sciendo

Proceedings on Privacy Enhancing Technologies ; 2019 (4):93–111

Elena Pagnin*, Gunnar Gunnarsson, Pedram Talebi, Claudio Orlandi, and Andrei Sabelfeld

TOPPool: Time-aware Optimized Privacy-Preserving Ridesharing

Abstract: Ridesharing is revolutionizing the transportation industry in many countries. Yet, the state of the art is based on heavily centralized services and platforms, where the service providers have full possession of the users' location data. Recently, researchers have started addressing the challenge of enabling *privacy-preserving ridesharing*. The initial proposals, however, have shortcomings, as some rely on a central party, some incur high performance penalties, and most do not consider time preferences for ridesharing. TOPPool encompasses ridesharing based on the proximity of end-points of a ride as well as partial itinerary overlaps. To achieve the latter, we propose a simple yet powerful reduction to a private set intersection on trips represented as sets of consecutive road segments. We show that TOPPool includes time preferences while preserving privacy and without relying on a third party. We evaluate our approach on real-world data from the New York's Taxi & Limousine Commission. Our experiments demonstrate that TOPPool is superior in performance over the prior work: our intersection-based itinerary matching runs in less than 0.3 seconds for reasonable trip length, in contrast, on the same set of trips prior work takes up to 10 hours.

Keywords: Privacy-preserving ride sharing, Location privacy, Private set intersection.

DOI 10.2478/popets-2019-0060
Received 2019-02-28; revised 2019-06-15; accepted 2019-06-16.

***Corresponding Author: Elena Pagnin:** Department of Computer Science, Aarhus University, E-mail: elena@cs.au.dk
Gunnar Gunnarsson: Chalmers University of Technology, E-mail: gunnarsson2@gmail.com
Pedram Talebi: Chalmers University of Technology, E-mail: pedramt@student.chalmers.se
Claudio Orlandi: Department of Computer Science, DIGIT, Aarhus University, E-mail: orlandi@cs.au.dk
Andrei Sabelfeld: Chalmers University of Technology, E-mail: andrei@chalmers.se

1 Introduction

Recent developments in information technology mark a paradigm shift in the transportation industry. *Location-Based Services* (LBS) for transportation have enabled a large and versatile variety of services like Uber [52], Lyft [29] and BlaBlaCar [5]. *Ridesharing* is a particularly appealing type of transportation service. Ridesharing companies, like BlaBlaCar [5], match ridesharing possibilities from a pool of trips advertised by its users. This kind of service is a typical sharing economy service that combines simplicity and travel cost efficiency for a more sustainable future. Yet, state of the art platforms are heavily centralized and allow the service providers to learn users' locations, commuting patterns and other data useful for individual profiling and tracking.

Need for privacy in ridesharing. While the appeal of ridesharing has led to remarkable developments, *privacy of ridesharing* is a key challenge that current platforms fail to address [12]. Indeed, the state of the art is based on heavily centralized solutions, where the service providers have full possession of the users' location data. Unfortunately, some centralized services have a record of abusing user privacy. For example, Uber and its employees have been allegedly involved in privacy-violating activities like stalking journalists, celebrities, and ex-girlfriends [4].

Even when a centralized service is well-minded, it becomes an attractive target for attacks to harvest the users' private location information. For example, Lyft's privacy safeguards have been allegedly bypassed by Uber while spying on double-dipping drivers [45]. This leads to a pivotal question of how to preserve privacy of passengers and drivers without sacrificing the functionality or performance of ridesharing services.

Privacy-preserving ridesharing. We set out to model flexible ridesharing, where either sharing whole rides or sufficiently long ride segments is possible. In this setting, we identify the following desiderata:

(i) Privacy wrt peers. Drivers and passengers should not learn more information about each other than is necessary by the ride matching protocol. This implies that no information is leaked when there is no match, and only information about the shared segment is leaked when there is a match on that segment.

(ii) Privacy wrt third parties. No third party should learn the users' location information. This includes any party involved in the matching process, including central parties when the protocol is centralized.

(iii) Realistic performance. For a ridesharing service to be usable, its performance is important. Users should be able to get quick answers to their match queries.

(iv) Time preferences. Users should be able to express time preferences and only get matches when the time preferences between riders and drivers agree.

Recently, researchers have started addressing the challenge of enabling privacy-preserving ridesharing [1, 2, 17, 21, 28, 34, 35, 44]. However, the initial efforts have shortcomings: most rely on a central party [1, 2, 21, 28, 34, 35, 44], some incur high performance penalties [17], and several do not fully support time preferences for ridesharing [2, 17, 28, 34, 35, 44]. Section 7 elaborates on the state of the art. The platform that lies closest to our goals is PrivatePool [17]. PrivatePool guarantees privacy, while not relying on a central party. It supports ride matching based on both proximity of end points and trajectories overlaps for segments larger than a chosen threshold. Thus, PrivatePool satisfies goals (*i*) and (*ii*), making it a suitable starting point. However, for trajectory matching, PrivatePool relies on a specially-designed threshold private set intersection construction, which incurs high performance overhead. In addition, PrivatePool has no support for setting the desired time of the ride. This implies that PrivatePool does not achieve goals (*iii*) and (*iv*).

TOPPool: a time-aware optimized approach to privacy-preserving ridesharing. This paper presents TOPPool, a decentralized platform for time-aware, optimized privacy-preserving ridesharing. TOPPool has two major improvements over PrivatePool: the inclusion of time in the ridesharing model and a faster trip matching mechanism. We offer two independent privacy-preserving approaches to determine feasible ridesharing: a time-aware proximity test of endpoints of a ride and a time-aware itinerary intersection. The former approach is extremely efficient in matching routes that go "in the same direction" though on different, possibly disjoint, ways. This solution, however, does not take into account the actual length of the shared ride. Itinerary intersection, on the other hand, identifies trips that "overlap" and suits ridesharing between cities or along specific routes like bridges or highways.

To ease the presentation, we first describe how to perform intersection-based matching in time linear on the trip length. We then extend both endpoint- and intersection-based matching to include the dimension of time. TOPPool should not be thought of as a self-standing interactive protocol, but rather as an approach or framework. The acronym is chosen to make a clear connection to the predecessor PrivatePool and highlight our contributions in terms of time-awareness and performance optimization

Intuitively, we include the time of the ride by augmenting the regular trips with one dimension and performing the usual matching on the spatial coordinates plus a time difference on the time coordinate. Our efficiency boost is due to more technical reasons. In a nutshell, PrivatePool performs intersection-based matching using a carefully crafted cryptographic tool named TPSI (Threshold Private Set Intersection). Looking at trips as sets of points and at the minimal itinerary overlap as a value c, TPSI loops through all of the possible sequences of c consecutive points along each trip and returns a match if one c-point set is present in both trips. This approach is quadratic in the trips' length. We leverage ordinary PSI (Private Set Intersection) techniques and perform intersection-based matching in time linear in the trips' length. The key point is to observe that the output of this matching process is a sequence of c *consecutive* points. Thus, we rely on an ideal routing model to identify one default itinerary between any two points. In this way, we simply check if both routes contain a pair of points (P, Q) with the ideal itinerary between P and Q being of length c. This can be done efficiently by running regular PSI on the sets of pairs (P, Q) generated by sliding P along the route.

Limitations. Our focus is on developing suitable cryptographic techniques for the emerging problem of private ridesharing. As such, our work does not constitute a fully-fledged ridesharing system. However, it paves the way for building such systems in the future. One limitation to note down is that we address privacy at the application layer, while not considering leaks that may occur at the system layer, such as leaks via IP addresses. More details on our setting are presented in Section 2.

Another limitation is scalability to large numbers of users. This limitation is shared with the entire line of work on privacy-preserving location proximity and ridesharing protocols (e.g., [17, 18, 31, 43, 53]), where matching among users is performed in pairwise fashion. Designing an efficient privacy-preserving marketplace where users can be matched in complexity less than quadratic (without sacrificing privacy) is a major open problem for this area of research.

protocol acronym	time awareness	cryptographic primitives	matching process complexity			
			ENDPOINT		INTERSECTION	
			computation	communication	computation	communication
PP [17]	×	AHE, TPSI	$\mathcal{O}(\lambda_{\max}^2)$	$\mathcal{O}(\lambda_{\max}^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n\log(n))$
O-PP (Sec.3)	×	AHE, PSI	$\mathcal{O}(\lambda_{\max}^2)$	$\mathcal{O}(\lambda_{\max}^2)$	$\mathcal{O}(n\log(n))$	$\mathcal{O}(n\log(n))$
TOPP (Sec.5)	✓	AHE, PSI	$\mathcal{O}(\lambda_{\max}^2+\tau_{\max})$	$\mathcal{O}(\lambda_{\max}^2+\tau_{\max})$	$\mathcal{O}(\tau_{\max}n\log(n))$	$\mathcal{O}(\tau_{\max}n\log(n))$

Table 1. Overview PrivatePool (PP), O-PrivatePool (O-PP) and TOPPool (TOPP); n is the maximum number of points in a trajectory; $\lambda_{\max}$ and $\tau_{\max}$ indicate respectively the maximum spatial and time deviation captured by the system. The complexities are only based on the input size and disregard the dependency on the security parameter. AHE, PSI and TPSI are the respective abbreviations of *additive homomorphic encryption*, *private set intersection* and *threshold* PSI.

Contributions. Table 1 provides a quick overview of our main results compared with PrivatePool. In detail:

We present O-PrivatePool, an optimized version of PrivatePool [17] for efficient and private intersection-based ridesharing. O-PrivatePool offers a novel way to test itinerary overlaps that reduces to private set intersection (instead of the inefficient *threshold* PSI of PrivatePool). We demonstrate the privacy guarantees of O-PrivatePool and discuss its asymptotic computational complexity (Section 3).

We present a general model for time-aware ridesharing that extends the one of PrivatePool to include users' time preferences (Section 4).

We present TOPPool a collection of two independent, efficient and privacy-preserving ways to match trips for ridesharing. Concretely, we describe a time-aware endpoint-based matching (timedEP) based on additive homomorphic encryption; and a time-aware intersection-based matching (timedIS) based on private set intersection (Section 5).

We evaluate the effectiveness of O-PrivatePool and TOPPool in finding ridesharing trips on real-world data from the New York's Taxi & Limousine Commission. Finally, we run a series of experiments to test and compare the efficiency of PrivatePool, O-PrivatePool and TOPPool (Section 6).

Fig. 1. The five ridesharing segments, for two trips.

2 Preliminaries

Notation. We denote by $[a,b]$ the set of all integer values between a and b, namely $[a,b]=\{a,a+1,\ldots,b\}$, $a,b\in\mathbb{Z}, a<b$. We denote by $|\mathcal{S}|$ the size of a set, i.e., the number of elements in $\mathcal{S}$. We denote by $\mathsf{Sym}[n]$ the symmetric group defined over the set $[1,n]$, consisting of all the possible permutation of n elements.

2.1 Ridesharing concepts

Ridesharing is the act of taking part in a journey in which one or more passengers travel in a private vehicle driven by its owner. In practice, drivers and passengers look for itineraries that match their own way, destination or direction. Given two itineraries, there are five segments to consider as depicted in Figure 1. For details on how to model ridesharing, maximum trip extension and ridesharing patterns we refer the reader to [17].

In this work, we adopt two independent approaches to match trips: *endpoint-based matching* (where rides are selected according to how far the respective starting/ending points are) and *intersection-based matching* (where rides are selected according to how large of an overlap there is between the two itineraries). In Section 2.2, we formalize the notion of feasible ridesharing in two flavors that closely model these methods. Below, we present our terminology and technical assumptions.

