Privacy-enhancing technologies for ridesharing

Ulrich Matchi Aïvodji

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *24/01/2018* **par :**

Ulrich Matchi Aïvodji

Technologies respectueuses de la vie privée pour le covoiturage

`Privacy-Enhancing Technologies for Ridesharing`

JURY

Cyril Briand	Professeur des universités	Université de Toulouse III
Josep Domingo-Ferrer	Professeur distingué	Universitat Rovira i Virgili
Dominique Feillet	Professeur des universités	École des Mines St-Etienne
Françoise Fessant	Ingénieure	Orange Labs
Marie-José Huguet	Professeur des universités	INSA Toulouse
Marc-Olivier Killijian	Directeur de recherche	LAAS-CNRS
Bertrand Le Cun	Ingénieur	Google
Benjamin Nguyen	Professeur des universités	INSA Centre Val de Loire

---

École doctorale et spécialité :
MITT : Domaine STIC : Sureté de logiciel et calcul de haute performance

Unité de Recherche :
LAAS-CNRS (UPR 8001)

Directeur(s) de Thèse :
Marie-José Huguet et *Marc-Olivier Killijian*

Rapporteurs :
Josep Domingo-Ferrer et *Dominique Feillet*

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisors Prof. Marie-José Huguet and Dr. Marc-Olivier Killijian, for giving the opportunity to work on fascinating research topics. Thanks for supporting me throughout these years. Working with you has been an honor and a great pleasure.

I am deeply thankful to my thesis committee members Prof. Cyril Briand, Prof. Josep Domingo-Ferrer, Prof. Dominique Feillet, Dr. Françoise Fessant, Dr. Bertrand Le Cun, and Prof. Benjamin Nguyen, for the time and efforts spent reviewing this thesis, for the valuable comments and the encouraging words.

I am thankful to my collaborators Prof. Sébastien Gambs, Prof. Kévin Huguenin, and Prof. Jean-Marc Robert, for their contributions to this thesis. Special thanks go to Prof. Sébastien Gambs and Prof. Jean-Marc Robert for welcoming me in their labs in Montreal during summer 2016, and for the fruitful discussions we had.

I am thankful to all the members of ROC and TSF teams in LAAS-CNRS who has provided a friendly environment during my thesis.

I am thankful to everyone who has helped me in one way or another during this thesis.

I am thankful to my family: my parents, my brothers, my sisters, my uncle Jean, and his wife Rose, for all their support and love. My heartfelt thanks go to Daniella for her continuous support during this thesis, for her love, and the happiness she has brought into my life.

Abstract — The emergence of mobile phones and connected objects has profoundly changed our daily lives. These devices, thanks to the multitude of sensors they embark, allow access to a broad spectrum of services. In particular, position sensors have contributed to the development of location-based services such as navigation, ridesharing, real-time congestion tracking...

Despite the comfort offered by these services, the collection and processing of location data seriously infringe the privacy of users. In fact, these data can inform service providers about points of interests (home, workplace, sexual orientation), habits and social network of the users. In general, the protection of users' privacy can be ensured by legal or technical provisions. While legal measures may discourage service providers and malicious individuals from infringing users' privacy rights, the effects of such measures are only observable when the offense is already committed and detected. On the other hand, the use of privacy-enhancing technologies (PET) from the design phase of systems can reduce the success rate of attacks on the privacy of users.

The main objective of this thesis is to demonstrate the viability of the usage of PET as a means of location data protection in ridesharing services. This type of location-based service, by allowing drivers to share empty seats in vehicles, helps in reducing congestion, CO_2 emissions and dependence on fossil fuels. In this thesis, we study the problems of synchronization of itineraries and matching in the ridesharing context, with an explicit consideration of location data (origin, destination) protection constraints.

The solutions proposed in this thesis combine multimodal routing algorithms with several privacy-enhancing technologies such as homomorphic encryption, private set intersection, secret sharing, secure comparison of integers. They guarantee privacy properties including anonymity, unlinkability, and data minimization. In addition, they are compared to conventional solutions, which do not protect privacy. Our experiments indicate that location data protection constraints can be taken into account in ridesharing services without degrading their performance.

Key words: privacy, ridesharing, privacy enhancing technologies

Résumé — L'émergence des téléphones mobiles et objets connectés a profondément changé notre vie quotidienne. Ces dispositifs, grâce à la multitude de capteurs qu'ils embarquent, permettent l'accès à un large spectre de services. En particulier, les capteurs de position ont contribué au développent des services de localisation tels que la navigation, le covoiturage, le suivi de la congestion en temps réel...

En dépit du confort offert par ces services, la collecte et le traitement des données de localisation portent de sérieuses atteintes à la vie privée des utilisateurs. En effet, ces données peuvent renseigner les fournisseurs de services sur les points d'intérêt (domicile, lieu de travail,

orientation sexuelle), les habitudes ainsi que le réseau social des utilisateurs. D'une façon générale, la protection de la vie privée des utilisateurs peut être assurée par des dispositions légales ou techniques. Même si les mesures d'ordre légal peuvent dissuader les fournisseurs de services et les individus malveillants à enfreindre le droit à la vie privée des utilisateurs, les effets de telles mesures ne sont observables que lorsque l'infraction est déjà commise et détectée. En revanche, l'utilisation des technologies renforçant la protection de la vie privée (*PET* pour *Privacy Enhancing Technologies*) dès la phase de conception des systèmes permet de réduire le taux de réussite des attaques contre la vie privée des utilisateurs.

L'objectif principal de cette thèse est de montrer la viabilité de l'utilisation des PET comme moyens de protection des données de localisation dans les services de covoiturage. Ce type de service de localisation, en aidant les conducteurs à partager les sièges vides dans les véhicules, contribue à réduire les problèmes de congestion, d'émissions et de dépendance aux combustibles fossiles. Dans cette thèse, nous étudions les problèmes de synchronisation d'itinéraires et d'appariement relatifs au covoiturage avec une prise en compte explicite des contraintes de protection des données de localisation (origine, destination).

Les solutions proposées dans cette thèse combinent des algorithmes de calcul d'itinéraires multimodaux avec plusieurs techniques de protection de la vie privée telles que le chiffrement homomorphe, l'intersection sécurisée d'ensembles, le secret partagé, la comparaison sécurisée dentier. Elles garantissent des propriétés de protection de vie privée comprenant l'anonymat, la non-chainabilité et la minimisation des données. De plus, elles sont comparées à des solutions classiques, ne protégeant pas la vie privée. Nos expérimentations indiquent que les contraintes de protection des données privées peuvent être prise en compte dans les services de covoiturage sans dégrader leurs performances.

Mots clés: vie privée, covoiturage, technologies renforçant la vie privée

Contents

List of Figures

List of Tables

Introduction

The emergence of mobile and connected devices has fostered the ubiquitous world in which we are living, where personal data like mobility data (GPS position), health data (weight, blood oxygen, heart rates, calories burned, sleep patterns, ...), movies preferences, and shopping activities, among many others, are gathered on a daily basis by services providers and shared with third parties like marketing or assurance companies. According to a recent *Cisco* report [31], the number of mobile-connected devices will hit 11.6 billion by 2021, with a monthly global mobile data traffic around 49 exabytes. The increasing usage of these devices has promoted the development of a wide range of services including ridesharing services as demonstrated by the popularization of platforms like *Uber* and *Blablacar*.

Ridesharing, which falls into the broad category of the *collaborative economy* [16], presents numerous advantages for both the drivers and the 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 [128] which are usually less crowded. Finally, ridesharing reduces CO_2 emissions [27]. However, in their current implementations, ridesharing service providers maintain highly sensitive data on their users, such as personally identifiable information (PII), location data, and financial information. The fine-grained nature of these data allows inference of sensitive information such as points of interest or social ties [53], and raises privacy issues that may disrupt the adoption of ridesharing services. For example, between 2014 and 2015, several privacy scandals have involved the ridesharing company Uber, including monitoring of the location of riders in real-time for entertainment [59], revenge attacks against journalists [123], and a security breach which disclosed the names and drivers license numbers of nearly 50,000 drivers [33].

Ridesharing has received many attentions from the research community over the last few years. In particular, Agatz et al. [2] formalize the ridesharing problem in a dynamic setting and propose several optimization techniques to solve it. Bit-Monnot et al. [21] 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. [111, 110] demonstrate that a reasonable increase in flexibility — regarding the desired departure, 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.

This thesis analyzes the research question *"Can ridesharing services be implemented in a privacy-preserving fashion?"*. To answer this question, we consider two essential problems in ridesharing, namely the computation of meeting points and the matching of drivers and riders. We formalize these two problems and design privacy-preserving protocols to solve them.

This thesis has been elaborated in the groups `Operations Research, Combinatorial`

Optimization and Constraints (ROC) and Dependable Computing and Fault Tolerance (TSF) of the Laboratory for Analysis and Architecture of Systems (LAAS-CNRS). It has been funded by the grant program called Axes Thématiques Prioritaires of the University of Toulouse III Paul Sabatier.

This thesis is divided into three parts, each of which is composed of two chapters:

Part I introduces preliminary concepts. **Chapter 1** gives general definitions of privacy, its threats, properties as well as existing legal and technical protection provisions. **Chapter 2** introduces graphs and multimodal routing notions that are employed in this dissertation. It also introduces ridesharing and analyzes its routing and matching aspects.

Part II presents our privacy-preserving protocols for ridesharing. In **Chapter 3** , we formalize the *Secure Meeting Points for Ridesharing (SMP4R)* problem, and provide two privacy-preserving protocols to solve it. The results of our experiments show that one can design privacy-preserving meeting points determination protocols with similar quality as existing (non-privacy-preserving) ridesharing protocols without introducing important computational and communication overheads. The findings of this work, done in collaboration with Sébastien Gambs (Université du Québec à Montréal), have been published in [7, 5, 6]. In **Chapter 4** we consider a generalized version of the SMP4R problem, in which we consider several drivers and riders and want to securely assign drivers to riders according to the ridesharing cost. We propose SRide, a privacy-preserving ride matching system which operates in three steps. First, it uses a secure filtering protocol to build the bipartite graph of feasible matches. Then, it relies on privacy-preserving meeting points determination protocols proposed inChapter 3 to obtain the cost of each feasible pair. Finally, it determines optimal assignments of drivers and riders. We evaluate our protocol and demonstrate its computational and communication efficiency. The results of this work, done in collaboration with Kévin Huguenin (Université de Lausanne), are submitted to [8].

Part III presents our empirical analysis of a real-world ridesharing service, and PlayMob: a platform that we develop during this thesis. **Chapter 5** is dedicated to the analysis of data collected from *Covoiturage-libre.fr* [1]: a popular, openly available ridesharing web service in France. This analysis allows us to get more insights about ridesharing in France and to use this information while generating synthetic data for our experiments. In **Chapter 6** , we present the main features of PlayMob.

Finally, we conclude by giving future research directions.

[1] http://covoiturage-libre.fr

Part I

Preliminaries

Chapter 1

Privacy

1.1 Definition

The *Oxford* dictionary defines the word privacy as "the state of being free from public attention" [95]. Privacy can be seen as the ability of a subject or community to keep their personal data away from the eyes of society. A widely used definition of privacy is by Alan Westin [125] who defines privacy as "the claim of individuals, groups, or institutions to determine for themselves when, how, and to what extent information about them is communicated to others".

In France, the Data Protection Act (*Loi Informatique et Libertés*) defines *personal data* as *any information relating to a natural person identified or that can be identified, directly or indirectly, by reference to an identification number or several elements of its own* [78]. The National Institute of Standards and Technology (NIST) defines *Personally Identifiable Information* (PII) as «any information about an individual maintained by an agency, including (1) any information that can be used to distinguish or trace an individual's identity, such as name, social security number, date and place of birth, mother's maiden name, or biometric records; and (2) any other information that is linked or linkable to an individual, such as medical, educational, financial, and employment information.» [89]. In general, any data relative to a subject can be classified as *Explicit Identifier* or *Quasi Identifier* (QID) or *Sensitive Attributes* or *Non-Sensitive Attributes* [25]. Explicit identifiers are attributes like name, social security number and biometric records that can directly identify a subject. Quasi Identifiers are attributes that, combined (*e.g.*, zip code, date of birth, sex), could potentially identify a subject. Sensitive attributes are sensitive information such as disease, salary, religion. Non-Sensitive Attributes are attributes that can not be classified in the three previous categories.

