TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 05/09 pp743–757
DOI: 10.26599/TST.2020.9010010
Volume 25, Number 6, December 2020

Preserving Personalized Location Privacy in Ride-Hailing Service

Youssef Khazbak*, Jingyao Fan, Sencun Zhu, and Guohong Cao

Abstract: Ride-hailing service has become a popular means of transportation due to its convenience and low cost. However, it also raises privacy concerns. Since riders' mobility information including the pick-up and drop-off location is tracked, the service provider can infer sensitive information about the riders such as where they live and work. To address these concerns, we propose location privacy preserving techniques that efficiently match riders and drivers while preserving riders' location privacy. We first propose a baseline solution that allows a rider to select the driver who is the closest to his pick-up location. However, with some side information, the service provider can launch location inference attacks. To overcome these attacks, we propose an enhanced scheme that allows a rider to specify his privacy preference. Novel techniques are designed to preserve rider's personalized privacy with limited loss of matching accuracy. Through trace-driven simulations, we compare our enhanced privacy preserving solution to existing work. Evaluation results show that our solution provides much better ride matching results that are close to the optimal solution, while preserving personalized location privacy for riders.

Key words: ride-hailing; location privacy; privacy preserving; Voronoi diagram

1 Introduction

Ride-hailing services, such as Uber and Lyft, have enabled drivers to offer rides using their own vehicles and have enabled riders to request and hail rides with their mobile devices. Millions of people enjoy such service due to its convenience and low cost. However, the current ride-hailing service significantly threatens riders' location privacy. Since rider mobility, including the pick-up and drop-off location information, is tracked, a Service Provider (SP) can infer sensitive information about the riders such as where they live and work[2]. Many real world incidents have been reported about the misuse of such data. For example, in November 2014, Uber investigated one employee who was reported to have tracked riders[3]. If this location privacy issue is not addressed, many users may not be willing to use such service despite its popularity.

Location privacy has been well studied in the literature, and researchers have proposed many location obfuscation mechanisms, such as location perturbation[4], spatial cloaking[5,6], dummy location generation[7,8], etc. In location perturbation, noise is added to locations to generate obfuscated locations. In spatial cloaking, a user reduces the granularity of his location so that his location can be hidden inside a cloaked region. The dummy location generation technique generates $k-1$ properly selected dummy locations to hide the user's actual location. These techniques increase users' location privacy as it would be more difficult for an adversary to know the actual locations of the users. However, these techniques cannot be directly applied to the current ride-hailing systems without affecting the system usability and the system performance.

Recently, researchers have looked into privacy issues

• Youssef Khazbak, Sencun Zhu, and Guohong Cao are with the Department of Computer Science and Engineering, The Pennsylvania State University, University Park, Philadelphia, PA 16802, USA. E-mail: ymk111@psu.edu; sxz16@psu.edu; gxc27@psu.edu.
• Jingyao Fan is with Amazon Corp., Seattle, WA 98121, USA. E-mail: jxf376@psu.edu.
• A part of this paper has been presented at 2018 IEEE Conference on Communications and Network Security (IEEE CNS 2018)[1].
∗ To whom correspondence should be addressed.
Manuscript received: 2020-03-16; accepted: 2020-03-19

in ride-hailing services. PrivateRide[9] is the first system that aims to enhance location privacy for riders. It relies on spatial cloaking algorithms to obfuscate locations of riders and drivers by replacing their actual locations with cloaked regions. Then, in order to match a rider to a driver, the SP measures the distances between the rider's pick-up location and the drivers' locations. Based on the calculated distances, the rider is assigned to the driver who is the closest to his cloaked region. Since the ride matching is based on the cloaked regions, instead of the actual locations, the chosen driver may not be the optimal one and has to drive extra distance for rider pickup, and the rider may wait extra time to get the ride.

In this paper, we propose location privacy preserving techniques that efficiently match riders and drivers while preserving riders' location privacy. We first propose a baseline solution that allows a rider to select the driver who is the closest to his pick-up location without revealing his location to the SP. However, with side information such as knowledge of ride matching and temporal cloaking techniques deployed in the baseline solution, the SP can launch location inference attacks. To overcome these attacks, we propose an enhanced scheme that allows a rider to specify his privacy preference. In this solution, the ride matching algorithm selects a set of drivers that are as close to the rider's location as possible, and meanwhile located within an area that meets rider's privacy preference. Then, the rider selects a driver among the set of drivers. The pick-up and drop-off times are further obfuscated in a way to prevent the SP from using the temporal information to improve the inference of the rider's location.

In summary, the main contributions of this paper are as follows:

(1) We present a baseline privacy preserving solution that protects locations of riders in ride-hailing services, and show potential inference attacks against it.

(2) To overcome the inference attacks against the baseline solution, we propose an enhanced privacy preserving solution which provides personalized riders' location privacy. It relies on novel obfuscation techniques that satisfy riders' privacy requirements without affecting the convenience of the service, and with limited loss in ride matching accuracy.

(3) We analyze a real dataset that contains 60 000 rides. Experimental results show that our enhanced solution outperforms existing solutions by achieving much better matching accuracy with negligible computational overhead.

The rest of the paper is organized as follows. In Section 2, we describe the related work. Section 3 presents the preliminaries, and Section 4 describes the baseline privacy preserving solution. We present the enhanced solution in Section 5, and present the evaluation results in Section 6. Section 7 concludes the paper.

2 Related Work

Privacy preserving ride-sharing service has received considerable attention[10–17]. For example, Friginal et al.[10] proposed a solution for protecting location privacy in dynamic ride-sharing services. They considered a distributed architecture which allows users' location information to be scattered around the network, hence preventing the SP from collecting and storing sensitive location information from ride-sharing users. PrivatePool[18] is another distributed ride-sharing solution which guarantees location privacy. It includes ride matching protocol based on users' proximity and trajectories overlaps. However, it constructs a threshold private set intersection which incurs high computation overhead. Priv-2SP-SP[14] allows a rider and driver to compute efficient pick-up and drop-off locations that match their constraints, without revealing their origins and destinations. However, this solution does not address the ride matching problem, and hence running the algorithm between every pair of driver/rider incurs high computation overhead.

SRide[15] is a dynamic ride-sharing service that provides privacy-preserving ride matching. The ride matching is done securely by using homomorphic secret sharing and secure two-party equality test, in order to compute feasible ride matches and calculate ride-sharing score for each feasible match. Drivers are assigned securely to riders based on their ride-sharing scores. Ni et al.[12] proposed a protocol to solve the contradiction between safety and privacy preservation of riders and drivers.

However, these proposed privacy preserving solutions are for ride-sharing which is different from ride-hailing. Ride-sharing services enable multiple individuals with similar trip schedules to share a single car along their route. On the other hand, ride-hailing services allow users to use their own cars as taxies, so they can pick up and drop off riders at their specified pick-up and drop-off locations.

Recently, researchers started to look into privacy issues in ride-hailing services. PrivateRide[9] is the first system that aims to enhance location privacy for riders.

It relies on spatial cloaking algorithms to obfuscate locations of riders and drivers by replacing their actual locations with cloaked regions. In order to match a rider to a driver, the SP measures the distances between the rider's pick-up location and the driver's locations. Based on the calculated distances, the rider is assigned to the driver who is the closest to his cloaked region. Since the ride matching is based on the cloaked regions, instead of the actual locations, the chosen driver may not be the optimal and has to drive extra distance for rider pickup, and the rider may wait extra time to get the ride. Moreover, the fare calculation is based on the cloaked regions, and then a rider can be either undercharged or overcharged for a ride. As another solution, ORide[19] relies on homomorphic encryption to encrypt riders' and drivers' locations before sending them to the SP. Then the SP computes the encrypted distances based on the encrypted locations and returns them to the rider. The rider decrypts the distance and selects the closest driver. Although ORide achieves better matching results compared to PrivateRide, its cryptographic solution incurs much higher computation overhead. Moreover, this solution encrypts drivers' locations; as a result, the SP does not know the locations of the drivers. Hence, this model is not compatible with the current ride-hailing services and not incrementally deployable. In addition, ORide computes distance between a rider and driver using the Euclidean distance, while the actual distance depends on roads connecting them, as drivers are constrained to move along the roads. To provide more accurate ride matching results, pRide[20] matches riders to drivers based on road network distances. However, pRide relies on homomorphic encryption and hence it incurs high computation overhead. Zhao et al.[21] studies privacy issues with drivers and presents attacks that can determine driver's sensitive information, including their locations and daily driving behaviors. The work relies on Application Programming Interface (API) reverse engineering techniques that enable attackers to extract nearby driver's information from the SP through the mobile app. Unlike this work, our solution focuses on protecting rider's location privacy.