Definition 1 (Trip, trajectory and segment)**.** *Given a graph $G=(V,E)$ a trip $\mathcal{T}$ is an acyclic sequence of consecutive vertices $v_i\in V$, where $P_s=v_0$ is the starting point (origin) and $P_f=v_{|\mathcal{T}|-1}$ is the final point (destination), such that $(v_i,v_{i+1})\in E$, for all $i\in[0,|\mathcal{T}|-2]$. The set $\mathcal{T}$ is also called users' trajectory and its vertices positions $P_i\in\mathcal{T}$. Any subset $\mathcal{S}\subseteq\mathcal{T}$ of consecutive vertices of a trip is called segment.*

Note that the points of a trajectory are naturally sorted along the graph path (see Figure 2 and 3 for examples).

Definition 2 (Segment length)**.** *Given a segment* $\mathcal{S} = \{P_0, \ldots, P_{s-1}\}$ *for some trip in a graph, we denote the length of the segment as* $\ell(\mathcal{S}) = \sum_{i=0}^{s-2} d_{xy}(P_i, P_{i+1})$*, where* $d_{xy}(\cdot,\cdot)$ *is the Euclidean distance between two points on a plane.*

Assumptions. In line with previous work [17], we make the following simplifying assumptions:

(1) The distance between two points is the *Euclidean distance* on the plane.
(2) There is a graph that models possible routes between points and a model for *ideal routing* which, given any two points on the graph, outputs an ideal route between them. The ideal routing model should be thought of as a way to generate a "canonical trajectory" between locations.
(3) Users travel at a *constant speed*; spatial and temporal cost of traversing a segment are equivalent.

Assumption (1) makes a realistic approximation of the route that is sufficiently good for most applications. More accurate solutions would require ad-hoc development of the geometry of the region considered, including physical barriers such as rivers and mountains, as well as actual disposition of the roads. We favored the more general approach that uses the Euclidean distance to provide a general-purpose model and leave the development of geography-specific models to future work.

Assumption (2) gives us a simplified yet realistic ground to work with. The ideal routing model provides a set of preferred (ideal) trips along the graph, intuitively selecting a set of common trajectories that efficiently approximates real traffic. We make use of this model to create and match routes, and exploit the fact that between any pair of points there is only one ideal itinerary. Note that the same graph may have several ideal models, and users may choose what model to use for matching according to their preference, e.g., "toll-free" or "shortest route". We leave this choice to the application layer. In our experiments we consider the ideal route to be the shortest path between two locations. We discuss how our platform can support trajectory matching while preserving privacy of the routing model chosen by the users in Section 4.

Assumption (3) states that users have constant speed when traversing their routes. In particular, if a passenger matches with a driver at the origin of the shareable segment at a given time, the constant speed assumption guarantees that the passenger reaches the end of the shareable segment at the expected time. This assumption simplifies our time-aware model and can be lifted without affecting user privacy. The idea is to associate time-weights to the edges of the graph and compute the travel time including these costs. We explain how to do it in Section 4.

Note that our model is based on an unweighted graph where all edges reflected by simply introducing additional nodes and edges along the congested paths. However, such modeling will trade accuracy for efficiency, since the introduction of nodes and edges will degrade the efficiency of the protocol.

2.2 Defining feasible ridesharing

To ease the exposition, we consider two users, Alice and Bob (denoted with the capital Latin letters, A and B), each having their own trip $\mathcal{T}^A$ and $\mathcal{T}^B$ respectively. We explain the matching processes from Alice's point of view and assume that Alice and Bob cooperate to find a segment for feasible ridesharing. Note that we do not fix Alice's role, so she can act as a driver or passenger. The feasibility of ridesharing depends on three ingredients that we explain below.

Users' itineraries. The possible itineraries are contained in an undirected unweighed graph G that models the network of streets in the considered city. Trips are trees in the graph, i.e., directed paths between two vertexes and are determined using the ideal routing model.

Users' willingness to deviate. Following the footprint of [17], we formalize the notion of "users' willingness to deviate from the planned trip" using two tools: a *deviation function* $\Delta(\mathcal{T}; P)$, that measures how willing is the user with trajectory $\mathcal{T}$ to adjust their trip around the point $P \in \mathcal{T}$; and a *threshold* value t that sets a lower limit to the length of the ridesharing segment. In detail, the deviation function Δ returns distance values, e.g., $\Delta(\mathcal{T}; P) = 500\mathsf{mt}$ implies that the user with trajectory $\mathcal{T}$ is willing to move of at most 500 meters around the point P on their route in order to match another user's itinerary. The threshold t is a percentage of the total trip length, e.g., $t = 50\%$ implies that the users' trajectories should overlap on at least half of Alice's itinerary for the ride to be considered feasible.

Matching trips. We define matching trips according to two independent approaches: endpoint-based and intersection-based. The methods are independent and incomparable, and no method is *better* than the other [17]. Below, we explain the strengths and weaknesses of the two processes and define feasible ridesharing for each approach.

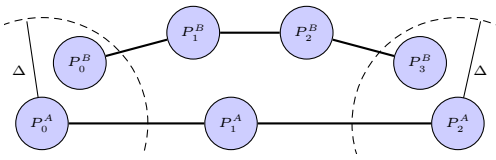

Fig. 2. Typical trajectories detected by endpoint-based matching.

Endpoint-based matching (EP). This matching process computes the distance from the starting (resp. ending) point of $\mathcal{T}^A$ to the starting (resp. ending) point of $\mathcal{T}^B$. If both distances are smaller than Alice's Δ around the endpoints of her trip, the matching process returns 1 (match found) otherwise it returns 0 (no match found). We formalize the notion of feasible ridesharing according to EP in the following definition.

Definition 3 (Endpoint-based ridesharing feasibility)**.** *For any fixed deviation function Δ, given two trips $\mathcal{T}^A = \{P_0^A, \ldots, P_{n_A-1}^A\}$ and $\mathcal{T}^B = \{P_0^B, \ldots, P_{n_B-1}^B\}$ for users A and B in a graph G, **ridesharing is feasible for A along B's itinerary** if and only if*

$$\begin{cases} d_{xy}(P_0^A, P_0^B) < \Delta(\mathcal{T}^A; P_0^A) \\ d_{xy}(P_{n_A-1}^A, P_{n_B-1}^B) < \Delta(\mathcal{T}^A; P_{n_A-1}^A)\ . \end{cases}$$

Endpoint-based ridesharing is especially useful when users care more about where the trip starts and ends than the actual way to go. In particular, this method matches trips that can be completely disjoint, as long as they start and end in about the same areas, as shown in Figure 2. We observe that EP by nature can be performed in constant time as the process is independent of the trip length and of the actual trajectory in-between the starting and ending points. While this matching mechanism is very natural and effective, it cannot detect possible ridesharing routes between trajectories that are close enough or even intersect on large segment but have starting and ending point that lie "far away" (see Figure 3). This is the main motivation to introduce an alternative matching mechanism that focuses on the actual itineraries and not only the endpoints of the ride.

Intersection-based matching (IS). This matching process looks for segments of length $t \cdot \ell(\mathcal{T}^A)$ present in both routes. Intuitively, intersection-based matching checks if any subset of consecutive locations builds a trip of the desired length within both Alice's and Bob's itineraries. If such a segment $\mathcal{S}$ is found, the matching process returns $\mathcal{S}$, otherwise it returns the empty set. We formalize the notion of feasible ridesharing according to IS in the following definition.

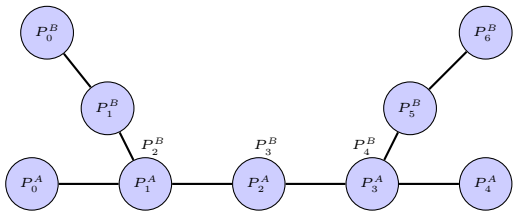

Fig. 3. Typical trajectories detected by intersection-based matching. In this example, $P_s = P_2^B$, $P_f = P_4^B$ and $t = 50\%$.

Definition 4 (Intersection-based ridesharing)**.** *For any threshold value t, given two trips $\mathcal{T}^A$ and $\mathcal{T}^B$ for users A and B in a graph G, **ridesharing is feasible for A along the segment $\mathcal{S} = \{P_s, \ldots, P_f\} \subseteq \mathcal{T}^B$** if and only if:*

1. *$\ell(\mathcal{S}) > t \cdot \ell(\mathcal{T}^A)$; and*
2. *there exist P_i^A, P_j^A in $\mathcal{T}^A$, with $i < j$ such that:*

$$\begin{cases} d_{xy}(P_i^A, P_s) = 0 \\ d_{xy}(P_j^A, P_f) = 0;\ . \end{cases}$$

Intuitively, Definition 4 ensures that at least a certain portion of Alice's trip is contained within Bob's itinerary. Therefore, intersection-based ridesharing is especially useful when users are flexible to be picked-up and drop-off at any point on their trajectory, provided that the shared ride will be long enough. This method is particularly effective when the city road plan has high-traffic roads such as bridges or tunnels.

As shown in Figure 3, intersection-based matching detects different ridesharing patterns than endpoint-based matching. Arguably the best approach to find more matching patterns would be to combine these two mechanisms. We elaborate on this in the next paragraph.

Optimal matching. EP and IS are valuable approaches to determine feasible ridesharing. In fact, Hallgren et al. [17] show that both methods excel, but on different rides. Endpoint-based matching quickly detects rides that are "in the same direction" but not necessarily identical, e.g., crossing a city on parallel roads (Figure 2); while intersection-based matching works best on "common" itineraries, e.g., crossing a bridge or along a highway (Figure 3). Therefore the optimal approach would be to have a definition of ridesharing feasibility that combines the flexibility of itinerary of EP with the mobility of pick-up and drop-off locations of IS. This is achieved in the following definition (taken from [17]):

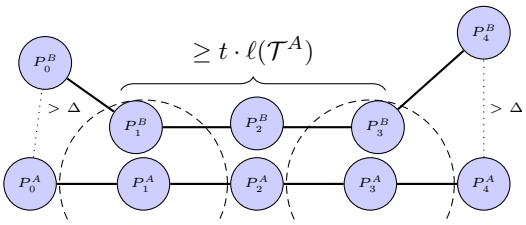

Fig. 4. Typical trajectories that satisfy threshold ridesharing (Definition 5) but are undetected by both EP and IS.

Definition 5 (Threshold ridesharing)**.** *For any fixed deviation function Δ and threshold percentage t, given two trips $\mathcal{T}^A$ and $\mathcal{T}^B$ for users A and B in a graph G,* ***ridesharing is feasible for*** *A* ***along the segment*** *$\mathcal{S} = \{P_s, \ldots, P_f\} \subseteq \mathcal{T}^B$ of B's itinerary if the following two conditions hold:*

1. *$\ell(\mathcal{S}) > t \cdot \ell(\mathcal{T}^A)$; and*
2. *there exist P_i^A, P_j^A in $\mathcal{T}^A$, with $i < j$ such that:*

$$\begin{cases} d_{xy}(P_i^A, P_s) < \Delta(\mathcal{T}^A; P_i^A) \\ d_{xy}(P_j^A, P_f) < \Delta(\mathcal{T}^A; P_j^A)\ . \end{cases}$$

Intuitively, Definition 5 states that at least a certain portion of Alice's trip (t) should be shared with user B, and the pick-up drop-ff locations should not deviate too much (Δ) from Alice's planned route. Note that Δ in Definition 5 parameterizes the user's spatial deviation preferences *at every point* along the route. Moreover, "small" threshold values make perfect sense for long rides (e.g., between cities) or to overcome particular geographical barrier (e.g., mountains, rivers etc.).

