ShiftRoute: Achieving Location Privacy for Map Services on Smartphones

Peng Zhang, *Member, IEEE*, Chengchen Hu, *Member, IEEE*, Di Chen, Hao Li, *Member, IEEE*, and Qi Li, *Senior Member, IEEE*

***Abstract*—Map services, e.g., Google Maps, are gaining popularity for vehicle navigation. However, map service users have to provide sensitive information like precise geographic locations or detailed addresses, which are susceptible to accidental leakage or even data mining in the future. We find existing general-purposed location privacy protection mechanisms (LPPMs) not effective, when applied to map service on smartphones. This paper presents ShiftRoute, a new LPPM specially designed for map services on smartphones. ShiftRoute enables smartphone users to query a route between two endpoints on the map, without revealing any *meaningful* location information. The basic idea is to strategically *shift* the endpoints to nearby ones, such that: 1) the semantic meanings encoded in these endpoints (e.g., their addresses) change *much*, i.e., location privacy is largely protected; 2) the routes returned by map services change *little*, i.e., service usability is preserved. Specifically, we design a protocol to allow a mobile client to retrieve point of interests (POIs) close to the original endpoints, and an algorithm that selects shifted endpoints from these POIs, that achieves the privacy property of *geo-indistinguishability*. We implement an application of ShiftRoute on Android, and conduct experiments with real traces from a production map service. Experimental results show that ShiftRoute strikes a good tradeoff between location privacy and service usability.**

***Index Terms*—Map service, smartphone, location privacy.**

I. Introduction

MAP *Services* are becoming indispensable on smartphones. Google Maps, a popular map App, reported 1 billion users in 2015 [1]. Due to the nice feature like up-to-date map, traffic report, multiple travel options, etc., map services on smartphones are gradually replacing dedicated navigation systems on vehicles as a better option for getting directions. Although offering many useful functions, map services also pose great threats to location privacy at the same time [2]. First of all, the advance of GPS/WiFi positioning technology allows map service providers to obtain very accurate locations of their customers [3]. Moreover, users are often required to provide sensitive information of where they are from or where they are going. For example, they should input detailed addresses when querying a route from a source to a destination. These location data can reveal a lot of sensitive information of users, including home address, religious belief, health condition, political view, etc. [4].

Many location privacy protection mechanisms (LPPMs) have been proposed these years [5]. However, we find they have several limitations when applied to map services on smartphones. First, approaches based on cloaking [6]–[9] and obfuscation [10], [11] often assume a trusted anonymizer, which poses a new point of privacy leakage. P2P-based cloaking approaches [12]–[15] attempt to remove the need of trusted anonymizers, however, they are not autonomous, i.e., the functioning of them relies on the participation of many users in the system. Secondly, approaches based on dummies [16]–[18] let clients generate dummy requests together with real requests. However, to generate indistinguishable dummy requests, a client needs to maintain a local database of statistics (e.g., road traffic histories, query statistics), which inevitably incurs a large storage overhead for smartphones. Finally, space transformation approaches [19], [20] require modifications of map servers, thus are not easy to be used in real scenarios. Table I summarizes the limitations of existing LPPMs.

The above limitations motivate us to propose `ShiftRoute`, a new LPPM that can effectively preserve location privacy for map services on smartphones. The basic idea of `ShiftRoute` is illustrated in Fig. 1. Suppose Alice is located at “The Museum of Modern Art (A)”, and needs directions to “Memorial Sloan Kettering Cancer Center (B)”. Instead of querying a route $A \rightarrow B$, Alice queries a route from “The Modern Restaurant (C)” to “The Rockefeller University (D)”. From Fig. 1, we can see that the route of $C \rightarrow D$ overlaps a lot with that of $A \rightarrow B$, and can still be used by Alice for directions. The map server only observes a query from a restaurant to a university, without knowing that Alice is now at the museum and is going to visit the cancer center. Note that Alice does not have to contact an anonymizer, or require other users to cooperate. In addition, she only needs to send one query consisting of shifted source/destination, without sending any dummy requests.

Manuscript received April 10, 2017; revised November 22, 2017; accepted January 5, 2018. Date of publication January 9, 2018; date of current version May 14, 2018. This work was supported in part by the National Key Research and Development Program of China (No. 2017YFB0801703, 2016YFB0800100), in part by the National Natural Science Foundation of China (No. 61772412, 61402357, 61572278, 61672425, 61702407), and in part by the R&D Program of Shenzhen (No. JCYJ20170307153259323). The work of P. Zhang was supported by the K. C. Wong Education Foundation. The review of this paper was coordinated by Dr. Z. Cai. *(Corresponding author: Hao Li.)*

P. Zhang, C. Hu, and H. Li are with the Department of Computer Science and Technology, Xi’an Jiaotong University, Xi’an 710049, China (e-mail: p-zhang@xjtu.edu.cn; huc@ieee.org; hao.li@xjtu.edu.cn).

D. Chen is with the Luoyang Electronic Equipment Test Center of China, Luoyang 471003, China (e-mail: chendi610@126.com).

Q. Li is with the Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China (e-mail: liqi@csnet1.cs.tsinghua.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2018.2791402

TABLE I
COMPARISONS OF SHIFTROUTE WITH OTHER LPPMS

LPPMs	Desired Properties			
	No Trusted Anonymizer	No LBS Modification	Autonomous	No Local Database
Cloaking [6]–[9], [21]	×	×	×	√
Obfuscation [10], [11]	×	×	√	√
P2P Cloaking [12]–[15]	√	×	×	√
Dummy [17], [18]	√	√	√	×
Space Transformation [19], [20]	√	×	√	√
`ShiftRoute` (this paper)	√	√	√	√

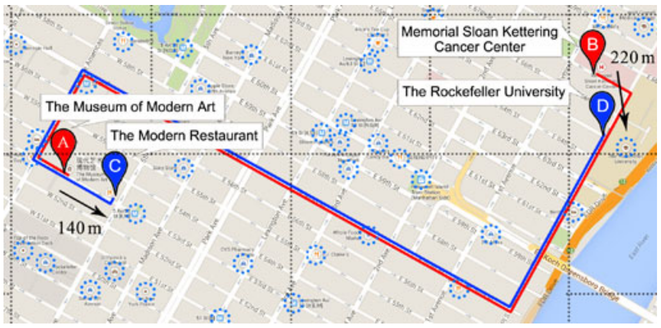

Memorial Sloan Kettering Cancer Center
B
220 m
The Rockefeller University
D
The Museum of Modern Art
The Modern Restaurant
A
C
140 m

Fig. 1. An example illustrating the basic idea of `ShiftRoute`.

As already noted from the above example, `ShiftRoute` is based on two key observations:

1) If the endpoints in a route query are geographically close to the endpoints in another route query, then the two routes would be quite similar. We will validate this observation with experimental results in Section II.
2) Even two locations are geographically close, their semantic meanings can be quite different. As seen in the above example, the cancer center and the university have quite different semantic meanings.

The key challenge here is how to find shifted endpoints without leaking users' real endpoints, while still guaranteeing service usability. We first dismiss the following straightforward approaches. (1) The user chooses arbitrary locations near the real endpoints for shifting. This approach may either fail to change semantic meanings of endpoints, or shift to endpoints that are farther than necessary. (2) The user queries map service to fetch places near the real endpoints. This requires the user to submit the real endpoints, which directly expose her location. (3) The user stores an off-line map on her device, and locally searches the map for shifted endpoints. This approach will inevitably impose additional storage overhead on smartphones.

`ShiftRoute` uses a novel two-stage approach to find shifted endpoints. In the first stage, the user sends a customized region r enclosing the real endpoint p to an anonymization server (AS), which returns a list of POIs that are within region r. In the second stage, the user strategically selects the shifted endpoints from these POIs, while ensuring the original endpoints are distinguishable from nearby POIs. For the first stage, since the user only sends regions instead of exact locations/addresses to the AS, the AS does not need to be trusted. In addition, we design a private POI retrieval method that allows users to only reveal necessary resolutions of their locations, based on the configured privacy level. For the second stage, we design a probabilistic endpoint selection method based on the recently proposed Geo-Indistinguishability model [22], [23]. Using this method, the expected distance from the original endpoints to the shifted ones can be minimized, thus the service usability is assured.

In sum, our contribution is three-fold:

1) We propose `ShiftRoute`, a new LPPM for map services on smartphones, featuring: (1) needless of trusted anonymizer; (2) tunable tradeoff between privacy and usability; (3) fully autonomous (needless of other users' participation).
2) We implement `ShiftRoute` as an Android application, and demonstrate its applicability in real life.
3) We evaluate `ShiftRoute` with real route queries from a large production map service, demonstrating that it can indeed achieve location privacy while maintaining service usability.