1.2 Location-based services

Location-based services (LBS) are part of a larger family of services: context-aware services [11], which, to provide services such as filtering or presenting information, will adapt to the *information context*. The information context itself is organized into two hierarchical levels: (1) a primary level containing raw data from the sensors (light sensor, microphone, accelerometer, GPS, ...) and (2) a finer level obtained by combining, filtering, inferring

information from the first level. In the case of location-based services, the primary context of the information is a location that can be refined to obtain a spatial or spatiotemporal secondary context. Location-based services include among other: *Check-ins services* which are online social networks that allow users to report locations that they have visited and share them with their friends, *Navigation services* which help in finding best itineraries and *Recommendation services* which allows users to find friends or places (like restaurant, public transport station, gas station ...) nearby.

1.3 Location data

Digital traces left by individuals using location-based services can be categorized into two families, namely mobility traces and contact traces [90]. A mobility trace is characterized by an **identifier** which allows to uniquely recognize the individual who interacts with the location based service (username, phone identifier, connected object identifier, ...), a **spatial component**, which can be a GPS position (latitude, longitude, altitude), a spatial zone (name of a street) or a semantic label (school, home), a **temporal component** which can be a precise time (9:30 am) or a time interval (morning) or frequency (every Monday morning) and **additional and optional information** such as speed and direction of movement, presence of other devices or individuals in the vicinity, accuracy of the measured position .... A contact trace is a set 2 identifiers, and a timestamp.

1.4 Location privacy

In location context, Beresford and Stajano define location privacy as «the ability to prevent other parties from learning one's current or past location» [19]. Shokri *et al.* [109] formally define location privacy as the estimation error of a malicious entity, hereafter referred to as the adversary, when trying to achieve his objective (*e.g.*, obtain the probability distribution of a user's locations at a specific time, obtain the number of users at a particular location at a specific time, ...) based on his prior knowledge.

1.5 Threats and Attacks against privacy

1.5.1 Adversary model and privacy properties

Generally speaking, any entity that seeks to recover personal data of somebody, without his explicit consent, to establish a profile or to infer his private data is referred to as an adversary. Deng *et al.* [39] notice eight main privacy properties classified in two categories: hard privacy properties and soft privacy properties, which we present in Tables 1.1 and 1.2.

Hard privacy properties are properties required at users' level to ensure that personal data

are not revealed. They are more expected in untrusted environments in which data subjects do not trust the data company in collecting and processing their personal information. On the contrary, soft privacy properties are properties required at service providers' level as users already lost control on their personal data and rely on service providers to ensure data privacy by the mean of policies explicitly mentioned and accepted. Therefore, depending on the setting (user-centric or centralized architecture), the adversarial environment varies. In hard privacy setting, adversaries include service providers, other users or external entities while in soft privacy, adversaries are essentially external entities or other users.

Property	**Definition**
Pseudonymity	Possibility to use pseudonyms as identifiers instead of real name.
Anonymity	Impossibility for the adversary to identify a subject within a group of subjects.
Unlinkability	Impossibility for the adversary to sufficiently distinguish at least two items of interest (subjects, actions, messages, ...) that are related.
Plausible deniability	Ability to deny having performed an action that other parties can neither confirm nor contradict. That is, the adversary should be unable to confirm that a subject knows, has said or has done something.
Unobservability	Impossibility to detect the absence or the presence of a particular subject and impossibility to discern a subject (if present) from the other subjects.
Confidentiality	Impossibility for non authorized persons to have access to an item.

Table 1.1: Hard privacy properties. Adapted from Deng *et al.* [39].

Property	**Definition**
Content awareness	Users are fully aware of the personal data they generate and are informed that they provide only the just necessary with regard to the service they expect
Policy and consent compliance	Service provider should implement the collection and processing of personal data in accordance with the legislation in force. Ability to the users to explicitly express their agreement to the collection and use of their personal data (*e.g.,* "I understand and accept the conditions")

Table 1.2: Soft privacy properties. Adapted from Deng *et al.* [39].

1.5.2 Privacy threats

In the light of privacy properties mentioned above, Deng *et al.* [39] propose seven potential threats to privacy, namely *Identification*, *Linkability*, *Non-repudiation*, *Detectability*, *Information disclosure*, *Content unawareness*, *Policy and consent noncompliance*.

Linkability indicates the possibility for an adversary to link two or more actions, messages to the same subject (for example, *Alice took the bus* 78 *at 5:30 pm and Alice went to cinema in the downtown at 7:00 pm*).

Identification indicates the possibility for an adversary to find a subject in a group of subjects (for example, *Alice, who lives in Toulouse, is a member of the ecologist group* France Verte*. In the anonymized list of actions carried out by members of* France Verte*, there is a single action carried out by a person living in the city of Toulouse*).

Non-repudiation means that an adversary can prove that a subject knows, has said or has done something.

Detectability indicates the fact that an adversary can conclude about the presence or the absence of a subject in a data set.

Information disclosure indicates the fact that unauthorized users have access to information.

Content unawareness indicates that users are not aware of the data they provide or produce more data than necessary regarding the expected service.

Policy and consent noncompliance indicates that the service provider does not respect the conditions for data collection and processing as read and explicitly accepted by users.

1.5.3 Privacy attacks

Inference Attacks. Most attacks against privacy are inference attacks, in which the adversary will extract personal data relating to a subject (hereafter referred to as the target) or will acquire new knowledge about the target.

Generally speaking, they are four types of attacks: the *record linkage*, the *attribute linkage*, the *table linkage* and the *probabilistic attack* [51]. Record linkage, attribute linkage and table linkage respectively happen when the adversary, based on the knowledge of the QID of his target, is able to link him respectively to a record in a published dataset, to a sensitive attribute in a published dataset, to the published dataset itself while a probabilistic attack happens when the adversary improves some beliefs on the target after obtaining the published dataset.

More precisely, in linkage attack, the adversary matches his target's QID to a small group of individuals and uses additional knowledge to uniquely identify the record of his target. In

an attribute linkage, the adversary learns sensitive values from his target based on the set of sensitive attributes values associated with the group the target belongs. For example, if the target belongs to a group whose sensitive attributes values for a disease are Hepatitis and HIV, then the attacker knows that his target has either Hepatitis or HIV, which, *per se*, is already damaging. In table linkage, the adversary learns the presence or the absence of his target in the published dataset. For example, if the adversary happens to detect that his target is present in a dataset of subjects and type of cancer they have, the fact of knowing his target has cancer (no matter what cancer) is already damaging. Finally, in a probabilistic attack, the adversary increases the difference between his prior and posterior beliefs about the target.

In databases, adversary will target anonymity and unlinkability properties, this is known as de-anonymization attacks. For this end, the adversary uses either prior knowledge on his target (*e.g.*, some habits, QID) to uniquely identify him in a group or uses QID to find him in another database. In 2000, Sweeney [116] demonstrated by crossing an electoral list with a pseudonymized medical database and using the triplet (zip code, date of birth, sex), which is known to be unique for 87% of the US population [117], that one could link medical data to individuals. In particular, she successfully obtained medical data of the governor of Massachusetts. In 2006, journalists of The New York Times successfully re-identify individuals in a pseudonymized list of 20 million Web search logs released by *AOL* [14]. Narayanan *et al.* demonstrate that by crossing the *Netflix Prize* dataset, a pseudonymized Netflix's movie rating list of $500,000$ subscribers, with the Internet Movie Database, they could identify the Netflix records of known users and *de facto* uncovering potentially sensitive information (opinions, sexual orientations, religion, ...) [86]. De Montjoye *et al.* show that the knowledge of only four location data is sufficient to re-identify 95% of individuals [37] in a pseudonymized mobile phone dataset that contains call information for nearly 1.5 millions users of a mobile phone operator. De Montjoye *et al.* [36] do similar research using three months of credit card records for 1.1 million people and show that four spatiotemporal points are enough to uniquely re-identify 90% of individuals. Furthermore, they demonstrate that knowing the price of a transaction increases the risk of re-identification by 22%.

In social networks, adversary will target users' anonymity by exploiting relationship between them. In [12], authors show that an honest but curious adversary can exploit his/her existing relationships with other users in a social network to de-anonymize them in the anonymous version of this social network in which a random ID replaces each node. Furthermore, they theoretically prove that a malicious adversary, by strategically creating a small subgraph (whose order is logarithmically proportional to the social network's order) of new accounts, and linking them the set of targeted users, will have high probability in de-anonymizing them in the anonymous network.

Adversarial model Many adversarial behaviors have been considered in the literature to analyze the security of the protocol (more details can be found in [68]). The two most common models of adversaries are: the *semi-honest* (or *honest but curious* or *passive*) adversary and the *malicious* (or the *active*) adversary. The former will follow the recipe of the established

protocol while trying to infer additional information from the output of the computation, its intermediary results and the ciphered inputs of other participants while the latter may actively cheat during the execution of the protocol to either infer information on other users or fail the protocol.

Attacks on Location privacy. Wernke *et al.* [124] distinguish four types of location privacy attacks (*c.f.*, Figure 1.1) according to the prior knowledge of the adversary. Namely, attacks on location privacy include (1) *single position attack*, (2) *context linking attack*, (3) *multiple position attack* and (4) *multiple position and context linking attack*. According to the same authors, the goal of an attack against location privacy is twofold: (1) the adversary learns the identity of the target and (2) the adversary learns the spatial or spatiotemporal information of the target's location. In *single position attack*, the adversary solely relies on a single mobility trace to achieve his goal. In *context linking attack*, the adversary uses external data sources in addition to the mobility trace. In *multiple position attack*, the adversary tracks and correlates several position updates or location queries of the target. The last type of attack combines multiple position attack with context linking attack to increase the success rate.

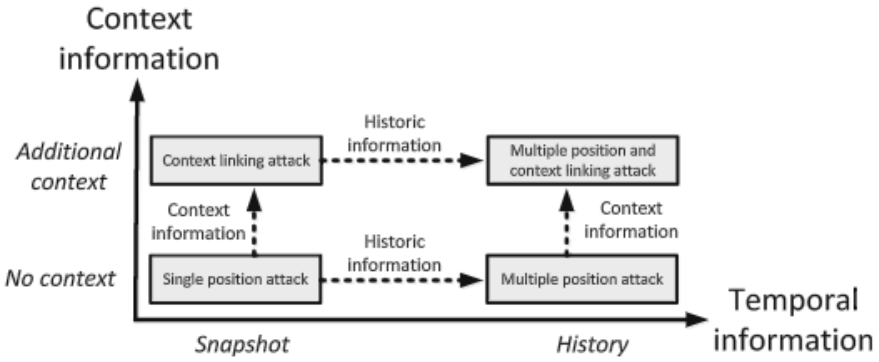

Figure 1.1: Adversary knowledge in location privacy attacks [124]

Inference attacks can be achieved with simple heuristics or machine learning algorithms such as clustering or classification.

Clustering. Clustering is a form of unsupervised machine learning algorithm that groups unlabeled items in clusters such that items belonging to the same cluster resemble each other more than items of different clusters. The measure of resemblance called similarity or distance metric expresses how far/similar two items are relative to each other. In location context, the Haversine distance or the shortest path length can be used to evaluate the similarity of two locations. That is, the closer they are the more similar they are to each other. The *k-means* algorithm is an iterative clustering algorithm that computes k clusters as well as their respective centroids (mean of all items within each cluster). Clustering algorithms can be used to discover points of interest of a target given the set of locations he has visited. For example, Hoh *et al.* [73] use *k-means* algorithm and some heuristics to find some individuals'

homes based on GPS traces of vehicles in Detroit region (Michigan, USA). First, they filter out GPS samples recorded at speeds higher than $1m/s$. Then, they apply a clustering until every centroid is 100 meters apart in average from all its elements. Finally, they eliminate clusters with no recorded points between $4p.m.$ and midnight as well as clusters whose centroids are outside residential areas.