The proposed location privacy preserving solution protects riders' pick-up and drop-off locations to overcome inference attacks that aim to infer where riders live and work. The preliminary version of this work appeared in Ref. [1]. To protect the full mobility traces, some researchers have proposed anonymization techniques that obfuscate the whole users' mobility traces[22, 23]. However, they can not be directly adapted to the ride-hailing system. Moreover, existing work[24, 25] highlights the privacy vulnerability of anonymized traces by exploiting users' side information such as users' observed locations and co-locations.

3 Preliminary

In this section, we introduce ride-hailing services, the security model, the design goals, and the basic concept of Voronoi diagram which is an essential part of our solutions.

3.1 Ride-hailing services

The ride-hailing service involves three parties: riders, drivers, and an SP. Riders are typically smartphone users who need to hail a ride. Drivers are car owners who are willing to offer rides. The SP receives ride requests from riders and matches the requests with available drivers.

Ride matching is primarily based on the locations of the rider and drivers. It is initiated when a rider sends a ride request to the SP that includes his pick-up and optionally drop-off locations. Then the SP selects among available drivers the closest driver to the rider's pickup location. To do so, drivers have to continuously report their locations to the SP.

After matching a driver to a ride request, the SP allows the driver and rider to coordinate the ride by sending each party the information of the other, i.e., name, reputation, and phone number. If both parties accept the ride, the SP continuously shares the driver's location with the rider. Once the driver picks up the rider, he notifies the SP with the start of the ride. During the ride, the driver continuously updates the SP with his actual location. At the end of the ride, the driver provides the SP with the distance traveled and the trip duration. The SP uses this information to calculate the ride's fare. The SP first charges the rider, then it deposits to the driver account an amount that equals the amount charged to the rider minus the service fees. Finally, the driver becomes available again to offer a new ride.

3.2 Security model

We assume the SP is an honest but curious adversary who has the incentive to track riders' precise locations. The SP can use collected location traces of the riders to profile and infer sensitive information about them, improve its own service, or to sell the collected data to other third parties (e.g., advertisement agencies). However, the SP has no incentive to attack the riders' and drivers' mobile devices by providing them with malicious application. Because such malicious behavior can be detected via reverse engineering the application, and hence harms its reputation and leads to business loss over its competitors.

In the system, drivers continuously update the SP with their actual locations, because they are workers who provide ride offers to riders and in return receive money for such a service. We also assume that drivers do not disclose riders' locations to the SP, because drivers are normally independent workers who do not have the incentive to collude with the SP.

3.3 Design goals

We have the following design goals for ride-hailing services:

(1) Preserving rider's personalized location privacy: The goal is to provide ride-hailing service without disclosing the riders' actual locations to the SP. Moreover, the service has to satisfy the riders' needs for personalized location privacy throughout the operation of the service.

(2) Maximizing the accuracy of ride matching: If privacy is not considered, ride matching matches a rider with the closest driver, since it minimizes the riders' waiting time and the drivers' driving distance. However, such matching will require a rider to reveal his accurate location information and sacrifice privacy. When considering privacy, the design goal is to select a driver who is as close as possible to the rider's pick-up location while preserving his personalized location privacy.

3.4 Voronoi diagram

In this paper, Voronoi diagram is leveraged for launching location inference attacks, and our enhanced privacy preserving solution addresses such location inference attacks by constructing Voronoi diagrams.

The Voronoi diagram[26,27] is a data structure in the field of computational geometry that represents the proximity information about a set of nodes. It partitions the space with a set of nodes into polygons such that each polygon contains exactly one node. Every point inside a given polygon is closer to the node inside this polygon than to any other nodes. Figure 1 shows a Voronoi diagram and its dual graph represented by the dashed lines. The dual graph of a plane graph G has a vertex for each face of G and an edge for every two faces of G that are separated by an edge. Each vertex lies inside one Voronoi polygon, and each edge

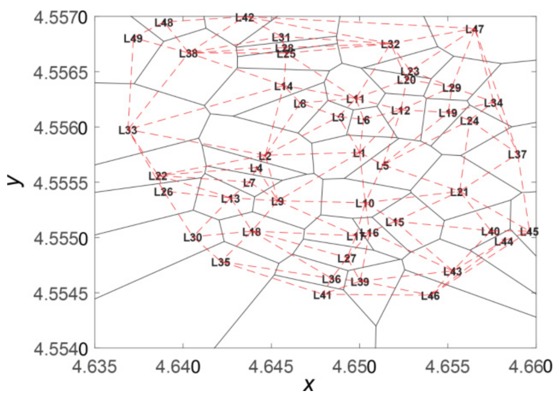


Fig. 1 Voronoi diagram of a set of drivers' locations. The dashed lines represent the dual graph for the Voronoi diagram of a set of drivers' locations, where the horizontal and vertical represent the coordinates of an area.

connects two vertices in two Voronoi polygons. In our context, Fig. 1 shows the Voronoi diagram of a set of drivers' locations. Each driver location is enclosed by a Voronoi polygon. Locations inside one polygon are closer to the driver inside this polygon than any other drivers. In the following sections, we will illustrate how Voronoi diagram can be leveraged for launching location inference attacks and how to defend against such attacks.

4 Baseline Privacy Preserving Ride-Hailing

In this section, we first introduce the baseline privacy preserving ride-hailing solution, and then explain why it suffers from location inference attacks.

4.1 System overview

Our system consists of three parties: riders, drivers, and SP, as illustrated in Fig. 2. The SP handles the incoming ride requests from riders and matches such requests to drivers who can serve them. When the SP receives a ride request which includes the rider's geographical region and the optionally obfuscated drop-off location, it constructs and sends the rider a list of drivers close to the rider, based on the provided geographical region. The rider receives the list, picks

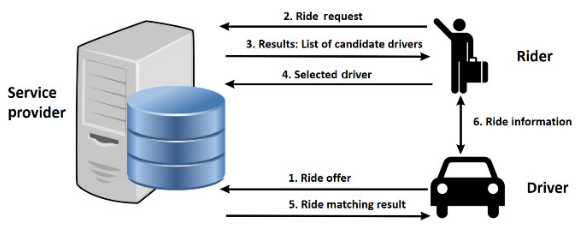


Fig. 2 System model.

a driver, and notifies the SP with the selected driver. The SP allows the driver and rider to coordinate the ride through direct communication. Through the established communication channel, they can share sensitive information such as actual pick-up location, drop-off location, and obfuscated locations. In addition, the driver continuously shares his actual location updates while heading to the pick-up location.

On reaching the cloaked region where the rider is located, he updates the SP with the same cloaked region of the rider. Then, the driver picks up the rider, and notifies the SP with an obfuscated pick-up time of the ride. During the ride the driver continuously sends SP the cloaked region until moving out of it. Afterwards, the driver continuously sends his actual location to the SP. After the end of the ride, the driver notifies the SP with an obfuscated drop-off time of the ride and becomes available again to offer a new ride. Next, we describe the details of the baseline solution, which includes three components: ride initiation, ride matching, and temporal cloaking.

4.1.1 Ride initiation

A rider initiates a ride request to the SP as follows. He sends a geographical region, denoted as Q, which contains his actual pick-up location and its cloaking region. As shown in Fig. 3, to construct Q, the rider specifies his privacy preference as a cloaking region[28], where his location is indistinguishable from other locations inside the region. Specifically, the rider specifies the cloaking region as a square of an area equals S^2 centered at the actual location L. Then a location, denoted as L', is chosen uniformly at random from points inside the cloaking region. Q is generated centering at L' with a diagonal $2R$. To ensure that the cloaking region lies inside Q, R should be greater than or equal to the diagonal of the cloaking region, i.e., $R \geqslant \sqrt{2}S$. Using the same method, the rider can generate an obfuscated drop-off location. Afterwards, the rider sends the ride request with the geographical

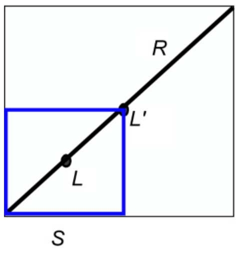


Fig. 3 Construction of geographical region *Q*.

area Q to the SP. Once the SP receives the ride request, it uses Q to filter out all drivers that are irrelevant to the rider's pick-up location and sends to the rider a list of drivers' locations in Q.

