PARTS – Privacy-Aware Routing with Transportation Subgraphs

Christian Roth, Lukas Hartmann, and Doğan Kesdoğan

University of Regensburg, Chair of IT Security Management,
Regensburg, Germany
{firstname.lastname}@ur.de

Abstract. To ensure privacy for route planning applications and other location based services (LBS), the service provider must be prevented from tracking a user's path during navigation on the application level. However, the navigation functionality must be preserved. We introduce the algorithm *PARTS* to split route requests into route parts which will be submitted to an LBS in an unlinkable way. Equipped with the usage of dummy requests and time shifting, our approach can achieve better privacy. We will show that our algorithm protects privacy in the presence of a realistic adversary model while maintaining the service quality. [1]

Keywords: Routing, Location Privacy, Anonymity

1 Introduction

In most areas of life, people are using smart devices and thereby generating data with personally identifiable information. Smart services collect personal data and base their quality of service (QoS) on this information. For this reason, data mining is on a broad research agenda, being able to extract precise user information from massive databases. With this data, the user experience and convenience of a service can be improved. For example, Apple's operating system iOS 9 studies a user's behavior to suggest the probable next locations the user wants to drive to [4]. However, this can be misused to control a user and massively violate his privacy. This was demonstrated by Facebook by experimentally changing the emotional perception of the user through news feed consumption [5].

Consequently, the user has the choice either not to use the smart service and lose the QoS improvements or use the smart device and resign their privacy. Since opting out of a service is less attractive for most users, the service provider almost always can collect highly sensitive user data without any significant consequences. On the first look, there is no possibility to resolve this dilemma. But nowadays smart phones are powerful enough to apply intelligent techniques to combine sensitive data in offline mode with additional online (real-time) information. With this approach, it will be much harder for an adversary, like data miners, to collect sensitive personal data.

[1] The final publication is available at Springer via https://dx.doi.org/10.1007/978-3-319-70290-2_6

To demonstrate our approach, we chose the well-known and popular field of route planning. Route planning services need location information from the user, which can be linked and analysed so that micro- and macroscopic profiles can be easily generated. It is shown that the start and target of a route are enough to deanonymise an otherwise anonymous user with a high probability [3]. Thus, the location information is very sensitive and protection algorithms have to be carefully designed. In addition, traditional obfuscation techniques from other scenarios dealing with location privacy cannot be applied because the route must be calculated for the exact given locations to provide the best user experience.

It would be theoretically possible to exclusively use the smart phone in offline mode by downloading all routing information from an area. But this means relinquishing any helpful third party information, i.e. any useful additional services. Consequently, to soften the trade-off between privacy and utility in the case of routing, we introduce privacy-aware routing with route parts using several interim destinations on the path to the overall target location.

1.1 Contribution

To the best of our knowledge, this is the first paper to investigate straightforward mechanisms to protect a user's route. We implement a basic set of these mechanisms (see section 4) in a routing algorithm which divides the route into several parts to hide location information. However, we can still guarantee navigation to an exact location from a specific start while also including real time data from the untrusted cloud. We believe that these are real crucial arguments to create a widely adopted application. In this paper, we follow the classical approach of the triple bottom line of security "algorithm, adversary and evaluation" and contribute with:

1. Our algorithm *PARTS* based on dynamical, unlinkable route parts using straightforward protection mechanisms.
2. A realistic adversary model in which a honest-but-curious location based service (LBS) provider wants to reconstruct full routes.
3. A detailed evaluation of our algorithm by means of a self generated data set based on our theoretical user model, where we derive the best combination of privacy protection methods.

Due to limited space, we plan to present a real smartphone application implementing our algorithm in future work. Therefore, we do not analyze extended performance figures such as battery impact and user experience.

1.2 Structure

The remainder of the paper is organised as follows: After a review of related work in section 2, the general system model is introduced in section 3. In section 4 we present techniques for privacy-aware routing, introducing our *PARTS* algorithm based on route parts. Section 5 contains the description of the adversary model who wants to reconstruct the navigated route. The results of our evaluation are depicted in section 6 where we analyse PARTS' overhead, its performance and ability to protect a user's privacy. We discuss and summarise the results in sections 7 and 8 respectively.

2 Related Work

Anonymity of location information is a relevant topic in academic research. The best discussed concept is the idea of *k-Anonymity* introduced by Sweeney [10]. A subject is anonymous within a set of k subjects reporting the same cloaked geographical region instead of the real locations of the subjects. Implementations of this concept tend to minimise the area of the cloaking region making it easy for adversaries with background knowledge to infer sensitive data [9]. An extension of this model called *l-Diversity* was introduced by Machanavajjhala et al. [6]. Here, the cloaking area should contain at least l different locations.

Wang and Liu [11] showed that anonymity and diversity of locations can be achieved by realising k-Anonymity with a graph based scalable model. However, this approach requires a certain number of active users. They state that different road segments per user can be used to realise l-diversity for disclosed locations.

Palanisamy and Liu [8] introduce their approach of *MobiMix* in which mix zones are placed on the road network to gain anonymity of locations. The users' location anonymity is ensured by unlinkable pseudonyms when entering or exiting a mix zone and within such a zone by non-traceability.