The rest of this paper proceeds as follows. Section II overviews the basic idea of `ShiftRoute`. Section III and Section IV present the detailed design and implementation of `ShiftRoute`, respectively. Section V introduces a location privacy metric, and some simulative analysis. Section VI evaluates the location privacy and service usability achieved by `ShiftRoute` with experiments. Section VII reviews some related work, Section VIII discussed some potential issues, and Section IX concludes the paper.

II. `ShiftRoute`: AN OVERVIEW

This section presents our key observation of route similarity; after dismissing a strawman approach, we introduce our approach.

A. *Key Observation: Route Similarity*

We observe that if two endpoints s and d are sufficiently close to s^* and d^*, respectively, then the shortest path from s to d, and that from s^* to d^* will be quite similar. In the following, we use experiments to validate this observation.

First, we randomly choose a source/destination pairs (s, d), and query a map service for the shortest route, say r. Here, the geographical distance between s and d is set to be the averaged value in our dataset. Then, we replace s with a random location s^* which is l meters away from s, and query to get the shortest

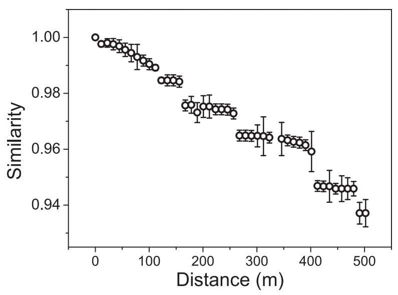


Fig. 2. Similarity of the original route r and the shifted route r^* for different values of l. The error bars indicate standard deviations.

route, say r^*. Here the queries are made through a public route query API from a production map service. The similarity of routes r and r^* is calculated by:

$$\texttt{Similarity}(r, r^*) = ||r \cap r^*||/\texttt{max}(||r||, ||r^*||)$$

Here, $r \cap r^*$ is the set of common road segments in r and r^*, $||r||$ is defined as the total length of road segments in r.

Fig. 2 shows the route similarity for different values of l. Note each point is averaged over 20 queries, and the error bars indicate standard deviations.

B. Strawman Approach

Based on the above observation, it is natural that shifting the endpoints to nearby locations can protect location privacy. Consider the strawman approach where the client can simply picks random points around the original endpoints as shifted endpoints. This approach, however, may either break location privacy or degrade service usability. For example, suppose the source s falls in a large area belonging to a specific POI, say a university. Then, we should not make the shifted endpoint s^* too close to s, since s^* may still fall within that POI, thus not effectively hiding the semantic meaning of s. On the other hand, we should not make s^* always too far from s either, since s may fall in a small area (e.g., a coffee shop) and a far-away s^* will unnecessarily degrade the usability of the shifted route. Thus, to make a better tradeoff between location privacy and user usability, the endpoint shifting method should be aware of the POI distribution. In the next section, we will show how to design such a method.

C. Our Approach

As noted in Section II-B, randomly shifting endpoints agnostic of POI distribution can either break privacy or degrade service usability. The basic idea of `ShiftRoute` to address this problem is shifting the endpoints to POIs (instead of random locations), in order to guarantee the shifted source s^* is always associated with a different POI than the original one. At the same time, `ShiftRoute` allows the user configure the number of candidate POIs that are used for shifting, so as to maintain the specified level of route usability.

At the highest level, `ShiftRoute` consists of two stages, i.e., *Private POI Retrieval* and *Private Endpoint Selection*. In the following, we will specify the task and outline the challenges for these two stages, respectively.

Stage 1. Private POI Retrieval: In this stage, the client retrieves two sets of POIs that are close to the original source/destination, respectively. Alternatively, the client can download the whole map from the map service provider, and can locally search for the POIs. However, this may incur prohibitive storage and bandwidth cost for smartphones due to large map size and POI updates. In `ShiftRoute`, we let clients interact with an external Anonymization Server (AS) to *privately* retrieve POIs. Specifically, the AS partitions the map into grids, with each grid containing a set of POIs. Then, the client submits the grids that the endpoints fall in, and the AS returns all POIs within that grid. Indeed, from the grids submitted by users, the AS can still learn the neighborhood of users. However, this coarse location is not very useful since a pair of geographically close POIs may have very different semantic meanings, as already illustrated in Fig. 1.

Stage 2. Private Endpoint Selection: In this stage, the client selects two POIs from the two sets as the shifted source/destination. For this stage, we will give a baseline approach where a shifted endpoint is uniformly chosen from all retrieved POIs. This approach is very simple to implement, while it does not take the service usability into consideration. As a result, the chosen endpoint may not be the nearest one that satisfies the privacy constraint, and the returned routes are suboptimal in terms of usability. Inspired by the notion of *Geo-Indistinguishability* [22], we propose a new strategy for shifted endpoint selection. This strategy can guarantee location privacy while minimizing the expected distance of shifted endpoints to original ones. We will use both simulations and experiments to demonstrate the advantage of this approach over simple uniform selection.

Fig. 3 shows the high-level message flow of `ShiftRoute`, where (1)–(3) corresponds to the POI retrieval stage, and (4) is the endpoint selection stage. In the following, we will present the design details of these two stages.

III. Design Details

In this section, we will present the design of `ShiftRoute`, which has two stages: *the private POI retrieval stage*, and *the private endpoint selection stage*.

A. The Private POI Retrieval Stage

In the grid-based private POI retrieval stage, the client retrieves *candidate* POIs by interacting with the anonymization server (AS). We first study how the AS constructs the grid-to-POI mapping, in prior to the client-side POI retrieval. The straightforward way is to assign a POI to a grid that geographically encloses it. However, this will result in that given a location p inside the grid g, the corresponding POIs of g are not necessarily the nearest ones for p. Take Fig. 4(a) for example. If the client needs to retrieve POIs near the location p, then she will query the AS with the top-left grid (A1), and the AS will return POIs numbered 1 through 4. However, the four nearest neighbors of p are actually 3, 4, 5, 6. Selecting from non-nearest

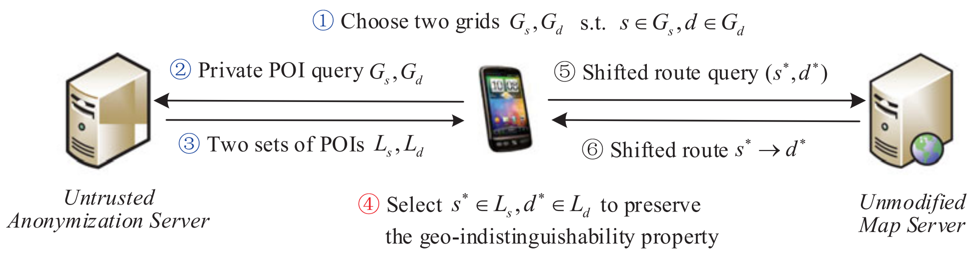

① Choose two grids G_s, G_d s.t. $s \in G_s, d \in G_d$
② Private POI query G_s, G_d
③ Two sets of POIs L_s, L_d
⑤ Shifted route query (s^*, d^*)
⑥ Shifted route $s^* \to d^*$
Untrusted Anonymization Server
④ Select $s^* \in L_s, d^* \in L_d$ to preserve the geo-indistinguishability property
Unmodified Map Server

Fig. 3. The high-level message flow of ShiftRoute. ① The client determines the grids G_s and G_d, which the source s and the destination d fall in, respectively. ② The client sends the identifiers of G_s and G_d to the AS. ③ The AS returns L_s and L_d, two sets of POIs falling inside G_s and G_d, respectively. ④ The client selects two POIs $s^\star \in L_s$ and $d^\star \in L_d$, ⑤ The client queries the map server for the route from $s^\star$ to $d^\star$. ⑥ The map server returns a route $s^\star \to d^\star$ to the client. ①–③ correspond to the private POI retrieval stage, ④ corresponds to the private endpoint selection stage, and ⑤–⑥ are standard route queries.

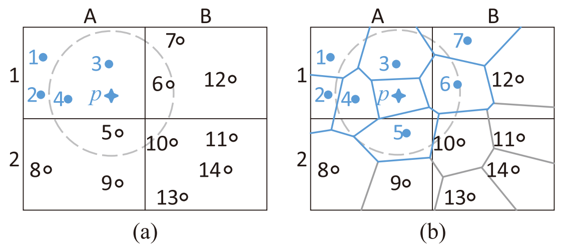


Fig. 4. An example for superimposing Voronoi diagram on top of grids.

neighbors will potentially cause the client to shift p to POIs that are farther away.