Classification. Classification is a form of supervised machine learning algorithm that predicts values (which take on a small number of discrete values referred to as *classes*) for an unknown item based on a model it trains with a set of labeled items referred to as *training set*. Classification can be used to learn the semantic attached to a point of interest. For instance, Subramanya *et al.* [112] use a dynamic probabilistic model trained using inputs from GPS and wearable sensors to classify a person's motion type (*e.g.*, walking, running, going upstairs/downstairs, driving) or environment (*e.g.*, outdoor, indoor, in vehicle).

1.6 Legal protection provisions

The Universal Declaration of Human Rights, through its 12^{th} article "No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor and reputation. Everyone has the right to the protection of the law against such interference or attacks." [120] highlights the universality of the right to privacy and the fact that should be respected everywhere in the world.

At European level, the *GDPR* (General Data Protection Regulation), which replaces the Directive 95/46/EC and will apply from 25 May 2018, is an initiative of the European Parliament, the Council of the European Union and the European Commission to unify data protection policies continent-wide. It will apply to data companies and data subjects based in the EU as well as to data companies based outside the EU, but that collect and process EU residents' data. GDPR recommends data companies, through its 25^th Article, to implement the so-called *privacy by design* as well as *privacy by default* policies. The former, introduced by Ann Cavoukian in the 90's, has as objective to proactively embed privacy directly into the design phase of information technologies, business practices, physical systems and networked infrastructures [29, 28]. The latter requires IT systems to achieve privacy by default. That is, a new user should have his/her privacy protected without having to make any configuration.

In France, the National Commission on Informatics and Liberty CNIL (Commission Nationale de lInformatique et des Libertés) is an independent regulatory agency whose mission is to ensure that all data subjects and data controllers are informed about their rights and duties and that the processing of personal data is carried out in conformity with the provisions of the Act 78-17 on information technology, data files, and civil liberty.

Surname	Age	Zip	Sex	Disease
Yoshida	23	33077	M	Asthma
Cohen	29	33300	F	Hypertension
Achebe	25	33400	F	Schizophrenia
Murphy	42	31200	M	Cancer
Bouchard	45	31400	F	Diabetes
Smith	55	31300	M	Influenza

Surname	Age	Zip	Sex	Disease
1	23	33077	M	Asthma
2	29	33300	F	Hypertension
3	25	33400	F	Schizophrenia
4	42	31200	M	Cancer
5	45	31400	F	Diabetes
6	55	31300	M	Influenza

Table 1.3: Example of a medical dataset (up) and its pseudonymised version (down)

1.7 Technical protection provisions

Privacy Enhancing Technologies (PETs) are defined in [121] as a system of *Information and Communication Technologies* (ICT) measures protecting informational privacy by eliminating or minimizing personal data thereby preventing unnecessary or unwanted processing of personal data, without the loss of the functionality of the information system. In [32], privacy methods are classified in two main categories: *law-based* and *technique-based* approaches. Although law-based approaches can deter an adversary from committing a privacy breach due to the threat of heavy penalties, their actions are perceptible *a posteriori*, when the offense (*i.e.*, privacy violation) is already committed. In the other hand, PETs approaches help in reducing the success rate of the adversary's attacks by the mean of several techniques. Possible privacy-preserving techniques include:

Pseudonymization. It is about replacing the identifier by pseudonyms like random identifiers. However, since the seminal attack in [116], pseudonymization, as such, is not considered as a viable privacy protection technique. An example of pseudonymization is given in Table 1.3.

Dummies generation. It is about hiding user data in a list of dummies while querying a service provider. That is, a set of dummies is generated and send along the real query and,

upon reception of the answers, the answer corresponding to the real query is filtered out.

In location context, the real query will contain user's actual location while dummy queries will contain random locations. However, it is important to generate realistic dummies so that the service provider cannot distinguish the real query from the dummies. For instance, in [75], Kido *et al.* propose a privacy-preserving and user-centric location query system that enables users to hide their locations in a set of dummies. The dummies generation algorithm is devised such that dummies behave like actual locations. In the first strategy proposed by the authors, dummies are generated in the vicinity of the previously generated dummies. The second strategy is a collaborative one, in which users generate dummies in regions containing other users. The solution is evaluated with real-world data and assessed through the impact of the number of dummies on the location anonymity which is defined as the maximum uncertainty that the LBS has on the actual location.

generalizations and suppressions. Generalizations and suppressions are used to hide details about a *QID* to prevent observation of uniqueness patterns. More precisely, a generalization replaces an attribute value by a more generalized parent value (*e.g.*, a zip code value 31400 will be replaced by $31**$ as both refer to places within the city of Toulouse). Suppression replaces an attribute value by a special value (*e.g.*, sex attributes $Female$ and $Male$ will both be replaced by $*$). A classic generalization and suppression technique is k-anonymization, which guarantees the *k-anonymity* property [115].

Definition 1.1. (k-anonymity) k-anonymity requires that each equivalence class (*i.e.*, set of items that are similar with respect to a QID) contains at least k records.

An example of a 3-anonymized dataset is given in Table 1.4. A dataset published using k-anonymization is privacy-preserving *w.r.t. record linkage* as each record is indistinctly matched to at least k record, however, it is not resistant to *attribute linkage*. To solve this problem, anonymization methods with *l-diversity* [80] and *t-closeness* [77] properties have been proposed.

Definition 1.2. (l-diversity) l-diversity requires that each equivalence class has at least l well-represented values for each sensitive attribute.

Definition 1.3. (t-closeness) t-closeness requires that for each equivalence class, the distance between the distribution of a sensitive attribute in this class and the distribution of the same attribute over the whole dataset is at most a threshold t.

Anonymization technique with l-diversity and t-closeness properties ensure that the published dataset is resistant to *attribute attack*. However they can not achieve privacy *w.r.t. table linkage* and *probabilistic attack*. To thwart this issue, the notion of *differential privacy* [44, 45] has been proposed.

Differential Privacy. Differential privacy aims at making the probability of any query output insensitive to the presence or absence of any individual in the dataset regardless the

Surname	Age	Zip	Sex	Disease
Yoshida	23	33077	M	Asthma
Cohen	29	33300	F	Hypertension
Achebe	25	33400	F	Schizophrenia
Murphy	42	31200	M	Cancer
Bouchard	45	31400	F	Diabetes
Smith	55	31300	M	Influenza

Surname	Age	Zip	Sex	Disease
*	< 30	33***	*	Asthma
*	< 30	33***	*	Hypertension
*	< 30	33***	*	Schizophrenia
*	≥ 40	31***	*	Cancer
*	≥ 40	31***	*	Diabetes
*	≥ 40	31***	*	Influenza

Table 1.4: Example of a medical dataset (up) and its 3-anonymized version (down)

adversary's background knowledge. That is, differential privacy achieves privacy against *table linkage* and *probabilistic attack*.

Definition 1.4. (Differential Privacy) A randomized mechanism $\mathcal{M} : \mathcal{X}^n \rightarrow \mathcal{Y}$ is (ϵ, δ)-differentially private if for every pair of adjacent databases (i.e., databases that differ only for the addition of one record) $\mathbb{D}, \mathbb{D}' \in \mathcal{X}^n$, and for every subset of output $S \subseteq \mathcal{Y}$,

$$Pr[\mathbb{M}(D) \in S] \leq exp(\epsilon) Pr[\mathbb{M}(D') \in S] + \delta$$

Cloaking. Cloaking is a particular use of k-anonymity in the spatiotemporal context. It ensures that at each period, each subject is located within a spatial area that is shared by at least $k - 1$ other subjects. For example, in [64], authors propose an algorithm that can be used by a centralized location-based service to enhance location privacy of its users. The algorithm can adjust spatial resolution of a location data to satisfy an anonymity requirement. More precisely, given a geographic area, a degree of anonymity k, the current position of the requester and the current positions of all other users in the considered area, the algorithm subdivides the area around the requester until the number of subjects it contains falls below k. The previous grid which still matches the constraint is returned. The proposed approach is evaluated with synthetic requests generated from transportation network information of the city of Denver, Colorado and the spatial resolution of the solutions are compliant with E-911 requirements.

Mix zone. Mix zones mimic the functionality of mix-nodes in communication systems. In fact, similarly to a mix node, which guarantees unlinkability between incoming and outgoing messages, a mix zone is an area in which mobile users can change their pseudonyms so that the mapping between their old pseudonyms and new pseudonyms are not revealed. Furthermore, each outgoing user is required to be k-anonymous regarding the mix zone she is leaving.

Mix zones have been initially introduced in [18] and aim to increase the unlinkability of the user. In this seminal work, authors first evaluate privacy as the size of the anonymity set before defining a solid measurement metric which takes into account correlation between each user's ingress and egress positions. The latter definition enables to consider more powerful adversaries that will exploit geographical context as well as movement characteristics.

Homomorphic Encryption. An encryption scheme allowing to perform operations (without decryption) on ciphertexts as on plaintexts is called *Homomorphic Encryption* (HE).

Let us consider a cryptosystem Π with an encryption function ϵ and two messages m_1 and m_2. Π is said to be an additive homomorphic encryption scheme *iff*: $\exists\ \Delta : \epsilon(m_1)\ \Delta\ \epsilon(m_2) = \epsilon(m_1 + m_2)$. Π is said to be an multiplicative homomorphic encryption scheme *iff*: $\exists\ \Delta : \epsilon(m_1)\ \Delta\ \epsilon(m_2) = \epsilon(m_1 * m_2)$. For example the *Paillier* [91] cryptosystem is an additive homomorphic cryptosystem while the *RSA* [100] and the *ElGamal* [46] cryptosystems are multiplicative homomorphic encryption schemes.

An homomorphic encryption supporting an arbitrary number of operations (additions and multiplications) is referred to as *Fully Homomorphic Encryption* (FHE) [57]. At a high level, an *FHE* scheme relies on a *Somewhat Homomorphic Encryption* (SHE) [48] to perform homomorphic operations and an additional building block –termed *bootstrapping* – which guarantees the correctness of the decryption, by reducing the inherent noise introduced by the homomorphic operations. *SHE* schemes allow an unlimited number of additions but a limited multiplication depth. In practice, whenever the number of homomorphic multiplications is small, *SHE* schemes provide better performances than *FHE*, with the same guarantee of correctness.

Secret sharing. Introduced in the 70's [22, 107], 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*.

Arithmetic 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 $r_A \in \mathbb{Z}_n$. Then, she computes $r_B \equiv s - r_A \mod n$ and sends it to the second party. For instance, in [60], authors propose a protocol, based on an arithmetic secret sharing, which allows two parties to compute the dot product of their private vectors obliviously. In the proposed protocol, the

first party uses an homomorphic encryption scheme to encrypt elements of his vector $x \in \mathbb{Z}^n$ before sending both the public key and the encrypted vector to the second party. Then, the second party computes the dot product with his private vector $y \in \mathbb{Z}^n$ in the ciphertext domain and adds a random number $-r_B$. Finally, the first party receives $r_A \equiv x.y - r_B \mod n$. Since r_A is computed in the ciphertext domain, only the first party knows its values. Likewise, only the second party knows the values of r_B. By putting their respective shares (r_A and r_B) together, both parties will be able to obtain the result of the dot product; that is, $r_A + r_B \equiv x.y \mod n$.

Boolean secret sharing. 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. Finally, by using the Goldreich-Micali-Widgerson (*GMW*) protocol [62] or the Yao's garbled circuits protocol [129] both parties can evaluate any function (represented as a boolean circuit) with their private boolean shares.

Secure Multiparty Computation. Introduced in the 80's [129, 62], 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. The gold standard would be the existence of a trusted third party that would perform the entire computation and returns the output to all the parties involved while erasing his memory afterward. The objective of SMC is to achieve the same functionality, but without relying on a trusted third party. Yao first defines the two-party comparison problem, now known as Yao's Millionaires problem, and developed a provably secure solution for this problem [129]. Since this seminal work, many works have been done in the field of secure multiparty computation. Secure multiparty computations are used for numerous tasks including coin-tossing, broadcasting, electronic voting, electronic auctions, electronic cash, contract signing, anonymous transactions and private information retrieval schemes.