Combining EP and IS may appear to provide higher effectiveness for determining ridesharing feasibility. The outcome, however, has a privacy issue: combining the results returned by endpoint-based matching and intersection-based matching leaks information any time we match trajectories as in Figure 2. Briefly, the reason for the information leakage is that when determining the actual pick-up and drop-off locations (namely P_0^B and P_3^B in Figure 2), an attacker in the role of Alice sees that these are not on her trajectory (as it happens with IS) and learns Bob's whereabouts, breaking the privacy requirement. Moreover, EP may return false positives on pathological patterns, as we discuss in the Appendix A. For completeness, we notice that there are pairs of trajectories that satisfy the definition of threshold ridesharing but remain undetected by both EP and IS as shown in Figure 4.

2.2.1 Toward building secure ridesharing systems

As mentioned earlier, our work does not offer a fully-fledged ridesharing system, but rather provides a foundation for building such systems in the future. Our model is meant to be general, leaving implementation-dependent choices to the service provider. This is a common approach shared with [2, 17, 44] which, e.g., does not include identify management.

There has been work on building fully-fledged systems from protocols like ours. Stirbys et al. [48] demonstrate how to implement identity and credential management and communication of the peers for a decentralized protocol with trust assumptions are similar to ours. Their system includes a mobile application for the Android operating system, an application server, and a messaging service used to send push notifications to smartphones, leveraging Firebase Cloud Messaging (FCM). In their implementation, the identify management is handled by the application server, while the communication is accommodated by an FCM server.

Random IDs and digital certificates can be jointly used to prevent identity leaks to service providers. For example, ORide [34] uses Anonymous Credentials Light (ACL) [3], a linkable anonymous credential system that prevents linking users with their transactions.

To keep the model simple we ignore variables such as the cost of the ride and available seats. These can be however easily added as additional conditions for defining ridesharing feasibility in our Definition 5 following Aïvodji at al.'s approach in [2]. Finally, generating the graph corresponding to a city's street planning and building trajectories between points is also left for implementation-specific decisions. In our case study, we use the open source routing software Routino [42] based on the mapping data from OpenStreetMap (OSM) [9].

A secure ridesharing system must ensure that sensitive information is not leaked by network attacks [34]. This is an achievable goal since the smartphones of the users do not have fixed public IP addresses, using their mobile Internet providers' gateways to access the Internet. Further, VPN proxies and Tor can be used to further protect network identifies and metadata.

2.3 Defining privacy-preserving ridesharing

Location privacy is gaining popularity to combat record abuses that may lead to people identification [6, 15], organized stalking [4] and even spying [45]. Privacy-preserving ridesharing aims to reduce the risk of such threats by enforcing that, in case ridesharing is feasible, only the common trip is revealed nothing else about users' itineraries is leaked. The following definition captures this concept [17].

Definition 6 (Privacy-Preserving ridesharing)**.** *An approach* Π *realizes privacy-preserving ridesharing if for any two trajectories* $\mathcal{T}^A, \mathcal{T}^B$ *in* G*, a given threshold value* t *and a deviation function* Δ *it securely implements the functionality described by the considered ridesharing feasibility definition.*

By "securely implement" we mean that the protocol satisfies the standard textbook definition of simulation based for secure two-party computation protocols in the presence of semi-honest adversaries (see e.g., [20]).

Endpoint functionality. Π is a privacy-preserving ridesharing protocol for EP if:

- $\Pi(\mathcal{T}^A, \mathcal{T}^B) \rightarrowtail 1$ if and only if $\mathcal{T}^A, \mathcal{T}^B$ and Δ satisfy Definition 3; and
- $\Pi(\mathcal{T}^A, \mathcal{T}^B) \rightarrowtail \emptyset$ otherwise.

Intersection functionality. Π is a privacy-preserving ridesharing protocol for IS if:

- $\Pi(\mathcal{T}^A, \mathcal{T}^B) \rightarrowtail \mathcal{S}$ if and only if $\mathcal{T}^A, \mathcal{T}^B, t$ and $\mathcal{S}$ satisfies Definition 4; and
- $\Pi(\mathcal{T}^A, \mathcal{T}^B) \rightarrowtail \emptyset$ otherwise.

Note that in intersection-based matching Π additionally leaks the total lengths of the two itineraries.

2.4 PrivatePool: a model for privacy-preserving ridesharing

PrivatePool [17] is the name of a suite of matching processes for determining ridesharing segments in a privacy-preserving way. The matching processes are EP and IS presented before and the privacy guarantees come from two cryptographic techniques. *Additive homomorphic encryption* (AHE) is used to perform proximity-testing of itineraries' starting and ending points via essentially the InnerCircle protocol [16]. Privacy-preserving trajectory-intersection is achieved using *threshold private set intersection* (TPSI) on the two trajectories (seen as sets of subsequent points).

Homomorphic encryption schemes [14, 41] are public-key encryption schemes with an additional functionality allowing anyone to "compute blindfolded", i.e., to perform meaningful manipulations on encrypted data without ever having to decrypt. A special case are AHE schemes, such as [8, 32], which allow to perform linear operations in the encrypted domain. In other words, given the encryption scheme $\mathsf{Enc}, \mathsf{Dec}$, there are operations $\oplus, +$ (over the ciphertext group and plaintex groups respectively) such that:

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, x) \oplus \mathsf{Enc}(\mathsf{pk}, y)) = x + y$$

Operators $(\ominus, -)$ for subtraction and $(\odot, \cdot)$ for multiplication by scalar can also be defined and evaluated similarly. AHE can be instantiated efficiently from standard computational assumptions.

Private set intersection protocols (PSI) are interactive protocols between a sender and a receiver, each holding a (private) set. At the end of the protocol the receiver learns which elements belong to both sets *and nothing else*. The exact security guarantees can be formally defined using the standard ideal world/real world paradigm, and their security can be proven using the simulation-based approach. In recent years, there has been a significant improvement in the performances of PSI protocols, mostly those based on *Oblivious Transfer* (OT) and *Garbled Circuit*. An excellent survey of techniques for PSI can be found in [39].

Threshold PSI was introduced by Hallgren et al. [17] as an extension of the standard PSI functionality. In TPSI, the receiver learns the intersection of the two sets only if this has size at least equal to a given threshold c, and nothing otherwise; formally:

$$\mathsf{TPSI}(A, B) \rightarrow (\perp, Z), \quad Z = \begin{cases} A \cap B & \text{if } |A \cap B| \geq \mathsf{c} \\ \emptyset & \text{otherwise} \end{cases}$$

The solution for TPSI provided in PrivatePool combines the OT-based PSI of [36, 38] with a threshold key encapsulation mechanism (TKEM) based on Shamir's secret sharing scheme. Unfortunately the TPSI protocol of [17] does not scale well in the set size, and it is the main source of inefficiency. of PrivatePool. Recently, a new approach for TPSI was introduced by [7]. In this paper, we depart from the approach of Hallgreen et al. and replace TPSI with standard (and more efficient) PSI.

3 O-PrivatePool

By nature, the computational complexity of intersection-based matching increases with the length of the considered trips. To give some benchmarks, on trajectories with 1024 edges, PrivatePool's endpoint-based matching terminates in 0.34 seconds, while intersection-based matching requires 96.78 seconds [17]. We address this disparity and propose O-PrivatePool, an optimized version of PrivatePool that offers a novel, efficient approach to IS. Concretely, we are able to replace the expensive TPSI protocol of [17] with an efficient *regular* PSI on a different set of points.

A simple and effective trick. Our optimization comes from observing that the output of the

intersection-based matching needs to be a sequence of c (or more) *consecutive* points. Let t denote Alice's threshold value for IS. From t we define the *sequence-threshold* value $\mathsf{c} > 1$ corresponding to the minimal number of consecutive points needed in the graph G to realize a segment of length $t \cdot \ell(\mathcal{T}^A)$. Let $\mathcal{T}^A = \{P_0^A, \ldots, P_{n_A-1}^A\}$ denote Alice's trajectory of length n_A, and $\mathcal{T}^B$ defined analogously for Bob. Establishing intersection-based ridesharing feasibility boils down to checking whether two points at position i and $i + \mathsf{c}$ on Alice's trajectory exist as well on Bob's. Indeed, if there exist an index $i_A \in [0, n_A - \mathsf{c} - 1]$, and an index $j_B \in [0, n_B - \mathsf{c} - 1]$ such that $P_{i_A}^A = P_{j_B}^B$ *and* $P_{i_A+\mathsf{c}}^A = P_{j_B+\mathsf{c}}^B$, then $P_{i_A+h}^A = P_{j_B+h}^B$ for all $h \in [0, \mathsf{c} - 1]$.

Intersection-based matching in O-PrivatePool. O-PrivatePool's privacy-preserving intersection-based matching is a two party protocol and works as follows.

Step 1: Alice extracts from her trajectory $\mathcal{T}^A = \{P_0^A, \ldots, P_{n_A-1}^A\}$ a set of $(n_A - \mathsf{c})$ pairs of points $\mathcal{A} = \{(P_i^A, P_{i+\mathsf{c}}^A)\}_{i \in [0, n_A - \mathsf{c} - 1]}$. Alice sends to Bob her sequence-threshold value c.

Step 2: Bob checks that his trip $\mathcal{T}^B$ is not too short by comparing $n_B \geq \mathsf{c}$. In case $n_B < \mathsf{c}$, Bob aborts. Otherwise, Bob performs the same procedure on his trajectory $\mathcal{T}^B$ and generates the set $\mathcal{B} = \{(P_j^B, P_{j+\mathsf{c}}^B)\}_{j \in [0, n_B - \mathsf{c} - 1]}$.

Step 3: Alice and Bob run a PSI protocol using the sets $\mathcal{A}$ and $\mathcal{B}$ as respective inputs.

We notice that the actual form of the PSI does not impact the structure of our intersection-based matching. If PSI returns no common intersection, we output $\emptyset$ (no match found); otherwise the PSI returns $\mathcal{S} = \mathcal{A} \cap \mathcal{B} \neq \emptyset$ (match found).

Security. We remark that the PSI protocol returns only the segments common to both routes. In particular, neither Alice or Bob learns how the other party's route is shaped outside the common piece(s).

Theorem 1. *Let* PSI *be a private set intersection protocol secure against semi-honest adversaries, then O-PrivatePool realizes privacy-preserving intersection-based ridesharing.*

We provide here a sketch of the security proof. By assumption, the underlying PSI protocol is secure, i.e., there exist a simulator that, given the input/output of a party in the PSI protocol, can produce a view which is computationally indistinguishable from the view of a semi-honest adversary in the protocol. Since we use the PSI protocol in a black box manner, it is enough to show that given the input/output of the intersection matching functionality, we can simulate the input/output of the PSI functionality. First, note that the intersection matching functionality reveals the length of the itinerary of Alice and Bob. Thus, if Bob's itinerary is shorter than the threshold, the simulator will abort as Bob would do in Step 2. Otherwise, our simulator will simply run the PSI simulator with the appropriate input as explained now: note that the output of the PSI protocol (i.e., a set of consecutive pairs $(P_i, P_{i+\mathsf{c}}), \ldots, (P_j, P_{j+\mathsf{c}})$) can be efficiently simulated given the desired output of the trajectory-intersection protocol (i.e., the common segment $(P_i, \ldots, P_{j+\mathsf{c}})$), simply by constructing a trajectory adding (once) each point appearing in the pairs used in the PSI protocol.