Inspired by [20], we superimpose *Voronoi diagram* [24] on top of grids. A Voronoi diagram (or V-diagram for short) is composed of Voronoi cells (or V-cells for short), each corresponding to a point centered at it. A nice property of V-diagram is that given a V-cell v of point p, the nearest neighbor for any point inside v is p. For example, Fig. 4(b) shows the V-diagram of Fig. 4(a). With the V-diagram, we can define the nearest neighbors for a grid as follows: *the nearest neighbors for a grid are those POIs whose V-cells are inside or intersecting with the grid*. Taking Fig. 4(b) for an example, the nearest neighbors for top-left grid A1 are POIs numbered 1 through 7.

To speed up the POI retrieval process, the AS constructs a table to record all internal or intersecting V-cells for each grid. We term this table as the *Grid V-cell Table (GVT)*. In the following, we show how to efficiently construct the GVT.

GVT Construction: The naive way to construct GVT, as adopted by [20], is to iterate over all POIs, and for each POI, determine whether its V-cell is inside or intersecting with each GVT. Clearly, this method has a complexity of $O(mn)$, where m and n is the number of grids and POIs, respectively. Even the construction of GVT can be done offline, this naive method may be very inefficient considering the large number of grids and POIs on a real map.

To make this construction of GVT more efficient, we propose a new method which has a complexity of $O(n \log m)$. The basic idea is divide-and-conquer: we recursively divide the map into four sub-grids, assigns V-cells to these sub-grids in each recursion. This details are summarized as Algorithm III-A, where the recursive function ConstructGVT takes a grid G and its V-cells set V as the input. First, SplitGrid splits the grid G into four sub-grids G_1 through G_4 (Line 1), and initializes the

Algorithm 1: ConstructGVT(G, V).

```
Input: $G$: The grid to be processed
Input: $V$: The V-cell set for $G$
1  $(G_1, G_2, G_3, G_4) \leftarrow$ SplitGrid($G$);
2  Initialize V-cell set $V_i \leftarrow \{\}$ for each $G_i, (1 \le i \le 4)$;
3  while $V \neq \emptyset$ do
4      $v \leftarrow$ a random V-cell in $V$;
5      foreach $1 \le i \le 4$ do
6          if $v$ is contained within $G_i$ then
7              $V_i \leftarrow V_i \cup \{v\}$, $V \leftarrow V \backslash \{v\}$;
8          else if $v$ intersects with $G_i$ then
9              $(v_{in}, v_{out}) \leftarrow$ SplitVcell($v$, $G_i$);
10             $V_i \leftarrow V_i \cup \{v_{in}\}$, $V \leftarrow V \cup \{v_{out}\} \backslash \{v\}$;
11         end
12     end
13 end
14 foreach $1 \le i \le 4$ do
15     AddToGVT($G_i$, $V_i$);
16     if $G_i.size \ge MinSize$ then
17         ConstructGVT($G_i$, $V_i$);
18     end
19 end
```

V-cell set for each sub-grid as empty (Line 2). In each iteration, a V-cell v is chosen from V (Line 4), and there are two possibilities for a V-cell v and a sub-grid G_i: (1) v is inside G_i, then v is assigned to the V-cell set of G_i (Line 6–7); (2) v intersects with G_i, then the function SplitVcell partitions v into two parts: v_{in}, the part overlapping with the G_i, and v_{out}, the remaining part not overlapping with G_i (Line 9). v_{in} is assigned to G_i, while v_{out} is put back to V (Line 10). When V becomes empty, the algorithm adds all the four sub-grids into the GVT (Line 15). If a sub-grid G_i is above a threshold (meaning it can be further partitioned), the algorithm recursively calls ConstructGVT for G_i and V_i (Line 17).

Since the depth of recursion is $\log_4 m$, and for each recursion, the algorithm iterates over the n POIs exactly once, Algorithm III-A has a computation complexity of $O(n \log m)$. We will evaluate the performance of Algorithm III-A numerically in Section VI-F.

POI Retrieval: When retrieving POIs, we allow a user to specify a privacy parameter Th, through which a user can personalize her protection level. Our POI retrieval aims to achieve

Algorithm 2: `RetrievePOIs`(p, Th).

```
Input: $Th$: the protection level
Input: $p$: an endpoint in route query
1  $level \leftarrow 1$ ;
2  while $level < MaxLevel$ do
3      $grid \leftarrow$ GetGrid($p$, $level$);
4      $sub-grid \leftarrow$ GetGrid($p$, $level+1$);
5      $count \leftarrow$ GetPOICount($grid$);
6      if $count[sub-grid] < Th$ then
7          break;
8      end
9      $level \leftarrow level+1$;
10 end
11 return GetPOIs($grid$);
```

the privacy constraint: *the grid that user specifies to the AS should always contain more than Th POIs, such that the AS cannot distinguish which POI the real endpoint is associate with.* Clearly, specifying a sufficiently large grid will always satisfy the condition. However, this will result in poor usability since the shifted endpoints can be too far away from the original ones.

Here, we propose an approach to enable a client to determine the smallest grid that satisfies her privacy constraint. We organize the GVT as a quad-tree, where each layer corresponds to a different resolution, i.e., grid size. Algorithm III-A summarizes the POI retrieval process. Suppose the client needs to retrieve nearby POIs for an endpoint p, with privacy parameter Th. The client starts from the first layer, i.e., $level = 1$ (Line 1). For each $level$, the client calls `GetGrid` to determine $grid$ at layer $level$ and $sub-grid$ at layer $level + 1$, both of which contain the endpoint p (Line 3-4). Then, the client calls `GetPOICount` to query the AS about the number of POIs in the four sub-grids of $grid$ (Line 5). If the $sub-grid$ has less than Th POIs, then the client calls `GetPOIs`($grid$) to query all POIs inside $grid$ (Line 6–8); otherwise, the client increments the $level$ by one and continues the above process (Line 9).

Since the map is recursively partitioned into four grids, the client can locally evaluate `GetGrid`. The other two functions `GetPOICount` and `GetPOIs` are evaluated by the AS. In the current implementation, the grid identifier is a tuple $\langle lat, lon, level\rangle$, where lat, lon are the latitude and longitude of the gird's top-left corner, and $level$ is the level of the grid.

Fig. 5 further illustrates the POI retrieval process, with the left side showing the quad-tree of grids, each of which is labeled with number of POIs in them. The client would first query the AS for the level-1 grid with 195 POIs, and find the sub-grid containing p has 45 POIs. Thus, the client moves one level down and chooses the level-2 grid with 45 POIs. Even though one sub-grid has only 4 POIs, the sub-grid containing p still has 16 POIs. Thus, the client moves on to level-3. This time the sub-grid has only 3 POIs, thus the client stops and queries the AS for the 16 POIs in the level-3 grid.

Note the above quad-tree based method is similar to R-Trees [25], an indexing method for geographical databases.

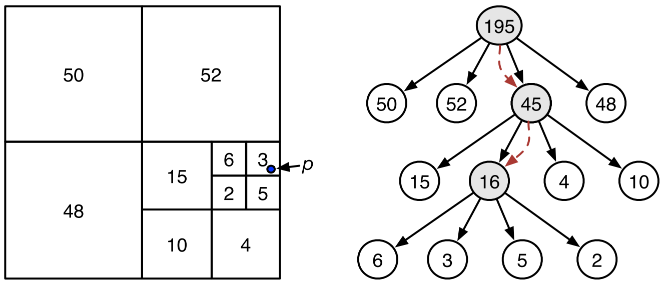


Fig. 5. An example illustrating the private POI retrieval process in `ShiftRoute`. The real endpoint p is marked as a blue circle on the left sub-figure.

The difference is that nodes in our quad-tree store all POIs within the corresponding grids, while nodes in R-Trees store different sizes of rectangle areas. In addition, the searching of the quad-tree is also different: our approach allows clients to probe for the right resolution of grids for querying.

B. The Private Endpoint Selection Stage

After collecting a bunch of POIs around both the source and destination, the client should select a pair of them as the shifted endpoints for query. In the following, we will discuss some POI selection strategies that respect both privacy and usability.

Uniform Selection: This is the simplest selection method, where all POIs are chosen with a uniform probability. Formally, let x be the original endpoint to query, and $G^t(x)$ be the grid that x falls in given the threshold value $Th = t$. Let $V(g)$ be the set of all POIs whose V-cells fall in or intersect with grid g. For simplicity, define $P^t(x) = V(G^t(x))$ as the set of all candidate POIs that can be selected for shifting. Then, we can uniformly select a POI y with probability $1/|P^t(x)|$, if $y \in P^t(x)$, and 0 otherwise.