An important building block for secure multiparty computation is the oblivious transfer (OT) protocol [98], where the sender inputs two l-bit messages (m_0, m_1) and the receiver inputs a selection bit $s \in \{0, 1\}$ and obtains one message m_s. The OT protocol guarantees that the sender does not learn the choice s of the receiver, while the receiver only learns m_s and nothing about m_{1-s}. This particular form of the OT protocol is referred to as the *1-out-of-2 oblivious transfer* protocol. A more generalized form of this protocol is known as the *1-out-of-n oblivious transfer* protocol [85], where the sender has n messages $(m_0, \ldots, m_{n-1})$, and the receiver has an index $i \in \{0, \ldots, n-1\}$. In this version of the protocol, the receiver wants to receive the ith message, without the sender learning i, while the sender wants to ensure that the receiver receives only the message m_i.

Private Set Intersection. In the field of SMC, series of work have been done [50, 76, 74, 67, 35, 42] on a cryptographic protocol referred to as *Private Set Intersection* (PSI) and a variant known as *Private Set Intersection Cardinality* (PSI-CA). In these tasks, several

(mutually-distrusting) parties jointly compute respectively the intersection or the cardinality of the intersection of their private inputs without leaking any additional information.

Authors in [50] propose *PSI* protocols based on *Oblivious Polynomial Evaluations* (OPE) in which additive homomorphic encryption is used to evaluate a polynomial encoded with private inputs obliviously. Figure 1.2 summarizes the main interactions of and OPE-based PSI which engages two parties Alice and Bob. First, Alice represents elements of her private set $X = (x_1, ..., x_n)$ as the roots of a nth degree polynomial $P(x) = \prod_{i=1}^{(} x - x_i) = \sum_{i=0}^{v} \alpha_i x^i$. Assuming pk to be Alice's public key of any additive homomorphic cryptosystem, Alice encrypts the coefficients of P with pk and sends both the encrypted coefficients and her public key to Bob. Then, Bob homomorphic-ally evaluates P with each of his private inputs $y_i \in Y$. Note that $P(y_i) = 0$ if and only if $y_i \in X \cap Y$. More precisely, for each $y_i \in (y_1, ..., y_n)$, Bob computes $z_i = E(r_i P(y_i) + y_i)$ (where r_i is random number) and sens it to Alice. If $y_i \in C \cap S$ then Alice learns y_i upon decryption. If $y_i \notin C \cap S$ then Alice decrypts a random value. For performance reasons, author apply the *Horner's* rule to evaluate the polynomial.

Inspired by the works presented in [50], authors in [76] explore the power of polynomial representation of multisets, using operations on polynomials to obtain composable privacy-preserving multisets operations such as set intersection, union, cardinality and over-threshold operations.

Committed Oblivious Pseudorandom Function Evaluation, another approach to implement PSI using secure pseudorandom function evaluations, has been introduced by [67] and lately improved by [74] and [35].

Finally, another line of work [42] proposed *Oblivious Bloom Intersection* in which private input sets are inserted in Bloom filters followed by the intersection of the Bloom filters. The proposed approach has a linear computational overhead. However, the use of hash functions with low and parameterizable probability in Bloom filters leads to the existence of false positives which impacts the accuracy of Bloom filter-based PSI.

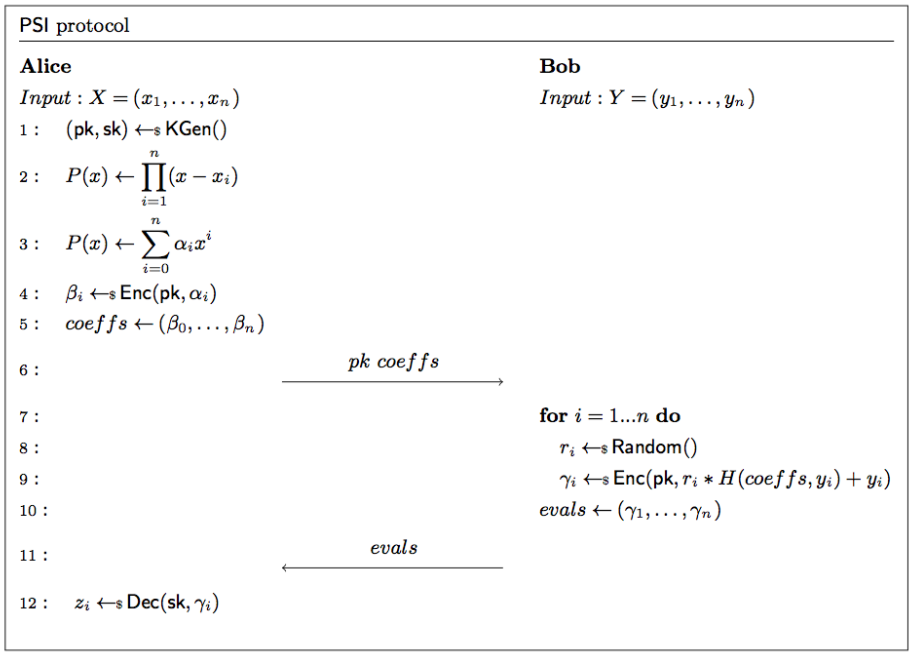

Figure 1.2: Private set intersection protocol as presented in [50]

Chapter 2

Multimodal routing and Ridesharing

2.1 Graphs

2.1.1 Basic definitions

Graphs are discrete mathematical structures used to represent relations between objects. In graph theory, these objects are referred to as *vertices* (or points or nodes), while relations between them are called *edges* (or arcs or lines). Graphs have numerous applications in Computer Sciences, Social Sciences, Engineering, Mathematics, Natural Sciences to name a few. For instance, graphs are used to model network communications, to detect communities in social networks or to find the shortest path to our favorite restaurant.

Definition 2.1. (Graph) A graph is represented as a tuple $G = (V, E)$ of a nonempty set of vertices V, a set E of edges that are one- or two-element subset of V.

A graph $G = (V, E)$ is said to be finite if its vertex set V is finite. A graph with an infinite set of vertices is an infinite graph. The number of vertices $|V| = n$ of a graph is referred to as its order, and the number of its edges $|E| = m$ is its size. In this thesis, we solely rely on finite graphs.

A graph $G = (V, E)$ is said to be directed if each edge $(u, v) \in E$ has a direction that indicates how it can be traversed. By contrast, in an undirected graph, edges can be traversed both ways.

A weighted graph is a graph with a function $w : E \to \mathbb{R}$ which associates to each edge $(u, v) \in E$ a cost $w(u, v)$. Weighted graphs are frequently used to model transportation networks in which edges' weights may represent time to reach a certain location via road segments represented by the edges.

A graph that has multiple edges connecting a pair of vertices is called *multigraph* while a graph without self-loop edges (*i.e.*, (u, u)) and that is not multigraph is referred to as a simple graph.

Definition 2.2. (Predecessors and successors) Given a directed graph $G = (V, E)$, for each

vertex $u \in V$, the set of predecessors of u is defined as $\mathcal{P}(u) = \{v \mid (v,u) \in E\}$. The set of successors of u is defined as $\mathcal{S}(u) = \{v \mid (u,v) \in E\}$.

Definition 2.3. (In-Degree, Out-Degree) Given a directed graph $G = (V, E)$, the in-degree and out-degree of a vertex $u \in V$ are respectively the cardinality of $\mathcal{P}(u)$ and $\mathcal{S}(u)$.

Definition 2.4. (Walk, Trail, Path, Circuit) Given a directed graph $G = (V, E)$, a walk is an alternating sequence of vertices and edges $(v_1, e_1, v_2, e_2, \ldots, v_n, e_n, v_{n+1})$, which starts and ends with a vertex and has the following condition hold: $\forall i = 1, \ldots, n, (v_i, v_{i+1}) = e_i$. A closed walk is a walk such that $v_1 = v_{n+1}$. A trail is a walk without repeated edges. A path is a trail without repeated vertices. A circuit is a closed path.

2.1.2 Special graphs

Definition 2.5. (Vertex-labeled Graph) A vertex-labeled graph $G = (V, E)$ is a graph such that each vertex $v \in V$ has a label.

In vertex-labeled graphs, labels are used to describe additional properties related to vertices. For instance, a vertex in a transportation network can be labeled as potential meeting station, parking station, bus stop, point of interest ...

Definition 2.6. (Edge-labeled Graph) An edge-labeled graph $G = (V, E, \Sigma)$ is a graph with a set of labels Σ associated to its edges.

In edge-labeled graphs, labels are used to describe the nature of relation between vertices. Edges are represented with the tuple (u, v, l) where u and v denote vertices and l the label assigned to (u, v). For instance, transportation networks can be modeled as edge-labeled graphs in which label are transportation modes; in this case, the edge (u, v, l) expresses the fact that one can move from location u to location v by using transportation mode l.

2.2 Shortest path algorithms

Shortest path problems (SPP) usually concern weighted graphs. Let us assume $G = (V, E)$ is a directed weighted graph in which each edge (u, v) is associated with a nonnegative weight $w(u, v)$. The length of a path P is the sum $w(P)$ of its edges' weights.

Generally speaking, there are two major types of shortest path problems: One-To-One and Many-To-Many shortest path problems.

Definition 2.7. (One-To-One shortest path problem) Given a directed and weighted graph $G = (V, E)$, a source vertex s and a target vertex t, find the path P from s to t such that for any path P' from s to t, $w(P) \leq w(P')$. A shortest path algorithm returns the shortest path P between s and t as well as its distance $dist(s, t) = w(P)$.

Definition 2.8. (Many-To-Many shortest path problem) Given a directed and weighted graph $G = (V, E)$, a set $S \subseteq V$ of source vertices and a set $T \subseteq V$ of target vertices, find for every pair $(s, t) \in S \times T$, the shortest path from s to t.

The One-To-All shortest path problem is a particular case of Many-To-Many shortest path problem where $S = \{s\}$ and $T = V$. In this case, the output is also known as *out-shortest path tree*, as for each vertex $u \in V$ the path from s to u in the tree is the shortest path from s to u.

The All-To-One shortest path problem is also a particular case of Many-To-Many shortest problem where $S = V$ and $T = \{t\}$, its output is also known as *in-shortest path tree*. Finally, *all pairwise shortest path* problem corresponds to the case $S = T = V$.

2.2.1 Standard techniques

The Dijkstra's algorithm [40] is widely used to solve One-To-All shortest path problems. The main idea of this algorithm is in updating a distance table $dist(s, v)$ from s to all the vertices $v \in V$. More specifically, it initializes by setting $dist(s, s) = 0$ and $dist(s, u) = \infty \ \forall u \neq s$ and by inserting s into a priority queue Q ordered by distance. When a vertex is extracted from Q it will be referred to as scanned vertex. At every iteration, the vertex u with the lowest distance $dist(s, u)$ is extracted from Q and for each outgoing edges (u, v) the following operation (also known as relaxation) is conducted:

- if $v \notin Q$
  - $dist(s, v) = dist(s, u) + w(u, v)$
  - $Q = Q \cup \{v\}$
- if $v \in Q$ and $dist(s, u) + w(u, v) < dist(s, v)$
  - $dist(s, v) = dist(s, u) + w(u, v)$
  - Update Q

In the case of One-To-One shortest path problem, thanks to the label-setting property (once a vertex u is scanned, its distance $dist(s, u)$ is correct), one can stop the algorithm as soon as the target vertex t is scanned. Dijkstra's algorithm runs in $\mathcal{O}((|V| + |E|)log|V|)$ with binary heaps. This complexity can be reduced to $\mathcal{O}(|E| + |V|log|V|)$ by using Fibonacci heaps.

The Bellman-Ford algorithm [17], even if it runs in $\mathcal{O}(|V||E|)$, can be used as an alternative to Dijkstra's algorithm when the considered graph has negative edge weights. Similarly to the Dijkstra's algorithm, it initializes the distance to every to ∞ except the source node s initialized to 0. Then, at each iteration, it relaxes all the edges in the graph. As the path from the source vertex to any other vertex in the graph can be at maximum $|V| - 1$ edges

long, the algorithm constructs the shortest path tree after $|V| - 1$ iterations provided there is no circuit of negative length.

