Ni WW, Zheng JW, Chong ZH. HilAnchor: Location privacy protection in the presence of users' preferences. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(2): 413–427 Mar. 2012. DOI 10.1007/s11390-012-1231-2

HilAnchor: Location Privacy Protection in the Presence of Users' Preferences

Wei-Wei Ni (倪巍伟), *Member*, *CCF*, Jin-Wang Zheng (郑锦旺), and Zhi-Hong Chong* (崇志宏)

School of Computer Science and Engineering, Southeast University, Nanjing 210096, China

Key Laboratory of Computer Network and Information Integration in Southeast University, Ministry of Education Nanjing 210096, China

E-mail: {wni, jwzheng, chongzhihong}@seu.edu.cn

Received June 29, 2011; revised January 16, 2012.

Abstract Location privacy receives considerable attentions in emerging location based services. Most current practices however either ignore users' preferences or incompletely fulfill privacy preferences. In this paper, we propose a privacy protection solution to allow users' preferences in the fundamental query of k nearest neighbors (kNN). Particularly, users are permitted to choose privacy preferences by specifying minimum inferred region. Via Hilbert curve based transformation, the additional workload from users' preferences is alleviated. Furthermore, this transformation reduces time-expensive region queries in 2-D space to range the ones in 1-D space. Therefore, the time efficiency, as well as communication efficiency, is greatly improved due to clustering properties of Hilbert curve. Further, details of choosing anchor points are theoretically elaborated. The empirical studies demonstrate that our implementation delivers both flexibility for users' preferences and scalability for time and communication costs.

Keywords location privacy, kNN query, minimum inferred region, users' privacy preferences

1 Introduction

Location-based services (LBS) in parallel with various applications of location-aware devices (e.g., GPS devices) have gained tremendous popularity, spanning a wide spectrum from sensor networks over online mapping services to geospatial information systems[1-2], to name a few. Queries in LBS usually include enquiring locations and other information pertaining to so called points-of-interest (POI). Most of these queries necessitate the query of the k nearest neighbors (kNN) such as the k nearest restaurants for a traveller[3]. While LBS enables a wide spectrum of location-based applications, they indeed threaten users' privacy as they force users to disclose their locations[4-5]. For example, the query of the nearest gas station by a traveller on his trip demands the disclosure of his current location. Such location leakage may hinder applications susceptible to users' privacy. In this sense, queries over public data introduce additional challenges when users' privacy protection is concerned.

Existing solutions to avoiding location exposure fall into three categories, namely, spatial cloaking[6-10], space transformation[11-12] or location obstruction[13-14]. These schemes all trade off among query performance, protection strength and query accuracy. The following practical cases highlight the needs of incorporating users' preferences into the design of privacy preserving scheme.

1) The protection the existing solutions provide cannot be absolutely guaranteed especially in the case where invasion clues such as attackers' background knowledge or their attack schemes cannot be prior determined. Hence, guarantees are only provided with prior assumptions. Therefore, it is naturally required that users can incorporate their preferences into these assumptions. For example, when the safety of encoding key is assumed safe, space transformation via cryptographic techniques is preferred, while users tend to choose spatial cloaking or location obstruction in the case where attackers have the chance of touching encoding key.

2) It is not surprising that both over-protection and under-protection do users harm since the former results in unnecessary cost overhead for high protection and the latter gives the chance of privacy leakage even with reduced cost overhead.

These requirements motivate us to emphasize the

Regular Paper

Supported by the National Natural Science Foundation of China under Grant Nos. 61003057 and 60973023.

*Corresponding Author

©2012 Springer Science + Business Media, LLC & Science Press, China

importance of users' preferences such that an efficient trade-off scheme between protection strength and protection cost can be controlled by users in a convenient way, as well as different protection modes it affords. Such preferences are called privacy preference, i.e., in different cases, users prefer different strengths of location privacy protections. For example, Bob feels very uncomfortable suddenly on his trip. At this emergent time, Bob would like to shift his concerns to query efficiency rather than high location protection in his privacy aware nearest hospital querying.

However, most existing solutions either ignore preferences or incompletely fulfill privacy preferences, say at the cost of high overhead.

Given the requirement that users' locations are not exposed to the server, the server delimits a region, denoted as RCA (region of candidate answer), that contains targeted POIs for users and that guarantees users' privacy[6]. To state briefly, except the users' locations, all other information, including the RCA and its creation procedure, is public to attackers. This exposure leaves attackers the clues of bounding the range of the users' possible locations[6,13]. The minimum bound of the range derived by attackers is called the minimum inferred region, referred to as MIR in most work[9]. In this paper, we continue to use MIR to measure privacy strength. Meanwhile, we allow users to specify MIR as their privacy preferences in queries. As a result, it is necessary to explore a new way of creating RCA. On the one hand, the creation of RCA should be pushed to client sides where MIR, strongly coupled with RCA, is specified. On the other hand, the creation of RCA heavily depends on POIs' distribution at server side. This dilemma makes the creation of RCA complicated, invalidating existing solutions. In this paper, we propose a strategy, called HilAnchor (Hilbert transformation and Anchor obstruction based method), in a users' preferences-driven fashion, while alleviating server's burden. In particular, the RCA is created at client side by initiating a false query, promising user specified MIR; the overhead workload from RCA determined at client side can be alleviated efficiently by specialized data compression.

HilAnchor, while sharing the implementation of false query in location obstruction based solutions, makes itself distinguishable in taking users' preferences into account by permitting the user to choose the MIR. A query in HilAnchor consists of two rounds; at the first round, it generates a square from the returned answers to cover both the targeted POIs and the users' preferred inferred region; at the second round, users resend the square and receive all POIs in it. The targeted POIs are immediately pinpointed at client side. Paralleling with spatial transformation, the overhead workload from transmitting and processing square is alleviated by leveraging Hilbert curve to transform 2-D space to 1-D space. The clustering properties of Hilbert curve enable to encode the square into discrete ranges at client sides so that the communication between two ends is compressed. Meanwhile, the time-expensive region query is converted into range query at the server and B^+-tree index structure further improves the time-efficiency. It is noted that Hilbert encoding can also secure the users' locations. Our contributions are summarized as follows.

1) We study the problem of incorporating preference into privacy protection by permitting users to specify their preferred minimum inferred regions. Therefore, we extend the trade-off space by adding preference into the previous one between performance and protection strength.

2) Two strategies are presented to overcome the problem of scalability originated from preference. The first is to devise a thin server model by pushing most workload down to client sides. The second is to reduce time cost further by Hilbert encoding. As a result, we provide two-level protections, one from the protection of the minimum inferred region and the other from Hilbert encoding.

3) Detailed analysis of anchor point selection is presented via a geometrical way and a new form of upper bound attack originated from the exposure of anchor point is identified. An improved solution is presented to deal with this rigorous case.

4) Empirical studies suggest that the proposed techniques are highly performant.

The rest of the paper is organized as follows. Section 2 presents overview of the related work. Section 3 discusses the user's option-driven framework HilAnchor. The thin-server enhanced version HilAnchor$^+$ based on Hilbert transformation is presented in Section 4. Section 5 discusses the guidelines of anchor parameter setting and proposes an improved version HilAnchor* in face of upper bound attack. Section 6 covers the experimental results of HilAnchor, HilAnchor* and HilAnchor$^+$. Finally, Section 7 concludes the paper and identifies research directions.

2 Related Work

Originated by the privacy threats of location-detection devices, recent attempts for providing location privacy in location-based service focus on location privacy-preserving queries, i.e. avoiding the exposure of users' locations while processing location based queries.

There has been a plethora of techniques to deal with these kinds of queries. Current practices to this problem are to hide users' locations through the following ways. 1) Location Obstruction[13-14]. The idea is that a user first sends a query along with a false location to the server, and the server keeps sending back the list of nearest POIs to the reported false location until the received POIs satisfy user's privacy and quality requirements. 2) Space Transformation[12,15]. This approach converts the original location of data and queries into another space. The transformation maintains the spatial relationship among the data and queries to provide accuracy. 3) Spatial Cloaking[6,9,16-18]. In this framework, a privacy-aware query processor is embedded in the database server side to deal with the cloaked spatial area received either from a querying user[16] or from a trusted third party[6,9]. Although these approaches would be valuable for protecting users' location privacy in LBS applications, the practicality in incorporating preference into location privacy protection is ignored or doubtful.

Cloaking-based solutions can provide user preference to location privacy by transmitting a user defined profile including user location and expected minimum inferred area to the anonymizer. Anonymizer then expands user location into a cloaked region with expected area to act as finial MIR, and sends the region to the server for retrieving candidate answers. In this way, complex server-side query processing is needed to determine RCA in terms of the given minimum inferred region, which affects system's performance seriously. It provides user preference in a brute-force way at the cost of complex server-side query processing and poor scalability. Besides, cloaking-based solutions usually suffer from the following forced requirements: 1) all users must trust the third party anonymizer, which becomes a single point of attack; 2) a large number of cooperating, trustworthy users are needed.