Uniform selection is simple and effective: it can guarantee that the original endpoint x is not distinguishable from any other POI within $P^t(x)$. However, uniform selection suffers from suboptimal performance in terms of route usability. The reason is it does not take the distance between x and y into account, which is a critical factor for the usability of shifted route. In this following, we present a novel selection strategy which can significantly improve route usability while only sacrificing a little privacy.

Differential-Private Selection: The selection strategy is based on *Geo-Indistinguishability* [22], a new notion of location privacy inspired by the differential privacy in database. Here, we simply refer to Geo-Indistinguishability as Differential Location Privacy (DLP). The privacy level of DLP is controlled by a single parameter ϵ: the smaller ϵ is set, the better location privacy is protected. Specifically, an LPPM is modelled as a probabilistic function that given a *real* location, outputs an *observed* location. Then, for two different locations x and x', the LPPM should produce two different probability distributions over all possible observations. The LPPM is said to satisfy DLP with parameter ϵ if the difference of these two probability distributions is no larger than $\epsilon d(x, x')$, where $d(x, x')$ is the geographical distance between x and x'.

In our setting, let $f_x^{(t,\epsilon)}(y)$ be the probability of choosing y as the shifted POI for x, given the threshold value t and DLP parameter ϵ, and let $\mathcal{X}$ be the set of POIs inside the grid $G^t(x)$. Then, we require the selection strategy $f^{(t,\epsilon)}$ should satisfy the following constraint:

$$f_x^{(t,\epsilon)}(y) \le e^{\epsilon d(x,x')} f_{x'}^{(t,\epsilon)}(y),\ \forall x, x' \in \mathcal{X}, y \in P^t(x) \quad (1)$$

Finding a selection strategy satisfying the above constraint can be formalized as the following optimization problem:

$$\begin{aligned} \min\ & \max_{x\in\mathcal{X}} \sum\nolimits_{y\in P^t(x)} f_x^{(t,\epsilon)}(y) d(x,y) \\ \text{s.t.}\ & f_x^{(t,\epsilon)}(y) \le e^{\epsilon d(x,x')} f_{x'}^{(t,\epsilon)}(y);\ \forall x, x' \in \mathcal{X}, \forall y \in P^t(x) \\ & \sum\nolimits_{y\in P^t(x)} f_x^{(t,\epsilon)}(y) = 1;\ \forall x \in \mathcal{X} \\ & f_x^{(t,\epsilon)}(y) \ge 0;\ \forall x \in \mathcal{X}, \forall y \in P^t(x) \end{aligned} \quad (2)$$

The above optimization problem is inspired by [23], where the objective is to minimize the expected distance from the original endpoint to the shifted one. The difference is that we are minimizing the maximum distance between the original and shifted endpoints. In this way, we can prevent the shifted endpoints from deviating too far away from the original ones. For the route query function of map services, this can better guarantee the usability of returned routes, compared with minimizing the expected distances as in [23]. The nonlinear problem (2) can be easily transformed into a linear program whose objective is min t, and whose constraints include those in (2) and an extra constraint $\sum_{y\in P^t(x)} f_x^{(t,\epsilon)}(y) d(x,y) \le t;\ \forall x \in \mathcal{X}$.

As the above linear program involves $\boldsymbol{O}(|\mathcal{X}|^2|P^t(x)|)$ constraints, solving it at the client side is clearly inefficient. To address this issue, `ShiftRoute` lets the AS solve the above problem and return results (a probability distribution $f_x^{(t,\epsilon)}$ for each $x \in \mathcal{X}$) to the client. Suppose the client's real endpoint is x_0, then she will choose the shifted endpoint y_0 with probability $f_{x_0}^{(t,\epsilon)}(y_0)$. Note here the AS does not know the user' real endpoint x_0, and thus location privacy is preserved. To prevent the AS from intentionally returning invalid $f_x^{(t,\epsilon)}$ in order to expose the user, the client should check whether $f_x^{(t,\epsilon)}$ satisfies (1) for each $x \in \mathcal{X}$.

IV. Implementation and Case Study

In this section, we will first give the implementation of `ShiftRoute`, and then present a case study based on the implementation.

A. Implementation

As shown in Fig. 6, our prototype implementation of `ShiftRoute` consists of an anonymization server (AS) and a mobile application (App).

Anonymization Server (AS): The AS is implemented with Python, exposing a REST API for the App. The core components include *the GVT constructor* and *the DLP solver*. The GVT constructor first computes the Voronoi diagram (V-diagram) from raw POI data that we collected, using the python version of Fortune's sweep line algorithm [26], [27]. The complexity of this algorithm is $\boldsymbol{O}(n \log n)$, where n is the number of all POIs. Then, based on the V-diagram, the GVT is constructed using Algorithm III-A (see Section III-A). The *DLP solver* calculates the selection probability distribution for each grid, by solving the linear program (see Section III-B) with the Matlab optimization toolbox. Note that the calculation is offline, and the probability distributions are stored in the GVT for satisfying user queries.

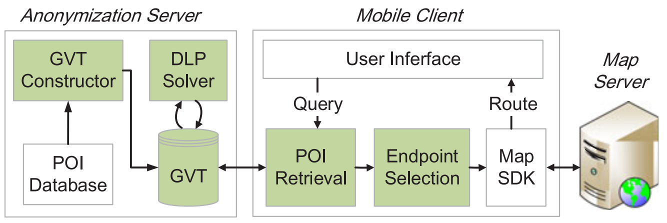


Fig. 6. System architecture, shaded parts belong to `ShiftRoute`.

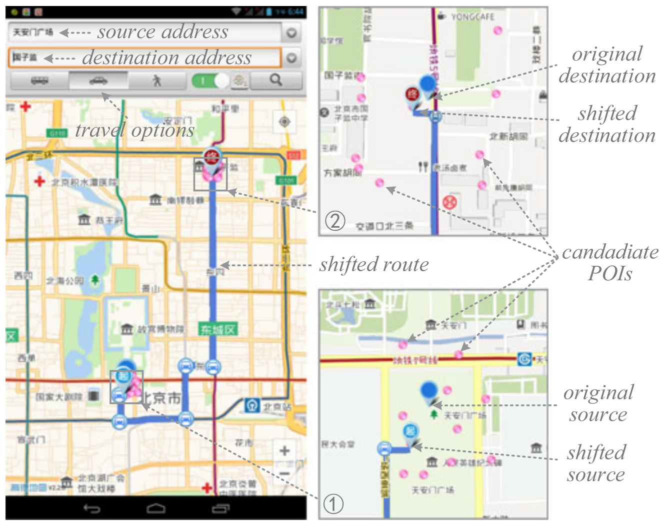


Fig. 7. Screenshot of the `ShiftRoute` App.

Mobile Application (App): The App is implemented as an Android application (version 4.4), using the map SDK released by Amap [28], the second largest map service provider in China. The core components include: (1) the *POI retrieval module* which interacts with AS to retrieve candidate POIs, according to Algorithm III-A; (2) the *endpoint selection module* which selects shifted endpoints from the candidate POIs, with either uniform or DLP selection strategy; (3) the *map SDK* which issues route queries and displays the returned routes on the user interface. Note that we only invoke the route query API in the map SDK, while disabling it to access the device's current locations.

B. Case Study

We continue to test the implementation of `ShiftRoute` with a real case. In this case, we set $Th = 6$, and search a route between two endpoints in the city of Beijing. Fig. 7 shows the screenshot of App, where the blue bubbles represent the original source and destination, and the pink circles are POIs retrieved

TABLE II
SERVICE USABILITY FOR THE CASE OF FIG. 7

Selection Strategy	Endpoint Deviation (m)		Route Deviation (m)		
	Source	Destination	By car	By bus	By foot
Uniform	177.8	181.5	−1027.5	−2289.0	256.7
DLP ($\epsilon = 0$)	118.9	171.3	−778.0	−2179.1	251.9
DLP ($\epsilon = 1$)	107.0	170.9	−903.5	−2484.2	207.3

from the AS. We can see that there are 11 and 10 POIs for the source and destination, respectively. The POIs selected as the shifted source/destination, and the route between them are also shown.

We use two selection strategies, i.e., uniform selection, DLP selection with $\epsilon = 0$ and $\epsilon = 1$. For each strategy, we run the search for 100 times, with travel options set to "by car", "by bus", or "by foot". Table II reports (1) the endpoint deviation, defined as the distance from the original endpoint to shifted endpoint, and (2) the route deviation, defined as the distance of shifted route minus that of the original route.

From Table II, we can observe that DLP selection has a smaller endpoint deviation compared with uniform selection. Increasing the parameter ϵ can further make the deviation smaller, but the improvement is not significant. An interesting observation is that the shifted routes are even shorter than the original ones for both "by car" and "by bus". The results are due to this specific setting, where most neighboring POIs of the source are closer to the destination than the source itself, for routes of "by car" and "by bus", as can be seen in bottom-right corner of Fig. 7). On the other hand, when traveling "by foot", the shifted route is slightly longer than the original one, but the difference becomes smaller with the increase of ϵ. The reason is that the routes for "by foot" have few road restrictions compared with the other two travel options, and highly depend on the Euclidean distances from the sources to destinations.

