# Preventing Restricted Space Inference in Online Route Planning Services

Florian Dorfmeister
LMU Munich
Oettingenstr. 67
Munich, Germany
florian.dorfmeister@ifi.lmu.de

Kevin Wiesner
LMU Munich
Oettingenstr. 67
Munich, Germany
kevin.wiesner@ifi.lmu.de

Michael Schuster
LMU Munich
Oettingenstr. 67
Munich, Germany
schusterm@cip.ifi.lmu.de

Marco Maier
LMU Munich
Oettingenstr. 67
Munich, Germany
marco.maier@ifi.lmu.de

## ABSTRACT

Online route planning services compute routes from any given location to a desired destination address. Unlike offline implementations, they do so in a traffic-aware fashion by taking into consideration up-to-date map data and real-time traffic information. In return, users have to provide precise location information about a route's endpoints to a not necessarily trusted service provider. As suchlike leakage of personal information threatens a user's privacy and anonymity, this paper presents PrOSPR, a comprehensive approach for using current online route planning services in a privacy-preserving way, and introduces the concept of $k$-immune route requests to avert inference attacks based on restricted space information. Using a map-based approach for creating cloaked regions for the start and destination addresses, our solution queries the online service for routes between subsets of points from these regions. This, however, might result in the returned path deviating from the optimal route. By means of empirical evaluation on a real road network, we demonstrate the feasibility of our approach regarding quality of service and communication overhead.

## Categories and Subject Descriptors

K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*

## General Terms

Algorithms, Experimentation, Security

## Keywords

Restricted Space Inference, Route Planning, Privacy, LBS

## 1. INTRODUCTION

The proliferation of mobile devices with built-in GPS receivers has recently led to a widespread adoption of so-called location-based services (LBS). These kind of services allow for a better user experience by adapting their behavior to a user's whereabouts, such as ordering query results by distance to her location. Some of the most popular applications of LBS today are online route planning services (RPS), such as Google Maps or OpenRouteService, which have superseded printed maps in everyday usage almost completely. Using such a service allows for retrieving the shortest or fastest route from a given start to a destination address on a road network. In contrast to offline implementations, which used to constitute the first generation of route planning applications, online RPS benefit from the availability of always up-to-date map data and real-time traffic information, e.g, about blocked roads, driving conditions, congestions or road works and can hence provide a better service to its users. Facilitated by the ubiquity and connectivity of smartphones, online RPS can also be used spontaneously by mobile users, i.e., the source of a requested route coincides with the user's current location provided by the sensors of her device.

Just as with any other LBS, however, receiving this kind of information enables a curious service provider and any eavesdroppers to track users and can hence be considered a threat to the latter's location privacy and anonymity: By using inference attacks, e.g., based on spatio-temporal clustering techniques, a service provider is likely to be able to find out about a user's important places, i.e., restricted spaces such as her home and workplace [9]. In the case of online route planning, the RPS provider typically is not only supplied with noisy location measurements, but exact addresses typed in by the user, rendering the realization of a reliable reverse lookup of an address in a white-pages database a trivial task. Having acquired such information, the RPS might succeed in identifying, i.e., de-anonymizing its users [4]. By recording and analyzing all of a user's route queries, the RPS provider can easily gain additional private information about its users. Being able to track a user's movements on an address level, one might not only be able to pinpoint

MOBIQUITOUS 2015, July 22-24, Coimbra, Portugal
Copyright © 2015 ICST
DOI 10.4108/eai.22-7-2015.2260037

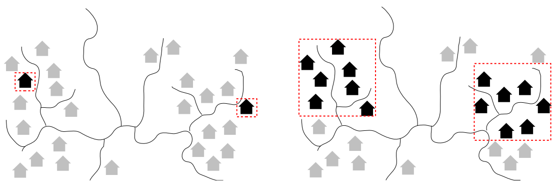

Figure 1: The different levels of information leakage using standard online route retrieval and our PrOSPR approach.

a user's current location, but also find out about her health status, personal and political interests, etc. as well as a company's clients, business partners or a doctor's patients [20].

Motivated by these observations, we aim at providing a privacy-preserving solution for the retrieval of nearly optimal routes from existing online RPS. In order to protect a user's location privacy, the RPS provider should not learn the actual source and destination of a query, as these might be restricted spaces that leak sensitive information about a user. As exemplified in Figure 1, we propose an approach relying on map-based location obfuscation [19] that lets the RPS provider at most learn two cloaked regions, which are guaranteed to contain at least $k$ restricted spaces. With current online RPS not supporting such region-to-region route queries, our algorithm lets the client issue route requests for several pairs of points from the cloaked regions' road networks to the RPS. As this procedure is not guaranteed to produce complete routes from the user's actual start to destination address, any missing subpaths are computed locally.

This paper hence presents a comprehensive approach for performing route queries using existing online RPS in a privacy-preserving way, making the following contributions: Firstly, we propose a map-based solution for the creation of meaningfully cloaked regions around a user's actual source and destination addresses, comprising all necessary steps such as geocoding and acquisition of map data. Secondly, we base our obfuscation approach on the means of $k$-immune route requests, which guarantees an adjustable level of ambiguity and works without the need of a trusted third party (TTP). Thirdly, we propose and evaluate different heuristics for the selection of nodes within the cloaked regions' road networks used as inputs for the point-to-point route queries to the RPS in order to find a suitable balance between the optimality of routes and minimization of communication overhead.

The remainder of this paper is structured as follows: Section 2 discusses related work in the field of location privacy. In Section 3, we define the problem statement and attacker model for this work. Our approach for the realization of *Private Online Shortest Path Retrieval* (PrOSPR) will be described in Section 4. In Section 5, we present the results of the empirical evaluation. In Section 6, we conclude.

## 2. RELATED WORK

Our work pertains to the fields of location privacy and privacy in LBS. Typically, *identity*, *position* and *time* are the primary information items [18]. Depending on the scenario and the user's requirements, different elements of this tuple are considered sensitive and thus have to be protected. An approach to hide a user's position is to present the service with *position dummies* [8] that detract the LBS provider from the user's actual position. However, the challenge is to create position dummies which cannot easily be recognized as fake positions using other sources of information such as map semantics or a user's location history [15]. One can also resort to *spatial obfuscation* techniques which intentionally reduce the accuracy of the user's position, e.g., by sending a circular area instead of a precise location to the LBS [1].