Michalevsky et al. [7] introduce *PowerSpy*, an application which is able to infer a user's location and his driving route by using the power consumption of his smart phone, without any permissions to access GPS, WiFi, or other location data on the phone. Their basic idea is to measure the power consumption of different routes in advance and train a machine learning algorithm.

While using semantic information to obtain a contextual service (e.g. recommendation of restaurants nearby) or to share information about a visited venue, Agir et al. [1] present a solution where the semantic dimension of a location can be protected by generalising the semantic tag and locations can be obfuscated.

To counteract location breaches by inference attacks or the reveal of semantic behaviour by membership inclusion attacks, Bindschaedler and Shokri [2] introduce synthesised plausible location traces which are separated into semantic and geographic features. This separation is required, because people with similar lifestyles share common semantic traces but differ in geographical patterns.

3 System Model

We assume that users utilise route planning devices that combine offline and online data to find the best possible route for a trip. The real start and target location should be hidden from a location based service (LBS). Therefore, a trusted application on the user's (mobile) device is used which employs anonymisation

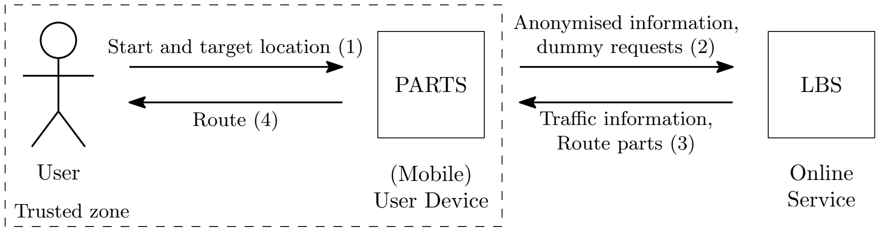


Fig. 1: Our *PARTS* framework works offline on the user's device and sends anonymised information to the LBS to protect the user's real start and target location (v_S, v_T). The LBS processes the requests and returns traffic information and route parts which are aggregated by our algorithm to the actual route.

techniques and/or dummy requests. This application uses static offline data and sends the anonymised request to the LBS. Here, the route request is processed and online real-time information like traffic jams are taken into account. For the sake of simplicity, we assume that an LBS has access to this information, even though it gets harder for an LBS to collect such data when *PARTS* is widely applied. The app on the user's device deanonymises the calculated route and presents the user with the routing information from the actual start to target location. The user sticks to the route recommended by the local device, as depicted in Figure 1. Here, the trusted application on the user's device is our *PARTS* framework (c.f. section 4) which uses route parts as anonymisation technique.

In this setting, we consider an honest-but-curious LBS that is interested in deriving the start and target point of a user's route.

3.1 Road Network

We model the road network in the well-known way of a directed graph $G = (V, E)$. The road transitions form the set of vertices $v \in V$ and are connected via road segments represented by the edges $e \in E$. Since G is directed, a route segment from v_1 to v_2 has two corresponding edges v_1v_2 and v_2v_1, whereas one-way streets are represented by a single edge. The degree of a vertex $\deg(v)$ corresponds to the number of different roads one could reach from this point. Thus, road intersections are vertices v with $\deg(v) \geq 3$, whereas vertices with $\deg(v) = 1$ are equivalent to the end of a dead-end street. In our setting, points which connect road segments but are neither dead-ends nor intersections are of little relevance. To simplify the algorithm, we use the minor G^* of G in which the degree 2 vertices have been deleted by contracting one of the adjacent edges.

3.2 Users

Simplifying the presentation, we focus on route planning from intersection to intersection. Therefore, each location can be identified uniquely with a vertex v in the graph G representing the road network.

The set of users in our model is described with $U = (u_1, u_2, \dots, u_m)$. There can be time intervals where no user moves and different users can travel to the same location at any time. Users submit their route requests with their current location v_S and their target v_T to the *PARTS* algorithm in a trusted zone, as depicted in Figure 1. Our algorithm processes the user's request and submits several (anonymised) route requests to the LBS. Each request results in an event $r_u(t) = (v_{S_i}, v_{T_i})$ from user u at time t recorded by the LBS. Hence a route (part) can be uniquely described with a pair of start and target (v_{S_i}, v_{T_i}).

3.3 LBS Provider

Additional information for a requested route can be provided by an LBS that has knowledge about context like real time traffic information or road blocks due to road constructions. Our algorithm *PARTS* uses this information to combine it with a locally generated route solely based on geographical information. By providing v_{S_i} and v_{T_i} as a result of a user request, an LBS collects personal data such as the request time. Furthermore, it is worthwhile to mention that the LBS has a similar amount of geographical information as our algorithm.

3.4 Adversary

The adversary in this paper is typically an LBS or an external observer who has access to all requests of all users, i.e. all received requests of an LBS. The adversary's main goal is to link the requests again and hence learn a route's overall start v_S and target vertex v_T. The adversary runs his attack *a posteriori*. Since he cannot be sure which requests are performed by a specific user, he collects a whole event log A for a specific amount of time and carries out his attack. It is reasonable for him to assume that A contains events from multiple users. However, it is possible that some of these events differ in time but share the same tuple of start and end vertices (v_S, v_T).