The Floyd-Warshall algorithm [49] solves the all pairwise shortest path problem in $\mathcal{O}(|V|^3)$. It can outperform $|V|$ calls to Dijkstra's algorithm on very dense graphs ($|E| \gg |V|$) with nonnegative edge weights.

2.2.2 Advanced techniques

To speed-up shortest paths algorithms and reduce the search space (*i.e.*, number of scanned vertices), several techniques have been proposed including bidirectional searches, goal-directed searches, hierarchical techniques and table-based techniques.

Bidirectional search [94] simultaneously launches a forward and a backward search respectively from the source vertex s and the target vertex t. The search terminates when the two search spaces intersect (*i.e.*, a vertex has been scanned by both search algorithms).

A* [66] is a well known goal-directed search algorithm. It uses a modified version of Dijkstra's algorithm which replaces the priority distance of a vertex u by $dist(s, u) + h(u)$ where $h : V \to \mathbb{R}$ returns the estimation of the shortest path from u to the target vertex t. This new priority distance forces the search algorithm to explore vertices that are near the target first. A* is guaranteed to output the shortest path provided the heuristic h is feasible (*i.e.*, $w(u, v) + h(v) - h(u) \geq 0 \ \forall (u, v) \in E$). Another goal-directed search algorithm: *ALT* [61] combines A*, the use of *landmarks* and *triangle inequality* to obtain better performances. First, it selects a subset of vertices called *landmarks* and pre-computes the distances $dist(l, u)$ between each landmark l and every vertex u of the graph. Then, during the online phase, the algorithm relies on triangle inequalities that hold between any vertex u, the target vertex t and a landmark l to compute lower bound of the the distance $dist(u, t)$ which in turn will be used by the A* search.

Hierarchical techniques rely on road networks structure to identify parts of the networks that are more likely to contribute to the shortest path (*e.g.*, high-speed ways). For instance highway hierarchy [103] exploits the natural hierarchy of roads (*i.e.*, primary, secondary, tertiary, residential, ...) and considers only roads with higher hierarchy along with roads in the immediate neighborhood of the source and target vertices. Contraction hierarchy [55] relies on the so-called *shortcuts* (edge representing the shortest path between two vertices) computed in a pre-computing phase to avoid non-important vertices during the online phase of long-distance queries. First, it orders vertices according to their importance (*e.g.*, vertices on primary roads are more important than those on secondary roads). Then, for each vertex v (from the less important to the most important), adjacent vertices whose priorities are higher are selected. For each pair (u, w) of such vertices, if the shortest path from u to w goes through v, v is removed and a shortcut (u, w) is added to the graph. Finally, in the online phase, a modified bidirectional Dijkstra's algorithm, which prioritizes edges incident to important vertices, is run.

Table-based techniques pre-compute all pairwise distances between a subset of important vertices and store distances in a table. During the online phase, they query the table to retrieve distances. For instance Transit Node Routing [104] considers a subset $T \subseteq V$ of transit nodes. Transit nodes could be nodes that are more likely to have higher centrality (*e.g.*, access point to high-speed roads or roads exits). Next, it computes all pairwise shortest paths within T. Then, for each vertex $u \in V \setminus T$, it computes its set of the so-called *access nodes* $A(u) \subseteq T$. More precisely, a transit node $v \in T$ is an *access node* for a vertex u if, given the *out-shortest path tree* $\mathcal{T}(u)$ rooted at u, v is the first visited transit node in a path included in $\mathcal{T}(u)$. Finally, during the online phase, the shortest path from s to t is obtained by selecting the path $a(s) - a(t)$ in T such that the length of the path $s - a(s) - a(t) - t$ is minimized.

2.2.3 Public transportation networks

Mobility constraints. In transportation networks, a highly desirable feature is the ability of users to take advantage of the wide spectrum of transportation modes (*e.g.*, car, walk, bicycle, bus, subway, tramway, ferry, rental bike, ...) while scheduling their journeys. In this case, in addition to minimizing traveling time, other mobility constraints can be expressed like avoiding car when the traffic is heavy, avoiding bike on rainy days, accepting only a limited number of transfers. To capture these constraints, the *Regular Language Constrained Shortest Path Problem* (RegLCSPP) has been formulated in [15]. It allows to express constraints as word of a regular language L. That is, a path P is valid if the sequence $\mathcal{W}ord(P)$ of edges' labels along P belongs to L.

Definition 2.9. (Regular Language Constrained Shortest Path Problem (RegLCSPP)) Given an alphabet Σ of transportation mode, a directed edge-labeled graph $G = (V, E, \Sigma)$, a regular language $L \in \Sigma^*$, a source vertex $s \in V$ and a target vertex $t \in V$, find a shortest path P from s to t in G such that $\mathcal{W}ord(P) \in L$.

A modified version of Dijkstra's algorithm, hereafter referred to as DRegLC, has been proposed in [15] to solve RegLCSPP. DRegLC first uses L and the user specified constraints to create a Non-deterministic Finite Automaton $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where Q denotes a finite set of states, Σ a finite set of inputs symbols corresponding to the available transportation modes, $\delta : Q \times \Sigma \rightarrow Q$ the transition relation, q_0 the initial state and $F \subseteq Q$ the set of accepting states. Then, the algorithm uses both G and $\mathcal{A}$ to compute a product graph $G^\times = (V^\times, E^\times) = G \times \mathcal{A}$ that will allow to navigate in the multimodal graph and the automaton at the same time. More precisely, a product vertex $v^\times = (u, q) \in V^\times \Leftrightarrow v \in V$ and $q \in Q$ and a product edge $e^\times = ((u, q_u), (v, q_v), l) \in E^\times \Leftrightarrow (u, v, l) \in E$ and $(q_u, q_v, l) \in \mathcal{A}$. Finally, the algorithm initialize at the product node (s, q_0) and select the path with minimum cost among the shortest paths from (s, q_0) to (t, q_f) for $q_f \in F$.

Time considerations. In transportation networks, the duration of a trip depends on its birth date. Thus, to obtain realistic trip scheduling, one need to consider timetable in-

formation while modeling public transportation networks. Generally speaking, there are two major approaches to model public transportation networks: the *time-expanded* and the *time-dependent* approaches. In the time-expanded model, vertices correspond to departure or arrival events at a station and edge costs are effective travel times. In the time-dependent model, vertices represent stations and edge costs are piecewise linear functions of time. In addition, time-dependent networks must have the FIFO property; that is, given a time-dependent cost function w, for any edge $e = (u, v) \in E$, for any time $\tau_1, \tau_2 \in \mathbb{R}_+$ $\tau_1 \leq \tau_2 \Rightarrow \tau_1 + w(e, \tau_1) \leq \tau_2 + w(e, \tau_2)$. Authors in [96] compare both approaches and conclude that the time-dependent model leads to better performance while the time-expanded can easily model, by nature, complex transfer scenarios (like train transfers). To make the time-dependent models more realistic, one can integrate a minimum transfer time at each station. In this thesis, we consider the time-dependent model.

Definition 2.10. (Time-dependent shortest path problem (TDSPP)) Given a directed graph $G = (V, E)$, its time-dependent cost function w, a source vertex $s \in V$, a target vertex $t \in V$ and a departure time τ_s, find a path P from s to t in G such that its cost $w(P, \tau_s)$ is minimum.

TDSPP can be solved with any forward relaxation-based approach (like Dijkstra or A*) provided the FIFO property hold for the graph. However, one needs to add an extra label to store the arrival time of each vertex and use it as input for the time-dependent cost function. On the other hand, using bidirectional is not straightforward since the exact arrival time at the target is unknown. To solve this issue, authors in [83] use a backward search from the target vertex using lower bounds on edges' costs to restrict the set of vertices that have to be explored by the forward search.

2.3 Dynamic Ridesharing

Ridesharing has widely benefited from academic results in operation research and progress in communication technologies. Dynamic ridesharing has been introduced in [2] and refers to an automated system that arranges one-time trips between riders and drivers in real-time. According to authors in [2], dynamic ridesharing has six main features:

- **Dynamic.** Trips are arranged in real-time, from a few minutes to a few hours before the departure time.
- **Independent.** Drivers are independent and do not work for a central entity that employs them or owns the cars.
- **Cost sharing.** Trip-related costs (*e.g.*, fuel expenses, parking costs, tolls, ...) are split between participants in order save money. For riders, it is competitive to traditional traveling mode (*e.g.*, airplane, train, ...), and drivers save money by sharing empty seat of their cars and splitting the cost with riders.
- **Non-recurring.** It focuses on single, non-recurring trips.

- **Prearranged.** Participants agree to share a ride in advance (*i.e.*, while they are not yet at the pickup location).
- **Automated matching.** It helps drivers and riders to find suitable matches with minimal effort. It also facilitates the communication between them (*e.g.*, a dedicated mobile app with in-box messaging feature).

2.3.1 Routing aspects

In [52], authors propose four patterns to characterize ridesharing *w.r.t.* the routing aspect:

- **Identical ridesharing.** Both rider and driver have the same origin and destination locations.
- **Inclusive ridesharing.** Both rider's origin and destination location are on the driver original route.
- **Partial ridesharing.** Pick-up and drop-off locations of the rider are on the driver original route. That is, ridesharing is only part of the rider's trip, who is willing to use other transportation modes to reach the pick-up and to get to his destination after the drop-off.
- **Detour ridesharing.** Similarly to partial ridesharing, it supposes that ridesharing is only part of the rider's trip. Additionally, pick-up and drop-off are authorized to not be on the driver's original route. Thus, the driver will make a detour to pick up and deliver the rider.

Detour ridesharing can be viewed as a generalization of identical, inclusive and partial ridesharing. In [21], authors solve the dynamic ridesharing problem in the detour ridesharing setting by introducing the 2 Synchronization Points Shortest Path problem (`2SP-SP`). In the `2SP-SP`, given a directed edge-labeled graph $G = (V, E, \Sigma)$, a driver d and a rider r, each having their respective origin and destination locations and departure times, the objective is to find the optimal pick-up point i^* and drop-off point j^*, such that the sum of arrival times for both users is minimized. Several variants were studied, such as the situation in which there is no limitation on the detour for both users and all points in the network may be a meeting point. In a second variant, the authors introduce detour limitations for the rider. The proposed method has a polynomial complexity and makes five calls to a time-dependent DRegLC algorithm (in forward and in backward search) and also solves a sub-problem: the so-called *Best Origin Problem*, which, given a set of vertices, selects the best origin vertex to be used in order to reach a target vertex. In [4], authors also consider the `2SP-SP` problem on road networks in the detour ridesharing setting. They propose heuristic methods based on several shortest path algorithms and a subgraph of potential pick-up and drop-off points. In another paper [122], public transport is taken into account for the rider, and the authors propose to limit the meeting points on the rider path from his origin to his destination and

still consider the detour constraint for the driver. In [111], authors show that considering meeting points other than origin and destination locations improve the number of matching and thus the overall functionality of the ridesharing system.

2.3.2 Matching aspects

Authors in [2] distinguish four types of matching scenarios namely: *single rider - single driver*, *single rider - multiple drivers*, *multiple riders - single driver* and *multiple riders - multiple drivers* scenarios.

Single rider - single driver. In this scenario, each driver will have at most one pickup and delivery during his/her trip. A classical solution used in this setting is a modeling of the system as a weighted bipartite graph of drivers and riders, in which weights of feasible edge correspond to either vehicle miles or traveling time. The underlining maximum-weight bipartite matching problem is solved to find optimal assignments for drivers and riders. Such approach has been used in [1] and assessed in a simulation environment, in term of matching success rate. In [56], authors consider the problem from rider's perspective and provide a method to find for a rider's request (s, t), the driver's offer with the smallest detour among a list of k offers (s_i, t_i). That is, the proposed solution outputs the offer such that $dist(s_i, s) + dist(s, t) + dist(t, t_i)$ is minimum. To do so, instead of running $2k + 1$ shortest path queries for each incoming rider's offer, the proposed method pre-computes for each s_i a forward search from s_i and stores the forward search space in a so-called *forward bucket* $B^{\uparrow}$; likewise, a *backward bucket* $B^{\downarrow}$ is computed by running a backward search from each t_i. Then, for each incoming rider's offer (s, t), all the distances $dist(s_i, s) + dist(s, t) + dist(t, t_i)$ are computed with a single backward (forward) search from s (t) that scans the forward (backward) bucket.