To prevent inference about individuals in a set of users, the concepts of $k$-anonymity [16] and $l$-diversity [12] have been adapted for location privacy: $k$-anonymity is tried to be achieved by *cloaking* a user's location, i.e., only reporting the user's position as a coarse-grained area containing at least $k - 1$ other users [3, 5]. However, $k$-anonymity fails to preserve a user's privacy with regard to location semantics in case all $k$ users are within a semantically similar area, e.g., all of them are in a hospital. This can be solved by requiring that the positions of the users within a cloaking area are different to such an extent that the location semantics are $l$-diverse, i.e., there are at least $l$ semantically different locations which are occupied by users in the cloaking area [2]. Gruteser at al. propose a single-user approach using a building-based $k$-area algorithm [7], which prevents position updates within a predefined zone from being sent to an LBS in case a user has entered one of $k$ sensitive areas in that zone. All of these approaches are yet not directly applicable to the problem of online route planning.

Specifically dedicated to this exact domain, Mouratidis introduces privacy-preserving shortest path queries based on private information retrieval (PIR) [14]. By using a tamper resistant, fully trusted secure co-processor (SCP) installed at the LBS and a thoroughly designed PIR protocol, the author provides two different PIR schemes for retrieving complete route information from an RPS in a privacy-preserving manner. This is the only approach promising full location privacy, i.e., the LBS will not even learn obfuscated information about the requested route. In return, however, this approach requires full cooperation of the RPS provider, i.e., having a SCP module installed. Also, the desired start and destination locations must be given as Euclidean coordinates, and this data cannot be expected to be generally available.

Lee et al. present OPAQUE [10], which relies on a TTP acting as a *Path Obfuscation Server*. The TTP receives route queries from clients, generates a user-defined number of position dummies, and sends an obfuscated path query to an RPS. To handle such queries, the LBS has to cooperate by providing a special module which has also been devised by the authors using a modified A* algorithm. A drawback of the system is the naïve randomized selection of the position dummies, which are points located in the vicinity of the actual start and destination locations [11]. Since the actual route is always contained in the obfuscated query, OPAQUE still delivers the optimal result. Also, no overhead is introduced at the client as it only sends one request to the TTP. The RPS yet inevitably gets to know the actual route, which might be identifiable from the dummy ones by means of frequency analysis or map knowledge. Moreover, the TTP is a single-point-of-attack, which we try to omit in our approach.

Vicente et al. describe an approach to be used with existing online RPS [17]. To enable privacy-preserving route queries, suitable map extracts of the start and destination regions are determined by a trusted location obfuscator and can be downloaded from a map provider. For each node in the start region then routes to each node in the destination region are requested from the RPS. In order to reduce the number of requests, the authors propose various heuristics which deliver good, albeit not always optimal results while drastically reducing the number of queries. The authors employ a grid-based partitioning of the map and analyze both to which extent the system is able to find the optimal route and the amount of communication overhead. They find that their system retrieves the optimal route more than 80% of the time, and only 3% of the routes deviate more than 500m from the optimal one. Using their default settings, roughly 50 calls to the RPS are generated for a single route query. The authors show that these settings result in only 3% fewer correct routes than when using the most precise settings, while needing only half of the otherwise required requests.

Only the last approach aims at off-the-shelf functionality without explicit cooperation of an unmodified online RPS. However, it produces considerable communication overhead and requires a TTP for location obfuscation. None of these approaches takes into consideration map information when calculating dummy positions or cloaked regions. Xue et al. introduce the concept of *location*-diversity [20], which deploys the concept of semantic locations for obfuscating the source location of a route request. However, their approach also depends on the existence of a trusted anonymizer and misses to protect the destination of a route request.

## 3. PROBLEM STATEMENT, ATTACKER MODEL AND DESIGN GOALS

We aim at preventing online RPS providers from being able to track its pseudonymously known users on an address level, which is problematic with regard to privacy as it allows for de-anonymization and inference about a user's personal life. Inference attacks are primarily based on *restricted space* information [6], which an attacker can misuse to gain knowledge about a user's lifestyle and interests by matching reported locations to places that are either inhabited by a limited number of people, such as family homes or sites with clear semantics, e.g., an office or medical practice [20].

Definition 1. *A restricted space is any spatial area that exclusively belongs to one subject or can unambiguously be linked to a clear semantical meaning.*

We hence base our work on an understanding of location privacy similar to the $k$-area approach [7], which tries to hide from a continuous LBS provider which of $k$ distinct sensitive areas a user has visited. W.l.o.g., in the following we consider every building having its own address to be a restricted space, even though one address might actually house several distinct restricted spaces at once and vice versa.

We seek to avoid the possibility of restricted space inference based on online shortest path queries by hiding a request's true source and destination location in a confusion set. The latter is created using an obfuscation approach derived from the concept of $k$-anonymity [16], which is commonly used for protecting mobile users' location and communication privacy in LBS. In contrast to the majority of existing works, which typically deploy a TTP for calculating spatio-temporarily-cloaked areas around $k$ of its users' actual positions [6][13], we aim at obfuscating the endpoints of a route request so that at least $k$ restricted spaces are equally likely to be the user's true source and destination address, respectively. We hence consider a route request to comply with a user's privacy requirements, if it is $k$-immune.

Definition 2. *An online shortest path query is $k$-immune against location inference, if both the request's true source and destination locations are indistinguishable from at least $k-1$ other restricted spaces.*

We believe this to be a viable approach for the field of online route planning, as requests can be issued individually for each user and do not depend on the availability of and synchronization with peers in the user's vicinity.

### 3.1 Attacker Model

Just as in [14] and [20], we expect the online RPS and map provider to be a semi-honest adversary curious about its users' current, visited and searched locations, but not behaving maliciously, i.e., working correctly and not providing forged results in order to gain knowlegde. In contrast to [20], however, we expect the RPS provider to be able to link all of a user's route requests, i.e., we expect pseudonymous usage of the routing service. By simply logging a user's route queries, the RPS provider can hence create lists of visited locations for each user and use these information for further analysis. Moreover, apart from full map knowledge, we expect the adversary to have some kind of white-pages database available, allowing him to perform reverse address lookups. On the other hand, the adversary does not have any additional background knowledge about its users, such as personal preferences. Additionally, with the RPS provider being a passive adversary only, he does not perform any active attacks such as observation identification [6].

### 3.2 Further Assumptions & Design Goals

We expect a user to request a route $P_{[S,D]}$ on a road network $G$ from a source $S$ to a destination $D$ from a standard online RPS. The user is either on a mobile phone with a built-in GPS receiver that provides her current position in terms of longitude and latitude or copies her desired source and destination addresses manually into the RPS' query interface, with a full address consisting of street, house number, postal code and city. However, we do not expect the client initially to be in possession of any offline map information.