Complexity. Generating the sets $\mathcal{A}$ and $\mathcal{B}$ for the PSI scheme takes $\mathcal{O}(n)$ where $n = \max\{n_A, n_B\}$, as users need to run through the points in their trajectories once, while pairing up the location P_i with the c hops away point $P_{i+\mathsf{c}}$. To be precise, the input set will consist of $n - \mathsf{c}$ pairs of points. After this, the complexity of O-PrivatePool is dominated by the choice of the PSI protocol. There are many PSI protocols in the literature, optimizing different parameters e.g., it is possible to optimize for computation or communication. Our implementation of O-PrivatePool uses the *BaRK-OPRF* PSI protocol of Kolesnikov et al. [25], which is among the fastest state-of-the-art custom PSI protocols for large batches of data. Asymptotically, the protocol requires work and communication $\mathcal{O}(n \log n)$. Importantly, the protocol uses a number of expensive cryptographic operations (e.g., exponentiations necessary for computing the Oblivious Transfers) proportional only to the security parameter (and not the input size). After that, the protocol mostly performs cheap symmetric-key operations (e.g., hashing, AES encryptions, etc.). The exact parameters to be used in *BaRK-OPRF* depend on the input size, and we refer to the original paper for a thorough discussion on the choice of parameters.

We stress again that due to the modular nature of O-PrivatePool, any improvements in efficiency of PSI protocols would immediately translate in a more efficient ride-sharing protocol. For instance, the first protocol for PSI with linear communication overhead has recently been announced [37]. Plugging their protocol into O-PrivatePool would give an overall linear complexity in the length of the itineraries.

4 Modeling time-aware ridesharing

The criteria PrivatePool and O-PrivatePool employ to define ridesharing feasibility are quite natural and expressive, however, they lack a fundamental component in ridesharing: the *time* of the ride. In realistic contexts, ridesharing depends not only on users' locations and routes but also –if not especially– on the time of the ride. To share a ride users need to be approximately at the same place at approximately the same time. Therefore, in what follows we extend Hallgren et al.'s [17] ridesharing model to include the users' time of the ride as well as their willingness to deviate from the planned time. To this end, we adjust the definitions given in Section 2.1 and propose a novel model for *time-aware ridesharing*.

Definition 7 (timed trip, trajectory and segment)**.** *A **timed trip** (or **timed trajectory**) $\mathcal{T}_{\mathsf{time}}$ is classical trip (as of Definition 1) in which each location P_i is augmented with a value T_i encoding the time at which the user wants to be to be at location P_i. In detail, the set $\mathcal{T}_{\mathsf{time}}$ is made of elements of the form (P_i, T_i). Any subset $\mathcal{S}_{\mathsf{time}} \subseteq \mathcal{T}_{\mathsf{time}}$ of consecutive locations and of a timed trip is called **timed segment**.*

Definition 7 simply augments the usual trip and segment definitions with one temporal component for each spatial coordinate. We remark that, in applications, users do not need to generate every single location and time along their trip. This can be done automatically by the client software given the following three ingredients: a starting point, an ending point, and a desired time for the ride to begin. Our time-aware ridesharing model uses the same definition of segment length as the previous model (Definition 2). Indeed, the only elements that contribute to the length of a segment are the spatial coordinates $P_i \in \mathcal{S}$ in the Euclidean plane.

We define time-aware feasible ridesharing using two deviation functions: Δ_{space} and Δ_{time}. Concretely, Δ_{space} is the same as in Section 2.1 and models the users' willingness to "move away" from the planned trajectory in terms of locations; while Δ_{time} models users' deviance in time. In addition to the 2-dimensional Euclidean distance d_{xy} to compare spatial coordinates on the plane, we will also employ the 1-dimensional distance d_z for measuring time differences. Formally,

Definition 8 (Time-aware threshold ridesharing)**.** *For any fixed threshold value t, and deviation functions Δ_{space} (modelling users' flexibility in space) and Δ_{time} (modelling users' flexibility in time), given two timed trips $\mathcal{T}^A_{\mathsf{time}}$ and $\mathcal{T}^B_{\mathsf{time}}$ for users A and B in a graph G, **time-aware ridesharing is feasible for A along the segment** $\mathcal{S}_{\mathsf{time}} = \{(P_s, T_s), \ldots, (P_f, T_f)\} \subseteq \mathcal{T}^B_{\mathsf{time}}$ if and only if the following conditions hold:*

- *$\ell(\mathcal{S}_{\mathsf{time}}) > t \cdot \ell(\mathcal{T}^A_{\mathsf{time}})$; and*
- *there exist $(P^A_i, T^A_i), (P^A_j, T^A_j)$ in the timed-trip $\mathcal{T}^A_{\mathsf{time}}$, with $i < j$, such that:*

$$\begin{cases} d_{xy}(P^A_i, P_s) & \leq & \Delta_{\mathsf{space}}(\mathcal{T}^A_{\mathsf{time}}; P^A_i), \\ d_{xy}(P^A_j, P_f) & \leq & \Delta_{\mathsf{space}}(\mathcal{T}^A_{\mathsf{time}}; P^A_j), \\ d_z(T^A_i, T_s) & \leq & \Delta_{\mathsf{time}}(\mathcal{T}^A_{\mathsf{time}}; T^A_i), \\ d_z(T^A_j, T_f) & \leq & \Delta_{\mathsf{time}}(\mathcal{T}^A_{\mathsf{time}}; T^A_j). \end{cases}$$

Comparing Definition 8 with Definition 5, we see that our time-aware model comprises all of the physical constraints from the un-timed model *plus* two conditions on the starting and ending time of the ride.

We derive the definition of *time-aware endpoint-based ridesharing* from Definition 8 by dropping the constraint given by the threshold and setting $P^A_i = P^A_0, P_s = P^B_0, P^A_j = P^A_{n_A-1}, P_f = P^B_{n_B-1}$. Analogously, we define *time-aware endpoint-based ridesharing* from Definition 8 by setting $\Delta_{\mathsf{space}} = 0$ for every trip and point. The definition of privacy-preserving ridesharing remains unchanged: the approach security implements its functionality.

Modeling dynamic delays. As mentioned in Section 2.1, our implementation of time-aware privacy-preserving ridesharing considers users to have constant speed (Assumption (3)). While this choice simplifies the presentation and analysis or our protocol, it clearly limits its accuracy. It is possible to lift Assumption (3) and accommodate for dynamic delays without braking our model or affecting user privacy, in the following manner. Consider the timed-trajectory $\mathcal{T}_{\mathsf{time}} = \{(P_0, T_0), \ldots, (P_{n-1}, T_{n-1})\}$ where T_0 denotes the desired initial time of the ride. Denote by k the 'constant speed factor' (in minutes per meter), i.e., the linear coefficient used to convert a spatial distance into transit time under normal traffic conditions. Concretely, this means that the estimated travel time between point P_i and point P_j (under Assumption (3)) equals $k_{i,j} = k \cdot \ell(P_i, P_j)$. Denote by $K_{i,j}$ the additional delay (in minutes) when transiting between P_i and P_j due to, e.g., rush hour, works on the road or similar conditions. We can include such dynamic delays in our model as weights on the edges and compute the estimated time of arrival $\tilde{T}_i$ at a point P_i in $\mathcal{T}$ as $\tilde{T}_j = \sum_{i=0}^{j-1} k \cdot \ell(P_i, P_{i+1}) + K_{i,i+1}$. Employing the dynamic time-values $\tilde{t}_j$ –instead of the standard ones without the $K_{i,j}$ addend– we improve the accuracy of our model.

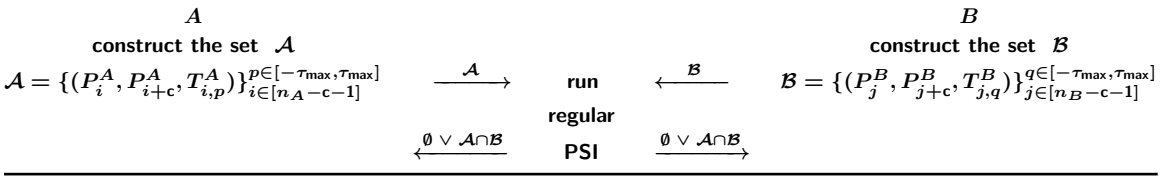

Fig. 5. Both users $U \in \{A, B\}$ hold a timed-trajectory $\mathcal{T}^U = \{(P_i^U, T_i^U)\}_{i\in[0,n_U-1]}$ and compute the time intervals $T_{i,p}^U$ for $p \in [-\tau_{\max}, \tau_{\max}]$ and $i \in [n_U - \mathsf{c} - 1]$ where c is a given sequence-threshold value and, for simplicity, we consider $\Delta_{\mathsf{time}} = \tau_{\max}$.

Supporting multiple routing models. There might be other user preferences to be taken into account, e.g., toll-free or shortest route. This can be easily done by treating routing preferences similarly to the time component. Concretely, if Alice wants to travel from point $(P, \mathsf{toll}-\mathsf{free})$ to a point $(Q, \mathsf{toll}-\mathsf{free})$ while Bob's itinerary includes $(P, \mathsf{toll}-\mathsf{allowed})$ to $(Q, \mathsf{toll}-\mathsf{allowed})$ our matching protocols will not return a match, since there is no common tuple as the routing tags differ. Note that such additional preferences would be kept private exactly in the same way as the time preferences of the user are protected, thanks to the guarantee of the underlying PSI protocol.

5 TOPPool

PrivatePool and O-PrivatePool perform privacy-preserving ridesharing matching in a basic way that does not capture the time of the ride. In this section, we introduce TOPPool, a decentralized platform for determining feasible ridesharing also according to time preference. In a nutshell, TOPPool extends O-PrivatePool to take into consideration the *time* of the ride as a parameter and therefore is a time-aware and optimized version of PrivatePool (thus the acronym). We remark that in our matching processes Alice's role can be either passenger or driver.

5.1 Time-aware, private intersection-based ridesharing (timedIS) in TOPPool

TOPPool's privacy-preserving time-aware intersection-based matching (timedIS) is a two party protocol that securely implements the following functionality:

timedIS **functionality:** *given as input two timed trips* $(\mathcal{T}_{\mathsf{time}}^A, \mathcal{T}_{\mathsf{time}}^B)$*, a threshold percentage* t*, and a time-deviation function* Δ_{time} *return the common segment* $\mathcal{S}$ *(match found) if the trips satisfy Definition 8 for* $\Delta_{\mathsf{space}} = 0$*; otherwise return* $\emptyset$ *(no match found).*

The idea behind TOPPool's timedIS can be summarized as follows. Borrowing the notation introduced in Section 3 (intersection-based matching in O-PrivatePool), we consider triplets of the form $(P_i, P_{i+\mathsf{c}}, T_{i,p})$, where the first two elements are as in O-PrivatePool, i.e., P_i is the spatial coordinate of the i-th point along the party's trajectory; the last component, $T_{i,p}$, represents the p-th possible temporal coordinate at which the user can be at location P_i. Concretely, $T_{i,p} = T_i + p$ for suitable values of p determined by the user flexibility in time around the i-th point on their route. Intuitively, the $T_{i,p}$ values span within an interval of radius $\Delta_{\mathsf{time}}(\mathcal{T}_{\mathsf{time}}; T_i)$ around the desired time, formalizing the concept of "at around T_i". We then let Alice and Bob run PSI between their sets of triples generated as described above. If the outcome is a non-empty intersection it means that the two trips have a long enough common segment and the time of the ride are compatible, i.e., there exist a moment in time at which both users can be at the first location on $\mathcal{S}$. Figure 5 provides an overview of timedIS, for a detailed description see the Appendix B.