From this case, we can draw a conclusion that DLP selection strategy can achieve a better service usability than the uniform selection strategy, in terms of endpoint deviation. There is, however, no necessary connection between selection strategy and route deviation, which depends on the relative position of original endpoints and POIs, as well as the travel options.

V. MODEL AND ANALYSIS

In this section, we first present a location privacy metric, and then use simulation to analyze the impact of endpoint selection strategies on location privacy and service usability.

A. Location Privacy Metric

This section defines a location privacy metric by adapting the model introduced in [29], where the authors study how to maximize the location privacy subject to service usability constraints. Here, we are only interested in quantifying the achievable location privacy, and as will be shown later, our location privacy metric has a simpler form than that in [29].

Let $\varphi(r)$ be the prior probability that the user chooses r as an endpoint in the route query. Define $f(r'|r)$ as the probability that given the real endpoint is r, the shifted endpoint is r'. Similarly, define $\Pr(r|r')$ as the posterior probability that given the shifted endpoint is r', the real endpoint is r. Then, we have:

$$\Pr(r|r') = \frac{\Pr(r, r')}{\Pr(r')} = \frac{f(r'|r)\varphi(r)}{\sum_{r^*} f(r'|r^*)\phi(r^*)} \tag{3}$$

Let $\hat{r}$ be the endpoint that is guessed by the adversary on observing r'. Suppose the shifted endpoint is r', and the adversary always chooses the optimal $\hat{r}$ that minimizes the user's location privacy. Thus, the user's privacy given the shifted endpoint is r' can be calculated as:

$$\min_{\hat{r}} \sum_{r} \Pr(r|r') d_p(r, \hat{r}) \tag{4}$$

Here, $d_p(r, \hat{r})$ denotes the *semantic distance* between the guessed endpoint $\hat{r}$ and the real endpoint r. A simple definition for $d_p(r, \hat{r})$ is $d_p(r, \hat{r}) = 0$ if $\hat{r} = r$, or 1 otherwise. Averaging over all possible r', the location privacy that can achieved by `ShiftRoute` is then:

$$LP(f, d_p, \varphi) = \sum_{r'} \Pr(r') \min_{\hat{r}} \sum_{r} \Pr(r|r') d_p(r, \hat{r}) \tag{5}$$

Suppose $d_p(r, \hat{r})$ is non-zero only when $\hat{r} \neq r$, we further reduce (5) as follows:

$$\begin{aligned} LP(f, d_p, \varphi) &= \sum_{r'} \Pr(r') \min_{\hat{r}} \sum_{r \neq \hat{r}} \Pr(r|r') \\ &= \sum_{r'} \Pr(r') \min_{\hat{r}} (1 - \Pr(\hat{r}|r')) \\ &= \sum_{r'} \Pr(r')(1 - \max_{r} \Pr(r|r')) \\ &= 1 - \sum_{r'} \max_{r} f(r'|r)\varphi(r) \end{aligned} \tag{6}$$

(6) is simpler and more intuitive compared with the equation given in [29]. The sum on the right of (6) corresponds to the adversary's advantage or success probability. We can see that to maximize its advantage, the adversary tries to find the endpoint r that is mostly likely to be shifted to the observation r'. The location privacy of the user is then one minus the adversary's advantage.

Let the total number of endpoints be n, then it is easy to verify that $0 \leq LP(f, d_p, \varphi) \leq 1 - 1/n$. By letting $f(r'|r) = 1$ when $r' = r$, and 0 otherwise (stands for no protection at all), we have $LP(f, d_p, \varphi) = 0$. By letting $f(r'|r) = 1/n$ for all r and r' (stands for uniformly random selection), we have $LP(f, d_p, \varphi) = 1 - 1/n$.

B. Simulative Results

In the following, we use simulation to analyze the location privacy and service usability achieved by the two endpoint selection strategies, i.e., uniform and DLP. In Section VI-E, we will further compare these two strategies based on real datasets.

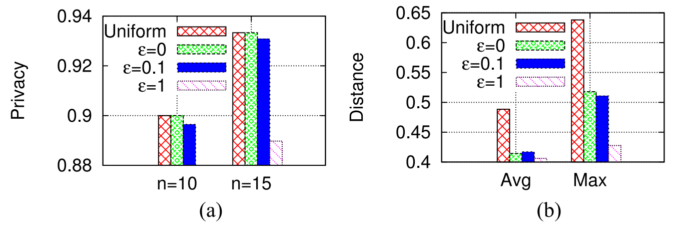


Fig. 8. Simulative results on location privacy and service usability, when the uniform and DLP endpoint selection strategies are used. n and m are the number of POIs and V-cells in the unit square. n is set to 10, 15, and $m = n + 5$. Note the unit for distance in (b) is 1 since we use a unit square. (a) Location Privacy ($n = 10, 15$). (b) Service Usability ($n = 10$).

First, we generate a unit square as the map, and randomly generate n points inside it as POIs. To simulate V-cells, we generate another $m - n$ random points inside the square. Then, we solve the DLP selection probabilities, and calculate the location privacy using the metric defined by (6). For service usability, we use the Euclidean distance between the shifted and original endpoints.

Fig. 8(a) reports the location privacy calculated with (6). n is set to 10 and 15 (m is set to $n + 5$). We can see that the uniform and the DLP strategy ($\epsilon = 0$) achieves roughly the same location privacy, and the location privacy drops for DLP selection when ϵ becomes larger. Specifically, for $n = 10$, the optimal value for privacy is 0.9, which is achieved by the uniform strategy, and the privacy achieved by DLP is very close to 0.9 when $\epsilon = 0$ or $\epsilon = 0.1$. Fig. 8(b) shows the service usability in terms of distance between the shifted and original endpoints when $n = 10$. We can see that DLP (even for $\epsilon = 0$) achieves a much smaller distance than uniform selection. The above results imply that DLP selection strikes a better tradeoff between privacy and usability than the uniform selection. Results based on real data sets in Section VI-E confirm this point.

C. Privacy Analysis

When a user only queries a specific route once, it is hard for the adversary to guess the real endpoints. However, when she queries routes between two endpoints multiple times (e.g., she may travel from home to work on a daily basis), the adversary may launch two attacks: *frequency analysis* and *intersection attack*. In the following, we show how `ShiftRoute` can defend against these two attacks.

Frequency analysis: In this attack, the adversary counts the number of times each shifted endpoint is observed. After collecting a sufficient number of shifted endpoints, the adversary guesses the endpoint with the highest frequency as the real endpoint. This attack cannot be totally prevented unless the uniform selection strategy is used. However, by employing differential location privacy (DLP) [22], [23], we can make the success probability small. According to (1), the probability of distinguishing whether x or x' are the real endpoint is bounded by $\epsilon d(x, x')$. Thus, we can make the success probability of frequency analysis arbitrarily close to that can be achieved by uniform selection, by choosing sufficiently small ϵ.

Intersection attack: For each observed shifted endpoint, the adversary draws a circle whose center is the shifted endpoint and whose radius is the privacy threshold. When there are multiple shifted endpoints, the adversary guesses POIs within the intersection of these circles as the real endpoint. This attack does not work since `ShiftRoute` chooses POIs that are within the same grid of the real endpoint, rather than in a circular area centered at the real endpoint. Thus, for the same endpoint e, it is always shifted to the same set of POIs (those corresponding to the V-cells of the grid enclosing e).

TABLE III
DATASET DESCRIPTION

	Map Size	# of POIs	# of queries
Beijing	16 km $*$ 17 km	14,607	367,194
Xi'an	4 km $*$ 3 km	828	131,989

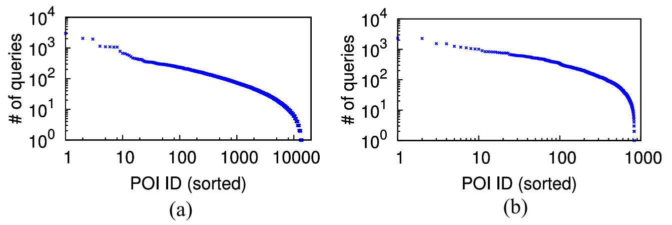


Fig. 9. Number of queries for each POI in the two cities. The POI IDs are sorted in descending order of # queries. (a) Beijing. (b) Xi'an.