Instead, transformation-based solutions generally have at least one of the following shortcomings. 1) Fall short in offering practical query accuracy guarantees. Query result may contain false hints. 2) User's location privacy relies wholly on the security of transformation key but does not afford user's preference at all. For example, in [12], Hilbert curve is used to map all POIs into 1-D space. A user initiates a query about Hilbert value of his position and receives the closest Hilbert value of POIs from the server-side. Finally, a decode operation is carried out at client side to get the targeted nearest POI. This solution achieves good query efficiency. While, query accuracy cannot be guaranteed since the fact that Hilbert curve does not completely preserve spatial proximity. [15, 19] propose a kind of novel solution based on private information retrieval theory. It can afford approximate or accurate query results via Hilbert encoding or Voronoi tessellation.

Alternatively, location obstruction based solutions commonly necessitate unpredictably rounds of communication between client and server sides to guarantee accuracy. The time of iterations is unpredictable, which deteriorates scalability and makes it difficult to support users' preferences. For example, SpaceTwist[13] can get the accurate result without any online or off-line trusted third party, while these advantages are achieved at the cost of multi-rounds communication which is inclined to occupy considerable communication resource and workload at the server. Meanwhile, as indicated in [13], in this framework, the MIR cannot be guaranteed because it depends on the location of the anchor and the data distribution at server side. This heavy dependence can be used to attack users' locations and therefore invalidate users' control of the MIR.

The key distinctions between these works and our proposed privacy-aware query solution are as follows.

1) Our query processor does not require any online trusted third party, while owing the ability to incorporate users' preferences instead of the brute-force way in cloaking-based solutions.

2) In order to achieve system scalability, we leverage Hilbert curve based transformation to trade off between scalability and privacy preferences. We focus on data compression effect of Hilbert curve transformation rather than its security protection effect as most space transformation based solutions rely on.

3) To alleviate overhead workload of location obstruction solutions, we determine a square covering targeted POIs at client sides: initiate a false query and finish handshakes within two rounds.

Table 1. Summary of Notations

Symbol	Description
p	Location of the user
s	Expected area of MIR
p'	2-D coordinates of the anchor point
kNN(p')	k nearest POIs to p' at server-side

3 HilAnchor

We next start with the framework of HilAnchor to illustrate how HilAnchor works and then show the details of creating RCA as the key step of HilAnchor framework.

3.1 Framework of HilAnchor

HilAnchor processes a kNN query in two rounds, detailed in Fig.1. In the first round, a user sends a false point p' of point p, called an anchor of point p, to server and receives k nearest neighbor answers, denoted as $k\mathrm{NN}(p')$, in terms of p'. In the second round, the client sends back RCA created from the returned answers. The server returns all POIs located inside the RCA. Finally, the actual result is pinpointed at the client side. During these two rounds, RCA needs to meet two-fold requirements. First, the region of RCA must cover the targeted POIs. Second, it promises users' preferences to MIR. The difficulty of RCA creation stems from the latter requirement; it is possible for adversaries to shrink the inferred region within a big RCA, invalidating its privacy protection. This observation contradicts usual institutions of enlarging RCA in a brute-force way, and lets alone the increasing cost with large RCA. Therefore, the realization of HilAnchor framework becomes difficult when it aims to allow for user-specified MIR.

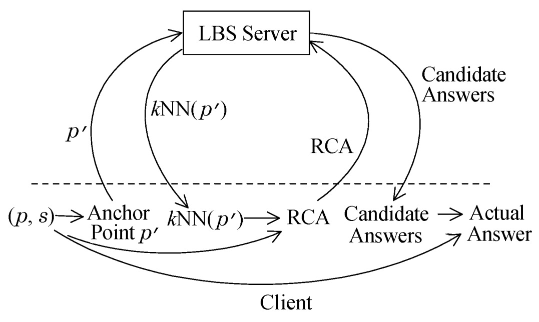


Fig.1. Framework of HilAnchor.

3.2 RCA Creation

For clarity, the 2-D space we discuss is defined with Euclidean distance $d(\cdot)$ and the dataset T contains all POIs at server. Recall the client side creates the RCA from the returned POIs for the anchor point. The creation is detailed into two phases as shown in Fig.2. At the first phase, an initial region of a circle is created to cover the targeted POIs. At the second phase, the initial region is blurred to meet the MIR requirement.

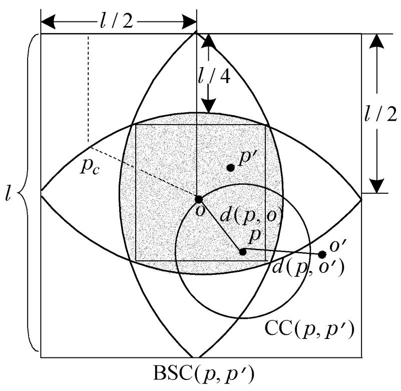


Fig.2. Inferring region of RCA.

For given point p, p' is its anchor. Point $o \in k\mathrm{NN}(p')$ is the farthest POI returned by the server to p. The client builds the initial region by creating a circle $\mathrm{CC}(p,p')$ shown in Fig.2, centered at point p with radius $d(p,o)$.

Theorem 1. *Given p, the area of the initial region is the minimum area that covers the k nearest POIs to p for any dataset T.*

Proof. After the first round handshake, there are at least k POIs ($k\mathrm{NN}(p')$) grasped by the user. Since o in $k\mathrm{NN}(p')$ is the farthest POI to p in $k\mathrm{NN}(p')$, there must exist at least $k-1$ other POIs whose distances to p are smaller than $d(p,o)$. Therefore, from the process of $\mathrm{CC}(p,p')$ creation, it can be deduced that the initial region $\mathrm{CC}(p,p')$ must cover $k\mathrm{NN}(p)$ and the maximum distance between p and any POI in $k\mathrm{NN}(p)$ is $d(p,o)$. Assume $o' \in T$ is an element in $k\mathrm{NN}(p)$ located outside the initial region $\mathrm{CC}(p,p')$. For o locates just on the boundary of the circle denoted in Fig.2, it must have $d(p,o') > d(p,o)$, which contradicts with the assumption that o' belongs to the POI set $k\mathrm{NN}(p)$. □

Theorem 1 indicates that any RCA must contain the initial region $\mathrm{CC}(p,p')$ such that the targeted POIs are returned to users in the second round. It is naturally assumed that point o and the creation of RCA are public to adversaries. This assumption facilitates the inference of user's location p, i.e., user's location p is determined at the center of $\mathrm{CC}(p,p')$. Hence, initial region $\mathrm{CC}(p,p')$ fails to guarantee the specified MIR if it directly serves as RCA. Our strategy is to expand $\mathrm{CC}(p,p')$ to a square, denoted as $\mathrm{BSC}(p,p')$ in Fig.2. To be precisely, o is set as the center of the square that covers the initial circle $\mathrm{CC}(p,p')$. For $\mathrm{BSC}(p,p')$, the following lemma is immediately obtained without proof.

Lemma 1. *Given the exposure of $BSC(p,p')$ and its creation algorithm, there are infinite number of $CC(p,p')$ that correspond to the $BSC(p,p')$, i.e., the probability of inferring $CC(p,p')$ for a query in a random way is zero.*

Further, the following theorem shows how $\mathrm{BSC}(p,p')$ leads to MIR guarantee.

Theorem 2. *For a specified area s, if the side length of $BSC(p,p')$ is no less than $l_{min} = max\{2.391\sqrt{s}, (\sqrt{2}+2)d(p,o)\}$, the area of MIR is no less than s.*

Proof. As shown in Lemma 1 that given point o on the boundary of circle $\mathrm{CC}(p,p')$, there are infinite number of circles centered at point p and containing o. It can be inferred that the centers for these possible circles must have equidistance from p to o and from p to the side of the concentric square inscribed to the circle within $\mathrm{BSC}(p,p')$. For a given side, these centers form a parabola with point o as its focus and the side as an alignment, e.g., in Fig.2, point p_c is a center on some parabola. MIR must be embodied by four parabolas w.r.t., four sides of BSC(p, p'), respectively, as the shaded shown in the figure. The parabolic equations can be normalized as $y^2 = lx^2$, where l is the side length of BSC(p, p'). The inferred region of the shaded can be calculated, $\psi = 8\int_0^{\frac{3-2\sqrt{2}}{4}l}\sqrt{lx}dx + (\sqrt{2}-1)^2 l^2 \approx 0.175l^2$. It requires that $\psi \geqslant s$ for location privacy guarantee. Thus, we have $l \geqslant 2.391\sqrt{s}$. Meanwhile, to guarantee that BSC(p, p') covers the initial region CC(p, p'), it requires $l \geqslant (\sqrt{2}+2)d(p, o)$, therefore $l_{\min} = \max\{2.391\sqrt{s}, (\sqrt{2}+2)d(p, o)\}$. □

```
Algorithm 1. HilAnchor-Client(p, s)
//p is the user location and s the area of MIR
1: create the initial region CC;
2: determine $l_{\min}$ by Theorem 2;
3: expand CC to BSC;
//the side length of BSC satisfies Theorem 2
4: send BSC back to the server;
5: pinpoint answers from the returned;
6: return the pinpointed answers;
//the k nearest neighbour POIs of p
```

Algorithm 1 running at client sides presents process details of our model at client. The inputs are query user's location and his preferred area of MIR. In line 1, the initial region is generated by initiating a k nearest neighbor query about an anchor point. Subsequently, the initial region is expanded to square BSC by Theorem 2 in lines 2∼3. The square BSC(p, p') serves as RCA and is sent back to the server in line 4. The server returns all POIs inside BSC(p, p') as candidate answers. Finally, the answers are pinpointed at client side and returned to the user.