4.1.2 Ride matching

Following the ride initiation process, the SP sends the rider a set of candidate drivers' locations which are within the geographical region Q. Then, the rider selects the closest driver based on his location and notifies the SP with the selected driver.

4.1.3 Temporal cloaking

The cloaking region protects rider's location privacy from the spatial perspective. Similarly, cloaking can also be done from the temporal perspective. To obfuscate the pick-up and drop-off times, the SP discretizes the time of the day into time intervals. Then, a driver reports to the SP the time interval where an event occurs. For example, if the time interval is 2 minutes, an event such as rider pick-up occurring between 0:10 and 0:12 would be reported to the SP at 0:12.

4.2 Location inference attacks

In this subsection, we first show how the SP can launch a location inference attack with the knowledge of ride matching. Then, we show another location inference attack which exploits knowledge of temporal cloaking along with information about driver's car speed to improve the inference of riders' locations.

4.2.1 Location inference attack with knowledge of ride matching

In the baseline approach, a rider selects the driver closest to his location. This gives the SP an opportunity to launch location inference attacks. The SP knows all drivers' locations and the selected driver location, and it knows that the rider's location is closest to the selected driver's location than any other driver's location. The SP can launch an attack by constructing a Voronoi diagram based on the drivers' locations. Every driver's location is enclosed by a Voronoi polygon which consists of all locations closer to that location than any other driver's location. By using the Voronoi diagram, the SP can infer the rider's location is within the Voronoi polygon of the selected driver, denoted by P_{D*}. When the cloaking region is larger than P_{D*}, the SP can reduce rider's cloaking region into a smaller region represented by P_{D*}.

Figure 4 describes the location inference attack. The "$\circ$" symbol represents the drivers' locations, and the

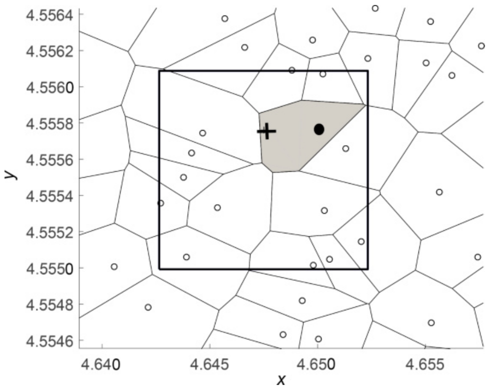


Fig. 4 Location inference attack with knowledge of ride matching, where the horizontal and vertical represent the coordinates of an area.

constructed Voronoi diagram partitions the region based on the distance to the drivers' locations. The square represents the cloaking region of size $A_R = S^2$. The gray polygon represents the Voronoi polygon P_D^* of the closest driver. The "•" symbol represents the location of the driver closest to the rider's pick-up location. The pick-up location is represented by a "+" symbol. Since the rider chooses the closest driver to his location, the SP infers that the rider's actual location lies within P_{D^*}.

To quantify the effect of the attack on location privacy, we introduce the following notations. Let O denote the cloaking region which is divided into a grid of K cells. Each location is represented by the cell it falls into. The probability of being in a cell i is denoted by p_i. Next, we quantify the effects of the attack using a metric called *Leaked Information*, which is defined as follows.

$$\textit{Leaked Information} = H(O) - H(O|P_{D^*}) \tag{1}$$

where $H(O) = \sum_{i=1}^{R} p_i \log \frac{1}{p_i}$. $H(\cdot)$ denotes the entropy of a random variable[29], where p_i represents the probability of the rider being located inside cell i. $H(O)$ evaluates the SP's prior knowledge about rider's location and it is the entropy of the prior probability. We assume that the SP does not have any side information before the attack and the best he can do is to reveal O. Hence, the actual rider's location can be within any cell inside O with probability equal to $\frac{1}{K}$. Here, the prior probability is represented by a uniform distribution which provides the maximum entropy before launching the attack, and hence $H(O) = \log K$. $H(O|P_{D^*})$ measures the SP's posterior knowledge after launching the attack and it is the entropy of the posterior probability. Hence, the privacy leakage is measured in terms of the change in SP's knowledge. Next, we quantify the attack using another metric called *Disclosed Area*, which is defined as follows:

$$\textit{Disclosed Area} = \frac{A_R - A_p}{A_R} \tag{2}$$

The *Disclosed Area* metric measures the proportion of the area of the cloaking region disclosed after running the attack. A_R denotes the size of the cloaking region and A_p denotes the size of P_{D^*}.

Using these two metrics, we quantify the effects of the attack based on a dataset consisting of mobility traces of taxi cabs collected in Shanghai, China[30] (more details in Section 6). Figures 5 and 6 show the cumulative distribution functions of the *Leaked Information* and *Disclosed Area*, respectively. It can be seen that both *Leaked Information* and *Disclosed Area* increase as S increases. This is because more drivers can be located inside O when A_R is larger. With many drivers inside O, many constructed Voronoi polygons lie inside O, and the SP can infer that the rider's location is only inside one of them which is P_{D^*}. As shown in Fig. 6, even with small O, a square with $S = 250$ m, only around 20% cases do not disclose any information about O. For cloaking regions with larger A_R, privacy breach is nearly inevitable. For example, when $S = 1000$ m, in 80% cases, the disclosed area is larger than 75%.

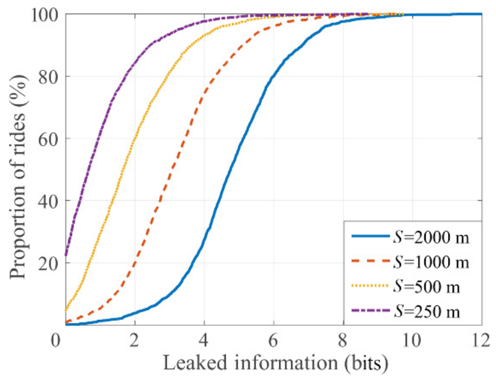


Fig. 5 ***Leaked Information*** **for the location inference attack.**

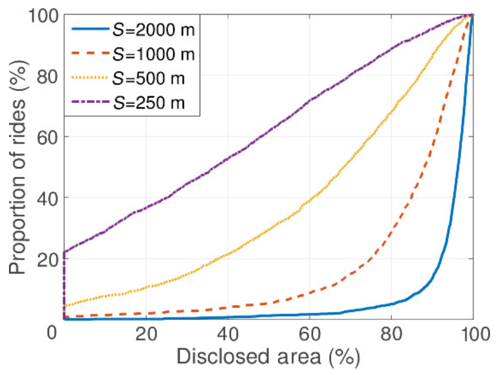


Fig. 6 ***Disclosed Area*** **for the location inference attack.**

4.2.2 Location inference attack with knowledge of temporal cloaking

The temporal cloaking technique does not prevent the SP from exploiting temporal information along with side information such as driver's car speed to infer more precise locations. The inference attack is described as follows. Let v denote the maximum driver speed, and T denote the time difference between the time of the last known driver's location and the obfuscated time for pick-up. Let the maximum distance between any two points inside the cloaking region P_{D^*} be $d_{\max}$. Then the SP may prune part of O by computing the maximum distance a driver can travel when moving at speed v. The maximum travel distance equals vT. Then the attack is successful if $d_{\max} > 2vT$.

As shown in Fig. 7, when $d_{\max}$ of P_{D^*}, i.e., the gray polygon, is larger than the maximum travel distance a driver can travel, it indicates that part of P_{D^*} is unreachable by the driver at the pick-up time, and hence the actual pick-up location cannot lie within the unreachable region, i.e., the region shown by dashed lines. Hence, the SP can further reduce the cloaking region P_{D^*}. Similar attacks can be used to improve the inference of the drop-off location.

5 Enhanced Privacy Preserving Ride-Hailing

To overcome the location inference attacks in the baseline solution, we propose an enhanced solution for privacy preserving ride-hailing in this section. We first present our solution and then give its privacy analysis.