VI. EVALUATION

We have presented simulative results on the endpoint selection strategies in Section V-B. This section continues to evaluate the overall performance of `ShiftRoute` with experiments based on real traces. The major metrics we consider include location privacy, service usability, and computation cost.

A. Dataset

Our dataset consists of two parts. The first part includes the route query logs from one of the largest map service providers in China. The logs contain over 3 million route queries made by users across the country, from 20 May to 6 Aug, 2013. We select the queries falling inside two large cities in China, i.e., Beijing and Xi'an. Specifically, we choose a rectangle area in each city, with one of size 16 km $*$ 17 km and the other of size 4 km $*$ 3 km. The second part of our dataset includes the POIs within these two rectangle areas. We crawled these POIs using the public map API from the same map service provider. The detailed description of the dataset is given in Table III. In order to obtain POIs' querying frequencies, we map both the endpoints of each route query in the query logs to their nearest POIs. Fig. 9 reports the number of queries mapped to each POI, in log-log coordinates. We can see that the distributions roughly conform to the Zipf's law [30].

B. GVT Construction and POI Retrieval

We first examine the effectiveness of GVT construction specified in Algorithm III-A. Fig. 10(a) shows the V-diagram for the

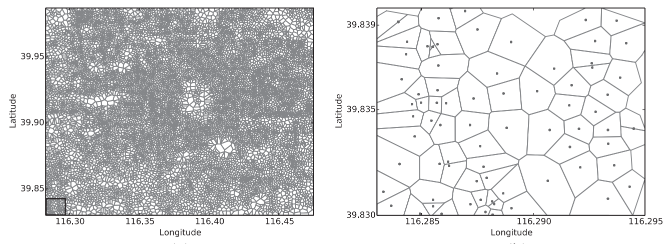


Fig. 10. The Vononoi diagram for the city of Beijing. (a) is the full diagram, and (b) is the zoom-in of the left bottom corner. (a) Full. (b) Small.

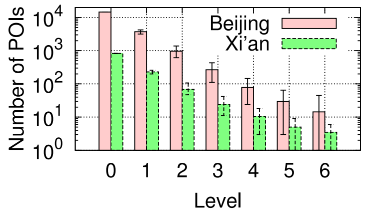


Fig. 11. Number of POIs per grid for different levels in the GVT.

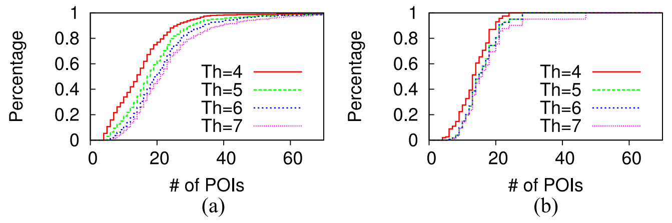


Fig. 12. Number of POIs in the gird specified by the user, for thresholds of $Th = 4, 5, 6, 7$. (a) Beijing. (b) Xi'an.

rectangle area of Beijing, and Fig. 10(b) is the zoom-in of the left-bottom corner of Fig. 10(a). The GVT for each city consists of 6 layers of grids, and Fig. 11 shows the average number of POIs associated with a grid for each layer.

Then, we evaluate the effectiveness of POI retrieval specified in Algorithm III-A. Recall that a user who is retrieving POIs near an endpoint p first specifies a parameter Th, and then uses Algorithm III-A to determine the smallest grid that contains p and no less than Th POIs. Here we are interested in how many POIs are in the grid specified by the user, and whether this number is always larger than the parameter Th. Fig. 12 reports the cumulative distribution for the number of POIs contained in a chosen grid, for Th ranging from 4 to 7. We can see that for both cities, the number of POIs is strictly larger than Th, concentrating in a relatively small range.

C. Location Privacy

In this experiment, we will evaluate the level of location privacy provided by ShiftRoute, using the location privacy metric specified by (6). First, we need to determine the prior distribution of POIs, i.e., $\varphi(p)$. Here, we approximate it using the query frequency of POIs, according to the route query logs. Specifically, let $\mathcal{P}$ be the set of all POIs within the area we consider (e.g., a city). For a POI $p \in \mathcal{P}$, let $N(p)$ be the number

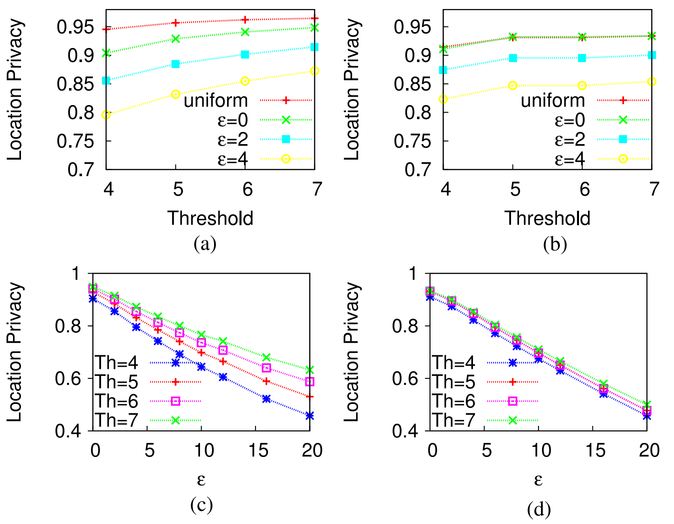


Fig. 13. The location privacy achieved by ShiftRoute for different values of Th and ϵ. (a) Beijing, Privacy vs. Th. (b) Xi'an, Privacy vs. Th. (c) Beijing, Privacy vs. ϵ. (d) Xi'an, Privacy vs. ϵ.

of queries whose sources or destinations are mapped to p. Then, we calculate $\varphi(p) = N(p)/\sum_{p' \in \mathcal{P}} N(p')$.

Fig. 13 reports the level of location privacy for different values of Th and ϵ. We can observe the privacy level is higher using uniform selection than using DLP selection, while the difference is not significant. In addition, the privacy level improves with the increase of threshold value Th. For DLP selection, the increase of ϵ will result in a lower level of location privacy. The above results echo our simulative analysis in Section V-B.

D. Service Usability

ShiftRoute inevitably degrades the service usability due to the usage of shifted endpoints for route query. To quantify the degradation of service usability, we define the following two metrics:

1) Endpoint Deviation: $Dev_e = ||e, e^*||$, defined as the shortest path from the original endpoint e to shifted endpoint e^*, where the endpoint e can be either a source or a destination. This metric reflects how convenient it is to find the shifted source when departing, and find the original destination when arrived.
2) Route Deviation: $Inc = (||s^*, d^*|| - ||s, d||)^+$, defined as the increased distance of the shifted route (from s^* to d^*), compared with the original one (from s to d). Here, a^+ equals zero if $a < 0$, or a if $a \geq 0$. If Inc is large, it will consume the user more time on the road, which clearly degrades service usability.

Note that endpoint or route deviation may not always reflect the real degradation of service usability. For example, for roads that are one-way, a small deviation may result in a rather different route. Also, even two route have roughly the same length, they may also be very different. We adopt the above two metrics as we are not aware of any better ones proposed in previous work to evaluate the usability of a shifted route.

Fig. 14 reports the above metrics achieved by ShiftRoute, for different values of Th and ϵ. We can observe that the endpoint

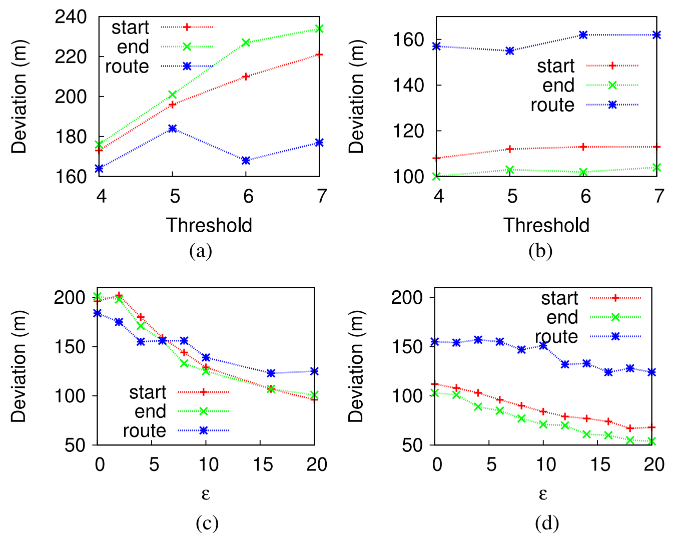


Fig. 14. The service usability achieved by `ShiftRoute` for different values of Th and ϵ. (a) Beijing, Usability vs. Th. (b) Xi'an, Usability vs. Th. (c) Beijing, Usability vs. ϵ. (d) Xi'an, Usability vs. ϵ.