The correctness of Algorithm 1 is guaranteed by Theorem 2. The users' preferences for protection strength are realized by parameter s. The problem arises that it is possible that the area of BSC created by HilAnchor-Client(p, s) is not just closely larger than that of the initial region. The number of candidate POIs within the expanded BSC may be large, introducing heavy time cost overhead especially in 2-D space at server side, as well as communication cost when they are sent back to client sides. To tackle this problem, we leverage the clustering property of Hilbert curve to devise the enhanced version HilAnchor$^+$ of HilAnchor in the following section.

4 HilAnchor$^+$

Before delving into the details of how Hilbert curve enhances the power of HilAnchor, we sketch the paradigm of HilAnchor$^+$. The main differences between the two versions are: 1) the POIs at the server are encoded into Hilbert indexes (a.k.a. Hilbert cells) and 2) all queries to the server are correspondingly encoded into Hilbert indexes and all returns from the server are decoded at client sides. Further, methods of processing encoded queries on encoded data are presented.

4.1 Hilbert Encoding Under Privacy Constraint

To alleviate the overhead workload due to users' preferences, we explore Hilbert curve. In particular, a square, say BSC(p, p') for example, in 2-D space is transformed to Hilbert indexes. Through this transformation, the time-expensive region query in 2-D space can be converted into range query in 1-D space. The Hilbert curve is used under privacy constraint. Both the service provider and the client users do not know parameters of the curve for privacy protection reason. The encoding and decoding functions at the users' ends are embedded in tamper-resistant devices without the third party's intervene.

Similar to work in [12], our enhancement requires an offline space encoding phase carried by a trusted third party. The curve parameters of SDK (space decryption key) include the curves starting point (x_0, y_0), curve orientation θ, curve order N and curve scale Γ. The whole space $\Re$ is encoded into 2^{2N} Hilbert cells by the specified Hilbert curve H_2^N. As a result, all POIs are encoded into corresponding Hilbert values and stored in a look-up table LUT.

Correspondingly, the client submits the Hilbert encode $h(p')$ of the anchor p' to the server rather than p' itself. The server returns the k nearest Hilbert values for POIs to $h(p')$. Subsequently, the client decodes the returned values of corresponding POIs and generates BSC(p, p'). Now, the problem boils down to encoding BSC of a square via Hilbert curve. For square S under Hilbert curve H_2^N, its Hilbert closure HC(S) is the minimum set of Hilbert cells that cover S.

For example, in Fig.3, the Hilbert closure HC(S) of square S is composed of 16 cells, i.e., HC(S) = {8, 9, 54, 55, 11, 10, 53, 52, 30, 31, 32, 33, 29, 28, 34, 35}, covered by the red line. It is time and communication prohibitive to directly represent BSC of square with its Hilbert closure. Therefore, we resort to compressing the Hilbert closure of a square. The intuition is to partition the cells in a Hilbert closure into consecutive ranges. Back to the above example, the HC(S) can be organized as three ranges, namely (8, 11), (28, 35) and (52, 55), which include all 16 Hilbert cells in HC(S).

For the square S, let BC$(S) \subset$ HC(S), containing all cells located at the boundary of square S.

Definition 1. *For square S, cell $u \in BC(S)$ is called in-cell of S if the Hilbert curve enters S via u.*

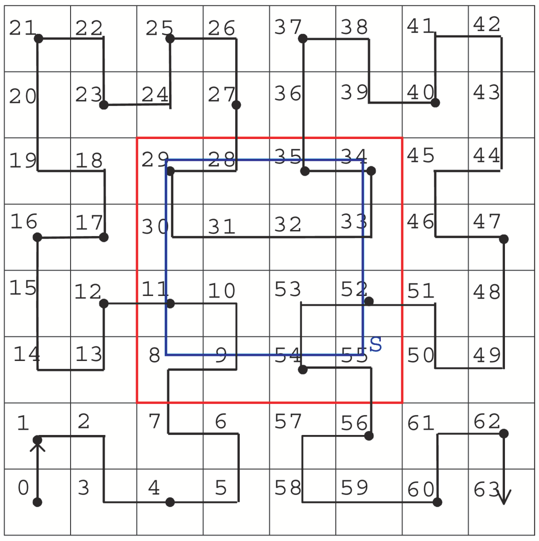


Fig.3. Hilbert curve transformation ($N = 3$).

Conversely, u is called out-cell of S if the Hilbert curve leaves S via u.

Hence, the cells in BC(S) can be partitioned into three types, namely called in-cells, out-cells and inner-cells. For example, in Fig.3, BC(S) = {8, 9, 54, 55, 11, 52, 30, 33, 29, 28, 34, 35}, 8, 28 and 52 are in-cells, 11, 35, 55 are out-cells and the others are inner cells.

Definition 2 (Hilbert Closure Range). *For given square S, sort its in-cells and out-cells in ascending order, resulting in a sequence of interleaved in-cells and out-cells. Each pair of two adjacent in-cell and out-cell forms a Hilbert range. The set of such ranges is called the Hilbert closure range of square S, denoted as HCR(S).*

For square S in Fig.3, the ascending sort of its in-cells and out-cells is {8, 11, 28, 35, 52, 55}. The set {(8, 11), (28, 35), (52, 55)} of intervals serves as HCR(S).

Theorem 3. *Given point $q \in \Re$, q belongs to square S, there must exist a Hilbert range (I_i, O_i) in HCR(S) such that $I_i \leqslant h(q) \leqslant O_i$.*

Proof. The curve H_2^N traverses each cell once and only once and encodes the cell on the way consecutively[20]. For $S \subset \Re$, the curve inside S must be partitioned into a series of continuous curve segment, which starts from an in-cell of S and ends at an out-cell of S. These curve segments correspond to HCR(S). Thus, Hilbert index of any cell in HC(S) must be included by one of ranges in HCR(S). For q belongs to square S, it must be indexed by some cell in HC(S). Therefore, $h(q)$ must be included by a range in HCR(S). □

Theorem 3 verifies that Hilbert closure range of BSC(p, p') must include all Hilbert indexes of targeted POIs w.r.t p. Therefore, the client can retrieve Hilbert indexes of target POIs by transmitting HCR(BSC(p, p')) instead of all cells in HC(BSC(p, p')). This would greatly reduce the query workload at server side and the communication cost.

While with unknown Hilbert curve due to the protection of privacy, it is non-trivial to decide the type of a cell in BC(S), even when its Hilbert index is obtained. This application constraint makes the computation of HCR(S) considerably difficult. A brute-force way is to calculate all Hilbert indexes of cells in HC(S) to determine the ranges at client. Although the client can encode 2-D coordinates into its Hilbert index within several steps (N steps), to deal with so many cells is time-consuming. Fortunately, the following Theorem enables to decide the types of cells efficiently.

Theorem 4. *For given square S, cell $u \in BC(S)$ and its outward adjacent cell u', if the indexes of u and u' are consecutive, u must be in-cell or out-cell of S. Specifically, if the index of u is larger than that of u', u is an in-cell, otherwise it is an out-cell.*

Proof. For the curve H_2^N traverses the whole space $\Re$ and encodes each cell on the way in an incrementally consecutive way. u is the boundary cell of S, if the curve leaves S from u and the next cell the curve visits must be just the outward adjacent cell of u. Similar conclusion can be drawn for in-cells. □

Back to Fig.3, for cell 28 in BC(S), cell 27 is its outward adjacent cell w.r.t S, so cell 28 is an in-cell of S. Similar verification can be made for other cells in BC(S). Therefore, only those indexes of cells in BC(S) and their outward adjacencies need computing.

Algorithm 2. HeC(S)
```
1: collect the Hilbert indexes of boundary cells in square $S$ into BC;
2: identify in-cells and out-cells in BC; //By Theorem 4
3: return all values of in-cells and out-cells in ascending
   order as HCR($S$);
```

By now, based on Theorem 4, we can present the algorithm to compress the Hilbert encode of a square. The input of the algorithm is a square to be compressed. The details are presented in Algorithm HeC(S). In line 1, the boundary cells of square S are generated. Subsequently, in-cells and out-cells are identified from these boundary cells to determine HCR(S). Finally, HCR(S) is returned to the user.

Optimization of HeC(S). The memory usage of Algorithm 2 can be further reduced since some boundary cells are not necessary to deal with after line 1. Some strategies can be adopted to prune those cells. For instance, cell u locates at the upper boundary of S, cells $u^1, u^2 \in$ BC(S), are left and right adjacent cells of u, if indexes of u^1, u and u^2 are consecutive, cell u is impossible to be in-cell or out-cell and can be pruned from BC.