Correctness. The correctness of timedIS reduces directly to the correctness of the employed PSI protocol and the way we generate the sets $\mathcal{A}, \mathcal{B}$. Observe that by Assumption (3), if there exist a point on two trips with temporal and spatial coordinates within the deviation limits of the driver, then it is guaranteed that the passenger will reach all the consequent points along the shared segment in due time. In particular, if the following conditions hold

$$\exists\, i_0, j_0, p_0, q_0 : P_{i_0}^A = P_{j_0}^B \wedge P_{i_0+\mathsf{c}}^A = P_{j_0+\mathsf{c}}^B \wedge T_{i_0,p_0}^A = T_{j_0,q_0}^B,$$

then the following condition is also satisfied:

$$\forall\, k \in [0, \mathsf{c}], \exists\, p_k, q_k \in [-\tau_{\max}, \tau_{\max}] \text{ such that}$$

$$P_{i_0+k}^A = P_{j_0+k}^B \;\wedge\; T_{i_0+k,p_k}^A = T_{j_0+k,q_k}^B.$$

This reasoning shows that any triple output by timedIS satisfies the desired functionality (given at the incipit of this section).

Security. Similarly to what we did for O-PrivatePool in Theorem 1, it can be easily argued that timedIS returns only the common ride segments (if any) and leaks nothing the users' trips outside the common portions. Formally,

Theorem 2. *Let* PSI *be a private set intersection protocol secure against semi-honest adversaries, then* timedIS *realizes privacy-preserving time-aware intersection-based ridesharing.*

Complexity. The running time of timedIS depends both on the trajectory length and on the maximal time deviation. Let $n = \max\{n_A, n_B\}$ and $\tau_{\mathsf{max}} = \max\{\tau^A_{\mathsf{max}}, \tau^B_{\mathsf{max}}\}$, to generate the sets of triples we only need $\mathcal{O}(n)$ elementary operations –for pairing up c-distant points– and $\mathcal{O}(\tau_{\mathsf{max}})$ elementary operations to generate the desired time interval *around each pair of points*. Thus, timedIS has complexity $\mathcal{O}(\tau_{\mathsf{max}} \cdot n)$ plus the complexity of the employed PSI, resulting in a $\mathcal{O}(\tau_{\mathsf{max}} \cdot n \cdot \log(n))$ in our implementation.

5.2 Time-aware, private endpoint-based ridesharing (timedEP) in TOPPool

TOPPool's privacy-preserving time-aware endpoint-based matching (timedEP) is a two party protocol that securely implements the following functionality:

timedEP **functionality:** *given as input two timed trips* $(\mathcal{T}^A_{\mathsf{time}}, \mathcal{T}^B_{\mathsf{time}})$*, a spatial-deviation function* Δ_{space}*, and a time-deviation function* Δ_{time} *return 1 (match found) if the trips satisfy Definition 8 for* $t = 0$*,* $P^A_i = P^A_0, P_s = P^B_0, P^A_j = P^A_{n_A-1}, P_f = P^B_{n_B-1}$*; otherwise return* $\emptyset$ *(no match found).*

TOPPool's timedEP can be seen as EP in PrivatePool (based on InnerCircle [16]) augmented with a time component (and its relative set). We provide here an intuition, for technical details see the Appendix A.

Alice uses her public key of an AHE scheme to encrypt the location and time of her starting and ending points and sends these data to Bob. In what follows, we denote by the subscript s (resp. f) values corresponding to the starting (resp. ending) points or times of user's trajectories. Bob uses the location and time of his starting and ending points to homomorphically compute the (encrypted) physical distances $D_{xy,s}, D_{xy,f}$ between his and Alice's locations (precisely as in InnerCircle and PrivatePool). In addition, Bob evaluates the (encrypted) temporal differences $D_{z,s}, D_{z,f}$. From each of the above values, Bob generates a set of ciphertexts using the InnerCircle technique. For example, from $D_{z,s}$ Bob computes:

$$\mathcal{W}_s = \{(D_{z,s} \ominus \mathsf{Enc}(\mathsf{pk}_A, j)) \odot \rho_j \mid j \in [-\tau_{\mathsf{max}}, \tau_{\mathsf{max}}]\}$$

where τ_{max} is as in timedIS and ρ_j are random values from the plaintext space. The set $\mathcal{W}_f$ and the location sets $\mathcal{L}_s, \mathcal{L}_f$ are generated analogously. Bob also shuffles the ciphertexts within each set. This procedure ensures that Bob's actual location and time preferences remain hidden to Alice. Nonetheless, if each set has one ciphertext that decrypts to 0 Alice finds out that their rides match. See the Appendix A for detailed explanations.

Correctness. The only difference between the EP in PrivatePool and timedEP is the addition of the time component. As we explained above timedEP treats time values in a very similar way to spatial coordinates therefore the correctness of timedEP comes for the one of PrivatePool's EP and InnerCircle [16].

Security. We want to show that timedEP securely implements the timedEP functionality. Formally,

Theorem 3. *Let* AHE *be a semantically secure additive homomorphic encryption scheme, then* timedEP *realizes privacy-preserving time-aware endpoint-based ridesharing.*

The proof of Theorem 3 is a straightforward generalization of the security proof of InnerCircle [16]. In particular, it is possible to efficiently simulate the views of Alice, Bob and an external entity (Claire). The addition of the time components does not impact the security, as they can be simulated using the same techniques as spatial coordinates.

Complexity. The running time of timedEP is quadratic in the flexibility of the system but independent of the trips length. Let λ_{max} and τ_{max} be maximum spatial and time deviations allowed by the system. Generating the sets $\mathcal{L}$ and $\mathcal{W}$ takes $\mathcal{O}(\lambda^2_{\mathsf{max}})$ and $\mathcal{O}(\tau_{\mathsf{max}})$ respectively. Shuffling the sets adds costs linear in $|\mathcal{L}| = \lambda^2_{\mathsf{max}} + 1$ and $|\mathcal{W}| = 2 \cdot \tau_{\mathsf{max}} + 1$ respectively. Thus, timedEP has complexity $\mathcal{O}(\lambda^2_{\mathsf{max}} + \tau_{\mathsf{max}})$. Regarding the communication complexity, let m denote the bit length of the ciphertexts. Alice sends 8 ciphertexts to Bob (3 for each spatial

point and 1 for the time preference at each point). Bob replies with 4 sets that add up to $\mathcal{O}((\lambda_{\mathsf{max}}^2+2\tau_{\mathsf{max}}+2)m)$ bits to transmit giving the asymptotic value provided in Table 1.

6 Experiments

This section collects our main results on the effectiveness and efficiency of O-PrivatePool and TOPPool. We show that, compared to PrivatePool [17], our proposals achieve an astonishing speed-up in intersection-based matching without compromising neither effectiveness nor user privacy.

6.1 Implementation details

In order to provide a fair comparison, we implemented PrivatePool, O-PrivatePool and TOPPool using the same programming language: Python. As PSI protocol, we deployed BaRK-OPRF [25] (using the open-source code available at [51]) as it is used within the TPSI procedure of PrivatePool and it is one of the fastest PSI to date. We implemented minor modifications to the original code to enable the program to read and parse the exported dataset and to return the result in an expected format.

The deviation functions Δ_{space} and Δ_{time}. We adopt the same approach as in PrivatePool to model the spatial deviation function Δ_{space} [17]. Concretely, given a trajectory $\mathcal{T}$ and a maximum spatial deviation value λ_{max}, we define the user's willingness to deviate at location $x \in [0, \ell(\mathcal{T})]$ along their route as:

$$\Delta_{\mathsf{space}}(x) = 4x^2 \frac{\lambda_{\mathsf{max}}}{\ell(\mathcal{T})^2} - 4x\frac{\lambda_{\mathsf{max}}}{\ell(\mathcal{T})} + \lambda_{\mathsf{max}} \qquad (1)$$

Equation 1 displays a curve that is λ_{max} far away from the trajectory's starting and ending points and it gradually approaches the actual trajectory until intersecting it in its middle point. This models the fact that users are willing to deviate at the start and end of their trip, but not in the middle; capturing the intuition that both passengers and drivers feel confident in moving around known places but are reluctant to change the "usual way" to go. Our time-aware model additionally considers a time-deviation function Δ_{time}. In our experiments, we adopt a static time-deviation, given a maximum delay value τ_{max}, we define Δ_{time} as:

$$\Delta_{\mathsf{time}}(x) = \tau_{\mathsf{max}}. \qquad (2)$$

We model deviance in time as a constant function in line with Assumption (3).

6.2 Effectiveness of O-PrivatePool and TOPPool

We evaluate the effectiveness of O-PrivatePool and TOPPool by comparing how many trips the respective intersection-based and endpoint-based matching processes find, against a plaintext implementation that finds all possible matches using bruteforce. Concretely, bruteforce matching checks all points (and times) of all the possible pairs of rides in the dataset and looks for segments that satisfy the *general* definition of ridesharing for the given values of threshold and spatial deviation Definition 5 (and 8). We report the percentages of how many trips IS, timedIS, EP and timedEP respectively detect, compared to all existing ridesharing routes detected via bruteforce.

The effectiveness of the matching methods were tested independently, on the same set of routes selected from the real-world data publicly available from Taxi & Limousine Commission (TLC) website [50]. TLC provides a large amount of data that can be used to generate endpoints of routes. However, the intermediate points of the trips are not included. In our experiments, we generated the trips using the open source routing software Routino [42] based on the mapping data from OpenStreetMap (OSM) [9]. To measure the effectivity of O-PrivatePool and TOPPool we employed the same set of trips as used to test PrivatePool's effectiveness: 1000 trips selected from each even month of the year 2015. We consider different values of threshold t, expressed as the minimum *percentage* of the user's total trip for considering ridesharing feasible; and of the maximal spatial deviation at any point of a trajectory (λ_{max}), in meters. The spatial deviation tolerated at intermediate points in the trajectories is derived from Equation (1). For the time-aware protocols timedIS and timedEP, we additionally consider different values for the maximal deviation in time, τ_{max}, expressed in minutes.

Table 2 reports the effectiveness of the privacy-preserving IS of O-PrivatePool compared to the timedIS of TOPPool. The values are average percentages of the ridesharing opportunities detected by our methods against the (non privacy-preserving) bruteforce approach. This means that the total of the ridesharing patterns (100%) is the output of the bruteforce comparison method between each individual point along two trips considering the optimal definition of (time-

Table 2. Effectiveness comparison: average percentages of privacy-preserving detecting ridesharing opportunities with **O-PrivatePool** and **TOPPool**. IS = intersection-based matching , timedIS = time-aware intersection-based matching.

λ_{max}		500 mt		1000 mt		2000 mt	
t	τ_{max}	IS	timedIS	IS	timedIS	IS	timedIS
20%	30 min	97.99	88.84	63.7	59.75	32.27	31.60
	45 min		93.26		60.95		31.91
	60 min		95.90		63.61		33.34
50%	30 min	79.42	76.58	44.55	41.46	15.88	15.72
	45 min		76.15		41.59		15.75
	60 min		76.22		42.90		16.72
80%	30 min	28.8	34.03	9.96	9.40	2.48	2.30
	45 min		33.14		11.21		2.81
	60 min		32.51		11.10		2.99

Table 3. Effectiveness comparison: average percentages of privacy-preserving detecting ridesharing opportunities with **O-PrivatePool** and **TOPPool**. EP = endpoint-based matching, timedEP = time-aware endpoint-based matching.