For his attack, the adversary takes into account geographical and temporal information described in section 5. With limitation, the attack can also be performed by third-party services such as an Internet service provider (ISP) who has access to a user's request log but lacks knowledge about, for example, geographical information.

For our setup, we assume that the adversary knows all system parameters, i.e. he knows which privacy protection methods (PPMs) are applied and what specific parameters are used. This results in an even stronger adversary to stress our algorithm. Together with the PPMs and the event log, he starts his attack to deanonymise a route R, respectively a user u.

4 Strategies for Privacy-enhanced Routing

Users request routing instructions from an LBS providing their geographical data v_S and v_T. To enforce their privacy goals, users apply PPMs, like our following *PARTS* framework, to hide their start and target location. To achieve this, *PARTS* can deploy different countermeasures which will be described in the following section.

In this work we focus on the route planning algorithm which resides in the application layer and tries to prevent information leakage from there. Hiding traffic data (such as IP addresses) is not the focus of our algorithm. However, this data can be used to link different route requests including dummy requests to a specific user. In this case, our algorithm does not provide any anonymity at all. We therefore assume that route requests will be submitted using an anonymous channel like Tor in order to hide information from lower communication layers.

The PPMs mentioned in this section may be combined to provide a stronger protection against an honest-but-curious LBS. We will evaluate different settings and combinations of the following PPMs in section 6.

4.1 Route Parts

PARTS splits every route R into multiple route parts R_i with start nodes v_{S_i} and target nodes v_{T_i}. The iterative combination of all R_i shall be the complete route R, thus $v_{S_{i+1}} = v_{T_i}$ holds. For each route part, a separate route request is necessary to obtain semantic information like traffic information. This request results in an event $r_u(t) = (v_{S_i}, v_{T_i})$ at the LBS. The LBS gains information that a user u wants to drive from v_{S_i} to v_{T_i} without knowing which of the start nodes $v_{S_1}, v_{S_2}, \ldots$ is the overall start v_S (resp. which v_{T_i} is the overall target v_T).

For each route part R_i, a subgraph G_i^* of G^* is constructed. The graph G_i^* shall contain all vertices having at most a specific distance to v_{S_i}. We call this adjustable distance *Hops* within our framework. Therefore, $dist_{G^*}(v_{S_i}, v) \leq Hops$ holds for all vertices v in G_i^*. The vertices v with $dist_{G^*}(v_{S_i}, v) = Hops$ are called *boundary vertices* $\partial V(G_i^*)$ and are candidates for being the target node v_{T_i}. The subgraph G_i^* contains at least one vertex with degree 3 by construction of G^* and therefore at least one intersection resulting in at least two candidates for v_{T_i}. It is obvious that a higher Hops count results in more target node candidates. To prohibit recursive routes, already used vertices are stored in a blacklist V_{BL} for the i-th iteration and will be ignored.

The node $v^* \in V_i^{cand} = \{v \in \partial V(G_i^*) \mid v \notin V_{BL}\}$ with the shortest straight line distance[2] $|v^* - v_T|$ to v_T is selected and used as target node v_{T_i} for the i-th iteration. A route request for v_{S_i} to v_{T_i} is submitted to the LBS and the calculated route forms the next route part R_i. For an example of a subgraph, see Figure 2. For the next iteration, the algorithm sets $v_{S_{i+1}} = v_{T_i}$ and calculates a new subgraph G_{i+1}^*. This process stops, until the overall target node v_T is contained in one of the subgraphs G_i^*. In this case, the last route part R_i will be the route from v_{S_i} to v_T.

[2] The straight line distance is used since it is locally computable and does not require any additional requests to a third party, thus not leaking any information.

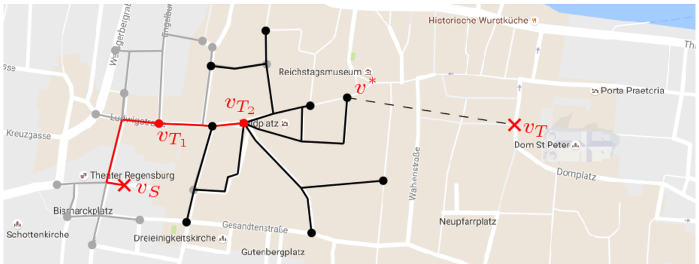


Fig. 2: The subgraph G_3^* is depicted in black, whereas the subgraphs G_1^* and G_2^* are grey. The already calculated routes R_1 and R_2 are red. The vertex v^* has the lowest straight-line distance $|v^* - v_T|$ to the overall target v_T and will be used as the next step v_{T_3}.

4.2 Dummy Traffic

We extend the PARTS algorithm from the previous section to include dummy requests to an LBS which may be needed in order to provide privacy. Using the previous setup, a route $R = \bigcup_{i=0}^{n} R_i$ consisting of n parts results in n events $r_u(t_i)$ at an LBS. It is trivial for an LBS to derive the complete route R by chronologically combining R_i. To oppose this, we introduce dummy requests in our system which should make it more challenging for an LBS to gain s and t.

Instead of submitting only one route request in the i-th iteration for the node v^* with the shortest straight-line distance to v_T, our algorithm produces one route request from v_{S_i} to v for some of the target vertex candidates $v \in V_i^{cand}$. Still, v^* is used as target vertex v_{T_i} and only the associated route is used as the route part R_i. In particular, the upper bound for dummy requests is $|V_i^{cand}| - 1$ since one request is always a legitimate request (v_{S_i}, v_{T_i}). The *PARTS* algorithm extended with the maximum number of dummy requests is depicted as Algorithm 1.