Our approach shall re-use existing RPS functionality as-is, hence we do not expect any form of cooperation from the RPS provider apart from responding to route queries, which rules out any changes on the LBS side. Furthermore, our approach should not rely on a TTP or any other trusted component apart from the user's device. Given the computational power of today's mobile devices, a part of the whole workload can yet be offloaded to the client. Apart from that,

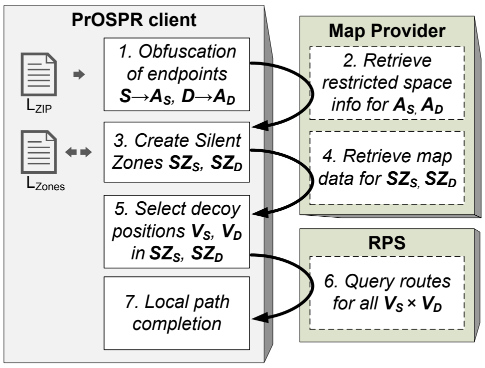
PrOSPR client
$L_{ZIP}$
$L_{Zones}$
1. Obfuscation of endpoints $S \rightarrow A_S$, $D \rightarrow A_D$
3. Create Silent Zones $SZ_S$, $SZ_D$
5. Select decoy positions $V_S$, $V_D$ in $SZ_S$, $SZ_D$
7. Local path completion
Map Provider
2. Retrieve restricted space info for $A_S$, $A_D$
4. Retrieve map data for $SZ_S$, $SZ_D$
RPS
6. Query routes for all $V_S \times V_D$

Figure 2: The basic steps of the PrOSPR algorithm.

we aim at providing an intuitive means for setting an individual level of privacy for each user. Finally, we are seeking to balance the trade-off between performance, i.e., the communication overhead produced by our approach and quality of service, i.e., being able to quickly answer each of a user's queries with a route that does not deviate much from the optimal one she would normally have retrieved.

## 4. PrOSPR SYSTEM OVERVIEW

Our approach for *Private Online Shortest Path Retrieval* (PrOSPR) aims at avoiding the possibility of inference attacks based on restricted space information on behalf of an online RPS provider by means of map-aware location obfuscation. As there is no TTP involved in our system, a user's exact location information will at no time leave her device, which is thus the only component deemed trusted.

For privacy-preserving route retrieval from an online RPS, our PrOSPR system takes the steps depicted in Figure 2: (1) Upon initiation of a route request, coarse location obfuscation is applied to the endpoints in order to create corresponding obfuscated areas. (2) Information about restricted spaces, i.e., addresses within these coarse areas are fetched from a map provider. (3) The client then checks if any of the two endpoints falls into a zone already stored in a local database $L_{\text{Zones}}$. If not, the algorithm refines the cloaked regions to enclose the request's endpoints less coarsely, yet still matching the user's privacy requirements. (4) Now the road networks for the refined zones are downloaded from the map provider. (5) According to different heuristics, variable subsets of nodes within these networks are selected, (6) which serve as inputs for the route queries to the RPS. (7) Finally, the returned results are locally completed by the client and the most appropriate one is presented to the user. In the following, these different steps will be described in detail.

### 4.1 Initial Obfuscation of Route Endpoints

As no precise information about the actual endpoints of a route request, $S$ and $D$, should leave the user's device, coarse location obfuscation has to be applied locally for obtaining two obfuscated areas $A_i$, $i \in \{S, D\}$. In order to not reveal a user's current or future locations, each obfuscated area has to contain at least $k$ restricted spaces. Given that the source or destination location of a request is provided as an address, such as *Oettingenstrasse 67, 80538 Munich*, some kind of client-side geocoding is needed in order to be able to specify a corresponding map section. Such initial location obfuscation can be achieved by relying on the hierarchical postal code system, which allows for a trivial mapping of an address to a coarse surrounding region. Using postal codes for location obfuscation naturally sets the upper limit for the maximum value of $k$. However, each postal code refers to several hundreds to thousands of addresses, which we believe to be an acceptable level of privacy. To enable corresponding region queries to a map provider, the PrOSPR client locally stores a precomputed list $L_{\text{ZIP}}$ containing the convex hulls of the geographic boundaries of all postal code areas. In case the user's location is given in the form of GPS coordinates, spatial obfuscation techniques such as described by Ardagna et al. [1] could be applied. However, similar to using a grid-based approach for location obfuscation [17], simple spatial obfuscation is not guaranteed to prevent the possibility of restricted space inference as it does not take into consideration map information. In both cases, it might happen that either an obfuscated area or a grid cell contains less than the required number of $k$ buildings and thereby violates a user's privacy expectations. PrOSPR therefore again relies on $L_{\text{ZIP}}$ for locally looking up the postal code area $A_i$ containing the user's current position, as this approach intrinsically warrants the existence of an uncritically high number of restricted spaces within the cloaked region. Either way, the client now is in possession of the geographical boundaries of two coarsely obfuscated areas, $A_S$, $A_D$, which both excel the user's privacy requirements.

### 4.2 Retrieval of Restricted Space Information

In line with [18], we argue that sensible location obfuscation requires consideration of relevant map information. In the context of restricted space inference, it is hence important that the obfuscated region contains at least $k$ such spaces, i.e., buildings the user is equally likely to be in or heading to. To be able to create such meaningfully cloaked areas, once the coarsely obfuscated map sections for the source and destination addresses have been found, our algorithm queries an online map provider for restricted space information $B_{A_i}$, i.e., the GPS coordinates of all buildings within an obfuscated area $A_i$ and their addresses in terms of street name and house number to enable both geocoding and mapping addresses to nodes on the road network. With $A_i$ being chosen to contain at least $k$ restricted spaces each, this does not interfere with a user's requirement for location privacy. As soon as these information have been acquired, PrOSPR continues with a refinement step of the obfuscated areas.

### 4.3 Silent Zone-based Region Refinement

In order to minimize the overall communication overhead, the coarsely obfuscated areas are refined to match the user's privacy requirements more closely before issuing requests to the RPS. Reducing the size of the cloaked areas is also beneficial for having the online RPS compute as much of the requested route in a traffic-aware fashion as possible. PrOSPR deploys a mechanism for map-aware location obfuscation called *Silent Zones* (SZ) presented by Wiesner et al. [19]. A SZ is defined as a privacy zone around a user's important places relying on the concept of building-based $k$-*anonymity* [16], i.e., a SZ contains at least $k$ buildings. Relative position and extent of a SZ are selected to not provide any hints towards which of the $k$ buildings the SZ was created for. To balance the trade-off between location privacy and quality of service, the SZ approach constructs a nearly minimal bounding box around a restricted space which still matches a user's privacy requirements in terms of $k$.