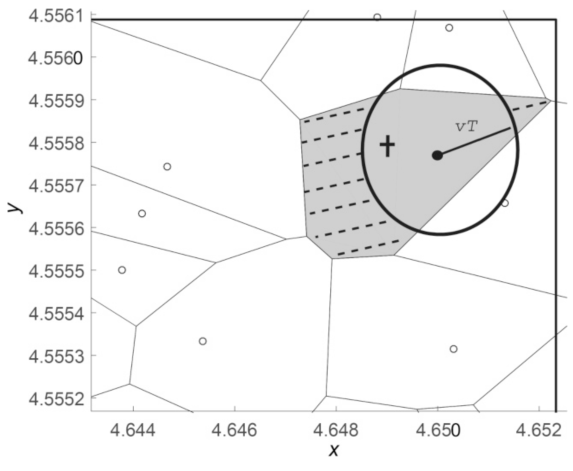


Fig. 7 Location inference attack with knowledge of temporal cloaking, where the horizontal and vertical represent the coordinates of an area.

5.1 Enhanced ride matching

Similar to the baseline solution, as shown in Fig. 2, after the ride initiation process, the SP sends a set of candidate drivers' locations within the geographical region Q to the rider. Then the rider runs the enhanced ride matching algorithm, which consists of two parts. The first part is a spatial cloaking algorithm that constructs a cloaking region based on drivers' locations and rider's privacy preference. The second part is a driver selection algorithm that picks a single driver among the set of drivers located within the cloaking region. The driver is chosen according to a probabilistic mechanism.

In the spatial cloaking algorithm, the actual pick-up location of a rider, denoted as L_R^p, is hidden inside a cloaking region. The rider specifies his privacy preference in the form of an area of size A_R. The privacy preference corresponds to hiding the rider's actual location inside an area of size at least A_R. With such privacy preference, the cloaking algorithm generates the cloaking region as follows. First, let $L_D = \{L_{D_1}, \dots, L_{D_n}\}$ denote the set of n drivers' locations provided by the SP. The algorithm constructs the Voronoi diagram and its dual graph from L_D. The Fortune's algorithm is used to generate the Voronoi diagram from the set of n drivers' locations in the region as follows. The algorithm maintains a sweep line and a beach line. The sweep line is a vertical straight line which moves from left to right as the algorithm progresses. The beach line is a piecewise curve to the left of the sweep line and composed of several parabolas. During the execution of the algorithm, at any point, the points to the left of the sweep line have been already included into the Voronoi diagram, while the other points to the right of the sweep line have not been considered yet. For every point to the left of the sweep line, a parabola of points equidistant from that point and from the sweep line is constructed. Then the beach line is the union of all the constructed parabolas. As the sweep line moves to the right, the vertices at which two parabolas cross form the edges of the Voronoi diagram.

The constructed Voronoi diagram divides the area into a set of Voronoi polygons represented by $\{P_1, \dots, P_n\}$. The dual graph of the Voronoi diagram corresponds to constructing a Delaunay triangulation[31] on the set L_D. To construct the Delaunay triangulation, we sort the set L_D of n drivers' locations in ascending order. Then we divide the set L_D into two subsets L_D^L and L_D^R, such that the points in L_D^L are lexicographically smaller than the points in L_D^R. Afterwards, we use the divide and conquer approach to recursively construct the Delaunay triangulations for the points in L_D^L and L_D^R. Then we merge the two solutions to obtain the final result. The merge is done by obtaining the union of the convex hulls of L_D^L and L_D^R. After obtaining the dual graph, we label each node L_{D_i} of the dual graph with two labels, which are the Euclidean distance between L_{D_i} and the L_R^p, denoted as $D_{D_i} = \text{dist}(L_{D_i}, L_R^p)$, and the area of the Voronoi polygon P_i, denoted as A_{P_i}.

The A_{P_i} is computed using the polygon triangulation method. The polygon triangulation method divides the polygonal area into a set of triangles. Then the area of the polygon is computed as the summation of the areas of all of the triangles within the polygonal area. A simple polygon is monotone with respect to a line, when any other line perpendicular to that line intersects the polygon at most twice. Assuming a polygon is monotone with respect to the y-axis, then a greedy algorithm can be used to scan the polygon from top to bottom while adding diagonals whenever it is possible. If a polygon is not monotone, it can first be partitioned into monotone sub-polygons using a sweep-line approach.

The enhanced ride matching problem can be formulated as follows.

Definition 1 Let the dual graph of the Voronoi diagram denote $G = (V, E)$, where $V = \{L_{D_1}, \ldots, L_{D_n}\}$ and each location L_{D_i} lies inside a Voronoi polygon P_i. Find a subgraph $G_S = (V_S, E_S)$, $E_S \subset E$ that satisfies the following: (1) the drivers' locations in V_S are as close as possible to the rider's location L_R^p; that is, for any driver's location $L_{D_i} \in V_S$ and driver's location $L_{D_j} \notin V_S$, $D_{D_i} \leqslant D_{D_j}$, (2) the area covered by the Voronoi polygons of the drivers in V_S, denoted as $A_T = A_{P_1} + \cdots + A_{P_{|V_S|}}$, must be equal to at least A_R.

The problem can be solved using a greedy algorithm. The intuition of the algorithm is to construct a subgraph that includes a set of drivers' locations by sequentially picking a driver's location that is closest to the rider's location until the size of the area A_T covered by the selected driver's polygons exceeds the required threshold A_R. Hence, the algorithm progressively expands the covered area by aggregating drivers' polygons until the privacy preference $A_R \leqslant A_T$ is satisfied.

To construct the subgraph, the algorithm traverses the dual graph G. However, the traversal of G cannot be done using a straight forward Breadth-First search. Because it is not always the case that immediate neighbor nodes are closer than faraway neighbors. For example, as shown in Fig. 8, the distance between a rider's location at node a and the location of its neighbor node c is larger than the distance between it and the location of its two-hop neighbor node b. Based on this observation, the algorithm uses a priority queue Q_S in traversing G and the traversal works as follows. Initially, V_S contains the location of the driver closest to the rider, i.e., the root node. Next, the algorithm explores and adds the neighbor nodes of the root node to Q_S. Then it picks a node among the neighbors from Q_S that has the smallest distance, inserts it into V_S, and explores and adds its neighbors to Q_S. Hence at every step, the algorithm chooses, among all nodes in Q_S, a node that is a neighbor to any node in V_S without favoring root's neighbors over others. The choice from neighbors is to preserve the connectivity of the cloaking region constructed by the drivers' polygons of V_S.

The second part of the enhanced ride matching is the driver selection algorithm, which uses a set S equivalent to V_S but ordered according to the distance from the root node. It divides S into two sets S_1 and S_2, such that S_1 contains locations that are closer to the rider's location when compared to locations in S_2. Then a single driver is selected among S_1 and S_2 with probability equal to $w \times p_i$ and p_i, respectively. The parameter w denotes a weight factor. Setting $w = 1$ is equivalent to selecting a driver uniformly at random from the two sets S_1 and S_2, every driver can be selected with probability $\dfrac{1}{|S|}$. Setting w to a value larger than one allows the driver selection algorithm to favor closer drivers. Finally, the parameter p_i is equal to $\dfrac{1}{w|S_1| + |S_2|}$.

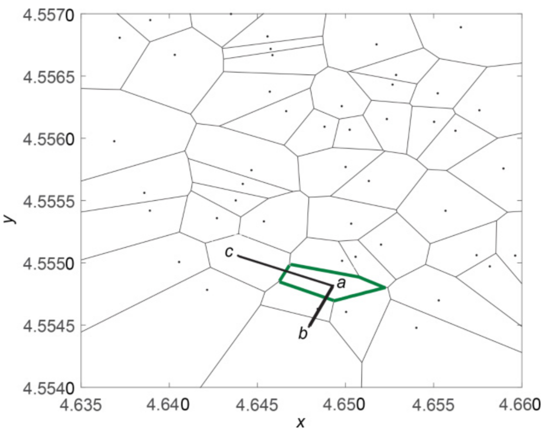


Fig. 8 Distance between a rider's location at node *a* and the location of its neighbor node *c* is larger than the distance between it and the location of its two-hop neighbor node *b*, where the horizontal and vertical represent the coordinates of an area.