4.3 Time Shift Requests

As stated in the previous subsection, an event always contains time information (e.g. request time of a route by a user) which is outside of a user's sphere. Therefore, it is easy for an LBS to use this information to reconstruct the complete route using all part requests R_i just by chronologically sorting the request log. The attack is even possible if dummy traffic is used if an adversary uses statistical attacks. By time shifting the subsequent route requests, the attack can be hindered. Different methods of time shifting are possible:

- All route requests are executed after a specific but static amount of time, not allowing any insight into the length of a route part.
- All route requests occur at a random time resulting in no correlation between the length of a route and the request time.

Algorithm 1: PARTS with dummy requests

```
Input: Start vertex $v_S$, target vertex $v_T$
Result: Route $R$ from $v_S$ to $v_T$
$v_{S_1} \leftarrow v_S$;
$V_{BL} \leftarrow \{v_S\}$;
while $v_{T_i} \neq v_T$ do
    $i \leftarrow i + 1$;
    construct $G_i^*$ with $V(G_i^*) = \{v \in G^* \mid dist_{G^*}(v_{S_i}, v) \leq Hops\}$;
    $V_i^{cand} \leftarrow \{v \in \partial V(G_i^*) \mid v \notin V_{BL}\}$;
    if $v_T \in V(G_i^*)$ then
        $V_i^{cand} \leftarrow V_i^{cand} \cup \{v_T\}$;
    foreach $v \in V_i^{cand}$ do
        submit route request $(v_{S_i}, v)$ to LBS;
    $v_{T_i} \leftarrow argmin_{v \in V_i^{cand}}\{|v - v_T|\}$;
    $V_{BL} \leftarrow V_{BL} \cup \{v_{T_i}\}$;
    $R_i \leftarrow$ route from $v_{S_i}$ to $v_{T_i}$ calculated by LBS;
return $R = (R_1, R_2, \ldots)$
```

– Route requests happen in batches, i.e. a specific number of requests are sent to the LBS at the same time to maintain unlinkability between requests.

5 Adversary's Inference Model

In this section, we will present our inference model for the adversary which he uses to deanonymise specific users from the event log of route requests. We will explain how an adversary can exploit the route requests to reconstruct the full route from route parts. Based on this adversary model, we evaluate the privacy provided by our route planning algorithm in section 6.

5.1 Background Knowledge

We assume that the adversary's inference model is based on the knowledge that users move along geographically valid routes using a valid behavior pattern. For example, users respect traffic regulations such as speed limits.

In our scenario, an adversary has extensive knowledge about the geographical structure of the road network represented. However, an adversary may utilise his own geographical database which can partially differ from the database on which our *PARTS* algorithm works (see section 6.1). In general, we assume that such deviations are not harmful to perform the attack illustrated in section 5.2. Thus, he is able to derive the same V_i^{cand} as our algorithm because all system parameters are known, as defined in section 3.4. As a consequence, the adversary can, for example, compute the average travel time $t(R_i)$ for route segments.

5.2 Empirically Improved Guessing

The adversary plans to reconstruct complete routes with the given route requests submitted by different users. Since the adversary is an LBS, he has access to the full event log A. He selects one of the requests for which he has a high interest in recovering the corresponding route. Starting with this route request, the LBS calculates a likelihood tree $\varGamma$ including possible past and future route parts.

Let $r_0 = (v_{S_0}, v_{T_0}) \in A$ be the route request for which the adversary is interested in reconstructing the full route. The adversary collects all route requests $A^{\rightarrow}_{r_0}$ whose start vertex is the target vertex v_{T_0} and which have been submitted to the LBS after r_0. For $r_1 = (v_{S_1}, v_{T_1}) \in A^{\rightarrow}_{r_0}$, we can define the function $f(r_1 \mid r_0)$ which models the likelihood that r_1 was the subsequent route request. Here, both temporal correlation and behavioural plausibility will be taken into account. With $t(R)$ being the travel time of the ideal route R from v_{S_0} to $v_{T_0} = v_{S_1}$ calculated by the LBS and t_0 and t_1 being the submission times of r_0 and r_1, respectively, the temporal correlation is denoted by $f_t(r_1 \mid r_0) = \exp\left(-c \cdot \frac{|(t_1 - t_0) - t(R)|}{t(R)}\right)$, where c is a scaling factor with $c \in [0, 1]$. If the request r_1 was submitted close to the travel time calculated by the LBS, it is more likely that r_1 will be the subsequent route request and $f_t(r_1 \mid r_0)$ increases.