The PrOSPR client first checks whether $S$ or $D$ are spatially contained in a SZ already stored in $L_{\text{Zones}}$. If so,the corresponding zone will be re-used in order to avoid the possibility of the RPS provider being able to narrow down a user's important places over time, e.g., by means of frequency analysis of several zones covering her home address. Otherwise, a new SZ for the given endpoint will be created. As a deterministic creation of the smallest possible zone might be reversible, the SZ approach seeks to find zones that are only nearly optimal in terms of the zones' size, but in return can be created more randomly. In order to create such meaningfully cloaked areas around a route's source and destination, a variant of the *RandomRect* (RR) algorithm proposed in [19] is deployed. In case $S$ or $D$ are given as addresses, restricted space information contained in $B_{A_i}$ is used to retrieve the geographic coordinates required for SZ creation. RR iteratively creates random square candidate zones $\text{SZ}'_i$, $i \in \{S, D\}$, containing the source (destination) as follows:

$$\begin{aligned}\text{SZ}'_i(l_i, a_j) = (&l_i.x - \alpha * a_j, l_i.y - \beta * a_j,\\ &l_i.x + (1-\alpha) * a_j, l_i.y + (1-\beta) * a_j) \quad (1)\end{aligned}$$

with $l_i.x$, $l_i.y$ being the location's longitude and latitude, and $a_j$ defining the side length of the cloaked region in the $j$-th round. $\alpha$, $\beta \in [0;1[$ are randoms used for shifting the zone's center off the actual location $l_i$. The initial side length $a_0$ can be chosen freely. When a candidate $\text{SZ}'_i$ is found, the algorithm proceeds by counting the number of buildings $c_j$ within this zone. While $c_j$ is smaller than the user specified value of $k$, the algorithm iteratively doubles the size of the area covered by the zone. Once the required value of $k$ is reached by $c_j$, the algorithm returns the boundaries of the $j$-th $\text{SZ}'_i$. In order to cater for cases where $S$ or $D$ is located near the border of the created zone, we expand these boundaries by adding a constant value S_BORDER and return that extended zone as $\text{SZ}_i$. The extra area does yet not contribute to the user's location privacy, as an attacker can simply remove this border. $\text{SZ}_S$ and $\text{SZ}_D$ are stored in $L_{\text{Zones}}$ for future queries for the same regions.

As soon as the exact SZs for both the source and destination regions of a request are known, PrOSPR queries the map provider for the zones' drivable road networks $G_i$, including all road segments that lead out of or into one of the zones. As both $\text{SZ}_S$ and $\text{SZ}_D$ are guaranteed to contain at least $k$ restricted spaces, this does not interfere with the user's privacy requirements. After downloading $G_S$ and $G_D$, $S$ and $D$ are mapped to the geographically closest nodes in the corresponding graph to enable later route completion.

## 4.4 Selection of Query Points

Our algorithm then heuristically selects a subset of nodes $V_i$ within each of the zones' road networks as inputs for the point-to-point route queries. This step is taken in order to further limit the number of queries that have to be sent to the RPS. Hence, neither will the actual endpoints of the request be communicated to the RPS provider [10][11], nor will routes from all nodes in $\text{SZ}_S$ to all those in $\text{SZ}_D$ be searched [17]. Instead, our approach is based on the concatenation of subroutes, with missing paths being computed locally by the client. In return, however, our algorithm is not guaranteed to find the optimal route and might instead produce differing results or even no result at all. The resulting quality of service will be evaluated in Section 5. For the selection of query point sets $V_i$, we propose the following heuristics:

- *Random node* (`R1`): Randomly choose one node on the respective zone's road network.
- *Center* (`C1`): Find the node on the road network closest to the geographic center of $\text{SZ}_i$.
- *Random nodes* (`RN`): Randomly choose $N$ nodes on the zone's road network, if present.
- *All Entry/Exit-Points* (`EEP`): Find all nodes that lead out of (into) the request's source (destination) SZ, i.e., nodes with a neighbor outside the respective zone.
- *Random Entry/Exit-Points* (`EEPN`): Randomly choose $N$ exit (entry) nodes, if present.
- *Random nodes, one reachable* (`RNr`): Randomly choose $N$ nodes and test if at least one of them is reachable from the request's source or has a route to the destination within the respective road network. Otherwise, replace one of the selected nodes by a new randomly chosen node until a reachable one is found.
- *Random Entry/Exit-Points, one reachable* (`EEPNr`): Randomly select $N$ exit (entry) nodes and test if at least one of them is reachable from $S$ (has a route to $D$) within $G_S$ ($G_D$). Else, replace one of the selected nodes by another randomly chosen entry (exit) node until a reachable one is found.

In order to not betray the exact location of the endpoints within $\text{SZ}_i$, the latter are not used in any of the first five approaches, i.e., each node in $G_i$ is equally likely to be selected. The *one reachable* heuristic increases the chance that our algorithm is able to obtain a correct result, as it prevents situations where no query point can be reached from an endpoint within the locally known road network. The `EEP` methods are based on the fact that all routes from source to destination must cross one of the SZ's entry or exit points. These strategies automatically adapt to small numbers of exit (entry) nodes and thus help to avoid waste requests.

## 4.5 Route Queries & Local Path Completion

For each pair of nodes in $V = V_S \times V_D$ one route query will be issued. With the user's real source and destination addresses not being communicated to the service provider, the result set is not guaranteed to contain a complete route from source to destination. As it might by chance, though, PrOSPR checks each result $P_{[A,B]}$, $A \in V_S$, $B \in V_D$ whether $P_{[S,D]}$ is a sub-route of the returned route. If so, our algorithm cuts the extra steps from $P_{[A,B]}$, as this necessarily is the shortest route from start to destination. In order to not impair the user's location privacy, however, PrOSPR proceeds until all pairs in $V$ have been processed. If the desired route is not contained in $P_{[A,B]}$, our algorithm tries to construct a complete path as follows: To avoid local path redundancies within $G_S$, PrOSPR searches the retrieved path

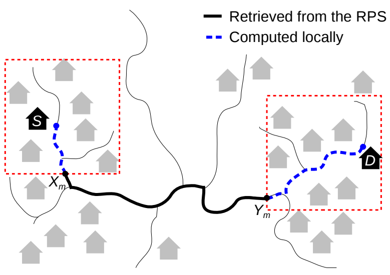

Figure 3: Local path completion within the SZs.

