SRide: A Privacy-Preserving Ridesharing System

Ulrich Matchi Aïvodji
Université du Québec à Montréal (UQAM)
Montréal, Canada
aivodji.ulrich@courrier.uqam.ca

Kévin Huguenin
UNIL–HEC Lausanne
Lausanne, Switzerland
kevin.huguenin@unil.ch

Marie-José Huguet
LAAS-CNRS, Université de Toulouse, CNRS, INSA
Toulouse, France
huguet@laas.fr

Marc-Olivier Killijian
CRM/UDeM and LATECE/UQAM, CNRS
Montréal, Québec, Canada
killijian.marc-olivier@uqam.ca

ABSTRACT

Ridesharing, in which drivers offer to share their rides, allows reduction of travel costs for both drivers and riders; such practice is increasingly popular. Modern ridesharing systems, enhanced with location-based features, have improved user experience by enabling drivers and riders to arrange a trip in near real time. However, the fine-grained nature of location data collected by the service providers and exchanged between users raises privacy issues that could disrupt the adoption of such systems. In this paper, we present *SRide*: a privacy-preserving protocol for ridesharing that addresses the matching problem for dynamic ridesharing systems. We design and implement a prototype of *SRide* that operates in four steps. First, it generalizes users spatiotemporal data of users. Next, it relies on a secure filtering protocol to compute feasible matches. Then, it uses an improved version of `Priv-2SP-SP-` a privacy-preserving protocol to compute meeting points for ridesharing- to compute a ridesharing score for each feasible pair. Finally, it computes the optimal assignment of drivers and riders based on their ridesharing scores. We conduct an experimental trace-driven evaluation of the proposed scheme to demonstrate its practical feasibility.

ACM Reference Format:
Ulrich Matchi Aïvodji, Kévin Huguenin, Marie-José Huguet, and Marc-Olivier Killijian. 2018. SRide: A Privacy-Preserving Ridesharing System. In *WiSec '18: Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks, June 18–20, 2018, Stockholm, Sweden.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3212480.3212483

1 INTRODUCTION

With the advent of mobile phones with geo-localization features, dynamic ride-sharing services, which arrange shared rides between a driver and one or multiple riders, have quickly grown over the last decade. *BlaBlaCar*, one of the major ride-sharing services, which operates mostly in Europe, boasts 35 million verified members. In a typical ride-sharing scenario, a driver who intends to drive from a given location to a destination offers to transport other potential riders for a part of, or the whole, journey. In exchange, the riders pay the driver a certain amount of money to (partially) cover the costs of the ride.

Ridesharing, which falls into the broad category of the *collaborative economy* [5], presents numerous advantages for both drivers and riders: It enables drivers to cut down their travel costs and offers a financially-competitive alternative for riders, compared to traditional means of transportations (e.g., train, plane). Also, ridesharing saves time as, in some countries, cars used for ridesharing can drive on so-called high occupancy vehicles lanes [30] which are usually less crowded. Finally, ridesharing reduces CO_2 emissions [8].

Ride-sharing services put drivers and potential riders in contact based on their respective offers and needs [1]; this is achieved by optimizing some criteria, including the drivers and riders total traveling time. To benefit from such services (in their current forms), however, drivers and riders need to communicate personal information to the service, including their departure locations and schedules as well as their destinations. Due to the sensitive nature of this data (e.g., related to monitoring of the location of riders in real-time, and inferencing sensitive information such as points of interest or social ties [15]), this raises serious privacy concerns.

Ridesharing has received substantial attention from the research community over the last few years. In particular, Agatz et al. [1] formalize the ridesharing problem in a dynamic setting and propose several optimization techniques to solve it, Bit-Monnot et al. [6] introduce `2SP-SP`, the two-synchronization points shortest path problem to determine the optimal meeting points (*pick-up* and *drop-off* locations), Stiglic et al. [24, 25] demonstrate that a reasonable increase in flexibility, in terms of desired departure and transit times and locations, results in a significant improvement of the overall performance of ridesharing services (e.g., matching rate).

Unfortunately, most works focus on the optimization problem underlying the matching of drivers and riders, and very few works focus on the privacy aspects of ridesharing. Aivodji et al. [3] propose `Priv-2SP-SP` which enables a driver and a rider to compute near-optimal pick-up and drop-off locations that match their constraints, without revealing their origins and destinations. However, they do not address the matching problem, that is, how to put in contact drivers with potential riders. Running this algorithm between every pair of driver/rider requires a prohibitively high running time; therefore, it is not a viable solution. Sanchez et al. [21] propose a privacy-preserving approach to solve the matching problem. However, they disregard the constraints related to the rider's travel from the origin to the pick-up location and from the drop-off location to the destination (i.e., the rider's *transit*).

WiSec '18, June 18–20, 2018, Stockholm, Sweden
2018. ACM ISBN 978-1-4503-5731-9/18/06.
https://doi.org/10.1145/3212480.3212483

In this paper, we propose SRide, a novel approach that addresses the matching problem for dynamic ridesharing in a privacy-preserving way, including the computation of the pick-up and drop-off times and locations, considering multiple means of transportation for the rider's transit, i.e., multimodal routing. More specifically, we propose a new privacy-preserving protocol, relying on existing well-established techniques, combined to implement data minimization at reasonable CPU and bandwidth costs: *homomorphic encryption*, *secure multiparty computation*, and *assignment or routing optimization methods*. The proposed solution, SRide, operates in four stages. In the first stage, riders and drivers apply spatiotemporal generalization to their private inputs. During the second stage, the set of potential drivers/rider matching pairs is reduced using a new secure filtering protocol. In the third stage, a two-party protocol is executed between feasible pairs, to determine meeting times and locations, as well as the overall quality of the match (i.e., compatibility scores), in a privacy-preserving way. Finally, in the last stage, a matching algorithm is run by the service provider to pair up drivers and riders, based solely on the scores computed in the third stage. The proposed approach offers desirable privacy properties; in particular, limited information disclosure to the service provider and between only a small number of drivers/riders pairs.

We analyze the privacy properties of our solution and evaluate its performance by using synthetic traces generated from a real dataset collected from a popular ridesharing service. In particular, our trace-driven experimental results show that our privacy-preserving protocol is an order of magnitude faster than a brute force approach that computes the secure meeting points protocol Priv-2SP-SP on the *complete* bipartite graph formed by the divers and the riders.

The rest of this paper is organized as follows. In Section 2, we survey the related works in the areas of ridesharing and privacy enhancing technologies for transportation. In Section 3, we describe the system model, and we formalize the private ridesharing problem. In Section 4, we give some background about the key techniques we use in our solution. We detail the proposed approach SRide in Section 5. In Section 6, we describe our experimental setup and methodology, including the datasets used, and we report on our experimental results. We conclude the paper in Section 8.

2 RELATED WORK

In this section, we survey related work, focusing on two areas: transportation and ridesharing.

2.1 Privacy in transportation

In the field of privacy-enhancing technologies for transportation, prior works include transportation modeling and secure navigation services.

Sun et al. [26] proposed a privacy-preserving mechanism to design fine-grained urban traffic modeling using mobile sensors. The proposed method ensures the unlinkability of mobility traces related to users (*i.e.*, it is difficult for an adversary to assign traces to specific users). In the same line of work, Ghasemzadeh et al. [17] devised an advanced anonymization technique to construct privacy-preserving passengers' flow graph based on trajectory data. The proposed approach relies on the so-called *lk-anonymity* which derives from *k-anonymity* [27] but considers that the adversary has prior knowledge, of length l, and ensures that any pattern of length l has at least k occurrences in the released dataset to thwart identity record linkages.