λ_{max}		500 mt		1000 mt		2000 mt	
t	τ_{max}	EP	timedEP	EP	timedEP	EP	timedEP
20%	30 min	0.30	0.00	1.39	0.76	5.95	3.63
	45 min		1.42		1.39		3.47
	60 min		1.03		1.36		3.97
50%	30 min	0.91	0.00	3.59	1.52	10.84	6.31
	45 min		3.81		2.89		5.91
	60 min		3.12		3.03		6.95
80%	30 min	2.51	0.00	6.11	2.38	12.89	7.28
	45 min		9.72		5.03		7.00
	60 min		7.04		4.97		8.24

aware) threshold ridesharing. We recall that in IS and timedIS the spatial deviation function set to a constant 0 value. Therefore, we do not expect to detect all of the ridesharing matches found via bruteforce. Nonetheless, our analysis shows that intersection-based matching is very effective, especially for shorter threshold values. We can observe that timedIS is only slightly less effective at detecting ridesharing opportunities than the time-insensitive IS protocol of O-PrivatePool.

Table 3 reports the effectiveness of the privacy-preserving EP of O-PrivatePool compared to timedEP of TOPPool. We recall that in EP and timedEP the threshold is set to 0, and only the staring and ending points of the rides are used for the comparison. The nature of the dataset [50] is reflected in the effectiveness numbers, as in the city of New York is more likely that rides have some intersection (e.g., along Brooklyn bridge) than the actual pick-ups and drop-offs being close to some fix points. Indeed, the effectiveness of both EP and timedEP increases with larger spatial deviation values (λ_{max}) and larger threshold.

Comparing Table 2 and 3 it is clear that endpoint-based matching is less effective than intersection-based matching. This is entirely due to the the geographical distribution of the routes considered in the experiment (LCT database [50]), where most trajectories have a large overlap, however, the pick-up / drop-off locations are actually sparse. Moreover, endpoint-based matching has an opposite behavior to intersection-based matching, thus confirming the intuition that two methods "compensate" each other. Finally, we remark that our time-aware matching mechanisms incur arguably little effectiveness drops with respect to time insensitive approaches.

6.3 Efficiency analysis of the intersection-based matching process

We tested the efficiency of O-PrivatePool and TOPPool in performing privacy-preserving intersection-based itinerary matching and compared the performances with the one of PrivatePool. In our experiments, we consider trajectories with the same number of nodes $n = 2^k$ for $k \in [5, 12]$. PrivatePool and O-PrivatePool were tested under identical conditions on a dedicated machine. The efficiency test suite comprises of the same tasks as the one for intersection-based matching in Pri-

vatePool, however, the trajectory datasets were randomly generated. Given the complexity of timedIS, we tested this process on a randomly generated dataset of routes with $\tau_{\max} = 12$ hours with time deviation precision were set to 30, 45 and 60 minutes. The motivation behind this choice is that users generally search for available rides the same day as they intend to make use of it. Since the number of possible time slots was considerably large and the standard deviation of the collected values was negligible, we limited the experiment to 10 repetitions and computed the average over the 30 values (10 per each deviation option).

Figure 6 displays the results, the concrete values are reported in Table 5 in the Appendix C. Figure 6 shows that O-PrivatePool clearly outperforms its predecessor. While PrivatePool's running time increases exponentially with the itinerary length, O-PrivatePool's runs constantly in less than 0.1 second, independently on the number of nodes in the trajectory. When the time of the ride comes into play, we expect performances to drop. Therefore it is no surprise that TOPPool's timedIS is less efficient than O-PrivatePool's IS and that its running time increases with the itinerary length. What is worth noticing is that the optimization trick that allows us to use regular PSI instead of TPSI is powerful enough to perform timedIS in less than 0.31 seconds, which is a legit *delay* in many application scenarios.

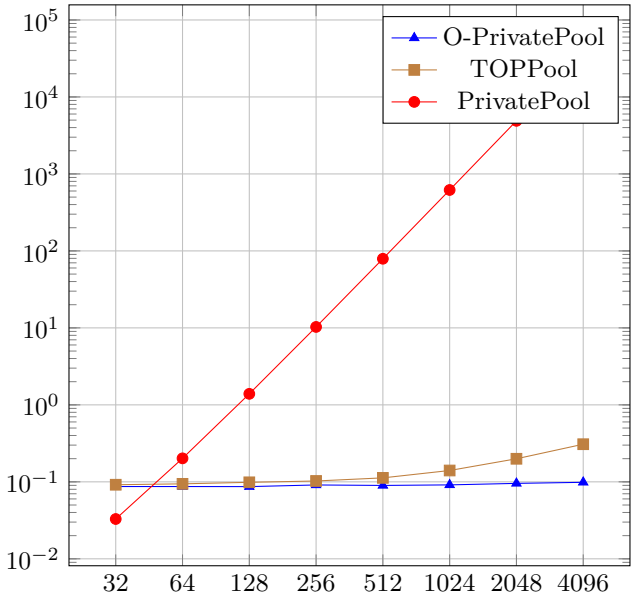

Fig. 6. Efficiency comparison of the running times of intersection-based matching in O-PrivatePool (blue-triangles), TOPPool (brown-squares) and PrivatePool (red-circles). The horizontal axis displays the value n representing the number of nodes in the trajectories, in *logarithmic scale*, while the vertical axis displays the average running times (in seconds) in *logarithmic scale*, where negative exponent values correspond to measurements below 1 second. The displayed values are average over 30 measurements for deviation precision set to 30, 45 and 60 minutes (10 measurements per each option).

Finally, our experiments (where threshold values were set randomly) indicate that the choice of the threshold value does not have a significant impact on the performance.

7 Related work

We discuss related work on re-identification, location proximity, privacy-aware ridesharing, and general private set intersection.

Re-identification. Location privacy is an increasingly important topic [27, 49]. When it comes to ridesharing scenarios, much prior research relates to re-identification [6, 15] in order to protect the privacy of a user's trajectory. The focus of hiding the identity of a user stems from the fact that many existing ridesharing services have access to users' location data and therefore can trace a user's whereabouts. As an example, consider public transport providers which use an electronic ticketing system to record where a user enters and exits a vehicle. While data on users' commutes could be valuable for statistical purposes about crowd movements and urban planning [24], they should not be misused and need not to be linkable to individuals. To tackle this and similar situations, researchers focus on collecting anonymized data. Our work takes a different approach and aims at protecting the privacy of a user's trajectory by minimizing the disclosure of location data, rather than the user's identity. This solution is relevant for cases where a service provider, who anyways needs to know the identity of their customers, wants to minimize the data disclosure to, e.g., reduce the risk of security breaches or to conform to privacy regulations, such as EU's General Data Protection Regulation (GDPR) [13].

Location proximity. There is extensive literature on the problem of how to test for the proximity of two points as provided by two different users, without revealing more than the proximity result [11, 18, 23, 30, 31, 43, 46, 47, 53]. However, most work considers a single proximity result in isolation. Further work uses additive homomorphic encryption for location proximity [18, 19, 31, 43, 53].

Privacy-aware ridesharing. Privacy-aware ridesharing [1, 17, 33] is now an active research area, boosted by real-life cases of location abuse by ridesharing plat-

Table 4. Comparison of privacy-aware ridesharing protocols.

Protocol	Decentralization	Privacy	Time
TOPPool	✓	✓	✓
PrivatePool [17]	✓	✓	
PrivateRide [35]		(✓)	(✓)
ORide [34]		✓	(✓)
He et al. [21]		(✓)	✓
Aïvodji et al. [2]		✓	
Li et al. [28]		(✓)	
SRide [1]		✓	✓
Sherif et al. [44]		(✓)	

forms [4]. Table 4 provides an overview and comparison of most approaches; checkmarks ✓ signify that a desired property is supported by the proposal, while checkmarks in parentheses (✓) indicate partial support.

PrivatePool [17] serves as a baseline for our approach. We substantially improve performance and generalize the approach to handle time, while preserving strong privacy guarantees and remaining in a fully decentralized setting with no trusted parties.

PrivateRide by Pham et al. [35] uses a combination of anonymous credentials, blind signatures as well as location and time cloaking to improve the privacy of passengers. However, privacy guarantees depend on the density of a cloaked area. Further, PrivateRide provides no guarantees of driver privacy. ORide [34] by Pham et al. builds on PrivateRide and uses somewhat-homomorphic encryption with optimizations such as ciphertext packing and transformed processing for privacy-aware ride-hailing. While it improves the privacy guarantees for both passengers and drivers compared to PrivateRide, it relies on a central party to compute on encrypted distances and leaks the distances of nearby drivers to the passenger.

He et al. [21] propose a central service that combines spatial preselection and Paillier cryptosystem to calculate the travel time saving, which is used to find a ride. However, some information, such as the region of riders and drivers is leaked to the service provider. Aïvodji et al. [2] use private set intersection to compute on possible pick-up and drop-off locations. Thus, they do not model possibilities of a successful ride as long as it is longer than a threshold. This model does not handle time preferences for a ride. Li et al. [28] use blockchain-assisted vehicular fog computing, involving private proximity testing and spatial cloaking. Fog computing achieves low latency due to local data processing. SRide [1] by Aïvodji et al. builds on the preceding work [2] and introduces a model that encompasses time preferences. However, SRide [1] looses precision due to the use of spatial cloaking techniques. Sherif et al. [44] focus on ridesharing for autonomous vehicles. They divide riding regions into cells and represent trips as binary vectors of such cells and use a server to decide similarity of encrypted binary vectors.

Private Set Intersection. Private set intersection is one of the most studied special cases of secure multi-party computation. In recent years, thanks to significant improvements in the performances of primitives for secure multi-party computation, the efficiency of PSI protocols has increased dramatically. Examples of such protocols can be found in [10, 22, 26, 36, 38, 40]. We refer the reader to the excellent survey by Pinkas et al. [39] for further background on PSI protocols.

8 Conclusions

With the rise in popularity of ridesharing systems, the risk of misuse and abuse of users' location data has become a real threat. We have presented TOPPool, an approach to prevent leakage of private information to ridesharing service providers. TOPPool encompasses a framework for modelling time-aware, flexible and optimized privacy-preserving ridesharing.

We have proposed two major improvements of the state of the art: O-PrivatePool exploits a simple yet powerful technique to break away from the tailored TPSI in PrivatePool and leverage a regular PSI protocol to achieve a remarkable speed-up in intersection-based itinerary matching. TOPPool extends O-PrivatePool to include the dimension of time in both intersection and endpoint-based matching. In realistic ridesharing contexts, indeed, the feasibility of a ride depends both on the users' locations and on the time of the ride. Therefore, in TOPPool we enable users to determine (*i*) the minimum length of a ride to consider sharing the trip, and more importantly (*ii*) a time interval for the ride to take place, and (*iii*) the maximum discrepancy between their ideal itinerary and the actual shared route.

We have tested the effectiveness and efficiency of our proposals on real-world data from New York's Taxi & Limousine Commission and demonstrated that both O-PrivatePool and TOPPool outperform PrivatePool while achieving the same privacy guarantees and precision level.