For the behavioural plausibility $f_b(r_1)$, we count how many requests in the event log A have the same points as r_1 and divide this number by the total number of requests. Additionally, we introduce a factor $\lambda \in [0, 1]$ indicating the weight of the temporal correlation f_t in the likelihood f. Hence, we obtain

$$\begin{aligned} f(r_1 \mid r_0) &= \lambda \cdot f_t(r_1 \mid r_0) + (1 - \lambda) \cdot f_b(r_1) \\ &= \lambda \cdot \exp\left(-c \cdot \frac{|(t_1 - t_0) - t(R)|}{t(R)}\right) + (1 - \lambda) \cdot \frac{|\{r \in A \mid r = (v_{S_1}, v_{T_1})\}|}{|A|} \end{aligned}$$

Since both values $f_t(r_1 \mid r_0)$ and $f_b(r_1)$ are in the range $[0, 1]$, we also have $f(r_1 \mid r_0) \in [0, 1]$. Here, a value close to 1 indicates a high likelihood that the route r_1 is the subsequent route part to r_0.

The adversary forms sets $A^{\rightarrow}_{r_1}$ for all $r_1 \in A^{\rightarrow}_{r_0}$ and evaluates the function $f(r_2 \mid r_1)$ for $r_2 \in A^{\rightarrow}_{r_1}$. He repeatedly continues with this approach and constructs a likelihood tree $\varGamma$ with root r_0 and $r_1 \in A^{\rightarrow}_{r_0}$ being the first level nodes, etc. An edge (r, r') in this tree is weighted with the likelihood function $f(r' \mid r)$. To construct the prior route parts w.r.t. r_0, the adversary uses a similar approach by forming sets $A^{\leftarrow}_{r_j}$ containing route parts whose target vertex equals the start vertex of r_j. If the value of f falls below a predefined threshold value ε, the process will not be continued within the related subtree of $\varGamma$. Finally, the adversary chooses the path in $\varGamma$ whose multiplied likelihood value along this path is the highest and uses the corresponding route requests as the reconstructed route R'.

5.3 Privacy Measurement

To evaluate the privacy of our algorithm, we compare the real route $R = (R_1, \ldots, R_k)$ from v_S to v_T from a user with the guess $R' = (R'_1, \ldots, R'_l)$ from $v_{S'}$

to $v_{T'}$ calculated by the adversary. His guess R' is the path in the tree Γ introduced in section 5.2 with the highest likelihood. The following metrics will be applied, each measuring a different privacy aspect.

Distance to start/target vertex One of the goals of our algorithm was the protection of the start and target location of a route request. For this reason, we measure the straight-line distance $dist(R, R') = |v_S - v_{S'}| + |v_T - v_{T'}|$ between the start vertices and target vertices from R and R'.

Fit of reconstructed route To protect movement patterns, reconstructing a route from route parts should not be easy for the adversary. Therefore, we measure $fit_{rou}(R, R')$, the percentage of the fit between guessed route and original one. Here, we count the number of correctly guessed route parts normalised over the total number of route parts in the original route.

Fit of continuous segments In order to measure the attained linkability protection of our algorithm, we calculate the fit of continuous segments $fit_{seg}(R, R')$. We count the number of route segments successfully linked together by an adversary without any error in-between and normalise this figure over the total number of route segments. For instance, if R has $k = 6$ segments and an adversary was able to link segments (R_1, R_2, R_3) and (R_5, R_6) but missed to link R_4, we have $fit_{seg}(R, R') = 0.5$.

6 Evaluation

This section explains how we generated data according to our system model and applied the adversary's inference model to evaluate the quality of our algorithmic approach based on route parts, presented in section 4.

6.1 Dataset and Simulator

PARTS is based on data from OpenStreetMap (OSM). In our setting we used a small portion of the whole dataset, more specifically a subregion of Bavaria, Germany. This region was converted to correspond to our graph setting.

Several users were created using a custom simulator. They move along the region in a predefined (but somewhat random) way[3]. Since our simulation should model users realistically, users follow different moving patterns resulting in route requests. All users submit route requests for activities which happen multiple times, like trips to a supermarket, whereas 80% of the users follow individual

[3] Since our users are exclusively moving within a city, we use the simplified assumption that travel speed is constant per road segment (homogeneous flow). Hereby, we use values from 37.5 kph to 62.5 kph. The adversary only knows that people will respect traffic regulations, therefore he uses a constant value of 50 kph for the whole route.

regular routines. Additionally, 30 % of the users make random trips. On average, there are three moving patterns per user. We simulated a whole month which led to 398 moving patterns (every person therefore travels between 25-50 times a month) resulting in 36,167 (part) route requests. These are summed up values for the different combinations of our PPMs in place including direct requests.

6.2 Experimental Setup

After generating the data as explained above, we applied the adversary's inference model to reconstruct the complete route from route segments (c.f. section 5). To prove the power of our inference model, we showed that the usage of temporal correlation improves the chance of reconstructing a route correctly.

We successively applied more PPMs to get an insight into the adversary's ability to reconstruct the whole route of a user from the event log. We also tested different values for the parameters in the inference model to obtain the strongest adversary. More precise, we evaluated $c \in \{0.01, 0.1, 1\}$ for the scaling factor in the temporal correlation function f_t and $\lambda \in \{0, 0.5, 1\}$ for the weighting of f_t in the likelihood function f (see section 5.2). Since the combination of $c = 0.01$ and $\lambda = 0.5$ results in the strongest adversary, we chose these values. Furthermore, we set $\varepsilon = 0.3$ for the threshold under which further route segments will not be considered in the likelihood tree Γ. If dummy traffic is used, we will send two dummy route requests per real request. A timeslot occurs every nine minutes.