Xi et al. [29] proposed a privacy-preserving shortest path algorithm based on the use of a cryptographic primitive known as *Private Information Retrieval* (PIR) [9]. The proposed approach allows users to query a navigation service provider to obtain the pre-computed shortest path between two locations A and B without disclosing A and B to the navigation service provider. In a relatively similar work, Wu et al. [28] have applied a graph compression algorithm on road networks to improve PIR-based privacy-preserving shortest path computation's runtime. Even though these works are related to ours, none of them can be directly applied to the case of ridesharing. In a *PIR*-based solution as in Xi et al. [29], the SP first builds a database, of drivers' offers, which will be securely queried by the riders. However, this type of solution is suitable only when drivers trust the SP on collecting their location data. In contrary, our solution can be used when the drivers do not trust the SP.

2.2 Privacy in ridesharing and ride-hailing

Aïvodji et al. [3] proposed an approach based on *private set intersection* and *multimodal routing* to securely compute pick-up and drop-off locations for ridesharing users in such a way that private information on users' origin and destination locations are not revealed to a centralized entity or other users. In this seminal work, authors have considered the case of one driver versus one rider and the proposed scheme can guarantee strong security and preserve privacy without sacrificing the usability of ridesharing services. The work presented in this paper considers the case of several drivers and several riders with the same objective to find for each user, the best pick-up and drop-off locations that minimize their trip costs while preserving peers' location privacy.

Sanchez et al. [21] proposed a fully decentralized approach to solve the matching between riders and drivers and also a privacy-preserving distributed protocol for reputation management in ridesharing. In this method, for the matching phase, space and time generalizations are used followed by a publish-subscribe [11] routine that allows drivers to subscribe to topics corresponding to the generalization of spatiotemporal doublets in their trajectory and receive a notification when a rider publishes on the corresponding topics. By contrast, our approach uses spatiotemporal generalization combined with private set intersection to help ridesharing users in searching for potential partners while taking into account the maximal distance they accept to travel before and after the ride.

Pham et al. [20] analyzed the privacy threats for a ride-hailing system and proposed PrivateRide, a solution that enhances location privacy for the riders *w.r.t.* the service provider and privacy for the drivers *w.r.t.* malicious outsiders, while preserving the convenience and functionality offered by the current system. Pham et al. later proposed ORide [19], a ride-hailing system based on somewhat-homomorphic encryption, which addresses most limitations of PrivateRide and provides stronger privacy and accountability guarantees. It should be noted that ride-hailing and ridesharing have fundamental structural differences: while in ridesharing, the vast majority of drivers plan a ride for themselves in the first place

and subsequently offer to *share* the ride with others, in ride-hailing, drivers are professionals and make on-demand rides based on riders' requests; therefore, drivers have relatively strong origin constraints and no route or destination constraints. These systems also differ in terms of use cases and properties of rides. Ride-hailing essentially replaces taxi cabs (short trips, e.g., intra-city) while ridesharing replaces trains and planes (medium/long trips, typically for weekend excursions, commuting or vacations). In this paper, our approach also relies on somewhat-homomorphic encryption to compute feasible matches for each rider, but unlike ORide, feasible matches are first computed as a shared secret between the service provider and the rider. To obtain her feasible matches, each rider engages in a secure comparison protocol with the service provider, in which they compare their secret share. This allows us to prevent the rider from learning information about drivers with whom she does not match. SRide also differs from ORide by the fact that it considers both spatial and temporal information about the users, while ORide does not check arrival time consistency.

3 SYSTEM MODEL

Our objective is to design a ridesharing system that provides strong privacy guarantees to both drivers and riders, without sacrificing the usability of the system. From a high-level perspective, our system involves three entities: drivers, riders, and the service provider.

3.1 Notations

We denote by $\mathcal{U}$, the set of users (drivers and riders), $\mathcal{D}$ the set of drivers, and $\mathcal{R}$ the set of riders ($\mathcal{D} \cup \mathcal{R} = \mathcal{U}$). Let $m_d = |\mathcal{U}|$ and $m_r = |\mathcal{R}|$ respectively denote the number of drivers and riders. Users have a single role: either driver or rider, that is: $\mathcal{D} \cap \mathcal{R} = \emptyset$. We will also denote by $\mathcal{L}$ the set of $n_L = |\mathcal{L}|$ locations and by $\mathcal{H}$ the time horizon for considered instances of ridesharing problem.

Each driver $d \in \mathcal{D}$ has a profile $\mathcal{P}^d = \{T^d, wt^d\}$ where:

- $T^d = \{(O_d, \tau^d_{O_d}), \ldots, (l_k, \tau^d_{l_k}), \ldots, (D_d, \tau^d_{D_d})\}$ is her trajectory containing an origin O_d, a destination D_d, and a set of intermediate locations l_k, along with their respective arrival times $\tau^d_{l_k}$.
- her maximum flexibility wt^d (*i.e.*, waiting time).

The maximum number of intermediate locations on the drivers' trajectories is denoted by $n_T = \max_{d \in \mathcal{D}}(|T^d|)$.

Each rider $r \in \mathcal{R}$ has a profile $\mathcal{P}^r = \{(O_r, \tau^r_{O_r}), D_r, \delta^r\}$ containing:

- her origin O_r with the expected departure time $\tau^r_{O_r}$.
- her destination D_r.
- her maximum transit distance δ^r.

3.2 Adversarial model

In the ridesharing system under study, we consider two types of adversaries:

- **System users (riders, drivers)** of the ridesharing system, who can try to infer private information (origin, destination, preferences) about other users.
- **Service provider** hereafter referred to as SP, who tries to learn the profile of its users and to collect their origins and destinations.

The security goals are as follows. The SP should not be able to identify a rider or a driver based on the information it receives from them; also, neither a driver nor a rider should be able to infer the origin and destination of other users. Finally, the drivers, the riders the service provider are assumed to be *honest-but-curious*. More precisely, the adversaries know the area on which the ridesharing system operates and will follow the protocol but they will try to obtain/infer additional information about other users by observing and reasoning about information exchanged during the protocol.

3.3 Types of ridesharing systems

Our approach is designed to support common types of ridesharing implemented by state-of-the-art matching services, namely *identical ridesharing* and *inclusive ridesharing* [14].

In the *identical ridesharing* setting, riders, and drivers have the same origin and destination locations. A match can occur between a driver d and a rider r, if:

(1) $dist(O_d, O_r) \leq \delta^r$
(2) $dist(D_d, D_r) \leq \delta^r$
(3) $\tau^r_{O_d} - \tau^d_{O_d} \leq wt^d$

where $dist(,)$ represents the distance between two locations (typically the traveling distance of the user given a transportation network and her transportation modes).

In the *inclusive ridesharing* setting, riders may be picked-up and dropped-off along drivers' itineraries. A match can occur between a driver d and a rider r, if $\exists\, (l_k, \tau^d_{l_k})$ and $(l_{k'}, \tau^d_{l_{k'}}) \in T^d$ with $\tau^d_{l_{k'}} > \tau^d_{l_k}$ such that:

(1) $dist(l_k, O_r) \leq \delta^r$
(2) $dist(l_{k'}, D_r) \leq \delta^r$
(3) $\tau^r_{l_k} - \tau^d_{l_k} \leq wt^d$

In both contexts, the first two conditions capture the fact that the prior (respectively posterior) transit of the rider must be less or equal to her maximum transit. The third constraint captures consistency between the rider's arrival time at the pick-up point and the driver's departure time.