Suppose the number of cells in HC(S) is $\varphi(S)$. Total number of boundary cells in BC(S) and their outward adjacent cells is approximately $8\sqrt{\varphi(S)}$. The computation cost of HeC(S) at client-side is $4N\sqrt{\varphi(S)}$ on average.

4.2 Algorithm HilAnchor$^+$

This subsection presents both client-side and server-side processing of HilAnchor$^+$.

Client-Side Processing. An adjustable RCA is built to contain MIR such that users' preferred MIR can be satisfied. Further, over-workload from RCA is alleviated by compressing RCA into HCR. Finally, query result can be pinpointed via decoding the returned Hilbert values inside HCR. Algorithm 3 summarizes the client-side process of HilAnchor$^+$ for a user to initiate a kNN query.

In lines 1~4, the initial region is created by one handshake with the server including encoding and decoding operations at client sides. In line 5, $l_{\min}$ is computed based on Theorem 2. The initial region is expanded to BSC in line 6. BSC is compressed into HCR(BSC) and sent back as the encoded RCA to the server, in lines 7~8, respectively. The server returns all values inside ranges of HCR. Finally, the client can pinpoint answers from returned Hilbert indexes by decoding these indexes and comparing them with p.

```
Algorithm 3. HilAnchor$^+$-Client($p$, $s$)
//$p$ is the user location and $s$ the area of MIR
1:  generate the anchor point $p'$ and encode it;
2:  initiate false query about Hilbert value of $p'$;
3:  decode and get POIs in $k$NN($p'$);
4:  create the initial region CC via $k$NN($p'$);
5:  determine $l_{\min}$ by Theorem 2;
6:  expand CC to BSC; //satisfying Theorem 2
7:  HCR = HeC(BSC); //recalling Algorithm 2
8:  send HCR back to the server;
9:  decode and pinpoint answers from the returned
    from the server;
10: return pinpointed answers;
//the $k$ nearest neighbour POIs of $p$
```

Remark 1. *The approximation error introduced in Hilbert decoding can be reduced by specifying suitable curve order. In our solution, function $h^{-1}()$ returns the 2-D coordinates of the input Hilbert cell's center. The approximation error will not exceed $\sqrt{2}/2$ times of the Hilbert cell extent. In reality, it is sufficient to set N above $8^{[12]}$.*

Remark 2. *The Hilbert encoding scheme used in HilAnchor$^+$ will not introduce false hints. In HilAnchor$^+$, a crucial step to query accuracy is to initialize region $CC(p,p')$, which must contain the targeted result kNN(p). As shown in Algorithm 3, for a kNN query, HilAnchor$^+$ first initiates a kNN query with Hilbert index $h(p')$ rather than with p' itself. Although, this may lead to false hints. That is, the returned k Hilbert values nearest to $h(p')$ may not be those Hilbert values of the actual k nearest POIs to p'. However, such inconsistency will not introduce error, which is proved in Theorem 5.*

Theorem 5. *Given anchor point p', the initial region $CC(p,p')$ based on the returned k nearest Hilbert values to $h(p')$ covers kNN(p).*

Proof. The returned k nearest Hilbert values must correspond to the existing k POIs stored at server side. As discussed in Subsection 3.2, CC(p,p') based on the k points decoded from the returned k nearest Hilbert values must cover these k POIs. Thus, CC(p,p') contains no less than $k+1$ points, definitely involving kNN(p), as well as p itself. □

Server-Side Processing. In our framework HilAnchor$^+$, two handshakes exist between client and server. First, client sends Hilbert value of the anchor point to the server for retrieving k nearest Hilbert values of POIs to that of the anchor. Second, the client sends the generated Hilbert closure ranges to the server and retrieves all POIs whose Hilbert values locate inside the ranges. Both of these queries are carried on the 1-D table LUT, and belong to 1-D range query. To improve time-efficiency at server side, a B$^+$-tree index is constructed on the table to accelerate range queries. Fig.4 illustrates the structure of the B$^+$-tree adopted, where leaf nodes are designed as a doubly linked list. Therefore, the time efficiency of such range query is $O(\log_2 n)$, the symbol n denotes the total number of POIs at the server.

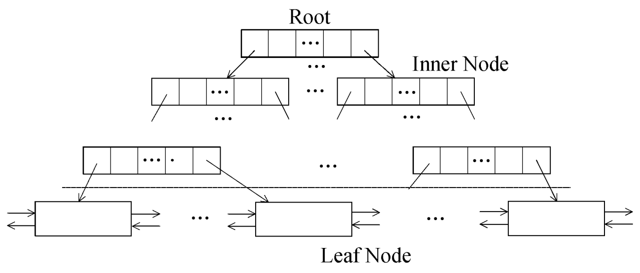


Fig.4. B$^+$-tree structure.

In conclusion, HilAnchor$^+$ can achieve trade-off between scalability and privacy preferences, allow users to specify their desired MIR as privacy preference and Hilbert encoding schemes are used to achieve scalability. In particular: 1) Instead of the brute-force way adopted in cloaking-based solutions, region RCA can be well regulated at client side and easily incorporate users' MIR preferences, which simplifies the server-side process. 2) The Hilbert encoding scheme converts range query in 2-D space into interval matching in 1-D one, which can further accelerate server-side process. 3) Rather than transferring 2-D coordinates of POIs inside RCA, only Hilbert values are required to return in HilAnchor$^+$.

5 Privacy Issue in Parameter Selection

A problem of choosing anchor point p' is naturally raised in HilAnchor$^+$. On the one hand, it seems that p' may expose the clue of user's location p when p' is too close to p. On the other hand, the size of candidate answer region increases sharply due to the enlargement of initial region $\mathrm{CC}(p, p')$ if p' deviates too far away from p. Hence, it is necessary that a pair of lower and upper bounds r and R should be carefully imposed on p' such that p' is distanced from p between r and R.

For the lower bound r, the operator $\max\{2.391\sqrt{s}, (\sqrt{2}+2)d(p,o)\}$ in Theorem 2 excludes the possibility of too small candidate answer region RCA even when p' approaches p. Therefore, the specified MIR can be guaranteed. However, for the upper bound R, except for the fact the large R results in work overhead, the assumption of public R as system parameter can be exploited by adversaries. The user's position must be located inside the circle centered at p' with radius R, denoted by $C_R(p')$. Therefore, in the following, we only concern the attack via R and anchor point p', which is called upper bound attack in this paper.

5.1 Upper Bound Attack

The exposure of R leaves the clue of p and therefore is utilized to initiate upper bound attack. First, p can be inferred in $C_R(p')$. Hence, the area of $C_R(p') = \pi R^2$ must be forced no less than user's specified minimum inferred area s. That is, $R \geqslant \sqrt{s/\pi}$. Otherwise, the region of $C_R(p')$ would directly invalidate s. Second, even if $R \geqslant \sqrt{s/\pi}$, the MIR depicted in Fig.2 might also be squeezed further because p must be located within the overlapped region of $C_R(p')$ and MIR as shown in Fig.5.

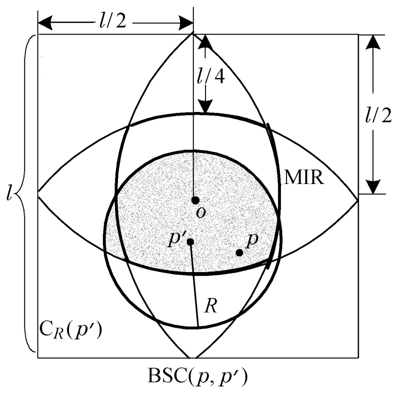


Fig.5. Upper bound attack.

For the simplicity of analysis, the MIR in Fig.5 can be approximated by its maximum inscribed circle centered at o with radius $0.25l$, as shown in Fig.6. The interactions between $C_R(p')$ and MIR can be classified into two cases.

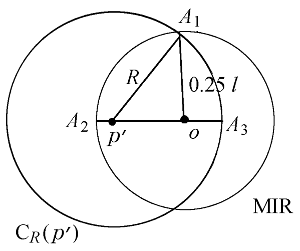


Fig.6. Details of the overlapped region.

1) *Containment Case*: $R \geqslant d(p', o) + 0.25l$ or $0.25l \geqslant d(p', o) + R$. It must be the case that $C_R(p')$ contains MIR or vice versa. For the reason that both $C_R(p')$ and MIR satisfy specified area of s, leakage of p' and R does not invalidate the MIR.

2) *Overlap Case*: $R < d(p', o) + 0.25l$ or $0.25l < d(p', o) + R$. As shown in Fig.5, $C_R(p')$ and MIR must overlap. In this case, the MIR is squeezed to the overlapped region (the shadowed in Fig.5). If the overlapped region is less than specified area s, user's privacy preference of MIR is unsatisfied.