Single rider - multiple drivers. In this scenario, also known as multi-hop ridesharing [63], a rider is transferred from one driver's car to another driver's car at transfer points (like public transport stops, shopping malls, ...) until he/she reaches his/her final destination. Authors in [69] consider this problem in the partial ridesharing setting (*i.e.*, drivers do not deviate from their original routes) and model drivers' offers as a time-expanded graph and solve for each rider's origin-destination query, a multi-objective shortest path algorithm is run to find a route that minimizes costs, time and number of transfers. Authors in [43], consider the problem in the detour ridesharing setting and solve it while supposing a fixed set of potential transfer points. The proposed approach takes into account detour constraints as well as waiting time constraints, and similarly to the approach proposed in [69], it models the problem as a time-expanded graph on which a multi-objective shortest path algorithm is run.

Multiple riders - single driver. In this scenario, the expected solution is a planning of the pick-up and drop-off of each rider. Baldacci *et al.* [13] formulate the problem as an integer

programming problem and propose both an exact and a heuristic method to solve it. Calvo *et al.* [26] also consider the problem and solve it with a heuristic based on local search.

Multiple riders - multiple drivers. In this scenario, the solution consists in finding routing for both drivers and riders. Drivers are allowed to take several riders. Furthermore, riders can be transferred from one car to another. Herbawi *et al.* [70] consider this problem as an optimization problem with a mono-objective function which is a weighted sum of the drivers' total traveling times and distances, the riders' total traveling time and the number of matches. They propose a genetic algorithm to solve it.

Part II

Privacy-preserving ridesharing systems

Chapter 3

Meeting Points in Ridesharing: a Privacy-Preserving Approach

The ubiquitous world in which we live has fostered the development of technologies that take into account the context in which users are using them to deliver high-quality services. For example, by providing personalized services based on the positions of users *Location-based services* (LBS) [9, 97] encourage the emergence ridesharing services.

In traditional ridesharing services, the main studied problem is to assign riders to drivers given various constraints or objectives [52, 2]. However, as stated in [2, 111], another issue arises when considering meeting points between riders and drivers to satisfy more assignments or to produce better solutions provided some objectives including reducing the travel time spent by the rider, partially using the public transportation system or merely walking before reaching the pick-up location. Instead of putting the burden of finding the meeting points on the users, the system itself should be able to identify the optimal meeting points to make the trip as short as possible for both participants.

The problem of meeting points for ridesharing is illustrated in Figure 3.1. We consider a driver and a rider, each having their origin and destination locations. Both users are looking for a pick-up and drop-off locations that will induce a small detour for the driver and that the rider can use to share a trip with the driver. The objective of the routing component of a ridesharing platform is to find for both users a pick-up and drop-off locations and optimal itineraries for their journeys. The optimization function considered is the arrival times of both users (*i.e.*, the sum of their arrival times).

Recent academic works [21] have considered this issue for multimodal transportation networks and proposed an exact method to automatically solve it with a centralized system in a polynomial time. In [122], some heuristic approaches are also proposed, that aim to reduce the computation time, by limiting the set of meeting points on the rider itinerary. However, these works do not consider privacy issues related to the sharing of location data.

In this chapter, we design a privacy-preserving protocol that a driver and a rider can use to compute their pick-up and drop-off locations, without revealing their origin and destination locations. More precisely, we focus on casual, one-time and irregular ridesharing scenarios. In regular ridesharing, a trust is supposed already established between riders and drivers. We also expect an almost real-time response for a better user experience.

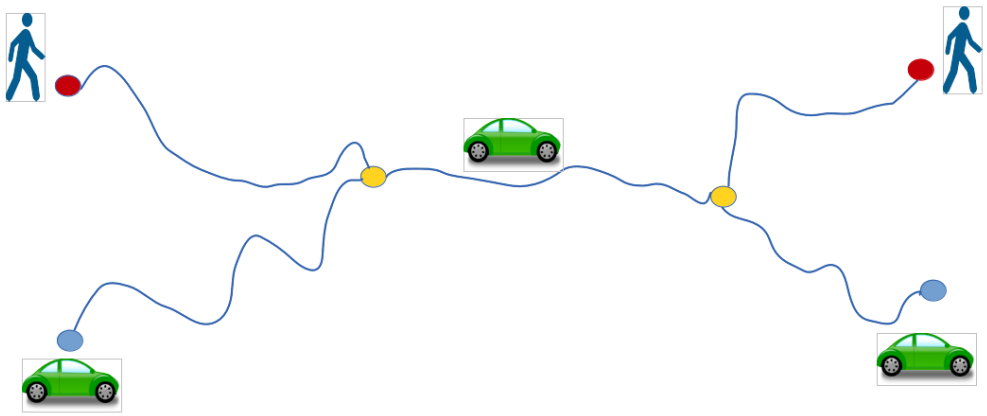

Figure 3.1: Ridesharing meeting point problem.

We formalize the secure meeting points for ridesharing problem and design two privacy-preserving protocols `Priv-2SP-SP` and `PPLD4R` to solve it. Both `Priv-2SP-SP` and `PPLD4R` are P2P protocols in which users make computations on their sensitive data in local and use secure two-party computations to determine the meeting points.

The remainder of this chapter is organized as follows:

In Section 3.1, we discuss recent related work on meeting points for dynamic ridesharing and location privacy-preserving mechanisms. In Section 3.2, we propose a model and a formulation of the secure meeting points for ridesharing (`SMP4R`) problem. In Section 3.3, we present `Priv-2SP-SP` a solution to compute meeting points for ridesharing in a privacy-preserving way. In Sections 3.4 and 3.5, we motivate and discuss security and privacy of the proposed solution as well as its performances from both analytic and experimental points of view. In Section 3.6, we highlight limitations of `Priv-2SP-SP` and provide an advanced protocol `PPLD4R` to get rid of these flaws. In Section 3.8, we provide in-depth evaluation of `PPLD4R`. In particular, we analyze both computation and communication overheads as well as the quality of the solutions. We compare it to a centralized and *non-secure* approach and show that introducing privacy constraints does not degrade the performance of a ridesharing system. In Section 3.9, we discuss some practical aspects of our solution, namely the monthly data plan, the consideration of malicious users, and the real world implementation of our solution. We conclude with a recapitulation of findings and future works in Section 3.10.

3.1 Related works

Related works to this chapter fall into two categories, namely meeting points in dynamic ridesharing and location privacy-preserving mechanisms.

3.1.1 Meeting points for dynamic ridesharing

The most relevant work, on dynamic ridesharing, to the present contribution is by Bit-Monnot *et al.* [21] who propose the so-called `2SP-SP` (2 Synchronization Points Shortest Path Problem) to compute optimal meeting points (pick-up and drop-off) for dynamic ridesharing. The proposed solution, as mentioned in Section 2.3.1, has a polynomial complexity and outputs the optimal pick-up and drop-off locations, as well as the itineraries that both rider and driver will use to complete their journey. The proposed approach takes into account the multimodal aspect of transportation network while computing travel time of the rider. The main limitation of this method is that it does not consider protecting location privacy of both the driver and the rider. We overpass this limitation by proposing a privacy-preserving protocol to compute meeting points for dynamic ridesharing.

3.1.2 Location privacy-preserving mechanisms

When users interact with location-based services, they need to share location information with service providers. However, the fine-grained nature of location data raises obvious privacy risks. To tackle this issue, location privacy-preserving mechanisms (*LPPMs*) aim at protecting users' location privacy during their interactions with location-based services [108].

LLPMs can be implemented with either a centralized or a distributed architecture. In a centralized setting, a trusted third party collects users' private inputs and applies data protection routines (*e.g.*, anonymization, dummy generation, encryption, ...) before sending obfuscated to service providers. In contrary, in distributed architectures, data protection techniques are applied directly at users' levels. Our secure protocol relies on the later to achieve location privacy for both driver and rider. Prior works related to distributed LPPMs include privacy-preserving location sharing systems [71], privacy-preserving spatiotemporal matching [113] and privacy-preserving *Fair Rendez-Vous Problem (FRVP)* [20].

In [71], authors devise a cryptographic privacy-preserving protocol for location-sharing based systems. They propose two variants, both based on *Identity-based Broadcast Encryption* [101]. Identity-based Broadcast Encryption schemes are public key encryption schemes that can use arbitrary strings as public keys, and allow senders to efficiently broadcast ciphertexts to a large set of receivers such that only non-revoked receivers can decrypt them. Authors also design a vector commitment scheme to allow service providers to collect aggregate statistics about users' check-ins. The proposed solution is evaluated in term of computation time, energy consumption and bandwidth overhead.

In [113], authors design a secure protocol to matches spatiotemporal profiles. Spatiotemporal profiles are vectors in which users continuously record their whereabouts in time. Matching spatiotemporal profiles have several applications, including ad targeting, participatory sensing, finding people with common interests, ... To protect users' location privacy, while matching spatiotemporal profiles, authors propose two solutions respectively based on the use *private set intersection cardinality* [50] and *Bloom filters* [42]. In both approaches, spatiotem-

poral profiles are generalized and then intersected. The proposed solutions are evaluated in term of efficacy and efficiency.

In [20], authors propose a privacy-preserving algorithm to compute optimal meeting location for a group of users. In the proposed approach, to determine the optimal meeting point, each participant uses a homomorphic cryptosystem to encrypt the coordinates of her preferred location before sending the ciphers to the service provider. Upon receiving all the ciphers from each user, the service provider computes the distances between each pair of potential meeting points obliviously and sends the result to the users. Finally, users select the location i^* that minimizes the maximum distance from any other place to i^*. In contrast, we propose a secure protocol to compute meeting points and separations for ridesharing. Besides, we consider actual travel time. To the best of our knowledge, our work is the first that relies on secure multiparty computations and multimodal routing algorithms to design privacy-protection mechanisms for ridesharing.

3.2 System model

We propose a secure ridesharing system to compute meeting points for ridesharing. Our objective is twofold: (*i*) the proposed protocol should provide privacy protection to both the driver and the rider and (*ii*) it should provide better or equivalent usability compared to `2SP-SP` and existing ridesharing systems. In this section, we present the system's participants, the problem formalization as well as the adversarial and system assumptions. In Table 3.1, we summarize the notations used throughout this chapter.

3.2.1 Participants

We assume a system consisting of two parties: a driver d and a rider r. Each user u is associated with a profile $[O_u, D_u, \tau^u_{O_u}, \Sigma^u]$ which includes the origin location O_u, the departure time $\tau^u_{O_u}$, the destination location D_u as well as the set of available transport modes Σ^u. A profile is said to be protected if any of its components O_u, $\tau^u_{O_u}$, D_u and Σ^u is not leaked to the service provider or the other participant.

While origin and destination locations can be easily considered as private information, as they may correspond to home or work address, the sensitivity of departure time and transportation modes is more ambiguous even though they can allow an adversary to learn origin and destination locations. For instance, as the distribution of the transportation infrastructure (*e.g.*, subway coverage, ferry location ...) is not uniform in a city, the way a target moves can help the adversary in improving his/her prior beliefs of the target's origin and destination. Likewise, the knowledge of the departure time can be used to perform a triangulation attack. In a triangulation attack, an adversary infers the location of the user based on the travel costs needed by her to reach a set of locations.