Furthermore, identical ridesharing is a particular case of inclusive ridesharing, in which the pick-up (respectively drop-off) point is the driver origin (respectively destination). In this paper, we will focus on inclusive ridesharing.

4 TECHNICAL BACKGROUND

In this section, we briefly describe the key techniques we use in the design and implementation of our system.

4.1 Homomorphic Encryption

A *Homomorphic Encryption (HE)* scheme [16] allows computations to be made on encrypted data without requiring access to the decryption key. *Fully Homomorphic Encryption (FHE)* schemes enable arbitrary computations on encrypted inputs, whereas *Somewhat Homomorphic Encryption (SHE)* schemes only support a bounded number of operations, after which the ciphertext becomes too "noisy", and can no longer be decrypted correctly. In this paper, we solely

rely on SHE schemes, as we only need to perform a limited number of homomorphic operations, and benefit from the better performance of SHE schemes over FHE schemes.

4.2 Secure multiparty computation and secret sharing protocols

Secure multiparty computation (SMC) protocols aim at computing a function depending on the inputs of several parties in a distributed manner, so that only the result of the computation is revealed while the inputs of each party remain secret. Secret sharing subroutines are widely used to implement SMC protocols. Introduced in the 70's [7, 23], secret sharing protocols are cryptographic protocols that allow a party to share a private input (called the secret) with other parties, each of which receives a share of the secret. The secret can be reconstructed only when all (or a subset of) the shares are combined. That is, individual shares are of no use on their own. In the following we discuss two commons secret sharing protocols in the two-party computation setting, namely the *arithmetic secret sharing* and the *boolean secret sharing*. In an arithmetic secret sharing protocol, the secret is an integer $s \in \mathbb{Z}_n$. In this setting, the secret owner generates a random integer $r_A \in \mathbb{Z}_n$, computes $r_B \equiv s - r_A \mod n$, and sends it to the second party. In a boolean secret sharing scheme, the secret is assumed to a binary number, and each bit of this binary number is shared in $\mod 2$ between both parties.

In this paper, for each rider, we compute the feasible matches as a secret shared between the service provider and the rider. To achieve this goal, we implement an arithmetic secret sharing protocol which relies on a SHE scheme for the computation of the secret shares of each party. Finally, a secure two-party equality test is used to compare secret shares of both the rider and the service provider, and to determine whether or not there is a match between the rider and a particular driver.

4.3 Secure determination of meeting points

Aïvodji et al. have introduced `Priv-2SP-SP`, a privacy-preserving protocol to securely compute meeting points for 2 participants in dynamic detour ridesharing systems [3]. By *securely*, we mean the origin and destination location are not revealed during the process. The proposed approach relies on a secure two-party computation protocol which combines private set intersection and multimodal routing, and allows a pair of driver and rider to compute optimal pick-up and drop-off location such that the overall journey duration of both users is minure variant of `2SP-SP` method [6], which allows computing optimal meeting points without privacy-related constraints.

The `Priv-2SP-SP` protocol runs in two major steps, namely computation of shared preferences and optimization. In the first step, both the driver and the rider use a multimodal routing algorithm to identify their respective potential meeting points as well as the corresponding travel time (considering the maximum driver detour and the maximum rider transit distance) and keep this data private. Then, they use a private set intersection (`PSI`) [13] protocol to determine common pick-up (respectively drop-off) locations. In the second phase, both users assign a score to each pair (i, j) of common pick-up and drop-off locations. The score associated with a pair reflects its contribution to the overall journey duration of each user and is based on shortest path computations, including multimodal aspects for the riders. That is, a pair (i, j) gets the highest score if and only if it minimizes the overall trip duration.

In this paper, we use `Priv-2SP-SP` to securely compute ridesharing cost for each feasible pair of driver and rider over a set of previously known ridesharing points.

5 PROPOSED APPROACH

A naive solution to address the privacy concerns related to the matching problem in ridesharing is to consider the *complete* bipartite graph formed by all the drivers and riders, and run the secure ridesharing protocol `Priv-2SP-SP` [3] between every pair. The corresponding bipartite graph has $m_d \times m_r$ arcs *i.e.,* one arc per pair of driver and rider weighted by the ridesharing cost. Finally, by using a minimum cost bipartite matching algorithm, we compute the optimal assignment for drivers and riders. However, running `Priv-2SP-SP` for the complete bipartite graph is too expensive. It implies m_d peer-to-peer communication for each rider to interact with the drivers, which can introduce important communication and computation overheads. For instance, it takes a given pair of driver and rider about 670 milliseconds in the intracity scenario studied in [3]. In order to address these limitations, the proposed algorithm reduces the size of the bipartite graph by securely computing feasible pairs of drivers and riders, *i.e.,* pairs satisfying conditions for inclusive ridesharing. More precisely, we propose a secure pre-filtering protocol to remove pairs that are quite unlikely to match. Then, we run `Priv-2SP-SP` on the *feasible* bipartite graph before computing optimal assignments by using ridesharing costs.

5.1 General overview

From a high-level perspective, the proposed approach `SRide` is composed of four modules (see Figure 1).

To use `SRide`, whenever a rider is looking for a ride, she uses the *generalization* module to generalize her private inputs (see Section 5.2). Then, she uses the *secure filtering* module to initiate a *secure filtering protocol* (see Section 5.3) with the service provider and the drivers. The secure filtering protocol determines the subset of drivers with whom the rider can travel. More precisely, potential drivers are drivers that visit the pick-up area of the rider at the same epoch, and whose itineraries pass through the rider's drop-off area. Once each rider learns her potential drivers, she relies on the *secure scoring* module to launch the `Priv-2SP-SP` protocol with each of her feasible drivers, and finally the `SP` uses the *matching* module to determine optimal assignments of drivers and riders (see Section 5.4).

5.2 Generalizations of inputs

5.2.1 Time and Space generalization. Our goal is to help drivers and riders in finding matches while keeping a minimal information disclosure. To this end, we propose a matching model that relies on two generalizations, namely *time generalization* and *space generalization*, to capture feasibility constraints.

Time generalization. The total time horizon $\mathcal{H}$ (a day for instance) is split into a set of epoch $\mathcal{E}$ of same length ω: $\mathcal{E} = \{e_t\}, \forall t \in$

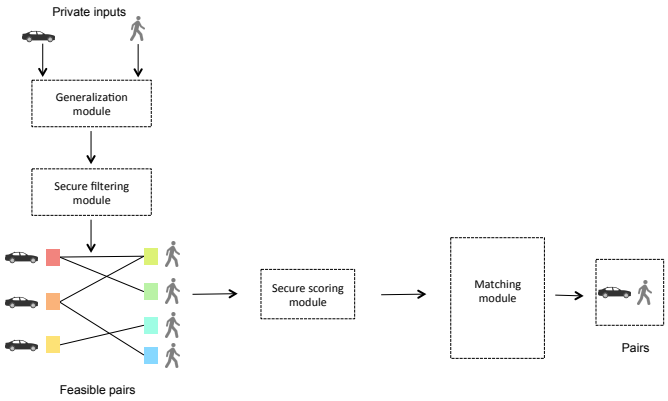

Figure 1: SRide Overview: Participants use their generalized inputs and the `Secure filtering` module to build a bipartite graph of feasible pairs. The `Secure scoring` module is used to securely compute ridesharing scores and meeting points. The `Matching` module takes the bipartite graph as an input and computes the matching that minimizes the total cost.