TOPPool's encouraging results pave the way for building practical privacy-preserving ridesharing systems in the future. An attractive avenue for future work is thus to develop a fully fledged ridesharing system based on TOPPool and evaluate its scalability on large

numbers of users. Directions for further development also include an investigation of more versatile space and time deviation functions and an analysis of the impact of deploying dynamic delays and multiple routing models in our platform.

Acknowledgments. This work was partly funded by the: Swedish Foundation for Strategic Research (SSF) and the Swedish Research Council (VR); the European Research Council (ERC) under the European Unions's Horizon 2020 research and innovation programme under grant agreement No 669255 (MPCPRO) and No 803096 (SPEC); the Danish Independent Research Council under Grant-ID DFF-6108-00169 (FoCC);

References

[1] U. M. Aïvodji, K. Huguenin, M. Huguet, and M. Killijian. Sride: A privacy-preserving ridesharing system. In *WISEC*, pages 40–50. ACM, 2018.

[2] U. M. Aïvodji, S. Gambs, M.-J. Huguet, and M.-O. Killijian. Meeting points in ridesharing: A privacy-preserving approach. *Transportation Research Part C: Emerging Technologies*, 72:239 – 253, 2016.

[3] F. Baldimtsi and A. Lysyanskaya. Anonymous credentials light. In *ACM Conference on Computer and Communications Security*, pages 1087–1098. ACM, 2013.

[4] C. Bessette. Does Uber Even Deserve Our Trust? http://www.forbes.com/sites/chanellebessette/2014/11/25/does-uber-even-deserve-our-trust/, Nov. 2014.

[5] BlaBlaCar - Trusted carpooling. https://www.blablacar.com/.

[6] R. Chen, B. C. M. Fung, and B. C. Desai. Differentially private trajectory data publication. *CoRR*, abs/1112.2020, 2011.

[7] M. Ciampi and C. Orlandi. Combining private set-intersection with secure two-party computation. In *Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings*, pages 464–482, 2018.

[8] I. Damgørd, M. Geisler, and M. Krøigard. Homomorphic Encryption and Secure Comparison. *Int. J. Appl. Cryptol.*, 1(1):22–31, Feb. 2008.

[9] O. Foundation. OpenStreetMap. https://www.openstreetmap.org/.

[10] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 1–19, 2004.

[11] D. Freni, C. R. Vicente, S. Mascetti, C. Bettini, and C. S. Jensen. Preserving location and absence privacy in geo-social networks. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 309–318, 2010.

[12] M. Furuhata, M. Dessouky, F. Ordóñez, M.-E. Brunet, X. Wang, and S. Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57:28 – 46, 2013.

[13] General Data Protection Regulation, EU Regulation 2016/679, 2018.

[14] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178, 2009.

[15] G. Ghinita. Private queries and trajectory anonymization: a dual perspective on location privacy. *Trans. Data Privacy*, 2(1):3–19, 2009.

[16] P. Hallgren, M. Ochoa, and A. Sabelfeld. InnerCircle: A parallelizable decentralized privacy-preserving location proximity protocol. In *2015 13th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6, July 2015.

[17] P. Hallgren, C. Orlandi, and A. Sabelfeld. PrivatePool: Privacy-Preserving Ridesharing. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 276–291, Aug 2017.

[18] P. A. Hallgren, M. Ochoa, and A. Sabelfeld. Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol. In *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21-23, 2015*, pages 1–6, 2015.

[19] P. A. Hallgren, M. Ochoa, and A. Sabelfeld. Maxpace: Speed-constrained location queries. In *2016 IEEE Conference on Communications and Network Security, CNS 2016, Philadelphia, PA, USA, October 17-19, 2016*, 2016.

[20] C. Hazay and Y. Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010.

[21] Y. He, J. Ni, X. Wang, B. Niu, F. Li, and X. Shen. Privacy-preserving partner selection for ride-sharing services. *IEEE Trans. Vehicular Technology*, 67(7):5994–6005, 2018.

[22] Y. Huang, D. Evans, and J. Katz. Private set intersection: Are garbled circuits better than custom protocols? In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

[23] K. Järvinen, Á. Kiss, T. Schneider, O. Tkachenko, and Z. Yang. Faster privacy-preserving location proximity schemes. In *CANS*, volume 11124 of *Lecture Notes in Computer Science*, pages 3–22. Springer, 2018.

[24] H. Kikuchi and K. Takahashi. Zipf distribution model for quantifying risk of re-identification from trajectory data. In *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21-23, 2015*, pages 14–21, 2015.

[25] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu. Efficient batched oblivious prf with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 818–829. ACM, 2016.

[26] V. Kolesnikov, R. Kumaresan, M. Rosulek, and N. Trieu. Efficient batched oblivious PRF with applications to private set intersection. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security,*

Vienna, Austria, October 24-28, 2016, pages 818–829, 2016.

[27] J. Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[28] M. Li, L. Zhu, and X. Lin. Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing. *IEEE Internet of Things Journal*, pages 1–1, 2018.

[29] Lyft. https://www.lyft.com/.

[30] S. Mascetti, D. Freni, C. Bettini, X. S. Wang, and S. Jajodia. Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. *VLDB J.*, 20(4):541–566, 2011.

[31] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location privacy via private proximity testing. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*, 2011.

[32] P. Paillier. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In J. Stern, editor, *Advances in Cryptology — EUROCRYPT '99*, pages 223–238, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[33] A. C. Pesara, V. Patil, and P. K. Atrey. Secure computing of gps trajectory similarity: A review. In *Proceedings of the 2Nd ACM SIGSPATIAL Workshop on Recommendations for Location-based Services and Social Networks*, LocalRec'18, pages 3:1–3:7, New York, NY, USA, 2018. ACM.

[34] A. Pham, I. Dacosta, G. Endignoux, J. R. Troncoso-Pastoriza, K. Huguenin, and J. Hubaux. Oride: A privacy-preserving yet accountable ride-hailing service. In *USENIX Security Symposium*, pages 1235–1252. USENIX Association, 2017.

[35] A. Pham, I. Dacosta, B. Jacot-Guillarmod, K. Huguenin, T. Hajar, F. Tramèr, V. D. Gligor, and J. Hubaux. Privateride: A privacy-enhanced ride-hailing service. *PoPETs*, 2017(2):38–56, 2017.

[36] B. Pinkas, T. Schneider, G. Segev, and M. Zohner. Phasing: Private set intersection using permutation-based hashing. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, pages 515–530, 2015.

[37] B. Pinkas, T. Schneider, O. Tkachenko, and A. Yanai. Efficient circuit-based psi with linear communication. In *Advances in Cryptology - EUROCRYPT 2019, International Conference on the Theory and Applications of Cryptographic Techniques*, 2019.

[38] B. Pinkas, T. Schneider, and M. Zohner. Faster private set intersection based on OT extension. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 797–812, 2014.

[39] B. Pinkas, T. Schneider, and M. Zohner. Scalable private set intersection based on ot extension. Cryptology ePrint Archive, Report 2016/930, 2016. https://eprint.iacr.org/2016/930.

[40] A. C. D. Resende and D. de Freitas Aranha. Faster unbalanced private set intersection. Cryptology ePrint Archive, Report 2017/677, 2017. https://eprint.iacr.org/2017/677.

[41] R. L. Rivest, L. Adleman, and M. L. Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, 1978.

[42] Routino : Router for openstreetmap data. http://www.routino.org/, 2018.

[43] J. Sedenka and P. Gasti. Privacy-preserving distance computation and proximity testing on earth, done right. In *9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, Kyoto, Japan - June 03 - 06, 2014*, pages 99–110, 2014.

[44] A. B. T. Sherif, K. Rabieh, M. M. E. A. Mahmoud, and X. Liang. Privacy-preserving ride sharing scheme for autonomous vehicles in big data era. *IEEE Internet of Things Journal*, 4(2):611–618, 2017.

[45] C. Shu. Uber reportedly tracked Lyft drivers using a secret software program named 'Hell'. https://techcrunch.com/2017/04/12/hell-o-uber/, 2017.

[46] L. Siksnys, J. R. Thomsen, S. Saltenis, and M. L. Yiu. Private and flexible proximity detection in mobile social networks. In *Eleventh International Conference on Mobile Data Management, MDM 2010, Kanas City, Missouri, USA, 23-26 May 2010*, pages 75–84, 2010.

[47] L. Siksnys, J. R. Thomsen, S. Saltenis, M. L. Yiu, and O. Andersen. A location privacy aware friend locator. In *Advances in Spatial and Temporal Databases, 11th International Symposium, SSTD 2009, Aalborg, Denmark, July 8-10, 2009, Proceedings*, pages 405–410, 2009.

[48] S. Stirbys, O. A. Nabah, P. A. Hallgren, and A. Sabelfeld. Privacy-preserving location-proximity for mobile apps. In *PDP*, pages 337–345. IEEE Computer Society, 2017.

[49] M. Terrovitis. Privacy preservation in the dissemination of location data. *SIGKDD Explorations*, 13(1):6–18, 2011.

[50] The City of New York. Taxi and Limousine Commission trip data. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page, 2016.

[51] N. Trieu. Github - osu-crypto/BaRK-OPRF: Efficient Batched Oblivious PRF with Applications to Private Set Intersection (CCS 2016). https://github.com/osu-crypto/BaRK-OPRF.

[52] Uber technologies inc. https://www.uber.com/.

[53] G. Zhong, I. Goldberg, and U. Hengartner. Louis, lester and pierre: Three protocols for location privacy. In *Privacy Enhancing Technologies, 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007, Revised Selected Papers*, pages 62–76, 2007.

A Details on timed endpoint-based matching

Before proceeding with the formal presentation of our time-aware endpoint-based matching, we clarify the statement given in Section 2.2 about the false positives matched returned by this process. Figure 7 depicts a special pair of trips and a specific choice of the values Δ and t for which endpoint-based matching would return match found, but condition 1 in Definition 5 is not satisfied.

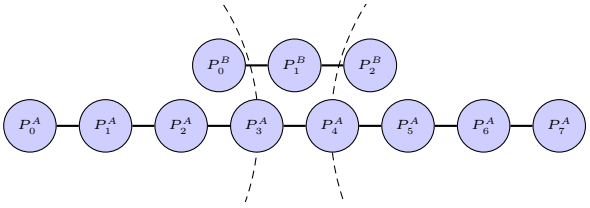

Fig. 7. Example of two rides that give a false positive match when performing endpoint-based matching. The two dashed sections of circles represent the spatial deviation values Δ set by Alice at her starting and ending points. Since Bob's starting and ending points lie within Alice's spatial deviation preferences, endpoint-based matching would return a match. However, if Alice's threshold value t is set anything higher than 40%, Bob's route is not a feasible segment for ridesharing.

The timedEP protocol. Time-aware endpoint-based matching aims at discovering whether two timed trips begin and end at points that are close enough in terms of space and time. Our timedEP is a two party protocol that works as follows.