Symbol	Meaning
$\mathcal{S}$	Set of ridesharing stations.
τ_l^u	Departure time of user u from location l.
$[O_u, D_u, \tau_{O_u}^u, \Sigma^u]$	Profile of a user u with origin location O_u, destination location D_u, departure time $\tau_{O_u}^u$ and transportation modes Σ^u.
π_{ij}	A path from location i to location j.
$W_{\Sigma^u}(\pi_{ij})$	Cost of the shortest path from location i to location j for user u while using transport modes Σ^u.
$W^t_{\Sigma^u}(\pi_{ij}, \tau_i^u)$	Cost of the time-dependent shortest path from location i to location j for user u while using transport modes Σ^u and starting the trip at τ_i^u.
$spt_\tau(l, \tau, m)$	Time-dependent shortest path tree rooted at location l at date τ with m as transportation modes. $spt_\tau(l, \tau, m)$ returns a set of locations with their respective minimal travel times from the origin (*i.e.*, $\{(l_i, \tau(l_i))\ \forall i = 0, \ldots, n\}$).
$spt_\pi(l, m)$	Estimated shortest path tree rooted at l. When there exists some time-dependent modes, the exact shortest path tree cannot be computed as starting time is unknown, but we can estimate it, for instance using the minimal travel time on each arc...
$\mathcal{M}_u^\uparrow$	Potential pick-up locations of user u.
$\mathcal{M}_u^\downarrow$	Potential drop-off locations of user u.
$sc^u(\pi_{ij})$	Score given by the user u to the path π_{ij}.

Table 3.1: Summary of notations.

3.2.2 Problem statement

Definition 3.1. Trip cost. Given a pick-up location $i \in \mathcal{S}$, a drop-off location $j \in \mathcal{S}$, a driver d with profile $[O_d, D_d, \tau_{O_d}^d, \Sigma^d]$ and a rider r with profile $[O_r, D_r, \tau_{O_r}^r, \Sigma^r]$, for a ridesharing scenario engaging the driver d and the rider r, we define the trip cost $Tr_{i-j}(d)$ (respectively $Tr_{i-j}(r)$) of the driver (respectively the rider) as follows :

$$Tr_{i-j}(d) = W_{\Sigma^d}(\pi_{O_d i}) + W_{\Sigma^d}(\pi_{ij}) + W_{\Sigma^d}(\pi_{jD_d})$$
$$Tr_{i-j}(r) = W^t_{\Sigma^r}(\pi_{O_r i}, \tau_{O_r}^r) + W_{\Sigma^d}(\pi_{ij}) + W^t_{\Sigma^r}(\pi_{jD_r}, \tau_j^r)$$

The trip cost of a user includes the transit time from her origin to the pick-up location, the duration of the shared trip between the pick-up and the drop-off locations, and her transit time from the drop-off location to her final destination.

Definition 3.2. Ride cost. Given a pick-up location $i \in \mathcal{S}$, a drop-off location $j \in \mathcal{S}$, a driver d with profile $[O_d, D_d, \tau_{O_d}^d, \Sigma^d]$ and a rider r with profile $[O_r, D_r, \tau_{O_r}^r, \Sigma^r]$, for a ridesharing

between the driver d and the rider r, we define the ride cost $\texttt{Ride}_{i-j}(d,r)$ as follows :

$$\textit{Ride}_{i-j}(d,r) = Tr_{i-j}(d) + Tr_{i-j}(r) + |\tau_i^d - \tau_i^r|$$

The ride cost of a pair of driver and rider includes the trip cost of each user and the minimum waiting time at the pick-up (defined as the difference between the arrival times of both users).

The secure meeting points for ridesharing problem hereafter referred to as SMP4R is defined as follows:

Definition 3.3. Secure Meeting Points for Ridesharing (SMP4R) problem. Given a driver d with profile $[O_d, D_d, \tau_{O_d}^d, \Sigma^d]$, a rider r with profile $[O_r, D_r, \tau_{O_r}^r, \Sigma^r]$, and the set $\mathcal{S}$ of ridesharing stations, find locations $(i^*, j^*) \in \mathcal{S} \times \mathcal{S}$, such that the ridesharing cost $\texttt{Ride}_{i^*-j^*}(d,r)$ is minimum and the profile of both driver and rider is protected.

The expected solution for the SMP4R problem is the tuple (i^*, j^*) of meeting and separation locations and five paths $[\pi_{O_r i^*}, \pi_{O_d i^*}, \pi_{i^* j^*}, \pi_{j^* D_r}, \pi_{j^* D_d}]$. Each path may have its restrictions. That is, the rider may want to use walking and tramway as transportation modes to reach the pick-up i^* from his/her origin and solely use walking to get to his/her destination from the drop-off j^*. The ridesharing cost is computed as the sum of trip costs of both users and the waiting time at the pick-up location i^*. The shared path $\pi_{i^* j^*}$ is considered twice as it is included in the itinerary of both the driver and the rider. Finally, the driver's traveling time is assumed time-independent as her paths are on the road network.

3.2.3 Adversarial and system assumptions

Adversarial assumptions. We assume that both driver and rider are *honest-but-curious* adversaries. Drivers and riders might attempt to learn private profile of each other.

Security We assume that the passenger and the driver are already in contact. A secure protocol for establishing such contact is presented in Chapter 4. The primary security requirement is the privacy of user data. Users should be able to control their personal data. In particular, no user should be able to access the profile of another user unless explicitly authorized. Users communicate over a secure channel and have sufficient computing resources on their platforms (*e.g.*, personal computer or smart-phone) to perform the tasks requiring local computations such as the cryptographic ones.

3.3 `Priv-2SP-SP` — A secure protocol for the computation of ridesharing's meeting points

In this section, we propose `Priv-2SP-SP` (*Privacy-preserving 2 Synchronization Points Shortest Path Protocol*) a privacy-preserving protocol to solve the `SMP4R` problem.

3.3.1 Overview

At a high level, `Priv-2SP-SP` is a $P2P$ protocol in which both users locally compute their potential meeting points and use a private set intersection to find common meeting points. Afterward, the pair of pick-up and drop-off locations that minimize the ride cost is selected as a solution for the ridesharing meeting point problem.

More precisely, the driver (respectively the rider) locally computes her potential pick-up locations and her potential drop-off locations. Then, both users obtain their common pick-up (respectively drop-off) locations by computing the intersection of their potential pick-up (respectively drop-off) locations. A privacy-preserving set intersection protocol is used to reduce the amount of information exchanged between both users. Simply put, users learn only shared preferences. Finally, the best pick-up (respectively drop-off) location is selected.

Rather than working in a centralized way as in `2SP-SP`, `Priv-2SP-SP` is a decentralized approach and consists of three main steps: (1) local computation of potential meeting points (2) secure calculation of shared pick-up and drop-off locations and (3) selection of the best pick-up and drop-off points. Figure 3.2 depicts the protocol.

3.3.2 Local computation of potential meeting points

This step is twofold.

In the first step, each user locally computes two isochrones, one from the origin location and the other from the destination location (*c.f.*, lines 1,2). For a user u, we denote by $\mathcal{I}_u^{\uparrow}$ (respectively $\mathcal{I}_u^{\downarrow}$) the isochrone rooted at his origin (respectively his destination).More precisely, $\mathcal{I}_r^{\uparrow}$ is computed with a time-dependent regular language constrained shortest path algorithm to capture both time-dependency and mobility aspects, and a regular language constrained shortest path algorithm is used to estimate $\mathcal{I}_r^{\downarrow}$. For the driver, as the road network is assumed not time-dependent, a regular language constrained shortest path algorithm is used to compute both $\mathcal{I}_r^{\uparrow}$ and $\mathcal{I}_d^{\downarrow}$.

In the last step, the isochrones computed in the previous step are used to filter out accessible ridesharing stations (*c.f.*, lines 3,4). In addition, for each user u, the travel costs of the trip from the origin location O_u to potential pick-ups $i \in \mathcal{M}_u^{\uparrow}$ (respectively from potential drop-offs $j \in \mathcal{M}_u^{\downarrow}$ to the destination location D_u) are stored in private database ($\mathcal{H}_u^{\uparrow}$ or $\mathcal{H}_u^{\downarrow}$) (*c.f.*, lines 5,6). For the computation of common meeting points, only the identifier of

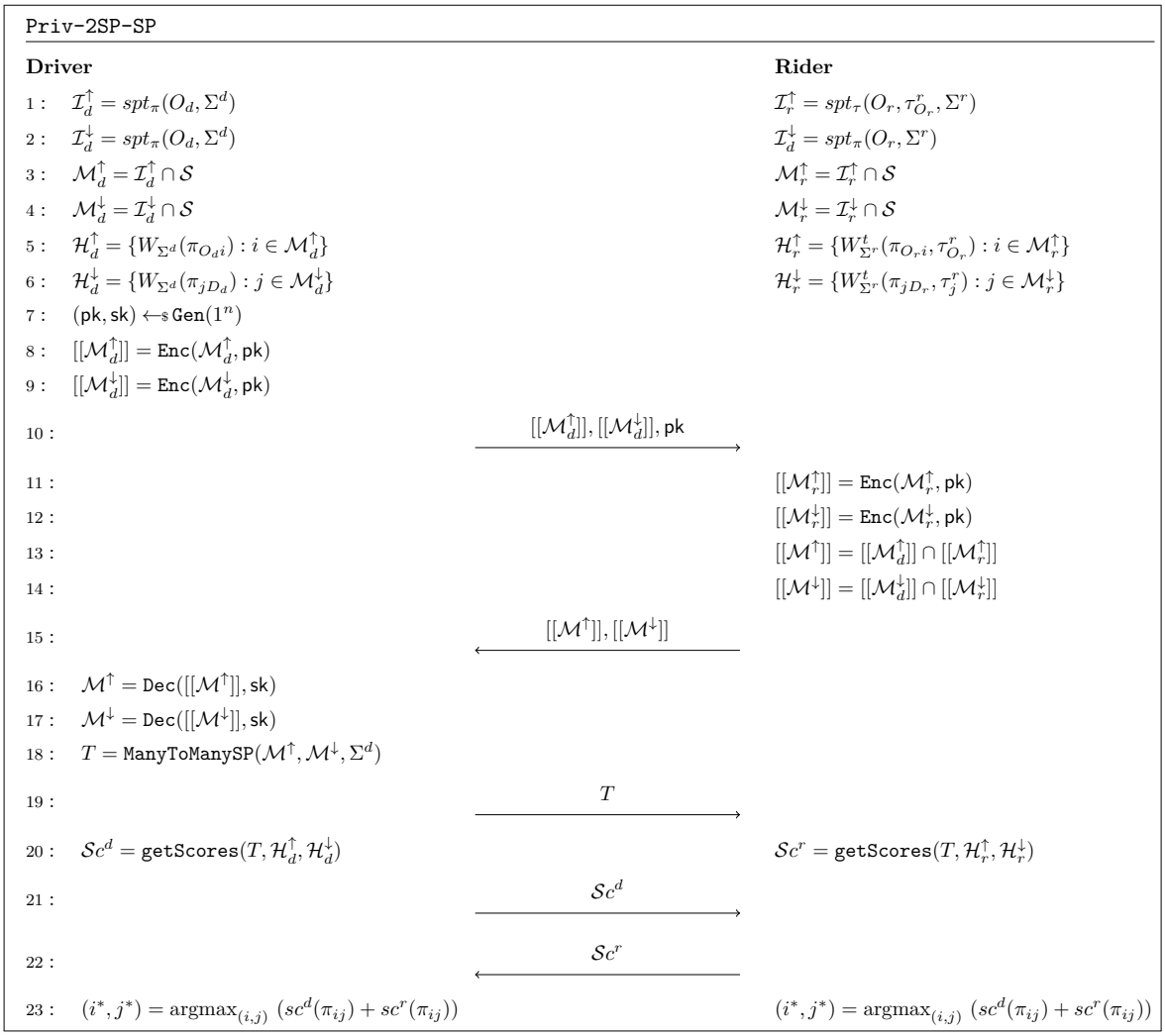

Figure 3.2: `Priv-2SP-SP` — A secure protocol for meeting points.

accessible ridesharing stations (stored in $\mathcal{M}_u^{\uparrow}$ and $\mathcal{M}_u^{\downarrow}$) will be used, and both $\mathcal{H}_u^{\uparrow}$ and $\mathcal{H}_u^{\downarrow}$ will remain confidential to avoid the possibility of *triangulation attacks*.