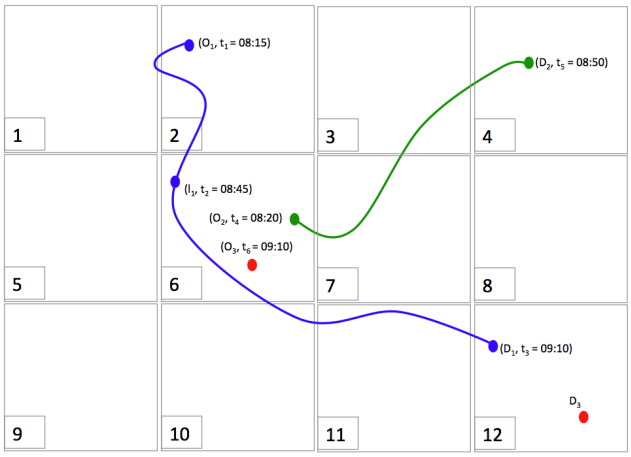

Figure 2: Illustration of the generalization approach: A scenario with one rider (red color) and two drivers (blue and green colors).

$1, \ldots, n_E$. We denote $\phi_\omega(\tau)$ the function that converts a time $\tau \in \mathcal{H}$ to its corresponding epoch $e_t \in \mathcal{E}$.

Space generalization. We consider that the set of locations $\mathcal{L}$ can be divided in a set $C = \{c_s\}_s$ of n_C polygons or cells. We denote $\phi_\theta(l)$ the function that converts a location point $l \in \mathcal{L}$ to its corresponding cell $c_s \in C$.

In these generalizations, we consider that the driver's maximum waiting time wt^d is lower than ω and that the area covered by the rider transit distance δ^r is included in the generalized cells.

Each user (driver d and rider r) computes a **generalized input vector**, denoted respectively by $\mathcal{I}^d$ and $\mathcal{I}^r$, based on their own profile $\mathcal{P}^d$ and $\mathcal{P}^r$.

For every location l_k on his trajectory T^d, a driver d first enumerates all the combination $(l_k, \tau^d_{l_k}, l_{k'})\ \forall k' > k$ of pick-up, departure time and drop-off where $>$ denotes the precedence relation on the trajectory. Overall, a driver d generates an input vector having $|T^d|\times(|T^d|-1)/2$ of such triplets. Finally, each driver computes his generalized input vector $\mathcal{I}^d$ by applying spatial (respectively temporal) generalizations on the spatial (respectively temporal) components of the triplets. The maximal size of drivers' generalized input vector is therefore $n_T \times (n_T - 1)/2$.

The rider's generalized input vector, $\mathcal{I}^r$, is composed of a single generalized spatiotemporal triplet corresponding to the generalization of her origin, departure time and destination.

5.2.2 Illustrative example. Let us consider the scenario of Figure 2 with one rider r and two drivers d_1 and d_2 with the following profiles:

- $\mathcal{P}^{d_1} = \{\{(O_1, t_1), (l_1, t_2), (D_1, t_3)\}, wt^{d_1}\}$
- $\mathcal{P}^{d_2} = \{\{(O_2, t_4), (D_2, t_5)\}, wt^{d_2}\}$
- $\mathcal{P}^r = \{(O_3, t_6), D_3, \delta^r\}$

In this example, arrival times are generalized to the corresponding 30-minute epochs (starting at 0:00). In addition, locations are generalized to the cell in which they fall. The computation of generalized inputs for each user produces the following vectors:

- $\mathcal{I}^{d_1} = \{(c_2, e_{17}, c_6), (c_2, e_{17}, c_{12}), (c_6, e_{18}, c_{12})\}$
- $\mathcal{I}^{d_2} = \{(c_6, e_{17}, c_4)\}$
- $\mathcal{I}^r = \{(c_6, e_{18}, c_{12})\}$

5.3 Secure filtering protocol

5.3.1 Characterization of feasible matches. During the secure filtering phase, the aim is to obtain pairs of riders and drivers that can travel together considering the inclusive ridesharing hypothesis (and thus the inclusive ridesharing).

A match is said feasible between d and r, denoted by $d \Longleftrightarrow r$, if the following conditions hold:

$\exists\ (c^d_s, e^d_t, c^d_{s'}) \in \mathcal{I}^d$ and $\exists\ (c^r_s, e^r_t, c^r_{s'}) \in \mathcal{I}^r$ such that:

(1) $c^d_s = c^r_s$
(2) $e^d_t = e^r_t$
(3) $c^d_{s'} = c^r_{s'}$

The single triplet of the rider's generalized input $\mathcal{I}^r = \{(c^r_s, e^r_t, c^r_{s'})\}$ corresponds to the generalized cell of its origin, the generalized epoch of its time at the origin and the generalized cell of its destination. Then, the first (respectively second) condition means that there is a location on the driver's trajectory that is in the same area as the rider's origin (respectively destination). The third condition means that the epoch of arrival time of the rider at this location is consistent with the driver's epoch of arrival time (and maximum waiting time).

At the end of the secure filtering protocol, one obtains the set of feasible pairs: $\mathcal{F}_\mathcal{M} = \{(d, r) | d \Longleftrightarrow r\}$.

In the example presented in Section 5.2.2, there is a feasible match between the rider r and the driver d_1. In fact, the triplet (c_6, e_{18}, c_{12}) is shared by both users and then satisfy the generalized ridesharing conditions.

We implement a secure filtering protocol to compute feasible matches. In our implementation, we encode private inputs (generalized spatiotemporal triplets) as coefficients of polynomials. From here on, we denote by $P[i]$ the i-th coefficient of a polynomial P. Furthermore, operations on polynomial are coefficient-wise.

5.3.2 Secure Filtering Protocol. The secure filtering protocol, summarized in Figure 3, engages a rider r with its generalized input vector $\mathcal{I}^r$, the service provider SP and all the drivers $d \in \mathcal{D}$ with their respective generalized input vector $\mathcal{I}^d$. To compute feasible matches, both the rider and all the drivers encode their generalized input to an integer to ease the comparison. More precisely, to encode a generalized spatiotemporal triplet (c_i, e_i, c_j), the components of the triplet are converted in their binary form, concatenated together, and the resulting binary number is converted in its decimal form. That is, $\mathsf{encode}(c_i, e_i, c_j) = [[c_i]_2 \mid [e_i]_2 \mid [c_j]_2]_{10}$.

The details of the protocol are summarized as follows:

- The rider r generates a public/private key pair (pk, sk) of a somewhat homomorphic cryptosystem. Then, she creates a m_dth degree polynomial P_r (the degree equals the number of drivers) whose coefficients are identical and correspond to the encoding of her generalized input. That is, $P_r[i] = \mathsf{encode}(\mathcal{I}^r), \forall i = 1 \ldots m_d$. Afterwards, she sends the encrypted version $[[P_r]]$ of her polynomial, and her public key pk to the service provider SP.
- The SP stores the encrypted polynomial of the rider, and forwards her public key to the m_d drivers.
- The j-th driver creates, for her q-th generalized input, a monomial $P^q_{d_j}$ such that $P^q_{d_j} = \mathsf{encode}(\mathcal{I}^{d_j}[q])x^j$, i.e. its j-th coefficient $P^q_{d_j}[j] = \mathsf{encode}(\mathcal{I}^{d_j}[q])$. Overall, $n_T \times (n_T - 1)/2$ of such monomials are created then encrypted with the rider's public key pk, and sent to the SP. For drivers d_j such that $|T^{d_j}| < n_T$, we set $\mathcal{I}^{d_j}[q] = (0, 0, 0)\ \forall q > |T^{d_j}| \times (|T^{d_j}| - 1)/2$.
- The service provider sums (obliviously) the qth encrypted monomial of each driver into a single encrypted m_dth degree polynomial $[[P^q_{\mathcal{D}}]]$ whose jth coefficient corresponds to the qth generalized input of the jth driver. Next, it computes the feasible matches of rider r on the q-th generalized input. The feasible matches are computed as a shared secret between the SP and the rider. More precisely, the SP generates a random m_dth degree polynomial P^q_{SP} corresponding to its share, and computes the share $[[P^q_{\mathcal{D}r}]]$ of the rider r as $[[P^q_{\mathcal{D}r}]] = [[P_r]] - [[P^q_{\mathcal{D}}]] + [[P^q_{\mathsf{SP}}]]$. The SP sends $[[P^q_{\mathcal{D}r}]]$ to the rider and keeps P^q_{SP}. The same operation is repeated for all the $n_T \times (n_T - 1)/2$ encrypted monomials $[[P^q_{d_j}]]$ of the drivers.
- The rider decrypts each of her shares $P^q_{\mathcal{D}r}$ and engages in a secure equality test with the SP to learn her feasible matches. Notice that, whenever $P^q_{\mathcal{D}r}[j] = P^q_{\mathsf{SP}}[j]$, $P_r[j] = P^q_{\mathcal{D}}[j]$, and there is a match between the rider r and the j-th driver d_j on the q-th generalized input. The secure equality test prevents the rider from learning information about drivers with whom she does not match. In fact, the secure equality test returns 1 whenever there is a feasible match, and 0 otherwise.

Overall, the secure filtering protocol is composed of two major steps. In the first step, hereafter referred to as the *secret-sharing subroutine*, the rider and the SP use a homomorphic arithmetic secret sharing protocol to secretly share the feasible matches of the rider. In the second step, hereafter referred to as the *secure two-party equality subroutine*, the rider and the SP use a secure equality test along with their private shares to compute the feasible matches of the rider.

5.4 Secure computation of ridesharing costs

In this section, we describe the privacy-preserving meeting points determination problem and show how we improve Priv-2SP-SP, the privacy-preserving protocol proposed in [3] to solve the problem.

5.4.1 Problem formulation. The problem introduced in [3] considers a driver d, a rider r and a set $\mathcal{S}$ of potential meeting points. Each user $u \in \{d, r\}$ has an origin location O_u, a departure time $\tau^u_{O_u}$, and a destination location D_u. In addition, the rider r is willing to use public transit before and after the ridesharing occurs. For each user $u \in \{d, r\}$ and a couple of meeting point $(i, j) \in \mathcal{S} \times \mathcal{S}$, the traveling cost $Tr_{i-j}(u)$ induced by (i, j) is the time it takes to the user u to complete her journey. The ridesharing cost $\mathsf{Ride}_{i-j}(d, r)$ induced by (i, j) for a couple of user (d, r) is $\mathsf{Ride}_{i-j}(d, r) = Tr_{i-j}(d) + Tr_{i-j}(r) + |\tau^d_i - \tau^r_i|$, where $|\tau^d_i - \tau^r_i|$ denotes the waiting time at the pick-up point i. The problem of privacy-preserving meeting points determination protocol consists in finding a couple of pick-up and drop-off locations $(i^*, j^*) \in \mathcal{S} \times \mathcal{S}$ such that $\mathsf{Ride}_{i^*-j^*}(d, r)$ is the minimum, and the location data (O_u and D_u of each user are protected.

5.4.2 Limitations of Priv-2SP-SP*.* We improve the performance of the Priv-2SP-SP protocol by addressing three main limitations, namely the scalability issue, the consideration of waiting time and the consideration of actual ridesharing duration. In fact, in its current form, waiting times for both the driver and the rider at the pick-up location are not captured by the trip cost model. Furthermore, the authors consider every node in the transportation network as a ridesharing station. Because this assumption increases the computational overhead of the protocol, they use an iterative approach which considers subsets of common meeting points while computing ideal meeting points. Furthermore, a scoring algorithm is used to hide real trip costs. Consequently, the solution found by the original Priv-2SP-SP protocol is not always the optimal one.

5.4.3 Details of our improvements. To tackle the aforementioned limitations, our improvements are threefold.

We first consider that the set of ridesharing stations is a subset of locations known by everyone and propose a speed-up technique based on table lookup. More precisely, given the set $\mathcal{S}$ of potential ridesharing stations, we run $|\mathcal{S}|$ shortest path algorithms to pre-compute the $|\mathcal{S}|^2$ shared paths $i \rightarrow j$ that interconnect the ridesharing stations. This is possible as we assumed that $\mathcal{S}$ is known in advance and $|\mathcal{S}| \ll |V|$, where $|V|$ represents. Then, we rely on a *secure comparison* protocol to integrate the waiting time in the trip cost model. More precisely, for each ridesharing station $i \in \mathcal{S}$ the driver and the rider use a secure comparison protocol to determine arrival order at the station i and update their traveling

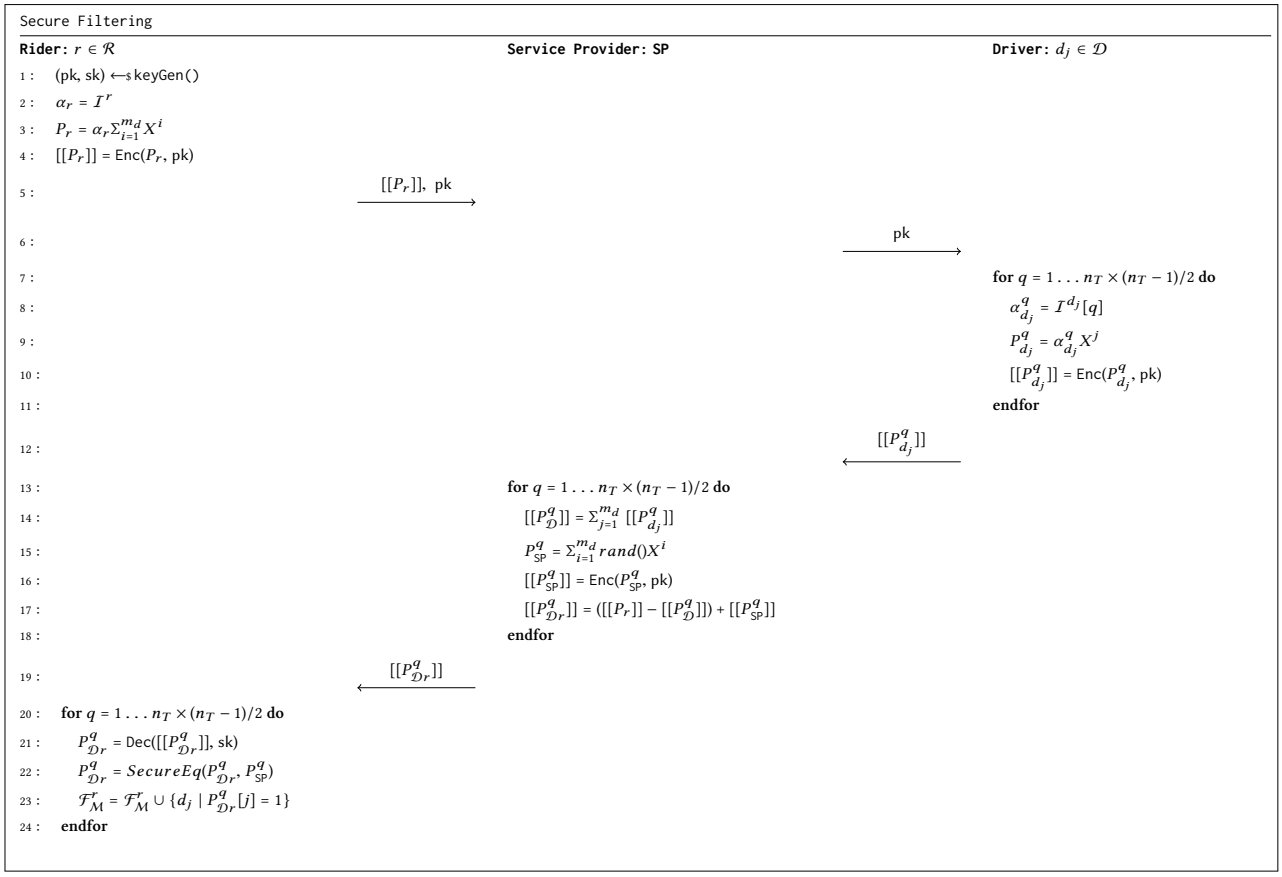

Figure 3: Secure filtering protocol.

