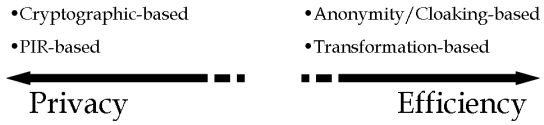


Fig. 12. The Privacy/Efficiency Spectrum

Figure 12 summarizes the privacy/efficiency tradeoffs of various location privacy approaches discussed in this chapter. While anonymity/cloaking and transformation-based approaches enable efficient spatial query processing, they suffer from various privacy implications. At the other end of the spectrum, cryptographic and PIR-based approaches provide significantly stronger privacy guarantees by incurring more costly query processing operations. Therefore, developing a framework that strikes a compromise between these two extremes remains

an interesting open problem. Such a framework should benefit from highly efficient spatial query processing while strongly protecting private user locations without any need for a trusted intermediary. Furthermore, expanding the current framework to efficiently support querying dynamic data remains a challenge. In addition to supporting dynamic queries, the aforementioned approaches can be generalized to support a wide range of spatial queries such as reverse nearest neighbor search and spatial joins.

References

1. Sweeney, L.: k-Anonymity: A Model for Protecting Privacy. Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems **10**(5) (2002) 557–570
2. Gruteser, M., Grunwald, D.: Anonymous usage of location-based services through spatial and temporal cloaking. In: MobiSys'03, San Francisco, CA
3. Gruteser, M., Liu, X.: Protecting privacy in continuous location-tracking applications. IEEE Security & Privacy **2**(2) (2004) 28–34
4. Mokbel, M.F., Chow, C.Y., Aref, W.G.: The new casper: Query processing for location services without compromising privacy. In: VLDB'06, Seoul, Korea 763–774
5. Bettini, C., Wang, X.S., Jajodia, S.: Protecting privacy against location-based personal identification. In: SDM'05, Trondheim, Norway 185–199
6. Gedik, B., Liu, L.: A customizable k-anonymity model for protecting location privacy. In: ICDCS'05, Columbus, OH 620–629
7. Beresford, A.R., Stajano, F.: Location privacy in pervasive computing. IEEE Pervasive Computing **2**(1) (2003) 46–55
8. Khoshgozaran, A., Shahabi, C.: Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In: SSTD'07, Boston, MA 239–257
9. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., Tan, K.L.: Private queries in location based services: anonymizers are not necessary. In: SIGMOD'08, Vancouver, BC, Canada (2008) 121–132
10. Khoshgozaran, A., Shirani-Mehr, H., Shahabi, C.: SPIRAL, a scalable private information retrieval approach to location privacy. In: The 2nd International Workshop on Privacy-Aware Location-based Mobile Services (PALMS) In conjunction with MDM'08, Beijing, China (2008)
11. Hengartner, U.: Hiding location information from location-based services. In: MDM'07, Mannheim, Germany 268–272
12. Yiu, M.L., Jensen, C.S., Huang, X., Lu, H.: Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In: ICDE'08, Cancún, México (2008) 366–375
13. Zhong, S., Li, L., Liu, Y.G., Yang, Y.R.: Privacy-preserving location-based services for mobile users in wireless networks. Technical report, Yale Univerisity (2004)
14. Indyk, P., Woodruff, D.P.: Polylogarithmic private approximations and efficient matching. In: TCC'06. 245–264
15. Zhong, G., Goldberg, I., Hengartner, U.: Louis, lester and pierre: Three protocols for location privacy. In: PET'07. Volume 4776., Ottawa, Canada (2007) 62–76
16. Khoshgozaran, A., Shahabi, C., Shirani-Mehr, H.: Location privacy; moving beyond k-anonymity, cloaking and anonymizers. Technical report, University of Southern California (2008)

17. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: FOCS. (1995) 41–50
18. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: FOCS. (1997) 364–373
19. Sion, R.: On the computational practicality of private information retrieval. In: In Proceedings of the Network and Distributed Systems Security Symposium, 2007. Stony Brook Network Security and Applied Cryptography Lab Tech Report. (2007)
20. Asonov, D.: Querying Databases Privately: A New Approach to Private Information Retrieval. Volume 3128 of Lecture Notes in Computer Science. Springer (2004)
21. Asonov, D., Freytag, J.C.: Almost optimal private information retrieval. In: PET'02, San Francisco, CA 209–223
22. Iliev, A., Smith, S.W.: Private information storage with logarithm-space secure hardware. In: International Information Security Workshops, Toulouse, France (2004) 199–214
23. Smith, S.W., Safford, D.: Practical private information retrieval with secure coprocessors. Technical report, IBM (August 2000)
24. Gertner, Y., Goldwasser, S., Malkin, T.: A random server model for private information retrieval or how to achieve information theoretic pir avoiding database replication. In: RANDOM'98, Barcelona, Spain (1998) 200–217
25. Beimel, A., Ishai, Y., Malkin, T.: Reducing the servers' computation in private information retrieval: Pir with preprocessing. J. Cryptology **17**(2) (2004) 125–151
26. Gertner, Y., Ishai, Y., Kushilevitz, E., Malkin, T.: Protecting data privacy in private information retrieval schemes. J. Comput. Syst. Sci. **60**(3) (2000) 592–629
27. Bhattacharjee, B., Abe, N., Goldman, K., Zadrozny, B., Chillakuru, V.R., del Carpio, M., Apte, C.: Using secure coprocessors for privacy preserving collaborative data mining and analysis. In: DaMoN'06, Chicago, IL 1–7
28. Jiang, S., Smith, S., Minami, K.: Securing web servers against insider attack. In: ACSAC'01, Washington, DC, USA 265–276
29. Kalashnikov, D.V., Prabhakar, S., Hambrusch, S.E.: Main memory evaluation of monitoring queries over moving objects. Distrib. Parallel Databases **15**(2) (2004) 117–135
30. Xiong, X., Mokbel, M.F., Aref, W.G.: Sea-cnn: Scalable processing of continuous k-nearest neighbor queries in spatio-temporal databases. In: ICDE'05, Tokyo, Japan 643–654
31. Yu, X., Pu, K.Q., Koudas, N.: Monitoring k-nearest neighbor queries over moving objects. In: ICDE'05, Tokyo, Japan 631–642
32. Hilbert, D.: Uber die stetige abbildung einer linie auf ein flachenstuck. In: Math. Ann. 38. (1891) 459–460
33. Faloutsos, C., Roseman, S.: Fractals for secondary key retrieval. In: PODS '89: Proceedings of the eighth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, New York, NY, USA (1989) 247–252
34. Flath, D.E.: Introduction to Number Theory. John Wiley & Sons (1988)
35. Berg, M.d., Kreveld, M.v., Overmars, M., Schwarzkopf, O.: Computational geometry: Algorithms and applications. Springer-Verlag (1997)
36. Ostrovsky, R., Shoup, V.: Private information storage (extended abstract). In: STOC'97, New York, NY, USA (1997) 294–303