Spatial cloaking and driver selection are summarized in Algorithm 1. We use a regular binary heap to implement the priority queue. The time complexity of the algorithm is as follows. Lines 1 and 2 construct the Voronoi diagram and its dual graph using the Fortune's algorithm, and the Delaunay triangulation. The Fortune's algorithm uses a binary search tree to describe the beach line, and a priority queue to store future events that may affect the beach line structure. The algorithm operates by progressively moving the sweep line to the right. At each step, the next event is retrieved from the priority queue, and the beach line is updated based on the changes caused by that event. As there are a total of $O(n)$ events, and each event requires $O(\log n)$ time, the total running time is $O(n \log n)$. The Delaunay triangulation algorithm is based on a divide and conquer approach in which a line is used to split the vertices into two sets. Then the algorithm computes the triangulation for each set, and the two sets are merged. The running time of such algorithm is $O(n \log n)$[32]. Lines 3 – 6 iterate on all nodes in V to label them, which

Algorithm 1 Enhanced ride matching

```
Input: Rider's pickup location $L_R^p$ and set of $n$ drivers' locations $L_D = \{L_{D_1}, \ldots, L_{D_n}\}$.
Output: An optimal driver location that satisfies the privacy requirement $A_R$.
1:  Construct Voronoi diagram from set $L_D$
2:  Construct the dual graph $G = (V, E)$ of the Voronoi diagram where $V = L_D$ and each location $L_{D_i}$ lies inside a Voronoi polygon $P_i$.
3:  for each node $L_{D_i}$ do
4:      Compute the Voronoi polygon area $A_{P_i}$
5:      Compute the distance $D_{D_i} = \text{dist}(L_{D_i}, L_R^p)$
6:      Label the node with $\{A_{P_i}, D_{D_i}\}$
7:  end for
8:  Set $V_S$ to the node $u$ with the smallest distance $D_{D_i}$ among all nodes in $V$
9:  Set $A_T$ to the area $A_{P_i}$ of the Voronoi polygon of $u$
10: Initialize priority queue $Q_S$ and push in $Q_S$ the children of $u$
11: while $A_T < A_R$ do
12:     Find and remove a node $v \in Q_S$ with the smallest distance among all nodes in $Q_S$
13:     Add to $A_T$ the area $A_{P_i}$ of the Voronoi polygon of $v$
14:     Add $v$ to $V_S$
15:     Push in $Q_S$ the children of $v$
16: end while
17: Set $S$ to $V_S$ and sort it in ascending order by distance
18: Divide $S$ into two sets $S_1$ and $S_2$
19: Pick a driver location $L_D^*$ from $S_1$ with probability $w \times p_i$ or from $S_2$ with probability $p_i$
20: return $L_D^*$
```

takes $O(n)$. The traversal of the dual graph in Lines 7 – 14 runs in time $O(|V_S|+|E_S|)$, where $|V_S| = O(n)$ and $|E_S| = O(n)$ as stated in the following lemma.

Lemma 1 Any planar graph on $n \geqslant 3$ vertices has at most $3n-6$ edges and at most $2n-4$ faces.

Proof In a maximal planar graph, no edges can be added to it without making it non-planar graph. All faces of a maximal planar graph are bounded by three edges, and each edge will be on the boundary of two faces. Let number of edges be e and number of faces be f, then $3f = 2e$. Substituting this into the Euler's formula, which states that for a connected planar graph $n = 2 + e - f$, then the number of edges is given by $e = 3n-6$, and the number of faces is given by $f = 2n-4$. Since any planar graph must have equal, or fewer edges than a maximal planar graph with the same number of vertices, then any planar graph will have $e \leqslant 3n-6$ and $f \leqslant 2n-4$. ■

Thus, the running time of the graph traversal is $O(n)$ and an iteration of it runs in $O(\log n)$ because each iteration consists of (1) finding and removing the node with the smallest distance among all nodes in Q_S, which requires $O(\log n)$, and (2) adding to Q_S the node's children, which requires $O(\log n)$. Therefore, Lines 7–14 require $O(n \log n)$ time. Lines 15 – 17 have the time complexity of $O(n \log n)$ for sorting V_S using merge sort algorithm. Hence, the time complexity of the greedy algorithm is $O(n \log n)$.

We use Fig. 9 to illustrate how our algorithm works. Figure 9 shows the constructed cloaking region, the selected driver by our algorithm, and one of the drivers that can be selected by PrivateRide[9]. V_S initially contains the location of the closest driver, and A_T equals to the area of the closest driver's polygon, represented by

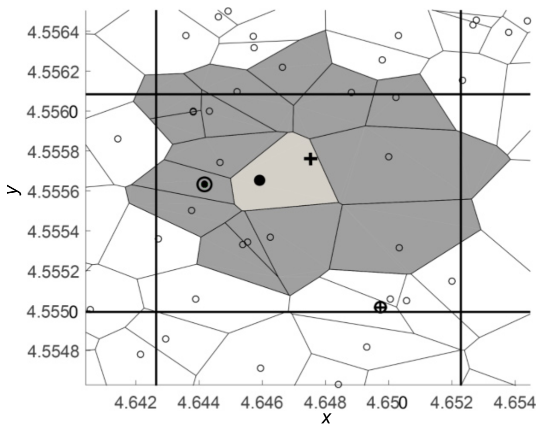


Fig. 9 Construction of the spatial cloaking region, where the horizontal and vertical represent the coordinates of an area.

the light gray polygon. The algorithm explores and adds the neighbor nodes of the root node, until it constructs the cloaking region that satisfies the privacy preference A_R. The cloaking region is represented by the light and dark gray regions. Then a single driver is selected among the set of drivers within this region. The driver selected by our algorithm is represented by a "$\odot$" symbol. The driver selected by PrivateRide is picked among the set of drivers inside the square region, i.e., the cloaking region represented by the square. Because PrivateRide considers all drivers co-located with the rider at the same cloaking region have a distance equals zero from that rider's location, it cannot distinguish between them. The driver represented by "$\oplus$" symbol, at the boundary of the cloaking region, can be selected by PrivateRide. The closest driver is represented by the "$\bullet$" symbol. As observed, our algorithm does not select the closest driver due to the privacy requirements, but it still selects a driver closer than that selected by PrivateRide. This is because PrivateRide may select the driver located in the boundary of the square region, while our algorithm uses nearby drivers to generate the cloaking region.

5.2 Enhanced temporal cloaking

We presented location inference attacks against the baseline temporal cloaking which obfuscates the actual pick-up and drop-off times into time intervals. It allows the SP to prune some parts of the cloaking region to launch attacks, by observing that a driver cannot reach them at his maximum speed. To protect against this attack, the temporal information should be obfuscated so that the possible travel distance of a driver within the given time covers the whole cloaking region.

Algorithm 2 shown the details of the enhanced temporal cloaking algorithm. In this algorithm, the cloaking region is represented by a set of points X, which represents the vertices of the Voronoi polygons lying inside the cloaking region. The SP with some statistic information can determine the maximum speed v of a driver within this region. In addition, the SP knows the time of the last actual location reported by the driver before entering the rider's cloaking region, denoted by t_d.

Algorithm 2 Enhanced temporal cloaking

```
Input: Cloaking region represented by a set of points $X$,
    maximum driver speed $v$, last known time where driver's
    actual location is reported $t_d$, and a random value $r$.
Output: An obfuscated time $t'$.
1: Compute the smallest enclosing circle for the set of points $X$
2: Set $d_{\max}$ to the circle diameter
3: Set $t$ to $\frac{d_{\max}}{v}$
4: Set $t'$ equals $t_d + t + r$
5: return $t'$
```

In Lines 1 and 2 of Algorithm 2, we use the randomized incremental algorithm proposed in Ref. [33] to compute the smallest enclosing circle of the points X. As the smallest enclosing circle has at least two points of X on its boundary, the maximum traveling distance between any two points inside the circle, denoted by $d_{\max}$, corresponds to the obtained circle diameter. In Line 3, t denotes the time delay which should be added to obfuscate the actual time and is computed according to $d_{\max}$. In Line 4, we set the obfuscated time. A random value r is added to the obfuscated time to prevent the SP from knowing $d_{\max}$ that reveals information about the cloaking region. The cloaking algorithm has a time complexity of $O(n)$.