6.3 Overhead of Segmented Routes

In this paragraph, we will investigate how the parameter $Hops$, as described in section 4.1, will influence the route quality of our routing algorithm, i.e. the distance overhead of a route built with route parts compared to the ideal route.

Even though a user should have the possibility to select the ideal $Hops$ size, not every option makes sense regarding user experience and privacy – the main factors for the QoS. It is obvious that a higher $Hops$ size results in larger route segments and thus may disclose more private information. However, it often results in a better user experience because routes constructed of fewer segments lead to routes more similar to thoroughly constructed and therefore ideal routes.

In order to find a good trade-off between privacy and user experience we analysed routes with different numbers of intersections. Figure 3 shows the ratio to the optimal route for routes with 10 to 25 intersections (100% is the optimum). The different colors indicate the different values for the $Hops$ parameter. *MIXED* uses a random value of $\{1, 3, 6, 12\}$ as the $Hops$ parameter for each iteration.

Figure 3 illustrates that the smaller the $Hops$ parameter gets the higher the distance overhead of a route is. Furthermore, $MIXED$ yields almost the same results as $Hops = 6$ does. In our simulation, on average all settings for the privacy enhanced routing resulted in an overhead to the optimal route. Obviously, there is no overhead if the complete route is shorter than the used $Hops$ size since the complete route equals the first and only route segment (resulting in no privacy).

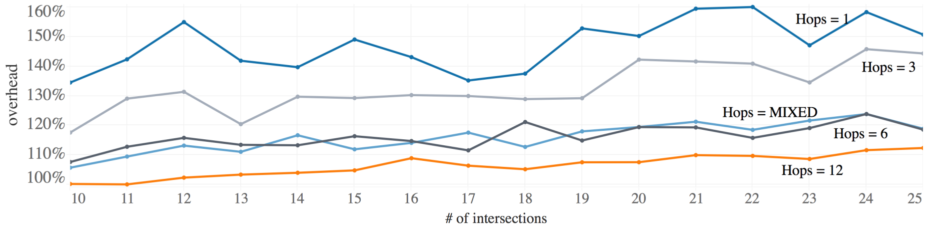


Fig. 3: Distance overhead for different values for the parameter $Hops$ with 250 iterations per number of intersections.

6.4 Privacy Related Results

We applied the privacy measures presented in section 5.3 to our simulated data, namely the number of times in which an adversary was able to reconstruct the route (fit_{rou} and fit_{seg}) and the distance between the reconstructed route's start/end point and the real ones ($dist$). We chose $Hops$ to be 6, 12 and $MIXED$.

In the remainder of this section and in the subsequent figures, we use the following abbreviations for the applied PPMs of our $PARTS$ algorithm: $DIRECT$ = route request without any PPMs applied, P = Route Parts, D = Dummy requests, Tb = Timeshift in batch mode, Tr = Timeshift in random mode, and Ts = Timeshift in timeslot mode.

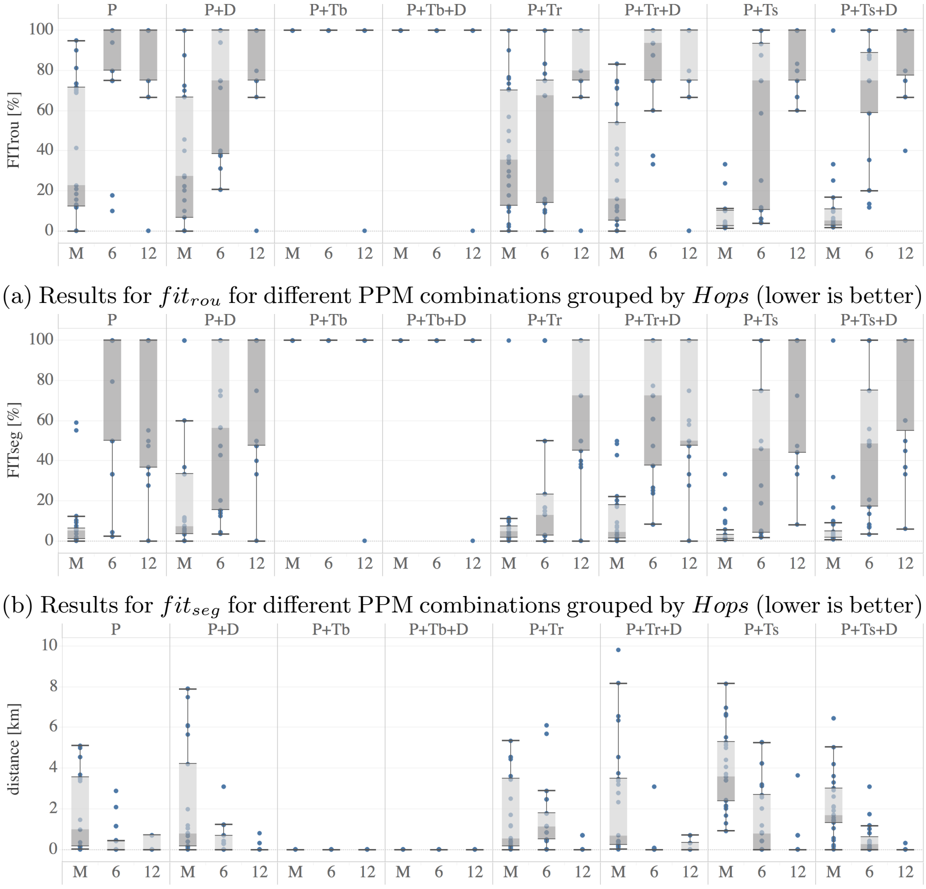
fit_{rou} and fit_{seg} Figure 4a shows the box plot for the metric fit_{rou} for different combinations of PPMs and $Hops \in \{6, 12, MIXED\}$ sizes, whereas Figure 4b illustrates the results for fit_{seg}.