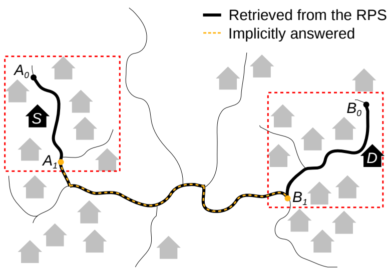

Figure 4: Redundant queries to the RPS can be avoided by discarding routes learned implicitly from previous requests.

for the first node $X_m \in P_{[A,B]}$ that is an exit node from $SZ_S$, i.e., $X_m \in G_S$. If no such node can be found, $X_m = B$. Using A* respecting one-way streets and turn bans, then the shortest path $P_{[S,X_m]}$ is computed. If, due to the former, $P_{[S,X_m]}$ must not be concatenated with $P_{[X_m,B]}$, the procedure is repeated with the predecessor $X_{m-1} \in P_{[A,B]}$ of $X_m$ until in the $n$-th round a valid route is found or no such node exists. In the former case, our algorithm then removes $P_{[A,X_{m-n}]}$ from $P_{[A,B]}$ and appends the resulting subpath to $P_{[S,X_{m-n}]}$, leading to $P_{[S,B]}$. The same procedure is then applied reversely to the last entry node $Y_m \in P_{[S,B]}$ to $SZ_D$ and its successor nodes from $P_{[S,B]}$ to find the missing path $P_{[Y_{m+n},D]}$. The final outcome of this procedure is depicted in Figure 3. When a correct route has been created or in case local route completion fails, e.g., because no valid route can be found, our algorithm continues with the next pair of nodes in $V$ until all pairs have been processed this way.

### 4.6 Optimizing Communication Cost

The number of requests that have to be issued to the RPS on average can be reduced without sacrificing privacy simply by avoiding any redundant queries, i.e., route requests that have implicitly been answered by previous requests. As shown in Figure 4, this is the case when a still scheduled query for a shortest path from $A_1$ to $B_1$ is the subroute of an earlier result $P_{[A_0,B_0]}$. Similar to [17], hence, upon receiving a result from the RPS we discard any pairs from $V$ that are found to be subpaths of the retrieved route.

In this section, we presented PrOSPR, a holistic and map-aware approach for the retrieval of nearly optimal paths from standard online RPS in a privacy-preserving way. Any external party involved in this process will at most learn the geographic boundaries of two meaningfully cloaked areas, which adapt to restricted space density and are hence guaranteed to contain at least $k$ distinct addresses each. Route requests using PrOSPR can thus be considered $k$-immune against location inference, as both the source and destination cannot be identified from at least $k$ restricted spaces. Not using a TTP, this requires additional data acquisition from a map provider and the RPS. Our approach hence also incorporates several steps aiming at limiting its communication overhead, which will be evaluated in the next section.

## 5. EVALUATION

We will now present the results of our empirical evaluation studying the feasibility of our approach in terms of quality of service, cost and privacy. Concerning quality of service, we evaluate to what extent the route results produced by PrOSPR deviate from the optimal results retrieved by using current route planning services as-is. Additionally, the number of requests that have to be sent to the RPS as well as the resulting overall communication overhead will be analyzed.

### 5.1 Experimental Setup

We base our experiments on map information for Upper Bavaria acquired from OpenStreetMap (OSM) as well as the YOURS routing server[1]. For the download of map material, we are making usage of the Overpass API[2]. $L_{ZIP}$ is created by extracting information about the convex hulls of all the postal code areas contained in the OSM material, too. For the region tested, this file used 394kB of disk space, or 620B per postal code area on average.

Map information available in OSM is not complete, i.e., buildings or addresses might be missing. In order to produce authentic results, we visually identified three differently populated regions with a high coverage of buildings, i.e., the cities of *Munich* (metropolis), *Rosenheim* (medium) and *Erding* (small). Our experiments are conducted using the settings from Table 1, performing 100 route requests for random addresses for each combination of parameters.

| Parameter | Values |
|---|---|
| $k$ | 25, 50, 100, 150, 200 |
| $a_0$ | 50m |
| S_BORDER | 20m |
| Query point heuristic | C1, R1, R5, R5r, EEP, EEP5, EEP5r |
| Routing scenario | withinMunich, withinRosenheim, withinErding, MunichToErding, MunichToRosenheim |

Table 1: Parameter Settings

### 5.2 Quality of Service

In a first step, we analyze the accuracy of routes found by our PrOSPR approach. We consider a route result to be *optimal*, only if its length deviates less than 1m from the original path's length, which can be attributed to rounding errors made when calculating distances between GPS coordinates.

Figure 5 shows the error distribution for the different selection heuristics, combined for all routing scenarios using $k = 100$, i.e., both the start and destination zones are guaranteed to contain at least 100 addresses each. The different heuristics are able to produce the optimal route with varying

[1]http://wiki.openstreetmap.org/wiki/YOURS
[2]http://wiki.openstreetmap.org/wiki/Overpass_API

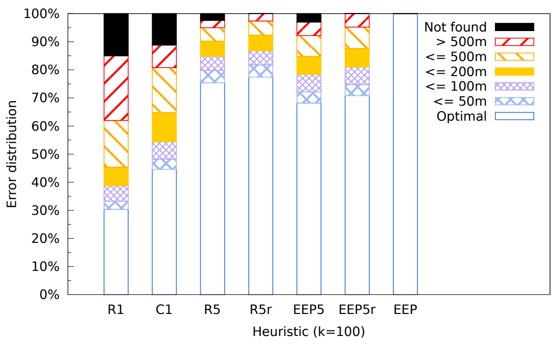

Figure 5: Error distribution of retrieved routes averaged over all scenarios using $k = 100$.

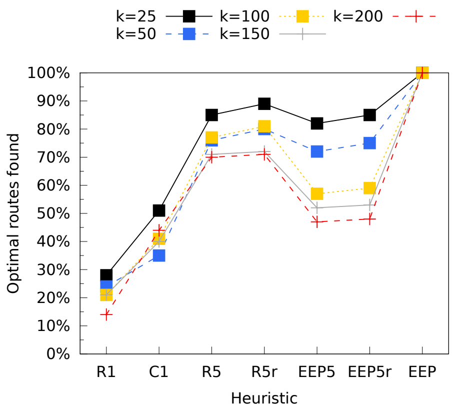

Figure 6: Optimal routes found *withinMunich*, varying $k$.