5.3 Fare calculation

The SP computes the ride's fare by using the pick-up and drop-off cloaking regions, the reported driver's mobility trace outside the cloaking regions, and the duration of the ride. With the privacy preserving solution, the SP calculates the fare based on obfuscated distance and duration. Thus, we consider a pricing mechanism where protecting rider's location privacy comes at a premium, and the riders are willing to pay more to protect their location privacy. Then, a ride's distance can be computed based on the real distance traveled by the driver plus the maximum distance a driver can drive inside the pick-up/drop-off cloaking region.

In the enhanced ride matching, to preserve rider's location privacy at the drop-off location, a driver should not directly report his actual location to the SP after the ride ends. This is because when the driver's actual location is revealed after the drop-off, the SP can infer rider's drop-off location. One solution to protect the rider's drop-off location is to allow the driver to drive outside of the rider's cloaking region before reporting his actual location. Another solution is to allow the driver to wait for some time proportional to the cloaking region size before reporting the actual location. In both solutions, the driver would not be able to accept new rides for some time, which may adversely affect his business. To compensate the drivers, the riders who want to preserve their location privacy should pay extra amount proportional to the time the driver has to wait or the extra distance he has to drive out of the cloaking region. More specifically, we consider the maximum distance/time a driver can drive inside the drop-off cloaking region. The maximum distance inside a cloaking region is obtained using the perimeter of the cloaking region, and the ride duration is computed based on the obtained maximum distance.

5.4 Privacy analysis

To better understand the privacy level of our enhanced solution, we compare the probability of rider being located in different Voronoi polygons within the cloaking region O, after observing the chosen driver's location L_D^*. Let P_i and P_j be two Voronoi polygons that lie inside O, then the ideal case is that $\frac{\text{Prob}(P_i|L_D^*)}{\text{Prob}(P_j|L_D^*)} \leqslant w$. For $w = 1$, the SP may deduce that the rider has equal probability of being located in P_i or P_j . The SP may have some side information on the prior probability of the rider to be within each polygon, represented by $\text{Prob}(P_i)$ and $\text{Prob}(P_j)$. For example, the probability for the rider to be in a shopping area may be higher than the probability of being at a lake. Hence, our privacy preserving solution has the following property.

Theorem 1 The enhanced privacy preserving ride-hailing solution guarantees strong privacy for rider's actual location inside a cloaking region O if it satisfies the following for all prior probabilities and any chosen driver L_D^* from S.

$$\frac{\text{Prob}(P_i|L_D^*)}{\text{Prob}(P_j|L_D^*)} \leqslant w \cdot \frac{\text{Prob}(P_i)}{\text{Prob}(P_j)}, \quad \forall P_i, P_j \in O \tag{3}$$

Proof The proof is by contradiction. Assume it is safe for a rider to release the cloaking region O to the SP. Then the SP knows the locations of drivers inside O, and it also knows that the algorithm uses the location information of nearby drivers to generate O. It can deduce that the rider tends to be close to the center of the cloaking region. With such information, the SP can improve the inference of the rider's actual location as follows. For each driver's location L_{D_i}, it computes the distances between that location and all other drivers' locations, denoted by $D_i = \{\text{dist}_j(L_{D_i}, L_{D_j})\}_{\forall j \neq i}$, and obtains the variance of the computed distances, represented by $\text{Var}(D_i) = \text{E}[(D_i - \mu)^2]$. Then, the SP chooses the Voronoi polygon of the driver's location with the smallest variance, i.e., $\arg\min_i \text{Var}(D_i)$. The chosen polygon contains the rider's actual location with high probability, and thus the inference of the rider's location can be improved.

However, in our solution, a rider releases the chosen driver's location L_D^* only to the SP, and no more information is released about the cloaking region. We obtain a contradiction and thus, the described inference attack is not possible without knowing the cloaking region O. Hence, the observed L_D^* has limited effect on the probabilities deduced by the SP. ■

6 Performance Evaluations

In this section, we evaluate the performance of the enhanced privacy preserving solution and compare it to PrivateRide[9]. We first introduce the evaluation setup and then present the evaluation results.

6.1 Evaluation setup

The evaluations are based on a real dataset consisting of mobility traces of taxies collected during a single day in Shanghai, China[30]. The dataset includes time-stamped location traces collected using GPS devices. Each record includes an *OCCUPIED* variable that indicates whether the taxi is vacant or occupied. It is equal to 1 when the taxi is occupied and 0 otherwise. Hence, the ride records are a sequence of consecutive records where *OCCUPIED* equals 1.

A ride is defined by the start time, end time, pick-up location, drop-off location, ride time, and ride distance. The start time and the pick-up location are obtained from the first record of the ride records. The end time and the drop-off location are obtained from the last record of the ride records. The ride distance is calculated using the haversine formula[34], as the distance between the pick-up and drop-off location. The total number of rides in the dataset is 60 000. We filter out rides that have a total distance of zero or last for less than 5 minutes, and only consider rides inside an area of about 44 km × 44 km. Finally, we have 25 000 rides in our evaluations.

We compare our enhanced solution with PrivateRide by evaluating the accuracy of their ride matching. In the evaluation, our enhanced ride matching is referred to as *our algorithm*. The performance of both solutions is evaluated based on a metric called *Relative Extra Distance*, which measures the extra distance a driver has to drive compared to the distance covered by the closest driver to the rider's location. We set $R = 4000$ m if not mentioned otherwise.

6.2 Evaluation results

In this section, we present the evaluation results. For comparison purpose, we implemented the cloaking algorithm in PrivateRide. We set the area of the cloaking region to A_R. More specifically, if $A_R = 62\,500\ \text{m}^2$, the region is divided into square cells of $250\ \text{m} \times 250\ \text{m}$.

6.2.1 Effects of privacy preference (A_R) and weight factor (w) on the ride matching accuracy

We first evaluate the effect of A_R and w on ride matching accuracy. The ride matching accuracy is measured by the *Relative Extra Distance*, which shows the extra cost for both drivers and riders. As shown in Fig. 10, by imposing higher privacy preference with larger A_R, the matching accuracy decreases. In addition, an increase in w leads to an increase in the matching accuracy, and hence a decrease in the extra distance.

Compared to PrivateRide, our algorithm, under the same A_R, achieves higher matching accuracy. As A_R increases, the accuracy gap between our algorithm and PrivateRide increases, which shows that our algorithm outperforms PrivateRide. As shown in Fig. 10a, when $A_R = 2000\ \text{m} \times 2000\ \text{m}$, in 80% of the rides, the *Relative Extra Distance* of our algorithm under different ws is up to 831 m for $w = 1$, 760 m for $w = 2$, and 675 m for $w = 4$, while it is up to 1560 m in PrivateRide. In this case, comparing our algorithm with $w = 4$ to PrivateRide, the savings in the extra cost can be calculated as follows: $\dfrac{1560 - 675}{1560} \times 100\% = 56.7\%$. As shown in Fig. 10b, when $A_R = 1000\ \text{m} \times 1000\ \text{m}$, in 80% of the rides, the *Relative Extra Distance* of our algorithm with $w = 1$ is up to 380 m, but up to 666 m in PrivateRide.

6.2.2 Computational overhead

In this subsection, we evaluate the computational overhead of the algorithm. As shown in Table 1, for a rider, the computational overhead introduced by our algorithm varies based on R. It can be observed that the computation overhead of our algorithm is small. When R ranges from 500 to 2000 m, it is only about 1.12 to 2.64 ms. Even when R increases to 5000 m, the computation overhead is till only about 8 ms. In PrivateRide, riders receive a single driver selected by the SP, hence there is no computational overhead. In addition, with higher privacy requirement, R goes higher and the download time increases. This is because more drivers are located within the considered region, and hence more drivers' information is downloaded by the rider.