The above analysis demonstrates that upper bound attack at least demands the knowledge of both R and p'. In the following, a rigorous version HilAnchor* of HilAnchor$^+$ is devised. In HilAnchor*, the area of the overlapped region is forcefully left up above user's specified area s even when both R and p' are grasped by adversaries (in the case that the key of Hilbert curve leaks).

5.2 HilAnchor*

Generally, HilAnchor* and HilAnchor$^+$ are the same at server side but a little different at client sides. HilAnchor* takes the overlap case into consideration. In particular, in containment case, HilAnchor* behaves in the same way as HilAnchor$^+$. On the other hand, in overlap case, as shown in Fig.5, the overlapped region is contained in the MIR, whose area is possibly below the specified threshold s. Given p', $C_R(p')$ is fixed in Fig.5. Hence, it is the only way to enlarge MIR such that the overlapped region is left above the specified threshold s. Specifically, enlarge side length l of $\mathrm{BSC}(p, p')$ in order to extend the overlapped region. We use $\gamma(l)$ to denote the area of the overlapped region.

In Fig.6, the half of $\gamma(l)$ is the sum of areas of sector $o\overline{A_1A_2}$ and sector $p'\overline{A_1A_3}$ minus the area of triangle $p'oA_1$. In more details, the areas of sectors $o\overline{A_1A_2}$,

$p'\overline{A_1A_3}$ and triangle $p'oA_1$ are denoted as $S_{o\overline{A_1A_2}}$, $S_{p'\overline{A_1A_3}}$, $S_{p'oA_1}$, respectively. They are calculated as follows.

$$S_{o\overline{A_1A_2}} = d(p',o)^2 \arccos\frac{d(p',o)^2+0.25^2l^2-R^2}{2\times 0.25l\times d(p',o)},$$

$$S_{p'\overline{A_1A_3}} = d(p',o)^2 \arccos\frac{d(p',o)^2+R^2-0.25^2l^2}{2\times R\times d(p',o)},$$

$$S_{p'oA_1} = 2\sqrt{2c(c-R)(c-0.25l)(c-d(p',o))},$$

where $c=\frac{1}{2}(R+0.25l+d(p',o))$. Consequently, $\gamma(l)$ can be approximated as follows.

$$\gamma(l) = 2(S_{o\overline{A_1A_2}} + S_{p'\overline{A_1A_3}} - S_{p'oA_1}).$$

Given R, p, p' and o, setting $\gamma(l) \geqslant s$, the lower bound $l'_{\min}$ of the side length of BSC(p,p') can be immediately obtained. Setting, $l \geqslant l'_{\min}$, HilAnchor*, a rigorous version of HilAnchor$^+$ is devised.

```
Algorithm 4. HilAnchor*-Client($p, s$)
//$p$ is the user location and $s$ the area of MIR
1:  create the initial region CC via initiating a query about
    encoded anchor point;
2:  if containment case then
3:      proceed in the same way as Hilanchor$^+$;
        //lines 5~6 in Algorithm 3
4:  else
5:      calculate $l'_{\min}$; //By Theorem 6
6:      expand CC to BSC with side length not less than $l'_{\min}$;
7:  end if
8:  compress BSC to HCR;
9:  send HCR back to the server;
10: decode and pinpoint answers from the returned from
    the server;
11: return pinpointed answers;
    //the $k$ nearest neighbour POIs of $p$
```

At the beginning steps, HilAnchor* works in the same way as HilAnchor$^+$ at client sides, creating initial region CC(p,p'). After that, HilAnchor* calculates its lower bound $l'_{\min}$ by considering two cases of containment and overlap, respectively, rather than $l_{\min}$ in HilAnchor$^+$. In containment case, HilAnchor* directly sets $l'_{\min}=l_{\min}$; in overlap case, HilAnchor* determines the minimum $l'_{\min}$ such that $\gamma(l) \geqslant s$. In the following steps, the two versions proceed in the same way. The details are elaborated in Algorithm 4.

The following theorem guarantees the correctness of HilAnchor* against upper bound attack.

Theorem 6. *For a specified area s, if the side length of BSC(p,p') is no less than $max\{l_{min}, l'_{min}\}$, where l_{min} satisfies Theorem 2 and $\gamma(l) \geqslant s$ for any $l \geqslant l'_{min}$, the BSC(p,p') can afford user specified s of minimum inferred region no matter whether upper bound attack happens.*

Proof. As discussed above, if MIR contains C$_R(p')$ or vice versa, upper bound attack cannot squeeze MIR. $l'_{\min}$ can be directly set with $l_{\min}$ by Theorem 2. Otherwise, if MIR and C$_R(p')$ overlap, HilAnchor* determines $l'_{\min}$ such that $\gamma(l) \geqslant s$. □

Although, HilAnchor* provides a strong protection against upper bound attack, HilAnchor$^+$ meets protection requirements in general cases; the underlying assumption is that the possibility of disclosing p' is small because the encoding and decoding functions in HilAnchor$^+$ are embedded in tamper-resistant devices at client sides. Hence, we mainly evaluate HilAnchor$^+$ after sufficient comparisons between the two versions.

6 Empirical Evaluation

We provide a sufficient experiment by comparing our solution with all representatives from three popular types of privacy preserving ones. Firstly, since in terms of preference, our work is most related with cloaking-based solutions, we focus on the comparison between them. In particular, Casper is chosen as the representation in cloaking-based solutions due to the following considerations. Cloaking-based solutions commonly consist of two main components, the location anonymizer and the privacy-aware query processor. The main differences among existing cloaking-based solutions lie in strategies adopted in location anonymizers, which are deeply influenced by real-time density of mobile users. Our privacy model does not require the knowledge of all users' real-time location distribution, which is hard to grasp for mobile users and snapshot query scenarios. Hence, cloaking-based solutions and our solution work in different settings and offer different kinds of privacy guarantees. Further, for comparison purpose, workload of location anonymizer in cloaking-based solutions is ignored.

Secondly, due to the similar mechanism with location obstruction, we further compare our solution with SpaceTwist[13], a representative location obstruction solution.

Finally, as mentioned in Section 2, transformation-based solutions commonly trade-off between query accuracy and query performance and do not afford users' preferences. Although, these solutions and our solution aim at different targets, we compare our solution with solution in [15] (denoted as ApproxNN) and solution in [12] (denoted as HilkNN), which both afford approximate answers via Hilbert encoding.

Before comparisons with others, we first verify the enhanced version HilAnchor$^+$ with respect to its initial version HilAnchor and strict version HilAnchor*.

We implemented all algorithms using C++ on Intel Xeon 2.4 GHz machine. The synthetic datasets UI were uniformly generated with 2 000 to 30 000 points and real datasets NE (http://www.rtreeportal.org) with 3 000 points (sampled from complete NE dataset randomly and serving as POI addresses). The coordinates of each points are normalized to the square 2-D space with extent 100 000 meters. The auxiliary structure is either R-tree or B$^+$-tree with 1 KB page size. As for the location parameter p', we choose it in two steps. The first step is to decide an anchor distance $d(p,p') \in [0,R]$ incorporating user's preference. In our experiments, R is set 5% of side length of the data space. In the second step, the anchor p' is randomly set to a location at distance $d(p,p')$ from p.

Scalability performance of each method is evaluated by submitting queries from a series of clients simultaneously to the server. In each experiment, we use a workload with M query points generated randomly from different clients and measure the value of the following performance metrics: 1) average communication cost of M queries, in numbers of TCP/IP packets as adopted in [13]; 2) total time cost at server side and 3) average time cost at client side.

Table 2 lists the default system setting of Hilbert curve key and TCP/IP packets.

Table 2. Default System Settings

Parameter	Setting
(x_0, y_0)	(0, 0)
θ	0
Curve Scale Γ	1
Curve Order N	10
TCP/IP Packet	576 B

6.1 Comparison Between HilAnchor, HilAnchor$^+$ and HilAnchor*

The workload of HilAnchor$^+$ is distributed over both client and server sides while that of HilAnchor mainly over server side. Because the bottleneck of location-based services lies in server side with a large number of requests, it is reasonable to mainly evaluate time cost on server side. For the comparison between HilAnchor$^+$ and HilAnchor*, each pair of user's position and its anchor point is stored to stimulate the exposure of SDK.

As depicted in Figs. 7(a) and 7(b), HilAnchor$^+$ achieves a sharp advantage than HilAnchor* in terms of time cost at server side with increasing M. Since $l'_{\min} \geqslant l_{\min}$, the area of RCA created in HilAnchor* is larger than that in HilAnchor$^+$, HilAnchor* consumes more time at server side than HilAnchor$^+$. Both HilAnchor$^+$ and HilAnchor* are clear winners compared with HilAnchor in terms of varying M and data size, the advantage of HilAnchor$^+$ and HilAnchor* stems from compression via Hilbert encoding. For upper bound attack, recall that an upper bound attack is possible only in the overlap case where the overlapped region is contained in MIR. The times of overlap cases are counted in an off-line way. Fig.7(c) depicts the ratio of times of possible upper bound attack to the number of queries. It shows the possibility of upper bound attack is obviously more than 20%.