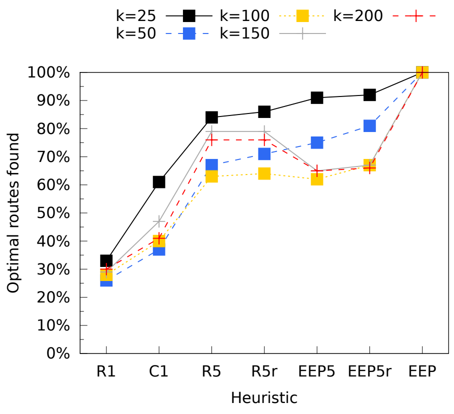

Figure 7: Optimal routes *withinRosenheim*, varying $k$.

success rates, ranging from 30.4% using `R1` to 77.4% using `R5r`, and 100% using `EEP`. `EEP` is the only strategy guaranteed to find the shortest path from $S$ to $D$, as it necessarily provides the optimal entry and exit nodes of the corresponding SZs. All other heuristics may result in minor or major detours compared to the optimal route. `C1` is able to produce the optimal result in 44.6% of situations, and 54.4% when accepting short detours of less than 100m. On the other hand, both `C1` and `R1` are not able to find a complete route from $S$ to $D$ at all in 11.2% and 15% of situations, respectively. This happens when $S$ or $D$ are located in another component of the locally known road network than the chosen query point, which seems less likely when choosing a node in the zone's center. The RPS has no means for distinguishing such requests from normal ones, yet users will not be keen on accepting such failure rates. The same problem partly applies to `R5` and `EEP5`, too. However, with five nodes being randomly selected here, the possibility of at least one query point being located in the same component as $S$ or $D$ increases, leading to failure rates of only 2.4% and 3.0%.

The heuristics which are allowed to choose query points from the whole of a zone's road network, i.e., `R5` and `R5r`, outperform the `EEP`-based strategies in the numbers of optimal routes found. Also, the positive effects of the *one reachable* strategy can be seen, as this ensures at least one correct route from $S$ to $D$ to be found, even though not necessarily the shortest one. Moreover, `R5r` and `EEP5r` also produce more optimal results than their pure counterparts and can hence be considered better fit for actual deployment, with `R5r` offering the best results throughout almost all of our experiments. Additionally, using `R5r` and $k = 100$, more than 97% of route requests result in a detour of less than 500m.

Figures 6 and 7 depict the relation between different values of $k$ and the number of correctly found optimal routes for different map extracts. In Figure 6, one can clearly see how a larger value of $k$ negatively influences the accuracy of our algorithm for all heuristics when applied to the tightly meshed road network of the *withinMunich* scenario. As a larger $k$ implies the creation of larger zones, the chance of choosing query points that leave or enter a zone using exit or entry nodes other than the ones belonging to the optimal route also increases. For small values of $k$, the random and entry-point-based strategies almost perform equally well. However, for larger values of $k$, the `EEP`-based heuristics degenerate at a much faster rate than `R5` and `R5r`. This is caused by growing sets of exit (entry) points `EEP` can choose from, so the likelihood of selecting the optimal nodes decreases. In contrast, query points selected by `R5` and `R5r` can be located all over the SZ, which increases the chance of a route retrieved from the RPS to contain the best exit/entry node.

From Figure 7, it can be observed that for the *withinRosenheim* scenario using $k \leq 100$, the `EEP`-based heuristics perform even better than the random ones. We attribute this to the fact that there are only few suboptimal exit or entry nodes our algorithm can select, caused by a less tightly meshed road network. In such situations, `R5` and `R5r` may choose several query points that produce routes leading out of a zone using the same suboptimal exit point, which is less likely to occur using `EEP5/r`. For $k \geq 150$, however, `R5` and `R5r` perform considerably better than the entry-point-based heuristics also in the *withinRosenheim* scenario. Surprisingly, using `R5` or `R5r` with $k \geq 150$ here produces even more optimal results than with lower values of $k$, apparently caused by the the underlying road network having more distinct clusters than in Munich. Overall, we hence conclude, that if there is a high number of exit and entry points – which commonly is the case for large values of $k$ and within tightly meshed road networks – the strategies based on random selection of several query points such as `R5r` are preferable, as they are the ones most likely to find the optimal route.

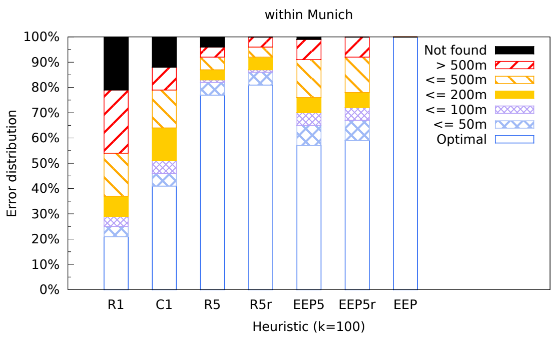

Figure 8: Error distribution *withinMunich* ($k = 100$).

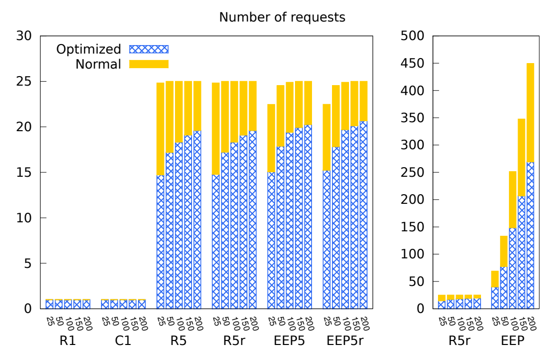

Figure 9: Number of requests to the RPS.

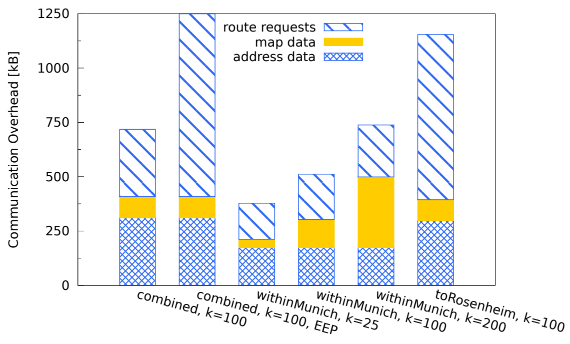

Figure 10: Mean communication cost using `R5r`.