costs accordingly. Finally, we use a *secure shared sum* protocol to consider real trip costs instead of scores. More precisely, for each couple of pick-up and drop-off locations (i, j), the ridesharing cost $\texttt{Ride}_{i-j}(d, r)$ is compute as a shared secret between the rider and the driver. Thus, the selection of the best solution is made using secret shares instead of using scores. This allows us to get optimal meeting points.

5.5 Putting it all together : The *SRide* protocol

SRide integrates the four previously detailed modules. First, the service provider SP initiates the system by setting the time and space granularity and generates an empty bipartite graph. Then, each participant generates its generalized inputs. Afterward, the privacy-preserving filtering protocol is used to produce the set of feasible matches. In this protocol, each rider relies on the service provider to compute feasible matches obliviously. Then, each pair of driver and rider having a feasible match securely computes its ridesharing cost with the peer-to-peer Priv-2SP-SP protocol after which both users learn the pick-up and drop-off locations. Finally, the SP centrally solves an assignment problem based on the weighted bipartite graph made of feasible pairs with their ridesharing costs, and notifies users on their matches.

6 PERFORMANCE EVALUATION

In this section, we evaluate the SRide regarding communication and computational overheads.

6.1 Experimental settings

Our experimental dataset was generated from data collected over a 19-month period on the *Covoiturage-libre* platform[1], a popular and openly available ridesharing web service operating in France. The collected data includes pick-up and drop-off cities and trip schedules. On average, there are 468 rides per day. On the busiest day, 1309 rides were scheduled. We generated inter-cities ride-sharing scenarios between the cities of Nantes and Rennes. These two cities (approximately 110 km apart from each other) were chosen because of the high number of rides observed between them in the dataset.

The multimodal graph was obtained by using data from OpenStreetMap[2] for the road network and from *Navitia*[3] (in the GTFS format) for the public transportation network {walk, bus, tramway, car}.

We generated $m_d = 1000$ drivers' offers and $m_r = 1000$ riders' requests. Departure locations are generated in the city of Nantes

[1] http://covoiturage-libre.fr
[2] http://www.openstreetmap.org
[3] https://www.navitia.io/datasets

and arrival locations in the city of Rennes. Each driver's ride offer is a trajectory composed of two locations points (origin and destination) with their corresponding arrival times ($n_T = 2$). Each rider's request is composed of an origin location with a departure time and a destination location. Departure times are uniformly generated between 5 p.m and 6 p.m for both riders and drivers. In our simulation, the number of ride requests, in a two hours period, is greater than what we observe on average for a day period in our real-world dataset.

The time horizons considered are weekdays between $[5p.m\ 8p.m]$. This range is then divided into equal-length epochs of 15 or 30 minutes. Locations are generalized to their district. The city of Nantes is composed of 11 districts while the city of Rennes has 12 districts leading to a total of 23 generalized locations.

For the homomorphic secret sharing protocol, we use the *FV-NFLlib* library,[4] which implements the *FV* scheme. As the degree of the polynomial has to be a power of 2 (see [2]), for the $1,000$ drivers we consider in our experiment, we use a 1024-th degree polynomial to integrate generalized inputs of all the drivers. The coefficients of the polynomial are coded on 64 bits and the modulus p on 124 bits. Therefore, a public key or a ciphertext takes up 31 KB, while a plaintext takes up 8 KB.

For the secure computation of feasible matches protocol, we use the *ABY* framework [10], which implements both the Goldreich-Micali-Wigderson (GMW) protocol [18] and the Yao's garbled circuit protocol [31], with security against passive adversaries. As suggested in [4, 22], we use the GMW protocol to have better runtime and communication performances for the secure two-party computation between the rider and the SP. The private shares used for the secure two-party equality test are 15-bit.

Our experiments are conducted on a Intel Xeon CPU E3-1271 v3 (3.60GHz, 32GB of RAM) running Linux 3.13.

6.2 Experimental results

Our results concern the two main steps of the SRide method: the secure filtering and the secure scoring. Then, we compare SRide to a brute-force approach.

6.2.1 Secure filtering. In this part, we report the communication overhead and the computational overhead of both the secret-sharing and the secure two-party equality subroutines used in the secure filtering protocol to compute feasible matches.

Communication overhead of the secret-sharing subroutine. The secret sharing of feasible matches engages the rider, the SP, and all the drivers. For each ride request, the rider sends to the SP a public key and a ciphertext for her encrypted generalized input. This requires a payload of $2 * 31 = 62$ KB. Next, the SP forwards the public key of the rider to each driver. This requires a payload of 31 KB. Then, each driver encrypts her generalized input with the rider's public key and sends the ciphertext to the SP. This requires a payload of 31 KB. Finally, the SP sends the ciphertext of the shared secret of feasible matches to the rider. This requires a payload of 31 KB.

Communication overhead of the secure two-party equality subroutine. The secure computation of feasible matches engages only the rider and the SP. In this protocol, the two parties run a secure equality on their private shares obtained with the secret sharing protocol. As each party has 1000 private shares, each of which is 15-bit length. Overall, the boolean circuit for this secure equality test has $15 * 1000 = 15000$ inputs, $14 * 1000 = 14000$ *AND* gates, and 1000 outputs. The *ABY* framework sends per *AND* gate 256 bits (128 bit per party) in the setup phase and 4 bits (2 bits per party) in the online phase. Since the circuit has 14000 *AND* gates, each party sends/receives $14000 * 128/8 = 224000$ bytes in the setup phase. In the online phase, each party sends 1 bit per input and 1 bit per output. Hence, we have $15000/8 + 1000/8 + 14000 * 2/8 = 5500$ bytes in the online phase.

Overall communication overhead. A detailed summary of the communication overhead for a ride request is given in Table 1. Overall, the secure filtering introduces a small communication overhead. In fact, the total bandwidth is under 70 KB (respectively 40 KB) for the rider (respectively the driver). As all the private inputs are encoded using the same number of bits, the time generalization does not impact the communication overhead.

Computational overhead of the secret-sharing subroutine. A summary of the different cryptographic operations of each party is given in Table 2. Overall, there are 5 cryptographic operations, namely the key generation (by the rider), the encryption of generalized inputs (by both rider and driver), the oblivious computation of the shared secret of feasible matches (by the SP) and the decryption of the shared secret of feasible matches (by the rider). To summarize, it takes about 519 ms for a rider to obtain secret shares of her feasible matches.

Computational overhead of the secure two-party equality subroutine. In the setup phase, the secure equality takes about 4 ms to complete. The online phase takes only 1 ms to complete.

Overall computational overhead. A summary of all the subroutines of the secure filtering protocol is given in Table 2. Overall, the computational overhead of the secure filtering protocol is very small (520 ms).