With increasing k, HilAnchor$^+$ works well as shown in Fig.8(a). Although the time costs for both versions increase when the distance between p and p' is raised, the time cost for HilAnchor$^+$ rises slowly, as shown in Fig.8(b). HilAnchor$^+$ processes queries in 1-D space via B$^+$-tree. Matching in 1-D space is efficient even when BSC is enlarged by increasing anchor distance. However, HilAnchor consumes more time since it works in 2-D space with R-tree. In terms of communication cost, HilAnchor$^+$ behaves similarly with HilAnchor for increasing distance between p and p', as shown in Fig.8(c). The communication cost is query and answer transmission between client and server sides. As for query transmission, HilAnchor outperforms HilAnchor$^+$ by sending two diagonal coordinates of the BSC while HilAnchor$^+$ a series of Hilbert values. Conversely, at the anchor distance of 600 meters, HilAnchor begins to lose its advantage. With increasing anchor distance, BSC covers more POIs. In this case, Hilbert encoding starts working because the cost

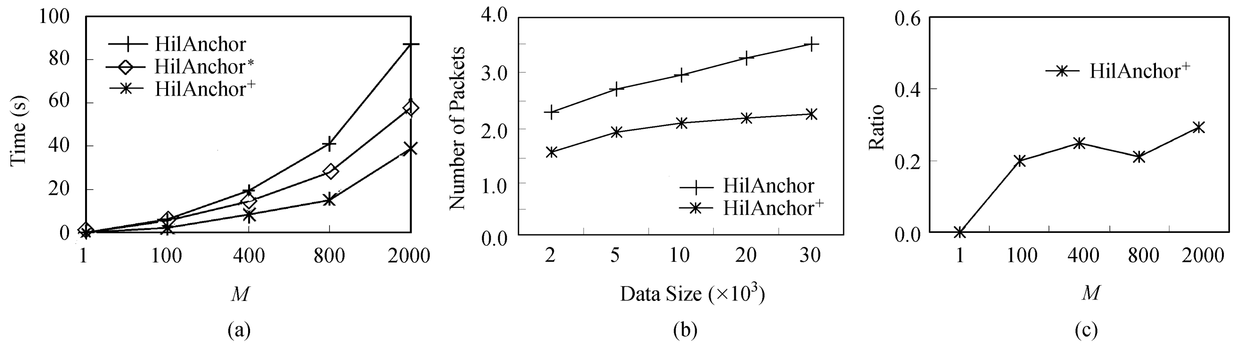


Fig.7. Scalability vs M and dataset size, $k = 1$, MIR is 1% of the data space. (a) Server cost on NE. (b) Communication on UI. (c) Ratio of upper bound attack.

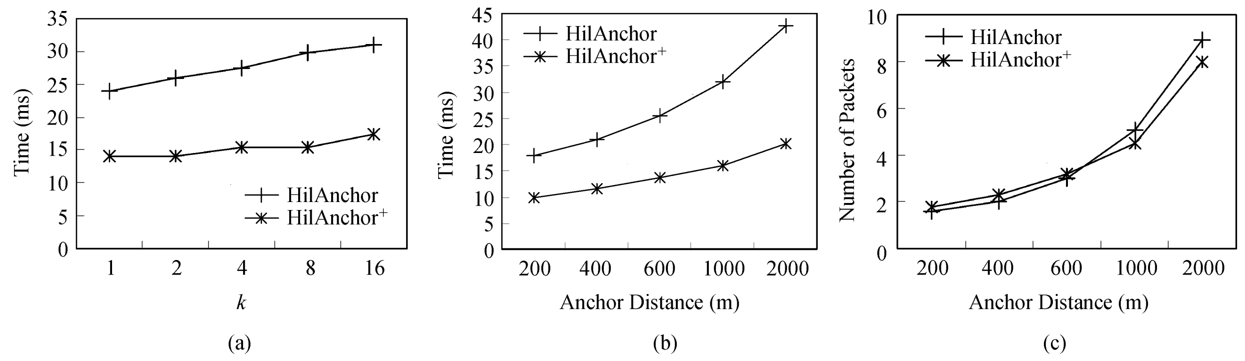


Fig.8. Performance vs anchor distance, $k = 1$, MIR is 1% of the data space. (a) Server cost vs k on NE. (b) Server cost on NE. (c) Communication on NE.

of transmitting Hilbert values is much less than transmitting 2-D coordinates of POIs. The gain obtained in HilAnchor$^+$ via Hilbert encoding scheme is not so large because of additional cost from transmitting HCR rather than original two diagonal coordinates of the BSC in HilAnchor.

It can be concluded that HilAnchor$^+$ is a good trade-off between performance and security. In the following experiments, we choose the version of HilAnchor$^+$ to compare with existing methods.

6.2 Comparison Between HilAnchor$^+$ and Casper

For comparison purpose, we implement solution Casper that generates the cloaked region, making its area the same to the area of MIR setting in HilAnchor$^+$.

Fig.9 depicts the scalability of Casper vs HilAnchor$^+$. Their performances w.r.t the number M of querying users are detailed in Fig.9(a). With increasing M, Casper consumes much more time. At server side, Casper needs to return all POIs which are k nearest neighbors to the region MIR. With the increasing number of concurrent queries, the server-side processing cost increases rapidly. In contrast, HilAnchor$^+$ just returns Hilbert values within submitted HCR via a B$^+$-tree.

Furthermore, HilAnchor$^+$ is a clear winner in terms of time cost shown in Fig.9(b) at server side and communication overhead shown in Fig.9(c). Although data size influences the density of POIs at server side, with fixed anchor distance, the both sizes of RCA and HCR are stable. Hilbert encoding can hide the increasing number of POIs. Therefore HilAnchor$^+$ is insensitive to POIs' density. As for Casper, it needs to search k nearest neighbors to a given region and return POIs. Hence, with varying data size, Casper behaves more sensitively than HilAnchor$^+$.

When parameter k varies, HilAnchor$^+$ shows its advantages in the time cost shown in Fig.10(a) and communication cost shown in Fig.10(b) on real dataset NE. Due to the fact that the compression 1-D structure HCR leads to the scalability to the extent of RCA, the cost of HilAnchor$^+$ is nearly independent of k. Figs. 11(a) and 11(b) show the comparisons of the server-side processing and communication cost in terms of MIR. The costs of both Casper and HilAnchor$^+$ increase with enlarging extent of MIR. However, even for large MIR, the server-side processing time of HilAnchor$^+$ increases quite more slowly than that of Casper. The underlying reason is that Casper

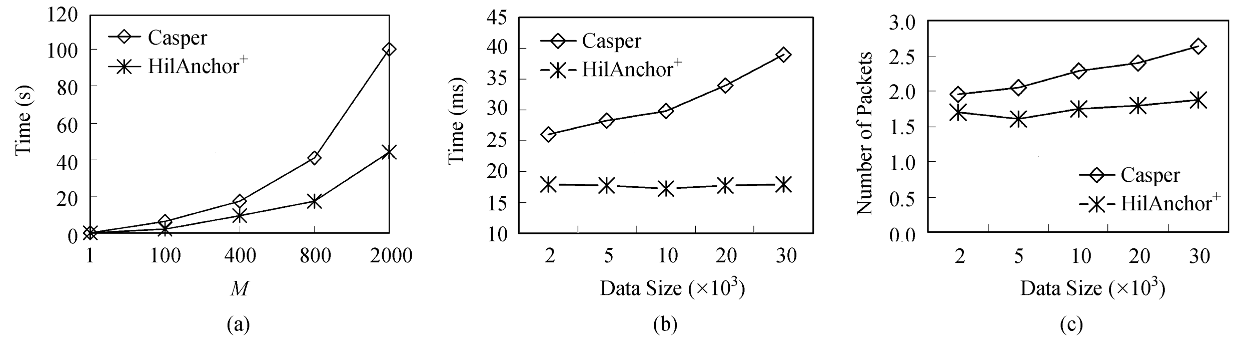


Fig.9. Scalability vs M and dataset size, $k = 1$, MIR is 1% of the data space. (a) Server cost on NE. (b) Server cost on UI. (c) Communication on UI.

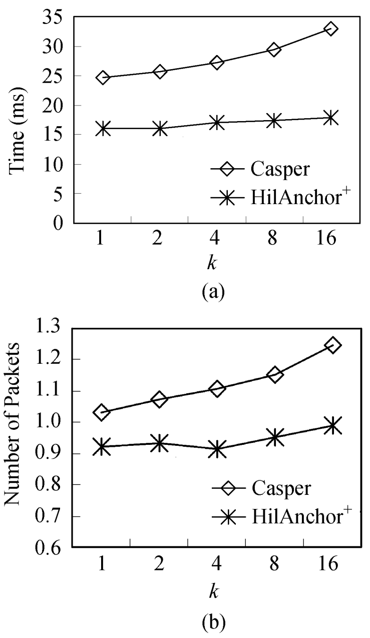


Fig.10. Performance vs k, MIR is 1% of the data space on NE dataset. (a) Server cost. (b) Communication.