6.2.3 Communication overhead

We evaluate the communication overhead of the algorithm which occurs as riders have to download a list of candidate drivers from the SP. We set $A_R = 62\,500\ \text{m}^2$ and $w = 2$, and vary the query range R. As shown in Fig. 11, the list of drivers needed to be downloaded from the SP that depends on the query range R. The parameter R is controllable, however, it depends on the privacy requirement specified by the rider, i.e., the size of the cloaking region. For instance, if the rider wants to hide his actual location within an area of $A_R = 250\ \text{m} \times 250\ \text{m}$, the query range should be $R \geqslant 353.55\ \text{m}$. With higher privacy requirement, R goes higher and the communication overhead increases. With a reasonable cloaking region of size $A_R = 1000\ \text{m} \times 1000\ \text{m}$, on average, the rider

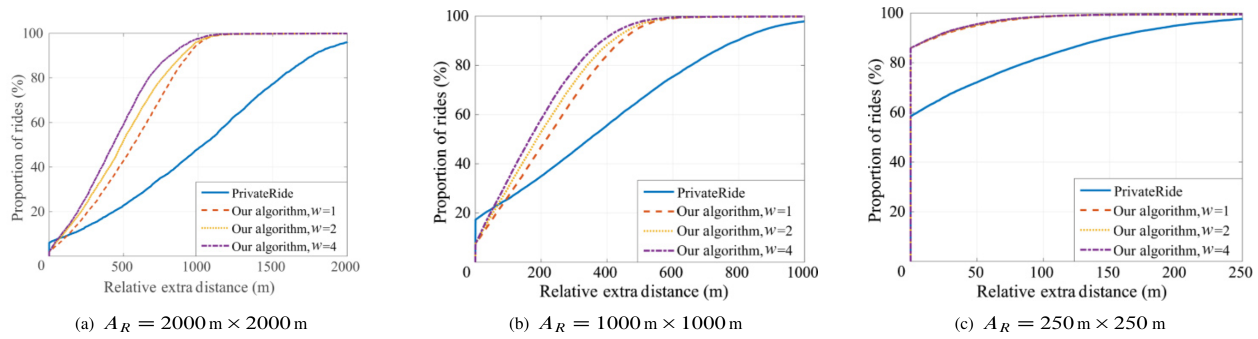


Fig. 10 Effect of A_R on ride matching accuracy (*Relative Extra Distance*).

Table 1 Computational time of our algorithm.

Setting (m)	Download time (ms)	Algorithm computational time (ms)	Upload time (ms)	Total time (ms)
$R = 500$	$(2.27 \pm 1.24) \times 10^{-5}$	1.12 ± 0.21	0.0143 ± 0.0076	1.15 ± 0.22
$R = 1000$	$(4.93 \pm 2.74) \times 10^{-5}$	1.41 ± 0.36	0.0143 ± 0.0076	1.47 ± 0.38
$R = 2000$	$(1.58 \pm 0.86) \times 10^{-4}$	2.64 ± 0.82	0.0143 ± 0.0076	2.81 ± 0.88
$R = 5000$	$(7.96 \pm 4.16) \times 10^{-4}$	8.66 ± 2.20	0.0143 ± 0.0076	9.47 ± 3.94

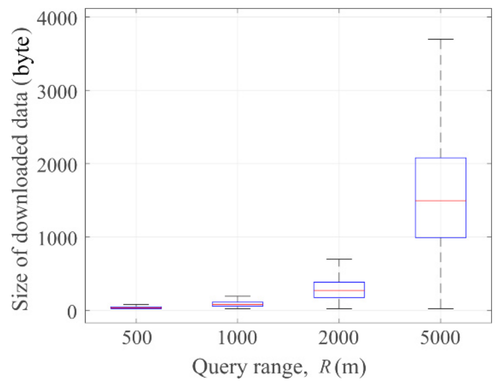


Fig. 11 Communication overhead to download the drivers' locations from the SP.

has to download 296 bytes. Even when the cloaking region is of size $A_R = 2500\,\text{m} \times 2500\,\text{m}$, on average, the rider has to download 1530 bytes.

6.2.4 Effect of privacy preference (A_R) on the delay due to temporal cloaking

We evaluate the time delay t generated by the enhanced temporal cloaking algorithm. We study the effect of A_R on the introduced time delay. As shown in Fig. 12, the time delay introduced by the temporal cloaking algorithm varies with A_R. The delay is small, especially when A_R ranges from $250\,\text{m} \times 250\,\text{m}$ to $500\,\text{m} \times 500\,\text{m}$. In about 90% of the cases, the delay is less than 2 minutes. In PrivateRide, the temporal cloaking introduces fixed delay. However, as shown in Section 3.2, the SP can improve the inference of riders' locations with the knowledge of such static temporal cloaking.

6.2.5 Effect of privacy preference (A_R) on fare calculation

In our solution, the SP calculates the fare of the ride based on the pick-up and drop-off cloaking regions. We consider protecting rider's location privacy that comes at a premium, and hence the calculated fare is higher than the original fare (computed by current ride hailing-service). We evaluate the percentage increase in the fare by the enhanced privacy preserving ride-hailing. We study the effect of A_R on the introduced fare increase. As shown in Fig. 13, the fare increase varies with A_R. When $A_R = 250\,\text{m} \times 250\,\text{m}$, in about 80% of the cases, the fare increase percentage is less than 11%. While in case $A_R = 500\,\text{m} \times 500\,\text{m}$, in about 80% of the cases, the fare increase percentage is less than 14%. Finally, in case $A_R = 1000\,\text{m} \times 1000\,\text{m}$, in about 80% of the cases, the fare increase percentage is less than 23%. In general, as A_R increases, the fare increases, which means that a rider with higher privacy requirement has to pay higher fare as a premium for protecting his location privacy.

7 Conclusion

In this paper, we addressed the problem of preserving riders' location privacy in ride-hailing services with two techniques, the baseline solution and the enhanced solution. Although the baseline solution can provide riders with personalized location privacy, we identified two inference attacks against it. To deal with these inference attacks, the enhanced solution relies on enhanced ride matching and temporal cloaking techniques. The enhanced solution provides riders with personalized location privacy while limiting the matching accuracy loss. Evaluation results showed that our solution outperforms previous work by providing more accurate matching results with negligible computational overhead.

References

[1] Y. Khazbak, J. Y. Fan, S. C. Zhu, and G. H. Cao, Preserving location privacy in ride-hailing service, in *Proc. 2018 IEEE*

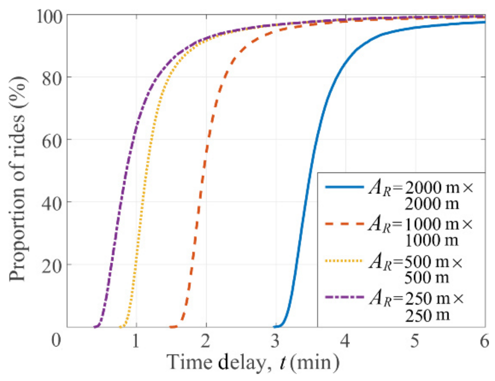


Fig. 12 Time delay introduced by the enhanced temporal cloaking algorithm for different A_R.

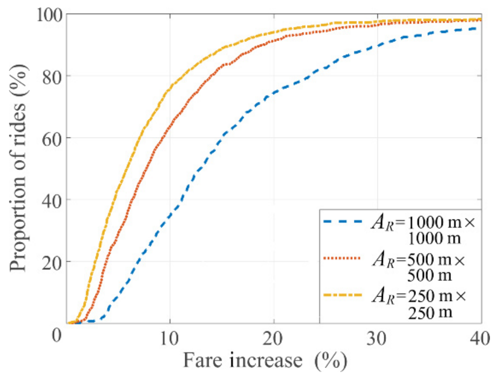


Fig. 13 Fare increase introduced by the enhanced privacy preserving ride-hailing for different A_R.

Conf. on Communications and Network Security, Beijing, China, 2018.

[2] P. Golle and K. Partridge, On the anonymity of home/work location pairs, in *Proc. 7th Int. Conf. on Pervasive Computing*, Nara, Japan, 2009.

[3] M. Feeney, Is ridesharing safe? https://www.cato.org/publications/policy-analysis/ridesharing-safe, 2015.

[4] R. Shokri, G. Theodorakopoulos, J. Y. Le Boudec, and J. P. Hubaux, Quantifying location privacy, in *Proc. 2011 IEEE Symp. on Security and Privacy*, Berkeley, CA, USA, 2011.

[5] M. L. Damiani, E. Bertino, and C. Silvestri, The probe framework for the personalized cloaking of private locations, *Trans. Data Privacy*, vol. 3, no. 2, pp. 123–148, 2010.

[6] M. Q. Xue, P. Kalnis, and H. K. Pung, Location diversity: Enhanced privacy protection in location based services, in *Proc. 4th Int. Symp. on Location- and Context-Awareness*, Tokyo, Japan, 2009.