Step 1: Alice identifies the starting and ending points and times of her trip:

$$P_s^A = P_0^A = (x_0^A, y_0^A),\ P_f^A = P_{n_A-1}^A = (x_{n_A-1}^A, y_{n_A-1}^A),$$
$$T_f^A = T_{n_A-1}^A = z_{n_A-1}^A,\ \ T_s^A = T_0^A = z_0^A.$$

In addition, Alice encrypts her components using an additive homomorphic encryption scheme and her public key. Concretely, Alice generates the ciphertexts:

$$\zeta_0^A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, z_0^A),\quad \zeta_{n_A-1}^A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, z_{n_A-1}^A),$$
$$\xi_0^A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, 2x_0^A),\quad \xi_{n_A-1}^A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, 2x_{n_A-1}^A)$$
$$\gamma_0^A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, y_0^A),\quad \gamma_{n_A-1}^A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, y_{n_A-1}^A),$$
$$\eta_0^A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, (x_0^A)^2 + (y_0^A)^2),$$
$$\eta_{n_A-1}^A \leftarrow \mathsf{Enc}(\mathsf{pk}_A, (x_{n_A-1}^A)^2 + (y_{n_A-1}^A)^2).$$

Note that Alice also encrypts the squares of her spatial components. This is to enable Bob to compute the distance between his endpoints and Alice's in a homomorphic way using AHE only (and not more expensive homomorphic encryption schemes supporting evaluation of higher degree functions). Computing directly the distance between Alice's and Bob's spatial and temporal coordinates, however, would leak too much information to Alice. Therefore, in this step Alice additionally sends her flexibility in terms of location and time at around her starting and ending points. In detail, she sends to Bob all of the above ciphertexts and the following four plaintext values:

$$\lambda_{\mathsf{max},s} = \Delta_{\mathsf{space}}(\mathcal{T}_{\mathsf{time}}^A; P_s^A),\ \tau_{\mathsf{max},s} = \Delta_{\mathsf{time}}(\mathcal{T}_{\mathsf{time}}^A; T_s^A),$$
$$\lambda_{\mathsf{max},f} = \Delta_{\mathsf{space}}(\mathcal{T}_{\mathsf{time}}^A; P_f^A),\ \tau_{\mathsf{max},f} = \Delta_{\mathsf{time}}(\mathcal{T}_{\mathsf{time}}^A; T_f^A).$$

Step 2: Bob receives Alice's ciphertexts and her flexibility values and uses his own plaintext coordinates to homomorphically evaluate the Euclidean distance between his starting (ending) point and Alice's, and the time differences. In detail, Bob identifies the starting and ending points and times of his trip:

$$P_s = P_0^B = (x_0^B, y_0^B), \qquad T_s = T_0^B = z_0^B,$$
$$P_f = P_{n_B-1}^B = (x_{n_B-1}^A, y_{n_B-1}^B), \quad T_f = T_{n_B-1}^B = z_{n_B-1}^B.$$

In addition, Bob computes:

$$\begin{aligned}
D_{xy,s} &= \eta_0^A \oplus \mathsf{Enc}(\mathsf{pk}_A, (x_0^B)^2 + (y_0^B)^2) \\
&\quad \ominus (x_0^B \odot \xi_0^A) \ominus (y_0^B \odot \gamma_0^A), \\
D_{xy,f} &= \eta_{n_A-1}^A \\
&\quad \oplus \mathsf{Enc}(\mathsf{pk}_A, (x_{n_B-1}^B)^2 + (y_{n_B-1}^B)^2) \\
&\quad \ominus (x_{n_B-1}^B \odot \xi_{n_A-1}^A) \ominus (y_{n_B-1}^B \odot \gamma_{n_A-1}^A), \\
D_{z,s} &= \mathsf{Enc}(\mathsf{pk}_A, z_0^B) \ominus \zeta_0^A, \\
D_{z,f} &= \mathsf{Enc}(\mathsf{pk}_A, z_{n_B-1}^B) \ominus \zeta_{n_A-1}^A.
\end{aligned}$$

Note that all of the ciphertexts are encrypted using Alice's public key. If, at this point, Bob sent back to Alice the four values he computed she would be able to determine the exact distance between their starting and ending points and the exact times at which Bob plans to be there. This obviously would break the privacy-preservation requirement we want to meet. To overcome this, we require Bob to perform one additional step.

Step 3: Bob protects his inputs by using a solution inspired to the InnerCircle two-party proximity-testing protocol by Hallgren et al. [16], with additional routines dedicated to hiding Bob's input time components. In detail, Bob creates four sets (one for each ciphertext he has) as follows:

$$\begin{aligned}
\mathcal{L}_s &= \{(D_{xy,s} \ominus \mathsf{Enc}(\mathsf{pk}_A, i)) \odot \rho_i \mid i = x \cdot y; x, y \in [0, \lambda_{\mathsf{max},s}]\} \\
\mathcal{W}_s &= \{(D_{z,s} \ominus \mathsf{Enc}(\mathsf{pk}_A, j)) \odot \rho_j \mid j \in [-\tau_{\mathsf{max},s}, \tau_{\mathsf{max},s}]\} \\
\mathcal{L}_f &= \{(D_{xy,f} \ominus \mathsf{Enc}(\mathsf{pk}_A, i)) \odot \rho_i' \mid i = x \cdot y; x, y \in [0, \lambda_{\mathsf{max},f}]\} \\
\mathcal{W}_f &= \{(D_{z,f} \ominus \mathsf{Enc}(\mathsf{pk}_A, j)) \odot \rho_j' \mid j \in [-\tau_{\mathsf{max},f}, \tau_{\mathsf{max},f}]\}
\end{aligned}$$

where the values $\rho_i, \rho_j, \rho_i', \rho_j'$ are taken uniformly at randomly from $\mathbb{Z}_q^*$. In addition, Bob 'scrambles' the values within each set, i.e., he selects four permutations $\sigma_s, \sigma_f \xleftarrow{R} \mathsf{Sym}[\lambda_{\mathsf{max}} + 1]$, $\sigma_s', \sigma_f' \xleftarrow{R} \mathsf{Sym}[2\tau_{\mathsf{max}} + 1]$ and computes:

$$\mathcal{L}_s' = \sigma_s(\mathcal{L}_s) \qquad \mathcal{L}_f' = \sigma_f(\mathcal{L}_f)$$
$$\mathcal{W}_s' = \sigma_s'(\mathcal{W}_s) \quad \mathcal{W}_f' = \sigma_f'(\mathcal{W}_f)$$

Finally, Bob sends to Alice the four sets $\mathcal{L}_s', \mathcal{L}_f', \mathcal{W}_s', \mathcal{W}_f'$.

size	32	62	128	256	512	1024	2048	4096
PP	0.0329	0.2019	1.3906	10.2894	79.0680	618.6747	4893.2446	38827.3026
O-PP (Sec.3)	0.0869	0.0871	0.0867	0.0911	0.0898	0.0913	0.0955	0.0986
TOPP (Sec.5)	0.0917	0.0942	0.0986	0.1026	0.1126	0.1404	0.1994	0.3081

Table 5. Average running times (in seconds) of performing intersection-based trajectory matching in PrivatePool, O-PrivatePool and TOPPool.

Step 4: Alice uses her secret key to decrypt (the elements in) the four sets. If she finds a 0 value in each of the (decrypted) sets, Alice outputs 1 (this is the output of timedEP) for 'match found'. In this case, indeed, it must hold that

$$d_{xy}(P_0^A, P_0^B) < \lambda_{\mathsf{max},s} \qquad d_{xy}(P_{n_A-1}^A, P_{n_B-1}^B) < \lambda_{\mathsf{max},f}$$

i.e., Alice's and Bob's starting and ending locations are close enough, and

$$|z_0^A - z_0^B| < \tau_{\mathsf{max},s} \qquad |z_{n_A-1}^A - z_{n_B-1}^B| < \tau_{\mathsf{max},f}$$

i.e., the desired departure and arrival times are close enough. If, on the other hand, the value 0 does not appear in at least one set, Alice outputs 0 (this is the output of timedEP) for 'no match'.

B Details on timed intersection-based matching

Before diving into the details of our construction, we collect some useful notation. Our definition of feasible ridesharing makes use of a threshold t for matching trajectories. In what follows we adopt the sequence-threshold c^{A} corresponding to t, defined as the minimal number of consecutive points needed in G to realize a segment of length $\ell(\mathcal{S}) \geq t \cdot \ell(\mathcal{T}^A)$. Moreover, to simplify the exposition, we assume Alice and Bob have a constant deviation function for space and for time. In other words, we define λ_{max}^A (resp. λ_{max}^B) to be Alice's (resp. Bob's) flexibility in space at any point on the trajectory and set $\lambda_{\mathsf{max}}^A = \Delta_{\mathsf{space}}(\mathcal{T}_{\mathsf{time}}^A; P_i^A)$ for all $i \in [0, n_A - 1]$ (λ_{max}^B is defined in an analogous way). We also set Alice's (resp. Bob's) time flexibility to be τ_{max}^A (resp. τ_{max}^B) defied as $\tau_{\mathsf{max}}^A = \Delta_{\mathsf{time}}(\mathcal{T}_{\mathsf{time}}^A; T_i^A)$ for all $i \in [0, n_A - 1]$ (τ_{max}^B is defined in an analogous way). These simplifications are only for exposition purposes, the generalization to the case where spatial and time flexibility is different for every point along Alice's trajectory is immediate.

Step 1: Alice extracts from her timed trajectory $\mathcal{T}_{\mathsf{time}}^A = \{(P_0^A, T_0^A) \ldots, (P_{n_A-1}^A, T_{n_A-1}^A)\}$ a set $\mathcal{A}$ of $(2\tau_{\mathsf{max}}^A + 1) \cdot (n_A - \mathsf{c}^{\mathsf{A}} - 1)$ triples of the form $(P_i^A, P_{i+\mathsf{c}}^A, T_{i,p}^A)$ where $i \in [0, n_A - \mathsf{c}^{\mathsf{A}} - 1]$ and $p \in [-\tau_{\mathsf{max}}^A, \tau_{\mathsf{max}}^A]$. Alice sends to Bob the her sequence-threshold value c^{A}.

Step 2: Bob checks that his trip $\mathcal{T}_{\mathsf{time}}^B$ is not too short by comparing $n_B \geq \mathsf{c}^{\mathsf{A}}$. In case $n_B < \mathsf{c}^{\mathsf{A}}$, Bob aborts and timedIS outputs 0 (no match found). Otherwise, Bob extracts from his timed trajectory $\mathcal{T}_{\mathsf{time}}^B = \{(P_0^B, T_0^B) \ldots, (P_{n_B-1}^B, T_{n_B-1}^B)\}$ a set $\mathcal{B}$ of $(2\tau_{\mathsf{max}}^B + 1) \cdot (n_B - \mathsf{c}^{\mathsf{A}} - 1)$ triples of the form $(P_j^B, P_{j+\mathsf{c}^{\mathsf{A}}}^B, T_{j,q}^B)$ where $i \in [0, n_B - \mathsf{c}^{\mathsf{A}} - 1]$ and $q \in [-\tau_{\mathsf{max}}^B, \tau_{\mathsf{max}}^B]$.

Step 3: Alice and Bob run a *regular* PSI using the sets $\mathcal{A}$ and $\mathcal{B}$ as respective inputs.
The output of timedIS is $\emptyset$ (no match found) if PSI returns no common intersection; otherwise the PSI returns $\mathcal{S} = \mathcal{A} \cap \mathcal{B} \neq \emptyset$ (match found).

C Implementation details

Table 5 reports the concrete values measured during our efficiency tests of TOPPool and displayed in Figure 6, Section 6. In detail, PrivatePool and O-PrivatePool were tested under identical conditions. The test suite was run on a dedicated machine and executed ten times without interruption, according to the experimental design. The collected values correspond to the actual running times of the intersection-based methods used in the two protocols, namely TKEM and BaRK-OPRF. Regarding TOPPool, the temporal proximity test was implemented following the deviation functions defined in Section 6 with τ_{max} set to 12 and time deviation precision of 30, 45 and 60 minutes. The first row (size) of Table 5 indicates the maximum number n of coordinates (points, or nodes) along the routes considered for the measurement.