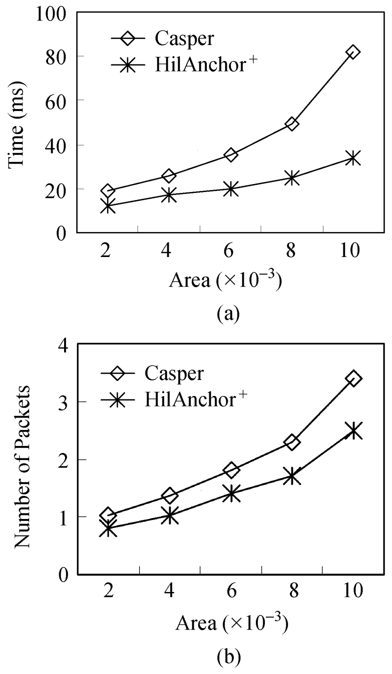


Fig.11. Performance vs area of MIR on NE dataset, k = 1. (a) Server cost. (b) Communication.

falls short of allowing for constraints between RCA and MIR, and its region of RCA determined at server side expands sharply with increasing MIR. From the above experiments, we conclude that HilAnchor$^+$ exhibits high scalability with data size and the number of users with constraints of specified MIR.

6.3 Comparison Between HilAnchor$^+$ and SpaceTwist

In this subsection, we proceed to investigate the scalability of HilAnchor$^+$ against SpaceTwist. SpaceTwist explores granular search strategy to retrieve data points from the server with a user-specified cell extent $\varepsilon/\sqrt{2}$. It improves its efficiency at the cost of query accuracy with error bound ε. We set ε to $\sqrt{2}/2$ times of Hilbert cell extent in HilAnchor$^+$ so that they are compared with the same accuracy.

In Figs. 12(a) and 12(b), HilAnchor$^+$ shows obvious advantage in terms of processing time for varying M, which is similar to the case of the comparison with Casper. Because of expensive queries in 2-D space other than 1-D query in HilAnchor$^+$, SpaceTwist takes more time. Although the client-side process of HilAnchor$^+$ is heavy, HilAnchor$^+$ exploits only two-round handshakes rather than multi-rounds handshakes in SpaceTwist, which can effectively reduce the time at client waiting for response. Therefore, HilAnchor$^+$ achieves both low server-side cost and low average client-side cost for a broad range of M, especially for larger M. Figs. 13(a) and 13(b) depict the performance with respect to parameter k. Due to its insensitivity for parameter k, HilAnchor$^+$ behaves approximately unchanged with increasing k.

Our solution shares strategies of the implementation of false query in location obstruction with SpaceTwist. Observe that these two solutions behave similarly in terms of client-side cost and communication overhead

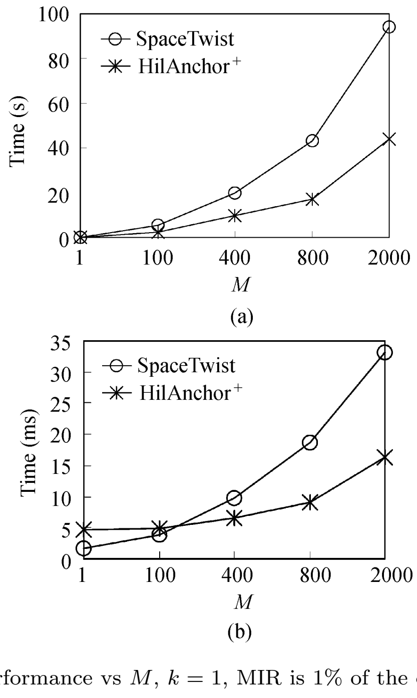


Fig.12. Performance vs M, k = 1, MIR is 1% of the data space, with same anchors. (a) Server cost. (b) Average client cost.

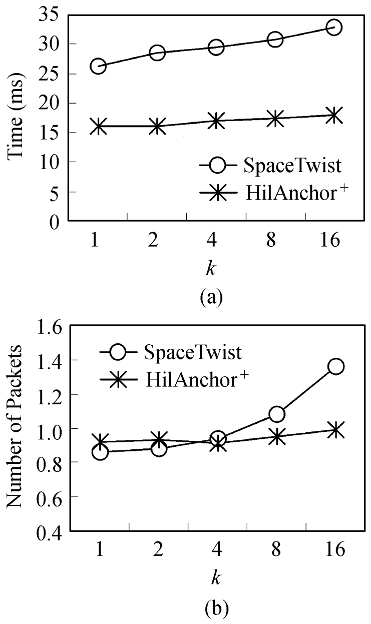


Fig.13. Performance vs k, MIR is 1% of the data space, with same anchors. (a) Server cost. (b) Communication.

as shown in Fig.14. Although, a larger anchor distance in common promises a large RCA in HilAnchor$^+$ or a larger supply space in SpaceTwist, encoding structure HCR of HilAnchor$^+$ can compress RCA sharply. Thus, the performance gap between them is enlarged gradually with increasing anchor distance. As for server-side overhead, SpaceTwist remains increasing much faster than HilAnchor$^+$.

Further, we compare the two solutions with respect to data size using synthetic datasets UI. As shown in Fig.15, HilAnchor$^+$ scales well.

6.4 Comparison Between HilAnchor$^+$, HilkNN, and ApproxNN

Finally, we investigate the performance of HilAnchor$^+$ against HilkNN and ApproxNN. The parameter of modulus bits in ApproxNN is set with 32 to simplify the computation. Since ApproxNN is restricted to the nearest neighbor query, we evaluate these solutions in terms of varying data size, assuming k with 1 as default.

In Fig.16(a), HilkNN shows prevailing advantage in client-side cost, since the client-side work in HilkNN is just making a couple of encoding and decoding operations, rather than a series of decoding and encoding operations in HilAnchor$^+$ to generate HCR. Instead, a complicated client-side matrix decoding is required in ApproxNN, which burdens its workload at client side.

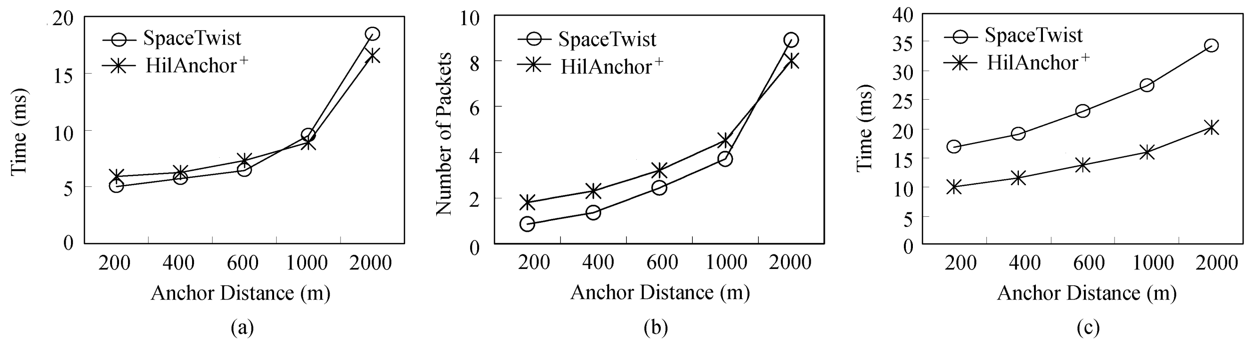


Fig.14. Performance vs anchor distance, $k = 1$, MIR is 1% of the data space. (a) Client cost. (b) Communication. (c) Server cost.

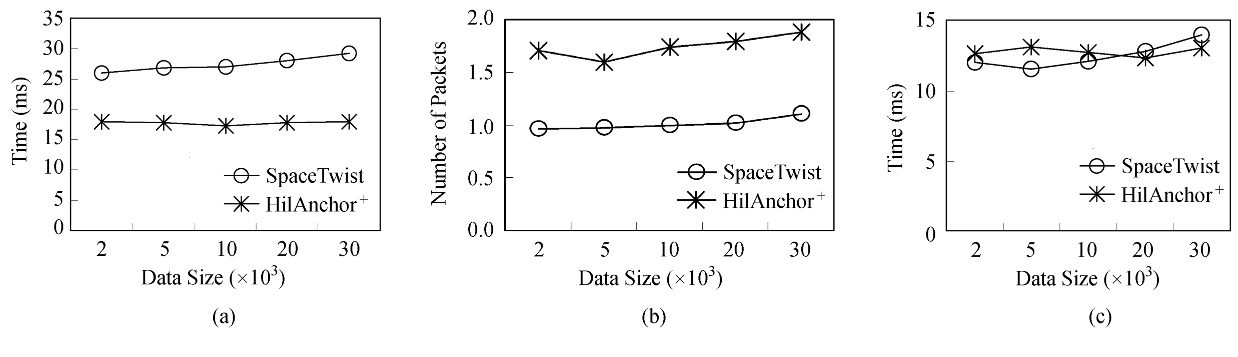


Fig.15. Performance vs data size, $k = 1$, MIR is 1% of the data space, anchor distances are the same. (a) Client cost. (b) Communication. (c) Server cost.