Scalability. As expected, the performance in terms of communication and computation behave well as the number of drivers rises. Table 3 shows that computation time for the three different entities during the secret sharing part of the protocol scales with $O(m_d \times m_d)$. Table 4 shows that the performance of the secure two-party equality test is linear with m_d.

6.2.2 Feasible Matches. The number $|\mathcal{F}^r_{\mathcal{M}}|$ of feasible matches per rider obtained after the secure filtering protocol for different temporal granularity is given Table 5. It presents the average total number of feasible matches per rider and its standard deviation (avg ± std). To summarize, in the two settings, the number of drivers with whom the rider can share a trip was significantly reduced. In fact, on average, for each ride request, only 4.3% (respectively 2.2%) of the drivers are candidates for ridesharing, when the temporal granularity is set to 30 (respectively 15) minutes. As expected, the finer the temporal granularity, the smaller the set of feasible matches for each rider.

6.2.3 Secure scoring. In this section, we discuss the communication and computational cost of the Priv-2SP-SP protocol used to find effective pick-up and drop-off locations and ridesharing costs.

[4] https://github.com/CryptoExperts/FV-NFLlib

Time	Rider		Driver	
	Up (KB)	Down (KB)	Up (KB)	Down (KB)
30	67.4	36.4	31	31
15	67.4	36.4	31	31

Table 1: Communication overhead of the filtering protocol

Time generalization (min)	Rider			SP	Driver	Rider & SP	Total
	KeyGen	Enc	Dec	HomAdd	Enc	SecureEQ	
15	33 ± 1 ms	4 ± 0 ms	1 ± 0 ms	477 ± 3 ms	4 ± 0 ms	1 ± 0 ms	520 ms
30	33 ± 1 ms	4 ± 0 ms	1 ± 0 ms	477 ± 3 ms	4 ± 0 ms	1 ± 0 ms	520 ms

Table 2: Computational overhead of the secure filtering protocol. The secure filtering protocol engages a rider, the SP, and the 1000 drivers at the same time. Statistics are computed over the 1000 riders.

m_d		100	200	500	1000	8000	10000
Rider	**KeyGen (ms)**	4	8	17	33	288	606
	Enc (ms)	<1	<1	2	4	36	76
	Dec (ms)	<1	<1	1	1	15	52
Driver	**Enc (ms)**	<1	<1	2	4	36	76
SP	**HommAdd (ms)**	6	23	122	477	31177	78103
Total		10	31	144	519	31552	78913

Table 3: Performances of the secret-sharing subroutine

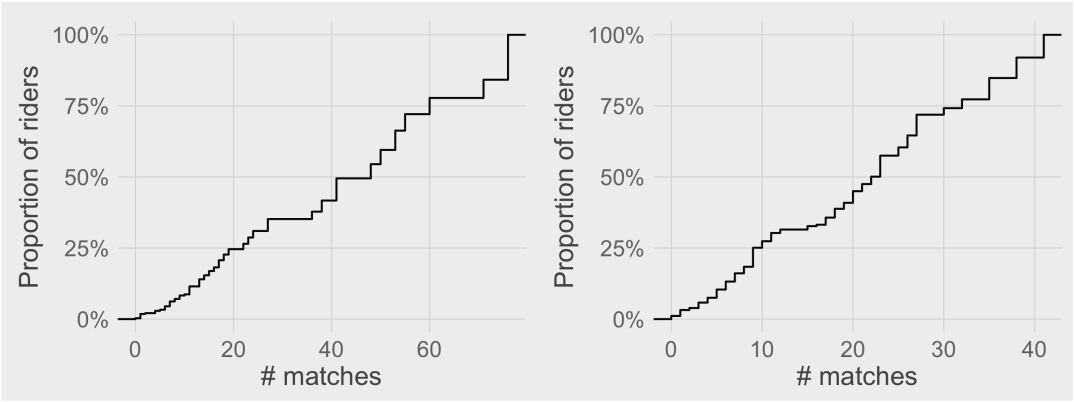

Figure 4: Number of feasible matches per rider for a temporal granularity of 30 minutes (left) and 15 minutes (right)

As suggested in [3], to speed up Priv-2SP-SP and reduce its communication overhead, we consider a small number of vertices as ridesharing stations (64 stations in our instances). By doing so, it takes less than 200 ms for a pair of rider and driver to run the improved Priv-2SP-SP protocol with a total data payload of 132 KB.

6.2.4 Comparison with the naive approach. Overall, using the secure filtering protocol and then the Priv-2SP-SP protocol, it will take about 9 seconds (respectively 5 seconds) to compute feasible matches and their corresponding ridesharing cost, for a time generalization of 30 (respectively 15) minutes. The communication overhead is about 6 MB (respectively 3 MB) for a time generalization of 30 (respectively 15) minutes. To compare, the naive approach (running the secure scoring between each pair of riders and drivers) requires about 3 minutes and 132 MB.

m_d	Comm. (bytes)	Comp. (ms)
100	616	0.394
200	1163	0.461
500	2816	0.535
1000	5554	0.556
8000	44054	1.131
10000	55063	1.286

Table 4: Performances of the secure two-party equality subroutine

Time granularity (min)	Number of feasible matches
30	43 ± 23
15	22 ± 12

Table 5: Number of feasible matches per rider. Statistics are computed over the 1000 riders.

Granularity \ # matches	> 5	> 10	> 15	> 20
15 min	89.6%	72.6%	67.3%	55%
30 min	96.7%	91.3%	83.1%	75.4%

Table 6: Anonymity set for a rider given the number of feasible matches

7 SECURITY AND PRIVACY ANALYSIS

We implement the homomorphic secret-sharing protocol by using the `Fan-Vercauteren (FV)` homomorphic encryption scheme [12]. As all *SHE* schemes, the FV scheme offers semantic security, i.e., it is computationally impossible to distinguish whether two different ciphertexts conceal the same plaintext. For the two-party secure equality test, we implement the secure computation of feasible matches protocol with the Goldreich-Micali-Wigderson (GMW) protocol [18] which provides security against passive adversaries.

7.1 Security of `Priv-2SP-SP` protocol

The protocol `Priv-2SP-SP` engages one driver and one rider. In the new version of this protocol, there are essentially two critical subroutines, namely the secure comparison subroutine used to integrate the waiting time at the pick-up location, and the secure shared sum subroutine used to consider actual trip costs when selecting the optimal meeting points. During the secure comparison protocol, both the driver and the rider learn nothing about the private shares of each other thanks to the privacy guarantee against passive adversaries of the GMW protocol [18]. During the secure shared sum subroutine, the rider receives encrypted inputs from the driver. Hence, she cannot learn anything about the driver's inputs because of the semantic security of the FV scheme [12]. On the other hand, the driver receives random-looking numbers. Hence, she cannot learn anything about the rider's input.

7.2 Confidentiality of the rider

During the secret sharing subroutine, the SP receives the encrypted generalized inputs of the rider. It cannot learn anything about the rider's generalized inputs because of the semantic security of the *FV* encryption scheme. Similarly, the drivers cannot learn anything about the rider except her public key, which is the only information they receive about her. During the secure computation of feasible matches, the SP learns nothing about the private shares of the rider thanks to the privacy guarantee against passive adversaries of the *GMW* protocol [18]. Finally, during the `Priv-2SP-SP` protocol, the driver cannot learn the rider's location data thanks to the security guarantee of the `Priv-2SP-SP` protocol.