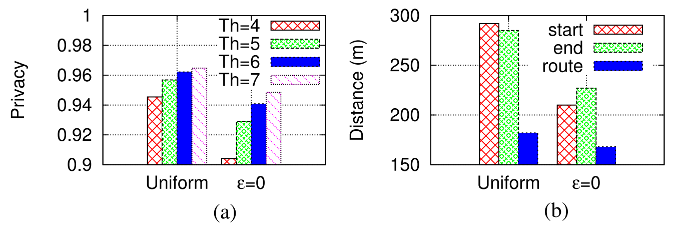


Fig. 15. Comparison of uniform selection and DLP selection, in terms of location privacy and service usability. (a) Privacy. (b) Accuracy ($Th = 5$).

deviation improves with the decrease of Th and the increase of ϵ, while there is no strong relationship between route deviation and these two parameters. The reason is that both Th and ϵ control how far away the endpoints can be shifted, and thus directly affect the endpoint deviation. However, even the shifted endpoints are far from the original ones, the shifted route may be shorter than the original ones, which has already be seen in Fig. 7.

E. Uniform or DLP?

This experiment will quantitatively compare two endpoint selection strategies, i.e., uniform and DLP. We consider one city and set the parameter ϵ for DLP to 0. Fig. 15 reports the level location privacy and service usability when using these two strategies. Form Fig. 15(a), we can observe that these two strategies achieve similar location privacy: above 0.9 for $Th \in [4, 7]$. On the other hand, DLP selection achieves a much smaller endpoint deviation than uniform selection, as shown in Fig. 15(b).

F. Computation Cost

In this experiment, we evaluate the computation cost of the AS (Anonymization Server). We run the AS (implemented with Python) on a Windows desktop with 3.2 GH Intel I5 processor and 8 GB DDR3 RAM.

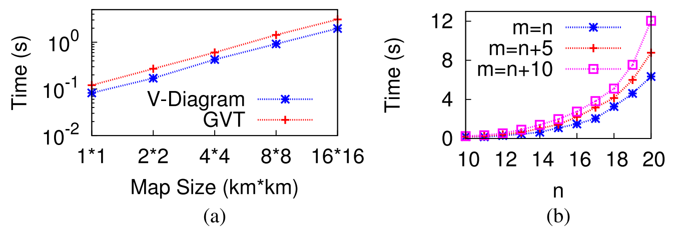


Fig. 16. Computation cost of the GTV constructor and DLP solver in `ShiftRoute`. (a) The GVT Constructor. (b) The DLP Solver.

Fig. 16(a) reports the time consumed by the GVT constructor for the city of Beijing. Two lines correspond to the time of generating the V-diagram and constructing the GVT, respectively. We can see that these two lines grow almost at the same rate. This is because the complexity of these two processes are roughly the same: Fortune's sweep line algorithm [26] for generating V-diagram has a complexity of $O(n \log n)$, and Algorithm III-A for constructing GVT has a complexity of $O(n \log m)$, where n is the number of POIs the m is the number of grids.

Fig. 16(b) reports the computation time cost by the DLP solver for different values of n and m, where n and m are the number of POIs and V-cells in a grid, respectively. We can see that when there are 20 POIs and 30 V-cells, the computation time is around 12 seconds, which is clearly infeasible if the AS performs the computation online. For this reason, we pre-compute the selection probabilities for all grids and typical values of $\epsilon \in \{0, 1, 2, \ldots, 20\}$, so that the AS can answer user queries with simple table lookup.

VII. Related Work

There are many Location Privacy Protection Mechanisms (LPPMs), and we roughly classify them into four categories.

Cloaking: The cloaking-based approaches aim to provide *location k-anonymity* [6], a variant of classic k-anonymity [21]. It requires that any query sent by a user cannot be distinguishable from another $k - 1$ users.

Casper [7] and CliqueCloak [8] allow users to personalize their requirements on location precision and query delay. PrivacyGrid [9] introduced location l-diversity as a complement of location k-anonymity. A problem with cloaking is that users must trust a centralized anonymizer. To address this issue, some P2P cloaking approaches [12]–[15] are proposed.

All cloaking approaches are designed for general LBSes like finding nearest POIs, while ill-suited for map services since commercial LBSes only accept locations rather than "cloaked regions".

Obfuscation: Ardagna *et al.* [10] proposed some obfuscation operators to transform locations into circular areas, so that LBS servers cannot identify a user's exact location. The obfuscation operators include `enlarge` and `reduce` the radius of the circle, and `shift` the center of the circle. In contrast to the above obfuscation method for the Cartesian plane, Duckham *et al.* [11] studied the obfuscation of road networks, where locations are modeled as vertices, and proximity of location are modeled as edges. Just as the cloaking approaches, these obfuscation-based approached also assumed that LBS servers could process the input of areas or regions, which is not the case for real life.

Dummy: With dummy-based approaches, users send multiple dummy queries in order to cover the real queries [16]. In SybilQuery [17], when a user is sending a query, another $k-1$ dummy queries are sent as well. To generate dummy queries, the user first specifies her start and destination before trip, and then consults a database recording the regional traffic histories. Dummy-Location Selection (DLS [18]) uses entropy as the privacy metric, and tries to spread the dummy locations as far as possible while maintaining the entropy. Similarly, DLS also requires mobile clients to store query statistics, in order to generate reasonable dummies. Lee *et al.* [31] proposed a dummy-based approach for route query services. When a user queries a route from s to t, she sends two sets S and T satisfying $s \in S$ and $t \in T$ to the map server, which returns the route for every start-destination pair. To reduce overhead, the authors designed an algorithm that could calculate shortest paths between two sets of locations. However, this required modification at the server side, and thus could not function with existing map services.

Space Transformation: In space transformation approaches, users transform the original space into another encoded one, and construct queries in the encoded space. Then, the LBS server evaluates user queries in the encoded space, and the user decodes the results to obtain the corresponding locations in the original space. For example, Khoshgozaran and Shahabi [19] used Hilbert space filling as the one-way transform function. Ghinita *et al.* [20] applied the technique of *Private Information Retrieval (PIR)* to enable users privately retrieve the k-nearest neighbors (kNN). Though pace transformation approaches are powerful tools for location query anonymization, they are only limited in functionalities (i.e., kNN search), and cannot be used for map services.

False Locations: Approaches like SpaceTwist [32] and Cover Location [33] let users send fake locations instead of real ones to the LBS server, and construct the right answers based on the results returned by the server. Similarly, Cover Location [33] also enables a user to retrieve nearby POIs by sending multiple fake queries. The advantage of the above two approaches is that they can be easily integrated with existing LBSes. However, to construct the true results, users often need to continuously interact with or send multiple queries to location servers, thus resulting in a large latency. In addition, we cannot see how they can be applied to provide privacy for map services.

VIII. Discussion

A. Usability of Shifted Routes

Similar to false location based LPPMs [32], [33] discussed in Section VII, `ShiftRoute` uses false endpoints/POIs for route query. Then, a natural question is "Is it possible that the user does not know the shifted endpoints/POIs, and then cannot use the shifted route?" First, the original and shifted endpoints are guaranteed to be close due to our endpoint selection algorithm (see Section III-B). The tradeoff between usability and privacy can be adjusted by the user through parameters Th and ϵ defined in Section III. In addition, as shown in Fig. 7), our mobile application will display both the shifted and original endpoints, as well as the current location of the user on the map. Thus, even if the selected endpoints are not familiar to the user, she can still find the route between the shifted and original endpoints easily.

B. Numerical Comparisons With Other LPPMs

In Section VI, we have not provided any numerical comparisons between our method and other LPPMs. The reason is as follows. First, it is not easy (if feasible) to find a common base to compare `ShiftRoute` which is designed for map services (route query) on smartphones with previous LPPMs that are designed for general-purposed applications or k-nearest neighbors (kNN) search. Second, even for the LPPM that is designed for map services (i.e., [31]), it uses custom algorithms (not released till now) to replace the original map services, which directly prevents its integration with production map services like Google Maps. Since we evaluate `ShiftRoute` using the public API provided by a production LBS provider, we think implementing a simple LBS with custom algorithms for comparison may not lead to valuable results. Thus, we choose only to compare `ShiftRoute` with other LPPMs qualitatively with Table I.

IX. Conclusion

This paper introduced `ShiftRoute`, a new LPPM for map services on smartphones. `ShiftRoute` enables mobile clients to query map services for routes, without exposing sensitive location information. The key idea is to strategically shifting the endpoints of route queries to nearby POIs. In realizing the idea, we addressed the challenge of selecting shifted endpoints with a novel two-stage method, i.e., private POI retrieval and private endpoint selection, which guarantees the privacy property of geo-indistinguishability. We implemented an application of `ShiftRoute` on Android, and conducted extensive experiments with real user traces. The results show that `ShiftRoute` can effectively preserve location privacy for map service users, while maintaining service usability at the same time. Our future work includes applying the current approach to other functions of map services, e.g., localization, navigation, POI search, etc.