Another interesting observation can be made from Figure 8, which indicates that in the *withinMunich* scenario, `EEP5` results in fewer failures than `R5`. The graph suggests that while `EEP5` is less likely to produce the optimal route *withinMunich*, it is better able to find at least one route from $S$ to $D$ than `R5`. We attribute this effect to the existence of several distinct components within a SZ's road network: If the zone's exit (entry) nodes are evenly distributed among the components, each of them is equally likely to house a query point using `EEP5`, thereby increasing the chance that a route can be found. With `R5`, however, the biggest component – which is not necessarily the one containing $S$ or $D$ – is likely to contribute a majority of query points, which will all lead to failure. Part of our ongoing research is hence to design more sophisticated heuristics for query point selection based on suchlike characteristics of the local road network.

| scenario | k | mean error | max error |
|---|---|---|---|
| *withinMunich* | 25 | 15.6m | 427.0m |
| | 100 | 48.5m | 917.5m |
| | 200 | 64.1m | 958.3m |
| *MunichToErding* | 25 | 22.0m | 548.6m |
| | 100 | 47.2m | 1,168.4m |
| | 200 | 48.3m | 1,097.7m |
| *combined* | 25 | 21.9m | 898.4m |
| | 100 | 47.2m | 1,168.4m |
| | 200 | 59.9m | 2,047.8m |

Table 2: Mean and maximum errors using `R5r`.

Table 2 has the errors in meters that can be observed for different scenarios and varying values of $k$ using `R5r`. Again, one can clearly see how a growing value of $k$ negatively influences the accuracy of our approach. Using $k = 100$, a maximum detour of more than 1.1km occurs. On average, however, our approach is able to produce a mean detour of only 47.2m per request, which we believe to be an acceptable tradeoff for privacy. Additionally, our results suggest that the accuracy of our approach is independent from the actual distance from $S$ to $D$, as the values for the different routing scenarios indicate almost no noticeable differences.

## 5.3 Number of Requests to the RPS

Apart from quality of service characteristics, also the number of requests that have to be sent to the online RPS should be optimized. Figure 9 depicts the average number of requests combined over all routing scenarios using the standard procedure and the optimization described in Section 4.6. In contrast to the trivial cases of `R1` and `C1`, which always only send a single query to the RPS, the methods based on selecting $N$ query points require up to $N^2$ queries, given that enough nodes can be found in a zone. As one can see, for small $k$ the `EEP`-based heuristics automatically adapt to the required minimum of requests also in normal mode. By using the proposed optimization, however, the number of requests that have to be sent to the RPS on average can be considerably lowered, e.g., from 25 to 18.6 using `R5r` with $k = 100$. It can also be seen, that for smaller values of $k$ a higher number of requests can be avoided. Additionally, the `EEP` strategy, which is the only one guaranteed to produce the optimal result, requires up to several hundreds of requests, increasing with larger values of $k$.

## 5.4 Combined Communication Cost

PrOSPR also needs to query restricted space information for the source and target postal code areas from the map server. The latter returns a list of all addresses and their GPS coordinates within the bounds of the requested region. Imperfect OSM information potentially lead to a smaller download size and to the creation of larger SZs. For the regions we chose for testing, however, address information seems to be quite complete. Figure 10 shows the mean amount of data combined for $S$ and $D$ that has to be downloaded to acquire complete address information for a route request, averaging at 174kB for the *withinMunich* scenario, and 437kB for *withinRosenheim*. Maximum individual file size is 400kB for a postal code area in Rosenheim. In a second step, map information has to be downloaded for the created SZs, which produces considerably less overhead. Figure 10 also depicts

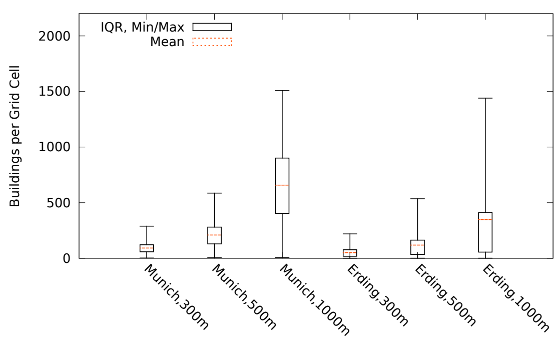

Figure 11: Number of buildings contained in grid cells with varying side length in Munich and Erding.

the communication overhead caused by different values of $k$ and routing scenarios using `R5r`, if not stated otherwise. Comparing the *withinMunich* scenario with *toRosenheim*, one can see how the query overhead increases with growing distance between source and destination, even though the numbers of requests remain exactly the same. Finally, `EEP` produces an overhead of 2.8MB only for the route requests.

## 5.5 Privacy Evaluation

Relying on the principle of location obfuscation, both our approach and the one in [17] do not directly betray the user's endpoints to the RPS. In contrast, OPAQUE [10][11] hides a user's true source and destination by injecting an arbitrary number of route requests for dummy locations in the user's vicinity. The level of location privacy hence only relies on the number of requests sent to the RPS, leading to massive communication overhead for high levels of anonymity. Additionally, here the actual source and destination addresses are communicated to the RPS, which in combination with inappropriately chosen dummy positions might lead to an easy identification of a user's actual locations. Vicente et al. [17] propose to facilitate fixed-size grid cells in order to conceal the user's true start and destination. In Figure 11, we analyze the resulting privacy level using grid cells with a side length $s$ of 300, 500 and 1000m. We generated 50 different randomly positioned grids for each setting, using the same addresses for the SZ approach. The grid cells contain a good amount of buildings on average, e.g., when applying a 300m grid in the Erding area, about 82.64 buildings are contained on average. However, in each of the tested scenarios it happens that there are almost no buildings contained within the corresponding cell. For instance, when applying a 500m grid in the Munich area, there is a case where only four buildings are contained. And even with a side length of 1000m, there are cells that only contain six buildings. The same problem can be expected to apply to more rural regions as well.

In contrast, PrOSPR utilizes map information in order to dynamically construct cloaked areas around a user's source and destination addresses, which guarantees a minimum number of $k$ restricted spaces being contained. Table 3 shows the average number of buildings within a SZ as well as the corresponding side length of the created zones. It can be seen that the required number of $k$ is easily reached in each of the tested scenarios, with a minimum amount of 100 buildings for $k = 100$ and a mean count of 170, which clearly overshoots the desired mark. One can also observe that the mean side length increases for larger values of $k$. Finally, it can also be seen how the side lengths behave differently for the *withinMunich* and *withinErding* scenario. In the latter, less densely populated area, the side length needed for reaching the same building count as in Munich is considerably longer. We hence argue that such an adaptive, map-aware approach for location obfuscation is indeed better suited for the field of private shortest path retrieval than the grid-based approach.

| **scenario** | **k** | min **c** | ∅ **c** | max **c** | ∅ **size** |
|---|---|---|---|---|---|
| *withinMunich* | 25 | 25 | 45.30 | 179 | 276.79m |
| | 100 | 100 | 163.54 | 1,180 | 680.54m |
| | 200 | 200 | 347.64 | 1,383 | 1,161.03m |
| *withinErding* | 25 | 25 | 48.39 | 218 | 365.87m |
| | 100 | 100 | 179.25 | 797 | 957.31m |
| | 200 | 200 | 303.78 | 797 | 1,359.00m |
| *combined* | 25 | 25 | 45.80 | 290 | 283.64m |
| | 100 | 100 | 170.39 | 1,241 | 715.34m |
| | 200 | 200 | 326.98 | 1,756 | 1,118.74m |

Table 3: Number of buildings and SZ side length.