First, fit_{rou} and fit_{seg} have a value of 100% for the combinations $DIRECT$, $P+Tb$ and $P+Tb+D$, i.e. an adversary is able to reconstruct the full route in these cases without having a single outlier. Thus, there is no protection applying any combination which uses time shifting in batch mode. This seems reasonable since batch mode apparently eliminates all benefits of using route parts.

Regarding the $Hops$ parameter, it can be stated that $MIXED$ performs best, followed by $Hops = 6$ and $Hops = 12$, independently from the used PPMs. This is especially pleasant, considering the little distance overhead added to a route when $Hops = MIXED$ is used (c.f. Figure 3). Furthermore, $Hops = 6$ creates more route parts compared to $Hops = 12$. Therefore, the adversary has to do more work to relink all parts, providing more privacy.

We also analysed how combinations of PPMs affect the user's privacy. It can be stated, that P increases the privacy with every $Hops$ parameter. Between the timeshifting modes, timeslot (Ts) performs best, followed by random (Tr) and trailed by the ineffective batch mode (Tb). Interestingly, the performance in $MIXED$ seems to lightly suffer from using dummy traffic (D) across the board.

Last, fig_{rou} and fit_{seg} show the same tendencies, although the spread of values for fit_{seg} is generally larger. This seems reasonable, since reconstructing a route in the correct order tends to be harder than finding used route segments.



(a) Results for fit_{rou} for different PPM combinations grouped by $Hops$ (lower is better)

(b) Results for fit_{seg} for different PPM combinations grouped by $Hops$ (lower is better)

(c) $dist$ in km between the real and guessed start and target vertices (higher is better)

Fig. 4: The figures show the protection performance of our algorithm against the described inference model w.r.t our metrics (M is short for $MIXED$).

dist Figure 4c shows the box plot for the distance metric for the different combinations of PPMs and $Hops$ size. Since an adversary is able to reconstruct the full route for $DIRECT$, $P+Tb$ and $P+D+Tb$, it is obvious that $dist$ is zero and that there is no difference between v_S and v_T and the guessed locations.

Overall, $dist$ follows the same trend as fig_{rou} and fit_{seg}: $MIXED$ yielded the highest distance across the board, i.e. an adversary guesses very distant points when reconstructing start and target. On the other hand, $dist$ is higher for $P+D$ than for P in contrast to fig_{rou} and fit_{seg} values.

It can further be seen that every combination of Ts with $Hops = MIXED$ shows the best results, even though D decreases $dist$.

		Mean	Min	Max
Time [ms]	Roundtrip	185	162	500
Size [byte]	Request	296	296	296
	Response	2,765	997	5,263

Table 1: Overview of 100 route requests to the Google API.

6.5 Performance Analysis

We also analysed the performance of our algorithm w.r.t. runtime and data size of a request. Because of its easy-to-use API and the overall quality of service, Google Maps was chosen as the LBS for our performance measurement. All traffic to Google Maps is end-to-end encrypted, hence we used a man-in-the-middle proxy to gain access to the data. For the sake of reproducibility, we did not simulate connection resets or data loss which can occur in mobile networks.

Runtime Table 1 shows the duration in milliseconds for a query to Google Maps. The mean duration is 185 ms per query (min 162 ms, max 500 ms). The bandwidth used is about 20 KB/s, a value easily achievable on mobile networks.

Executing requests in parallel does not seem to impact the duration. This is important since it proves that sending dummy requests does not significantly influence the performance and user experience. Furthermore, the time Google Maps needs to find a route is not significantly affected by the length of a route segment. In addition, we were unable to measure any difference regarding the time between real requests and dummy requests. This is obvious because both kinds of requests are using the same API calls. We thus assume that an LBS cannot distinguish real and dummy requests on a time basis.

Data size In a next step, we analysed the data size of requests for the different combinations of PPMs. It is obvious that *DIRECT* uses the least amount of traffic. However, using route parts has almost the same data size. This seems reasonable since a direct route has a similar amount of routing instructions like the same route constructed from combining different route parts (i.e. number of navigation instructions). There is a slight amount of overhead since every Google Maps API call provides additional metadata. Table 1 shows that each request has the same size since it only contains start and target vertices encoded as coordinates. However, the resulting response differs in size due to a different number of instructions.

It is interesting that the growth of data is linear, i.e. every additional request is roughly the same in size. Arguing that an average route request needs about 3 KB, a route has 4 segments and 2 more dummy requests are sent, *PARTS* requires acceptable 24 KB per privacy enhanced route request in addition.

We skipped the analysis of performance figures such as CPU and RAM usage because our proof of concept implementation is running on a desktop client.