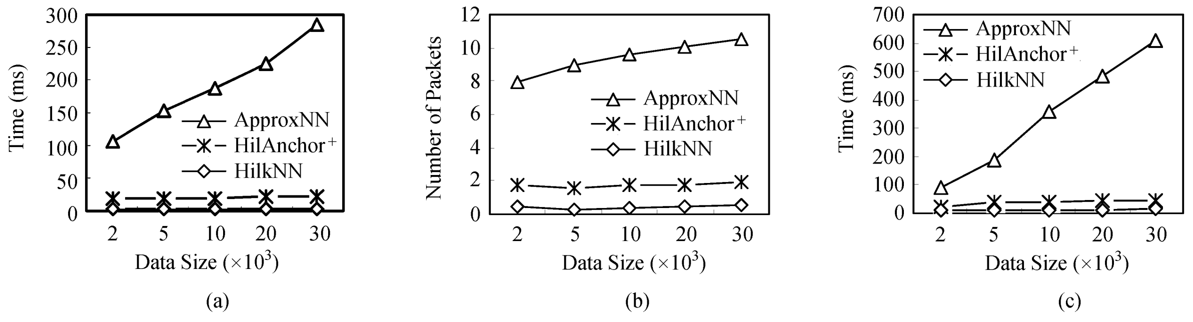


Fig.16. Performance vs data size, $k = 1$, MIR is 1% of the data space. (a) Client cost. (b) Communication. (c) Server cost.

As for communication cost, since the HCR structure in HilAnchor$^+$ is insensitive to data size, its communication cost increases slightly with enlarging data size. Instead, HilkNN only needs to transfer a single Hilbert value, its communication cost is far less than others, as shown in Fig.16(b).

Fig.16(c) depicts the server-side cost with respect to varying data size. The cost of ApproxNN increases proportionally with data size, since the number of multiplication is proportional to the number of "1" bits in the data. Due to the insensitivity, HilAnchor$^+$ and HilkNN behave similarly. The cost of HilAnchor$^+$ keeps twice of HilkNN approximately, since twice of matching on B$^+$-tree are needed in HilAnchor$^+$ rather than only once in HilkNN.

From above comparisons, we conclude that performance of our solution is comparable with existing space transformation based ones. Although, a Voronoi tessellation based method ExactNN is proposed in [15] to improve ApproxNN and guarantee query accuracy, the benefit is gained at the cost of much more overhead workload. A dual Hilbert curve based solution is proposed in [12] to reduce the possibility of false-hints. It can improve query accuracy effectively, but cannot avoid false-hint thoroughly. Most space translation based solutions seek trade-off among privacy protection strength, query accuracy and query performance. Instead, our solution pursues for more purposes among users' privacy preferences, query performance and location privacy protection.

7 Conclusions

This paper concerns the privacy preference support for location-based service while guaranteeing user-defined minimum inferred region. Existing location privacy solutions either ignore the privacy preference or consider it in a brute-force way with high workload and poor scalability at server side. We propose a trade-off among preference, accuracy and scalability by leveraging Hilbert curve based compression. Furthermore, the details of choosing anchor points are theoretically analyzed. Sufficient empirical studies with real-world and synthetic datasets verify our proposals. We intend to extend our proposals to cover real data distributions, involving influence of some non-reachable regions. Besides, it is expected to support continuous queries.

References

[1] Gruteser M, Schelle G, Jain A, Han R, Grunwald D. Privacy-aware location sensor networks. In *Proc. the 9th Workshop on Hot Topics in Operating Systems* (*HotOS 2003*), Hawaii, USA, May 18-21, 2003, pp.163-167.

[2] Beresford A R, Stajano F. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2003, 2(1): 46-55.

[3] Ronssopoulos Nick R, Kelley S, Vincent F. Nearest neighbor queries. In *Proc. the ACM SIGMOD International Conference on Management of Data* (*SIGMOD 1995*), San Jose, California, USA, May 22-25, 1995, pp.71-79.

[4] Xiao Z, Meng X F, Xu J L. Quality aware privacy protection for location-based services. In *Proc. the 12th International Conference on Database Systems for Advanced Applications* (*DASFAA 2007*), Bangkok, Thailand, April 9-12, 2007, pp.434-446.

[5] Bettini C, Wang X S, Jajodia S. Protecting privacy against location-based personal identification. In *Proc. the 2nd VLDB Workshop on Secure Data Management* (*SDM 2005*), Trondheim, Norway, September 2-3, 2005, pp.185-199.

[6] Mokbel M F, Chow C Y, Aref W G. The new Casper: Query processing for location services without compromising privacy. In *Proc. the 32nd International Conference on Very Large Data Bases* (*VLDB 2006*), Seoul, Korea, September 12-15, 2006, pp.763-774.

[7] Li P Y, Peng W C, Wang T W, Ku W S, Xu J, Hamilton J A. A cloaking algorithm based on spatial networks for location privacy. In *Proc. IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing* (*SUTC 2008*), Taichung, Taiwan, China, June 11-13, 2008, pp.90-97.

[8] Duckham M, Kulik L. A formal model of obfuscation and negotiation for location privacy. In *Proc. the 3th International Conference on Pervasive Computing* (*Pervasive 2005*), Munich, Germany, May 8-13, 2005, pp.152-170.

[9] Kalnis P, Ghinita G, Mouratidis K, Papadias D. Preventing location-based identity inference in anonymous spatial queries. *IEEE Trans. Knowl. Data Eng.*, 2007, 19(12): 1719-1733.

[10] Ghinita G, Kalnis P, Skiadopoulos S. PRIVE: Anonymous location-based queries in distributed mobile systems. In *Proc. the 16th International Conference on World Wide Web* (*WWW 2007*), Banff, Alberta, Canada, May 8-12, 2007, pp.371-380.

[11] Indyk P, Woodruff D P. Polylogarithmic private approximations and efficient matching. In *Proc. the 3rd Theory of Cryptography Conference* (*TCC 2006*), New York, NY, USA, March 4-7, 2006, pp.245-264.

[12] Khoshgozaran A, Shahabi C. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *Proc. the 10th International Conference on Advances in Spatio and Temporal Databases* (*SSTD 2007*), Boston, MA, USA, July 16-18, 2007, pp.239-257.

[13] Yiu M L, Jensen C S, Huang X G, Lu H. SpaceTwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In *Proc. the 24th International Conference on Data Engineering* (*ICDE 2008*), Cancún, México, April 7-12, 2008, pp.366-375.

[14] Gong Z Q, Sun G Z, Xie X. Protecting privacy in location-based services using k-anonymity without cloaked region. In *Proc. the 11th International Conference on Mobile Data Management* (*MDM 2010*), Kanas City, Missouri, USA, May 23-26, 2010, pp.366-371.

[15] Ghinita G, Kalnis P, Khoshgozaran A, Shahabi C, Tan K L. Private queries in location based services: Anonymizers are not necessary. In *Proc. the ACM SIGMOD International Conference on Management of Data* (*SIGMOD 2008*), Vancouver, BC, Canada, June 9-12, 2008, pp.121-132.

[16] Cheng R, Zhang Y, Bertino E, Prabhakar S. Preserving user location privacy in mobile data management infrastructures. In *Proc. the 6th Workshop on Privacy Enhancing Technologies* (*PET 2006*), Cambridge, UK, June 28-30, 2006, pp.393-412.

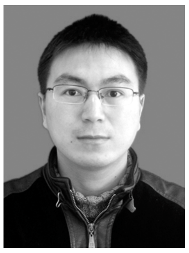
[17] Pan X, Xu J L, Meng X F. Protecting location privacy against location-dependent attacks in mobile services. *IEEE Transactions on Knowledge and Data Engineering*, 2011, http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.105.

[18] Xu J L, Tang X Y, Hu H B, Du J. Privacy-conscious location-based queries in mobile environments. *IEEE Trans. Parallel Distrib. Syst.*, 2010, 21(3): 313-326.

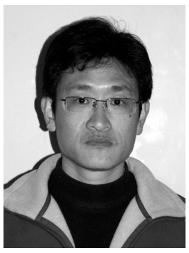
[19] Papadopoulos S, Bakiras S, Papadias D. Nearest neighbor search with strong location privacy. *PVLDB*, 2010, 3(1): 619-629.

[20] Moon B, Jagadish H V, Faloutsos C, Saltz J H. Analysis of the clustering properties of the Hilbert space-filling curve. *IEEE Trans. Knowl. Data Eng.*, 2001, 13(1): 124-141.

Wei-Wei Ni received the Ph.D. degree in computer application from Southeast University in 2005. He is now an associate professor of Southeast University and a member of the China Computer Federation. His research interests include data privacy protection and data mining.



Jin-Wang Zheng is now a Master candidate of Southeast University. His research interests include data mining and privacy protection in location-based service.



Zhi-Hong Chong received the Ph.D. degree in computer software from Fudan University in 2006. He is now an associate professor of Southeast University. His research interests include data management and analysis.