[7] B. Niu, Q. H. Li, X. Y. Zhu, G. H. Cao, and H. Li, Achieving k-anonymity in privacy-aware location-based services, in *Proc. IEEE INFOCOM 2014 - IEEE Conf. on Computer Communications*, Toronto, Canada, 2014.

[8] B. Niu, Q. H. Li, X. Y. Zhu, G. H. Cao, and H. Li, Enhancing privacy through caching in location-based services, in *Proc. 2015 IEEE Conf. on Computer Communications*, Kowloon, China, 2015.

[9] A. Pham, I. Dacosta, B. Jacot-Guillarmod, K. Huguenin, T. Hajar, F. Tramèr, V. Gligor, and J. P. Hubaux, PrivateRide: A privacy-enhanced ride-hailing service, *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 2, pp. 38–56, 2017.

[10] J. Friginal, S. Gambs, J. Guiochet, and M. O. Killijian, Towards privacy-driven design of a dynamic carpooling system, *Pervasive Mobile Comput.*, vol. 14, pp. 71–82, 2014.

[11] E. Pagnin, G. Gunnarsson, P. Talebi, C. Orlandi, and A. Sabelfeld, TOPPool: Time-aware optimized privacy-preserving ridesharing, *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 4, pp. 93–111, 2014.

[12] J. B. Ni, K. Zhang, X. D. Lin, H. M. Yang, and X. S. Shen, AMA: Anonymous mutual authentication with traceability in carpooling systems, in *Proc. 2016 IEEE Int. Conf. on Communications*, Kuala Lumpur, Malaysia, 2016.

[13] C. Stach and B. Mitschang, Privacy management for mobile platforms—A review of concepts and approaches, in *Proc. 2013 IEEE 14th Int. Conf. on Mobile Data Management*, Milan, Italy, 2013.

[14] U. M. Aïvodji, S. Gambs, M. J. Huguet, and M. O. Killijian, Meeting points in ridesharing: A privacy-preserving approach, *Transp. Res. Part C Emerg. Technol.*, vol. 72, pp. 239–253, 2016.

[15] U. M. Aïvodji, K. Huguenin, M. J. Huguet, and M. O. Killijian, SRide: A privacy-preserving ridesharing system, in *Proc. 11th ACM Conf. on Security & Privacy in Wireless and Mobile Networks*, Stockholm, Sweden, 2018.

[16] D. Sánchez, S. Martínez, and J. Domingo-Ferrer, Co-utile P2P ridesharing via decentralization and reputation management, *Transp. Res. Part C Emerg. Technol.*, vol. 73, pp. 147–166, 2016.

[17] Y. Y. He, J. B. Ni, X. Y. Wang, B. Niu, F. H. Li, and X. M. Shen, Privacy-preserving partner selection for ride-sharing services, *IEEE Trans. Veh. Technol.*, vol. 67, no. 7, pp. 5994–6005, 2018.

[18] P. Hallgren, C. Orlandi, and A. Sabelfeld, PrivatePool: Privacy-preserving ridesharing, in *Proc. 2017 IEEE 30th Computer Security Foundations Symp.*, Santa Barbara, CA, USA, 2017.

[19] A. Pham, I. Dacosta, G. Endignoux, J. R. T. Pastoriza, K. Huguenin, and J. P. Hubaux, ORide: A privacy-preserving yet accountable ride-hailing service, in *Proc. 26th USENIX Security Symp.*, Vancouver, Canada, 2017.

[20] Y. C. Luo, X. H. Jia, S. J. Fu, and M. Xu, pRide: Privacy-preserving ride matching over road networks for online ride-hailing service, *IEEE Trans. Inf. For. Secur.*, vol. 14, no. 7, pp. 1791–1802, 2019.

[21] Q. C. Zhao, C. S. Zuo, G. Pellegrino, and Z. Q. Lin, Geo-locating drivers: A study of sensitive data leakage in ride-hailing services, in *Proc. 26th Annual Network and Distributed System Security Symp.*, San Diego, CA, USA, 2019.

[22] C. Y. Chow and M. F. Mokbel, Trajectory privacy in location-based services and data publication, *ACM SIGKDD Explorations Newsletter*, vol. 13, no. 1, pp. 19–29, 2011.

[23] V. Bindschaedler and R. Shokri, Synthesizing plausible privacy-preserving location traces, in *Proc. 2016 IEEE Symp. on Security and Privacy*, San Jose, CA, USA, 2016.

[24] Y. Khazbak and G. H. Cao, Deanonymizing mobility traces with co-location information, in *Proc. 2017 IEEE Conf. on Communications and Network Security*, Las Vegas, NV, USA, 2017.

[25] C. Y. T. Ma, D. K. Y. Yau, N. K. Yip, and N. S. V. Rao, Privacy vulnerability of published anonymous mobility traces, *IEEE/ACM Trans. Network.*, vol. 21, no. 3, pp. 720–733, 2013.

[26] S. Fortune, Voronoi diagrams and Delaunay triangulations, in *Computing in Euclidean geometry*, D. Z. Du, ed. Singapore: World Scientific, 1992.

[27] F. Aurenhammer, Voronoi diagrams—A survey of a fundamental geometric data structure, *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, 1991.

[28] R. Cheng, Y. Zhang, E. Bertino, and S. Prabhakar, Preserving user location privacy in mobile data management infrastructures, in *Proc. 6th Int. Workshop on Privacy Enhancing Technologies*, Cambridge, UK, 2006.

[29] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York, NY, USA: John Wiley & Sons, 2012.

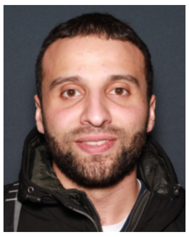
[30] S. Y. Liu, Y. H. Liu, L. M. Ni, J. P. Fan, and M. L. Li, Towards mobility-based clustering, in *Proc. 16th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Washington, DC, USA, 2010.

[31] D. T. Lee and B. J. Schachter, Two algorithms for constructing a Delaunay triangulation, *Int. J. Comput. Inf. Sci.*, vol. 9, no. 3, pp. 219–242, 1980.

[32] G. Leach, Improving worst-case optimal Delaunay triangulation algorithms, in *Proc. 4th Canadian Conference on Computational Geometry*, Newfoundland, Canada, 1992.

[33] E. Welzl, Smallest enclosing disks (balls and ellipsoids), in *New Results and New Trends in Computer Science*, H. Maurer, ed. Springer, 1991.

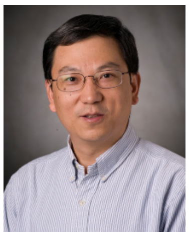
[34] C. Veness, Movable type scripts, http://www.movable-type.co.uk/scripts/latlong.html, 2017.



Youssef Khazbak received the MS degree from Cairo University in 2014. He is currently pursuing the PhD degree with the Department of Computer Science and Engineering, Pennsylvania State University. His research interests include security and privacy in mobile networks, mobile systems, and Internet of Things security.

Jingyao Fan received the BS degree from Tsinghua University in 2011. She is currently a software engineer in Amazon, Seattle. Her research interests include security and privacy in mobile networks.

Sencun Zhu received the BS degree from Tsinghua University, Beijing, China, in 1996, the MS degree from the University of Science and Technology of China in 1999, and the PhD degree from George Mason University, Fairfax, VA, USA, in 2004. He is an associate professor with Penn State University. His research interests include wireless and mobile security, network and systems security, and software security. Among his many academic services, he is the editor-in-chief of *EAI Endorsed Transactions on Security and Safety* and an associate editor of *IEEE Transactions on Mobile Computing*.



Guohong Cao received the PhD degree from the Ohio State University in 1999. Since then, he has been with the Department of Computer Science and Engineering, Pennsylvania State University, where he is currently a distinguished professor. He has published more than 200 papers in the areas of wireless networks, mobile systems, wireless security and privacy, and Internet of Things, which have been cited over 20 000 times. He has served on the editorial board of *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Wireless Communications*, and *IEEE Transactions on Vehicular Technology*, and has served on the organizing and technical program committees of many conferences, including the TPC Chair/Co-Chair of IEEE SRDS, MASS, and INFOCOM. He has received several best paper awards, the IEEE INFOCOM Test of Time award, and the NSF CAREER award. He is a fellow of the AAAS and a fellow of the IEEE.