For future work, it is planned to create an Android application to measure performance figures on a real mobile device as well.

7 Discussion

One can observe that by using route parts, the overall privacy increases. However, the route part approach yields more privacy, as more requests of different users overlap. The required level of simultaneous requests can also be achieved by generating dummy traffic. One can see that the combination of route parts with dummy traffic always performs better than solely using route parts.

Another finding regarding dummy traffic is that it is sometimes easy for an adversary to filter. This fact may be connected to the number of requests at the same time and should improve if more users use the system and have overlapping route requests. Ideally, one dummy request from a user could be a real request from another user. It may also help to choose dummy target locations and run *PARTS* in parallel for both the real and dummy locations instead of creating artificial requests per iteration.

In addition, a constant $Hops$ parameter adds a static component to *PARTS* detectable by an adversary. Thus, it performs worse than $MIXED$ mode. This setting uses different $Hops$ parameters for each iteration and therefore weakens the adversary's ability to use behaviour knowledge by diluting his data pool.

A surprising result is that by sending route requests in batch mode, there is no privacy at all, regardless if dummy traffic is included or not. Hence, it is easy for an adversary to filter dummy requests and combine the different route segments just by comparing start and target vertices of each request since this indicates a chronological order. The attack is independent from the number of users in the system because it is very unrealistic that two requests from different users occur at the exact same time.

8 Conclusion

We presented the routing algorithm *PARTS* which protects a user's movement pattern by splitting each route request into several route segments without revealing the real start and target locations v_S and v_T of the overall route. In this way it is possible to combine local offline knowledge, such as a geographical layout, with global online real-time data, like traffic information.

A simulation further emphasised the need for such an algorithm and revealed that it is trivial for an adversary to derive a movement pattern for a user. Our simulation has shown that route parts can provide additional privacy but need to be combined with further PPMs to achieve their full potential. Therefore we extended our algorithm to use dummy traffic and time shifting. It was shown that the application of time shifting with timeslots was very powerful to protect a user's privacy in almost every case. The usage of *MIXED* mode for the $Hops$ parameter provides a good overall user experience, since it offers a very high privacy level and produces a reasonable distance overhead compared to the ideal route. In general, the *PARTS* algorithm has no significant influence to the user experience in terms of performance and data consumption.

For future work, we plan to include semantic background information in our scenario. On the one hand, we want to strengthen our adversary with these capabilities. On the other hand, we want to improve the way dummy traffic is constructed since our experiments have identified that randomly generated dummy traffic is not that powerful. Furthermore, we plan to evaluate *PARTS* against real-world datasets such as Microsoft Geolife [12] to prove its feasibility on a daily usage. In addition, we want to implement the algorithm as a mobile application to further elaborate its usability.

References

1. Ağır, B., Huguenin, K., Hengartner, U., Hubaux, J.P.: On the privacy implications of location semantics. Proceedings on Privacy Enhancing Technologies 2016(4), 1 (2016)
2. Bindschaedler, V., Shokri, R.: Synthesizing plausible privacy-preserving location traces. In: Institute of Electrical and Electronics Engineers (ed.) 2016 IEEE Symposium on Security and Privacy (SP). pp. 546–563. IEEE (2016)
3. Golle, P., Partridge, K.: On the anonymity of home/work location pairs. In: Tokuda, H., Beigl, M., Friday, A., Brush, A.J.B., Tobe, Y. (eds.) Pervasive computing, pp. 390–397. Lecture Notes in Computer Science, Springer, Berlin and Heidelberg (2009)
4. Hughes, N.: Inside ios 9: Apple's maps app gets smarter with automatic directions based on user habits. Apple Insider (2015)
5. Kramer, A.D.I., Guillory, J.E., Hancock, J.T.: Experimental evidence of massive-scale emotional contagion through social networks. Proceedings of the National Academy of Sciences 111(24), 8788–8790 (2014)
6. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: L -diversity: Privacy beyond k -anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) 1(1), 3 (2007)
7. Michalevsky, Y., Schulman, A., Veerapandian, G.A., Boneh, D., Nakibly, G.: Powerspy: Location tracking using mobile device power analysis. In: 24th USENIX Security Symposium (USENIX Security 15). pp. 785–800 (2015)
8. Palanisamy, B., Liu, L.: Mobimix: Protecting location privacy with mix-zones over road networks. In: IEEE 27th International Conference on Data Engineering (ICDE), 2011. pp. 494–505. IEEE, Piscataway, NJ (2011)
9. Shokri, R., Troncoso, C., Diaz, C., Freudiger, J., Hubaux, J.P.: Unraveling an old cloak: k-anonymity for location privacy. In: Al-Shaer, E., Frikken, K. (eds.) Proceedings of the 9th annual ACM workshop on Privacy in the electronic society. p. 115. ACM, New York, NY (2010)
10. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(05), 557–570 (2002)
11. Wang, T., Liu, L.: Privacy-aware mobile services over road networks. Proceedings of the VLDB Endowment 2(1), 1042–1053 (2009)
12. Zheng, Y., Xie, X., Ma, W.Y.: Geolife: A collaborative social networking service among user, location and trajectory. IEEE Data(base) Engineering Bulletin 33, 32–39 (2010)