References

[1] "Google has 7 products with 1 billion users," 2016. [Online]. Available: http://www.popsci.com/google-has-7-products-with-1-billion-users

[2] A. R. Beresford and F. Stajano, "Location privacy in pervasive computing," *IEEE Pervasive Comput.*, vol. 2, no. 1, pp. 46–55, Jan.–Mar. 2003.

[3] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. IEEE INFOCOM*, 2000, pp. 775–784.

[4] S. B. Wicker, "The loss of location privacy in the cellular age," *Commun. ACM*, vol. 55, no. 8, pp. 60–68, 2012.

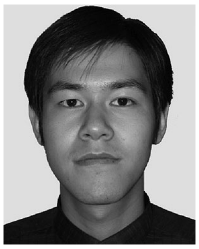
[5] M. Wernke, P. Skvortsov, F. Dürr, and K. Rothermel, "A classification of location privacy attacks and approaches," *Pers. Ubiquitous Comput.*, vol. 18, no. 1, pp. 163–175, 2014.

[6] M. Gruteser and D. Grunwald, "Anonymous usage of location-based services through spatial and temporal cloaking," in *Proc. 1st Int. Conf. Mobile Syst., Appl. Services*, 2003, pp. 31–42.

[7] M. F. Mokbel, C.-Y. Chow, and W. G. Aref, "The new casper: Query processing for location services without compromising privacy," in *Proc. 32nd Int. Conf. Very Large Data Bases*, 2006, pp. 763–774.

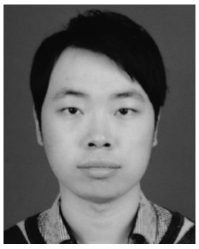
[8] B. Gedik and L. Liu, "Protecting location privacy with personalized k-anonymity: Architecture and algorithms," *IEEE Trans. Mobile Comput.*, vol. 7, no. 1, pp. 1–18, Jan. 2008.
[9] B. Bamba, L. Liu, P. Pesti, and T. Wang, "Supporting anonymous location queries in mobile environments with privacygrid," in *Proc. 17th Int. Conf. World Wide Web*, 2008, pp. 237–246.
[10] C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. Di Vimercati, and P. Samarati, "Location privacy protection through obfuscation-based techniques," in *Proc. 21st Annu. IFIP WG 11.3 Working Conf. Data Appl. Security*, 2007, pp. 47–60.
[11] M. Duckham and L. Kulik, "A formal model of obfuscation and negotiation for location privacy," in *Proc. 3rd Int. Conf. Pervasive Comput.*, 2005, pp. 152–170.
[12] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "Mobihide: A mobilea peer-to-peer system for anonymous location-based queries," in *Proc. 10th Int. Conf. Adv. Spatial Temporal Databases*, 2007, pp. 221–238.
[13] G. Ghinita, P. Kalnis, and S. Skiadopoulos, "Prive: Anonymous location-based queries in distributed mobile systems," in *Proc. 16th Int. Conf. World Wide Web*, 2007, pp. 371–380.
[14] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in *Proc. 14th Int. Symp. Adv. Geograph. Inf. Syst.*, 2006, pp. 171–178.
[15] C.-Y. Chow, M. F. Mokbel, and X. Liu, "Spatial cloaking for anonymous location-based services in mobile peer-to-peer environments," *GeoInformatica*, vol. 15, no. 2, pp. 351–380, 2011.
[16] H. Kido, Y. Yanagisawa, and T. Satoh, "An anonymous communication technique using dummies for location-based services," in *Proc. Int. Conf. Pervasive Services*, 2005, pp. 88–97.
[17] P. Shankar, V. Ganapathy, and L. Iftode, "Privately querying location-based services with sybilquery," in *Proc. 11th Int. Conf. Ubiquitous Comput.*, 2009, pp. 31–40.
[18] B. Niu, Q. Li, X. Zhu, G. Cao, and H. Li, "Achieving k-anonymity in privacy-aware location-based services," in *Proc. IEEE INFOCOM*, 2014, pp. 754–762.
[19] A. Khoshgozaran and C. Shahabi, "Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy," in *Proc. 10th Int. Conf. Adv. Spatial Temporal Databases*, 2007, pp. 239–257.
[20] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan, "Private queries in location based services: Anonymizers are not necessary," in *Proc. Int. Conf. Manage. Data*, 2008, pp. 121–132.
[21] L. Sweeney, "k-anonymity: A model for protecting privacy," *Int. J. Uncertainty, Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, 2002.
[22] M. E. Andrés, N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Geo-indistinguishability: Differential privacy for location-based systems," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2013, pp. 901–914.
[23] N. E. Bordenabe, K. Chatzikokolakis, and C. Palamidessi, "Optimal geo-indistinguishable mechanisms for location privacy," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2014, pp. 251–262.
[24] M. De Berg, M. Van Kreveld, M. Overmars, and O. C. Schwarzkopf, *Computational Geometry*. New York, NY, USA: Springer-Verlag, 2000.
[25] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1984, vol. 14, pp. 47–57.
[26] S. Fortune, "A sweepline algorithm for voronoi diagrams," *Algorithmica*, vol. 2, no. 1–4, pp. 153–174, 1987.
[27] "The python version of Voronoi diagram calculator," 2010. [Online]. Available: https://svn.osgeo.org/qgis/trunk/qgis/python/plugins/fTools/tools/voronoi.py
[28] "Amap," 2017. [Online]. Available: http://ditu.amap.com/
[29] R. Shokri, G. Theodorakopoulos, C. Troncoso, J.-P. Hubaux, and J.-Y. Le Boudec, "Protecting location privacy: Optimal strategy against localization attacks," in *Proc. ACM Conf. Comput. Commun. Security*, 2012, pp. 617–627.
[30] L. A. Adamic and B. A. Huberman, "Zipfs law and the internet," *Glottometrics*, vol. 3, no. 1, pp. 143–150, 2002.
[31] K. C. Lee, W.-C. Lee, H. V. Leong, and B. Zheng, "Navigational path privacy protection: Navigational path privacy protection," in *Proc. ACM Conf. Inform. Knowl. Manage.*, 2009, pp. 691–700.
[32] M. L. Yiu, C. S. Jensen, X. Huang, and H. Lu, "Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services," in *Proc. IEEE 24th Int. Conf.* Data Eng., 2008, pp. 366–375.
[33] S. T. Peddinti, A. Dsouza, and N. Saxena, "Cover locations: Availing location-based services without revealing the location," in *Proc. 10th Annu. ACM Workshop Privacy Electron. Soc.*, 2011, pp. 143–152.

Peng Zhang received the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2013. He is currently an Associate Professor with the Department of Computer Science and Technology, Xi'an Jiaotong University, China. He has been a visiting student with The Chinese University of Hong Kong and Yale University. His research interests include network security and software-defined networks.

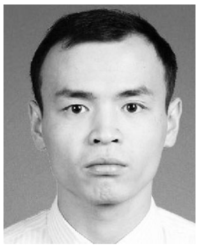


Chengchen Hu received the B.S. degree from the Department of Automation, North-Western Polytechnical University, Xi'an, China, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2003 and 2008, respectively. He worked as an Assistant Research Professor with Tsinghua University from July, 2008 to December, 2010. After that, he joined the Department of Computer Science and Technology, Xi'an Jiaotong University, where he is currently a Full Professor. His main research interests include computer networking systems and network measurement and monitoring.

Di Chen received the B.S. degree and the Master's degree both in computer science from Xi'an Jiaotong University, Xi'an, China, in 2013, and 2015, respectively. She is currently an Assistant Engineer with Luoyang Electronic Equipment Test Center of China. Her main research interests include location privacy protection and communication countermeasure.



Hao Li received the B.S. degree and the Ph.D. degree both in computer science from Xi'an Jiaotong University, Xi'an, China, in 2010 and 2016, respectively. He is currently an Assistant Professor with the Department of Computer Science and Technology, Xi'an Jiaotong University. His main research interests include network measurement, big network data, and software-defined networking.



Qi Li received the Ph.D. degree from Tsinghua University, Beijing, China. He is currently an Associate Professor with the Graduate School at Shenzhen, Tsinghua University. He is working on different projects in security and networking, and has worked with ETH Zurich, University of Texas at San Antonio, The Chinese University of Hong Kong, Chinese Academy of Sciences, and Intel. His research interests include various topics including security, Internet, and large scale distributed systems.