During the matching step, the SP receive ridesharing cost of each feasible matches. Similarly to [20], we use the anonymity set to evaluate the k-anonymity of the riders. Figure 4 shows the empirical cumulative distribution function of the number of feasible per rider for a temporal granularity of 30 minutes and 15 minutes. Overall, with a granularity of 30 minutes 50% of the riders have at least 48 feasible matches, and with a granularity of 15 minutes 50% of the riders have at least 22 feasible matches. That is, the finer the temporal granularity, the smaller the anonymity set. In Table 6, we show variations of the anonymity set for a rider given the number of her feasible matches. Overall, 96.7% (respectively 89.6%) of the riders have at least 5 feasible matches with time granularity of 30 minutes (respectively 15 minutes).

7.3 Confidentiality of the driver

During the secret sharing subroutine, the SP receives the encrypted generalized inputs of the drivers. However, as the FV scheme has semantic security, the SP cannot learn anything about drivers' generalized inputs. During the secure computation of feasible matches, the rider learns the generalized input of a driver if and only if there is a match between them; that is, the rider learns nothing about drivers with whom she does not match. Finally, during the `Priv-2SP-SP` protocol, the rider cannot learn the driver's location data thanks to the security guarantee of the `Priv-2SP-SP` protocol.

8 CONCLUSION

In this paper, we have proposed SRide, a practical solution to implement matching in ridesharing systems while protecting the privacy of users against both the service provider and other curious users. We propose a secure filtering subroutine, which relies on homomorphic arithmetic secret sharing and secure two-party equality test, to compute feasible matches. Then, each feasible pair uses the `Priv-2SP-SP` protocol to compute its ridesharing cost which will be used to compute the optimal assignment of riders and drivers. Our experimental analysis shows that our privacy-preserving protocol has acceptable performances for real-world applications.

Future works include more extensive experiments with more realistic ridesharing hypothesis, for instance, drivers' detour or fares, several intermediate locations on the drivers' trajectory, multiple origin locations for the riders.

REFERENCES

[1] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. 2012. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research* 223, 2 (2012), 295–303.

[2] Carlos Aguilar-Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrede Lepoint. 2016. NFLlib: NTT-based Fast Lattice Library. In *RSA Conference Cryptographers' Track*.

[3] Ulrich Matchi Aïvodji, Sébastien Gambs, Marie-José Huguet, and Marc-Olivier Killijian. 2016. Meeting points in ridesharing: A privacy-preserving approach. *Transportation Research Part C: Emerging Technologies* 72 (2016), 239–253.

[4] Gilad Asharov, Daniel Demmler, Michael Schapira, Thomas Schneider, Gil Segev, Scott Shenker, and Michael Zohner. 2017. Privacy-Preserving Interdomain Routing at Internet Scale. *Proceedings on Privacy Enhancing Technologies* 3 (2017), 1–21.

[5] Russell Belk. 2014. You are what you can access: Sharing and collaborative consumption online. *Journal of Business Research* 67, 8 (2014), 1595–1600.

[6] Arthur Bit-Monnot, Christian Artigues, Marie-José Huguet, and Marc-Olivier Killijian. 2013. Carpooling: the 2 synchronization points shortest paths problem. In *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS)*. 12–p.

[7] George Robert Blakley et al. 1979. Safeguarding cryptographic keys. In *Proceedings of the national computer conference*, Vol. 48. 313–317.

[8] Brian Caulfield. 2009. Estimating the environmental benefits of ride-sharing: A case study of Dublin. *Transportation Research Part D: Transport and Environment* 14, 7 (2009), 527–531.

[9] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. 1998. Private information retrieval. *Journal of the ACM (JACM)* 45, 6 (1998), 965–981.

[10] Daniel Demmler, Thomas Schneider, and Michael Zohner. 2015. ABY-A Framework for Efficient Mixed-Protocol Secure Two-Party Computation.. In *NDSS*.

[11] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. The many faces of publish/subscribe. *ACM Computing Surveys (CSUR)* 35, 2 (2003), 114–131.

[12] Junfeng Fan and Frederik Vercauteren. 2012. Somewhat Practical Fully Homomorphic Encryption. *IACR Cryptology ePrint Archive* 2012 (2012), 144.

[13] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. 2004. Efficient private matching and set intersection. In *Advances in Cryptology-EUROCRYPT 2004*. Springer, 1–19.

[14] Masabumi Furuhata, Maged Dessouky, Fernando Ordóñez, Marc-Etienne Brunet, Xiaoqing Wang, and Sven Koenig. 2013. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological* 57 (2013), 28–46.

[15] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. 2014. De-anonymization attack on geolocated data. *J. Comput. System Sci.* 80, 8 (2014), 1597–1614.

[16] Craig Gentry et al. 2009. Fully homomorphic encryption using ideal lattices.. In *STOC*, Vol. 9. 169–178.

[17] Moein Ghasemzadeh, Benjamin CM Fung, Rui Chen, and Anjali Awasthi. 2014. Anonymizing trajectory data for passenger flow analysis. *Transportation Research Part C: Emerging Technologies* 39 (2014), 63–79.

[18] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1987. How to play any mental game. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM, 218–229.

[19] Anh Pham, Italo Dacosta, Guillaume Endignoux, Juan Ramón Troncoso-Pastoriza, Kévin Huguenin, and Jean-Pierre Hubaux. 2017. ORide: A Privacy-Preserving yet Accountable Ride-Hailing Service. In *Proc. of the 26th USENIX Security Symposium*. 1235–1252.

[20] Anh Pham, Italo Dacosta, Bastien Jacot-Guillarmod, Kévin Huguenin, and Jean-Pierre Hubaux. 2017. PrivateRide: A Privacy-Preserving and Secure Ride-Hailing Service. *PoPETs* (2017). To appear.

[21] David Sánchez, Sergio Martínez, and Josep Domingo-Ferrer. 2016. Co-utile P2P ridesharing via decentralization and reputation management. *Transportation Research Part C: Emerging Technologies* 73 (2016), 147–166.

[22] Thomas Schneider and Michael Zohner. 2013. GMW vs. Yao? Efficient secure two-party computation with low depth circuits. In *International Conference on Financial Cryptography and Data Security*. Springer, 275–292.

[23] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[24] Mitja Stiglic, Niels Agatz, Martin Savelsbergh, and Mirko Gradisar. 2015. The benefits of meeting points in ride-sharing systems. *Transportation Research Part B: Methodological* 82 (2015), 36–53.

[25] Mitja Stiglic, Niels Agatz, Martin Savelsbergh, and Mirko Gradisar. 2016. Making dynamic ride-sharing work: The impact of driver and rider flexibility. *Transportation Research Part E: Logistics and Transportation Review* 91 (2016), 190–207.

[26] Zhanbo Sun, Bin Zan, Xuegang Jeff Ban, and Marco Gruteser. 2013. Privacy protection method for fine-grained urban traffic modeling using mobile sensors. *Transportation Research Part B: Methodological* 56 (2013), 50–69.

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

[28] David J. Wu, Joe Zimmerman, Jérémy Planul, and John C. Mitchell. 2016. Privacy-Preserving Shortest Path Computation. *arXiv preprint arXiv:1601.02281* (2016).

[29] Yong Xi, Loren Schwiebert, and Weisong Shi. 2014. Privacy preserving shortest path routing with an application to navigation. *Pervasive and Mobile Computing* 13 (2014), 142 – 149.

[30] Hai Yang and Hai-Jun Huang. 1999. Carpooling and congestion pricing in a multilane highway with high-occupancy-vehicle lanes. *Transportation Research Part A: Policy and Practice* 33, 2 (1999), 139–155.

[31] Andrew Yao. 1986. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*. IEEE, 162–167.