## 5.6 Discussion

In case a correct route from $S$ to $D$ cannot be found – which might happen using `C1`, `R1`, `R5`, and `EEP5` – a practical implementation would have to start over and send additional route request to the RPS. By analyzing the set of seen route requests, however, a RPS provider can narrow down a user's actual locations by extracting all addresses that are mapped to those segments of $G_i$, that did not yet contain a node in $V_i$ of the original request. For this reason, we argue that also from a privacy perspective, the *one reachable* heuristic should be used, as it prevents the need to start over the whole process and the leakage of valuable information. On the other hand, if the RPS provider is aware of the deployment of the *one reachable* strategy, he may try to remove those addresses from $B_{A_i}(\mathrm{SZ}_i)$ which map to components of $G_i$, that did not contribute a node to $V_i$. An analysis of how likely this might affect a user's location privacy by reducing the effective value of $k$ will be examined in our future work.

In this section, feasibility, quality-of-service and overhead of our approach have been evaluated. The results indicate that PrOSPR is able to effectively perform $k$-immune online shortest path retrieval at the cost of slightly decreased mean accuracy and noticeable communication overhead. For $k = 100$, however, the latter has been shown to be smaller than 750kB on average, which we consider to be acceptable.

## 6. CONCLUSION

This paper introduced PrOSPR, a novel approach for performing shortest path retrieval from an online RPS in a privacy-preserving way. Our system aims at preventing location tracking and de-anonymization of a pseudonymously known mobile user based on precise address information by the means of $k$-immune shortest path requests. Our approach neither requires collaboration from the RPS provider, nor is it based on the existence of a TTP. Instead, it can be used with standard online route planning services. Restricted space inference is counteracted by deploying a $k$-anonymity-based means for map-aware location obfuscation. Based on empirical evaluation, we analyzed the feasibility of our approach in terms of accuracy, privacy and communication overhead. We have shown that different heuristics can be used to balance the trade-off between quality of service and cost, e.g., allowing us to retrieve the optimal route in up to 77% of situations with less than 19 requests to the RPS on average using `R5r` and $k = 100$, and a mean detour of only 47.2m. Moreover, we have shown that our system is able to effectively protect its users against attacks based on restricted space inference by constantly providing $k$-immune online shortest path queries.

We see this as a first step only. Directions for future work on this topic can be seen in finding techniques able to further reduce the communication overhead of our approach, such as optimizing the number of query points, and increasing quality-of-service by eliminating detours of more than 100m length. Additionally, privacy implications of using the *one reachable* heuristics deserve further attention. Another interesting question concerns the types of additional contextual information about restricted spaces, such as opening hours, which have to be considered for effectively creating $k$-immune route requests against an attacker who also uses such information in order to infer a user's possible locations.

## 7. REFERENCES

[1] C. Ardagna, M. Cremonini, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. Location privacy protection through obfuscation-based techniques. In S. Barker and G.-J. Ahn, editors, *Data and Applications Security XXI*, volume 4602 of *Lecture Notes in Computer Science*, pages 47–60. Springer Berlin Heidelberg, 2007.

[2] B. Bamba, L. Liu, P. Pesti, and T. Wang. Supporting anonymous location queries in mobile environments with privacygrid. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 237–246, New York, NY, USA, 2008. ACM.

[3] C. Chow, M. Mokbel, and X. Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based services. *GIS '06 Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems*, 2006.

[4] P. Golle and K. Partridge. On the anonymity of home/work location pairs. In *Proceedings of the 7th International Conference on Pervasive Computing*, Pervasive '09, pages 390–397, Berlin, Heidelberg, 2009. Springer-Verlag.

[5] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, MobiSys '03, pages 31–42, New York, NY, USA, 2003. ACM.

[6] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, MobiSys '03, pages 31–42, New York, NY, USA, 2003. ACM.

[7] M. Gruteser and X. Liu. Protecting privacy, in continuous location-tracking applications. *Security Privacy, IEEE*, 2(2):28–34, Mar 2004.

[8] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *Pervasive Services, 2005. ICPS '05. Proceedings. International Conference on*, pages 88–97, July 2005.

[9] J. Krumm. Inference attacks on location tracks. In *Proceedings of the 5th International Conference on Pervasive Computing*, PERVASIVE'07, pages 127–143, Berlin, Heidelberg, 2007. Springer-Verlag.

[10] K. Lee, W.-C. Lee, H. Leong, and B. Zheng. Opaque: Protecting path privacy in directions search. In *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on*, pages 1271–1274, March 2009.

[11] K. C. Lee, W.-C. Lee, H. V. Leong, and B. Zheng. Navigational path privacy protection. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 691–700, New York, NY, USA, 2009. ACM.

[12] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), Mar. 2007.

[13] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new casper: A privacy-aware location-based database server. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 1499–1500. IEEE, 2007.

[14] K. Mouratidis. Strong location privacy: A case study on shortest path queries. In *Data Engineering Workshops (ICDEW), 2013 IEEE 29th International Conference on*, pages 136–143, April 2013.

[15] P. Shankar, V. Ganapathy, and L. Iftode. Privately querying location-based services with sybilquery. In *Proceedings of the 11th International Conference on Ubiquitous Computing*, Ubicomp '09, pages 31–40, New York, NY, USA, 2009. ACM.

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

[17] C. Vicente, I. Assent, and C. S. Jensen. Effective privacy-preserving online route planning. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 1, pages 119–128, June 2011.

[18] M. Wernke, P. Skvortsov, F. Dürr, and K. Rothermel. A classification of location privacy attacks and approaches. *Personal and ubiquitous computing*, 18(1):163–175, 2014.

[19] K. Wiesner, S. Feld, F. Dorfmeister, and C. Linnhoff-Popien. Right to silence: Establishing map-based silent zones for participatory sensing. In *IEEE ISSNIP 2014 - Symposium on Participatory Sensing and Crowdsourcing*, pages 339–344, Singapore, Singapore, Apr. 2014.

[20] M. Xue, P. Kalnis, and H. Pung. Location diversity: Enhanced privacy protection in location based services. In T. Choudhury, A. Quigley, T. Strang, and K. Suginuma, editors, *Location and Context Awareness*, volume 5561 of *Lecture Notes in Computer Science*, pages 70–87. Springer Berlin Heidelberg, 2009.