3.3.3 Secure collaborative computation of common meeting points

Once potential ridesharing locations are locally identified, both users collaboratively compute common pick-ups $\mathcal{M}^{\uparrow}$ and common drop-offs $\mathcal{M}^{\downarrow}$ with the *PSI* protocol. The private set intersection is chosen instead of the regular set intersection to follow the data minimization strategy as recommended by the *privacy-by-design* approach [29]. Consequently, at the end of this step users will get only common ridesharing locations.

To achieve this goal, the driver first generates a pair of public and private keys $(\mathsf{pk}, \mathsf{sk})$ that will be used in the private set intersection protocol (*c.f.*, line 7). Then, she encrypts her potential pick-up (respectively drop-off) locations $\mathcal{M}_d^{\uparrow}$ (respectively $\mathcal{M}_d^{\downarrow}$) and sends the encrypted items $[[\mathcal{M}_d^{\uparrow}]]$ and $[[\mathcal{M}_d^{\downarrow}]]$ to the rider along with her public key pk (*c.f.*, lines 8 — 10). Upon receiving of the public key and the encrypted potential meeting points, the rider obliviously computes the intersections of potential pick-up (respectively drop-off) locations (*c.f.*, lines 11—14) and sends the ciphers $[[\mathcal{M}^{\uparrow}]]$ and $[[\mathcal{M}^{\downarrow}]]$ to the driver (*c.f.*, line 15). Finally, the driver decrypts the ciphers and obtains the set of common pick-ups $\mathcal{M}^{\uparrow}$ and the set of common drop-offs $\mathcal{M}^{\downarrow}$ (*c.f.*, lines 16,17).

Notice that, as the *PSI* protocol is asymmetric, only the initiator of the protocol gets the final output. That is, in our case, only the driver will learn the common meeting points. To get shared meeting points, the rider also needs to initiate a *PSI*. However, as we are in the *honest but curious* setting, the driver will send the output to the rider without cheating.

3.3.4 Computation of shared paths

In this step, the driver computes the set T of all the costs (when using her transportation modes Σ^d) of the shared paths π_{ij} between the set of common pick-ups $\mathcal{M}^{\uparrow}$ and the set of common drop-offs $\mathcal{M}^{\downarrow}$ (*c.f.*, line 18). The set of all the shared path is then sent to the rider (*c.f.*, line 19).

The subroutine `ManyToManySP`$(\mathcal{M}^{\uparrow}, \mathcal{M}^{\downarrow}, \Sigma^d)$ requires the computation of $\min(|\mathcal{M}^{\uparrow}|, |\mathcal{M}^{\downarrow}|)$ shortest path algorithms to get all the $|\mathcal{M}^{\uparrow}| * |\mathcal{M}^{\downarrow}|$ paths between $\mathcal{M}^{\uparrow}$ and $\mathcal{M}^{\downarrow}$. If the number of common pick-ups and drop-offs is important, this subroutine may induce high computational overhead as it requires the computation of several shortest paths. We deliberately leave the driver in charge of this computation for scenarios in which we would like to take into account more complex itinerary's constraints associated with the driver, that he only can express (*e.g.*, taking a particular highway or using a particular parking station). In the absence of such constraints, both driver and rider can compute the shared path locally.

To reduce this overhead, both users iteratively compute the intersection of their isochrones by gradually growing the radius. Iterations may stop as soon as a condition on the size of the

intersection set is reached. This strategy helps in reducing the number of shared paths to be computed.

3.3.5 Local computation of ridesharing scores

Once the set T of shared path is obtained, each user u computes the trip cost $Tr_{i-j}(u)$ induced by each shared path $\pi_{ij} \in T$ by using the private databases $\mathcal{H}_u^{\uparrow}$ and $\mathcal{H}_u^{\downarrow}$ previously computed. That is, for each shared path π_{ij}, the driver (respectively the rider) computes its corresponding trip cost $Tr_{i-j}(d)$ (respectively $Tr_{i-j}(r)$) as follow:

$$Tr_{i-j}(d) = \mathcal{H}_d^{\uparrow}(i) + W_{\Sigma^d}(\pi_{ij}) + \mathcal{H}_d^{\downarrow}(j)$$
$$Tr_{i-j}(r) = \mathcal{H}_r^{\uparrow}(i) + W_{\Sigma^d}(\pi_{ij}) + \mathcal{H}_r^{\downarrow}(j)$$

At the end of this step, the shared path π_{ij} such that the trip cost $Tr_{i-j}(u)$ is minimum will be the most relevant for each user u. As travel cost in isochrones can reveal enough information to enable a triangulation attacks, trip costs can give enough details to infer the area in which the origin and destination locations are located. Therefore, rather than relying on the trip costs to elect the ideal shared path, each user first sorts the set of shared paths T *w.r.t.* the trip cost. Then, she associates to each shared path a score (by calling the `getScores` subroutine) which depends on its rank (*c.f.*, line 20).

More precisely, the driver (respectively the rider) outputs for each shared path π_{ij} a score $sc^d(\pi_{ij}) = \texttt{d.getScores}(\pi_{ij})$ (respectively $sc^r(\pi_{ij}) = \texttt{r.getScores}(\pi_{ij})$ reflecting her willingness to use it as a subpath during the ridesharing trip. For simplicity we define `getScores` as:

$$\texttt{getScores} : \left| \begin{array}{ccc} \mathcal{P} & \longrightarrow & \mathbb{N} \\ \pi_{ij} & \longmapsto & |\mathcal{M}^{\uparrow}| * |\mathcal{M}^{\downarrow}| - rank(\pi_{ij}) \end{array} \right.$$

In both cases, the better the rank of a shared path, the higher its score. We design the scoring function as the complementary of the rank. The conversion of the trip cost to a score has the advantage of adding more privacy on the choice of each user because it discloses less information than the trip cost itself. However, further privacy improvements can be made by randomizing the scoring function to make it more difficult for an adversary to infer the rank from the score.

3.3.6 Election of ideal pick-up and drop-off locations

In this step, each user communicates the score she attributes to each π_{ij} (*c.f.*, lines 21 — 22). Then, a simple voting procedure is applied in which instead of taking into account the trip costs, the focus is made on the scores only (*i.e.*, the values of costs are kept confidential for

privacy reasons).

Instead of using the simple voting procedure, other voting strategies would be possible provided they not be computationally expensive. Note that the same process must be employed by the two users to obtain the same result. `2SP-SP` sought to minimize ridesharing cost while `Priv-2SP-SP` reduces the problem to the maximization of the scores.

To achieve this goal, each user selects the shared path π_{ij} such that $(sc^d(\pi_{ij}) + sc^r(\pi_{ij}))$ is maximized. More precisely, the pick-up and drop-off locations corresponding to the solution are obtained the following manner: $(i^*, j^*) = \text{argmax}_{(i,j)}\ (sc^d(\pi_{ij}) + sc^r(\pi_{ij}))$ (*c.f.*, line 23).

By default, the election works in a fair manner by deciding that a score assigned to a given path has the same weight for both parties. However, it is easy to design an unbalanced version of this vote by using a positive multiplicative factor reflecting the importance given to one party over the other. In this case, the best path will be the one such as $\alpha \times \mathsf{sc}^{\mathsf{d}}(\pi_{\mathsf{ij}}) + \beta \times \mathsf{sc}^{\mathsf{r}}(\pi_{\mathsf{ij}})$ is maximized with α and β representing respectively the influence of the driver and of the rider on the choice of a shared path.

3.3.7 Waiting time

In the proposed approach, there is no guarantee that the waiting time of each participant at the pick-up location will be the lowest one. To ensure that the waiting time meets the constraints of the schedule of each user, one can securely check if $|\tau_i^d - \tau_i^r| \leq \theta$, where θ represents the maximal waiting time the users are willing to tolerate at the pick-up location i. To take into account the limitation of each user on the waiting time, in this last step it is important to check if the waiting time corresponding to the best-selected path is valid using a secure comparison method. If this condition is not met (*i.e.*, the time is not valid), users have to select the second path and so forth until the waiting time of the path considered matches their constraints.

3.4 Analysis of `Priv-2SP-SP`

3.4.1 Security analysis

Since we solve `Priv-2SP-SP` by using a *PSI* method, the security of our scheme depends mainly on the security of the PSI method used. In [50], the authors proved the security of PSI in the semi-honest model. We recall hereafter the sketch of the scheme and the security proof.

Consider a party A running a PSI method with another party B. To summarize A starts by generating a public and private encryption keys of an homomorphic encryption scheme before defining a polynomial whose roots are her inputs. Then, she sends the public key and the encryption of the polynomial coefficients to B. Once B receives the public key and the

polynomial, he evaluates, in an oblivious fashion, the polynomial with his inputs and returns the result to A. Finally, A decrypts an item if it is included in the intersection of the two sets or a random number otherwise.

The *correctness* of the PSI method is ensured by the fact that it successfully evaluates with high probability. The *privacy of the input of A* is guaranteed if the encryption scheme used is semantically secure. Indeed, in this case, the views of B for any two inputs of A are indistinguishable. The *privacy of the input of B* comes from the fact that for every party A' operating in the real world, there is a party A operating in the ideal model, such that for every input of B, the views of the parties A and B in the ideal model are indistinguishable from the views of A' and B in the real world.

3.4.2 Privacy analysis

Anonymity of the driver. In `Priv-2SP-SP`, the rider receives encrypted inputs from the driver. Hence, she cannot learn anything about the driver's inputs, as the *PSI* protocol's encryption scheme offers semantic security, i.e., it is (computationally) impossible for the rider to distinguish whether two different ciphertexts conceal the same plaintext [50].

Anonymity of the rider. After receiving the results of the *PSI* protocol, the driver can only learn the items that she and the rider have in common (and nothing else), as the *PSI* protocol is secure in the honest but curious setting [50].

3.4.3 Communication and computational complexities of `Priv-2SP-SP`

We respectively denote by $|V|$ and $|E|$, the number of vertices and the number of edges in the transportation network.

The complexity of the proposed approach is directly proportional to the complexity of the shortest path algorithms. More precisely, each participant runs twice the DRegLC algorithm (which has a complexity of $\mathcal{O}((|E| + |V|) \times \log |V|)$) to get their potential pick-up and drop-off locations. Finding the common POIs using *PSI* has a linear complexity of $\mathcal{O}(|V|)$. Once common pick-up and drop-off locations are discovered, each participant makes at most $|V|$ calls to the DRegLC algorithm to get all the $|V|^2$ possible shared paths. The scoring and ranking step require the insertion of $|V|^2$ paths into a binary heap. All these steps lead to a global computational complexity of $\mathcal{O}(|V| \times |E| \times \log |V|)$.

Finally, the communication complexity is $\mathcal{O}(|V| \times \log |V|)$ as the identifier of each node may have to be exchanged, and the use of an efficient homomorphic encryption adds a constant overhead.

The respective communication and computational complexities of the exact and centralized approach (`2SP-SP`) and the privacy-preserving approach (`Priv-2SP-SP`) are summarized in Table 3.2.

	Communication cost	Computational cost
2SP-SP	$\mathcal{O}(1)$	$\mathcal{O}(\lvert E\rvert \times \lvert V\rvert^2)$
Priv-2SP-SP	$\mathcal{O}(\lvert V\rvert \times \log \lvert V\rvert)$	$\mathcal{O}(\lvert E\rvert \times \lvert V\rvert \times \log \lvert V\rvert)$

Table 3.2: Computational and communication complexities of 2SP-SP and Priv-2SP-SP.

The 2SP-SP protocol is a centralized protocol. Therefore, it has a constant communication cost. On the other hand, the Priv-2SP-SP protocol has a quasi-linear (in the number of vertices) communication cost. The Priv-2SP-SP protocol is faster than 2SP-SP.

3.5 Evaluation of Priv-2SP-SP

In this section, we report on the experiments that we have conducted to evaluate the effectiveness of our approach.

3.5.1 Experimental settings

Transportation network. The tests have been performed using a multimodal transportation graph of the city of Toulouse in which we consider any vertex as potential ridesharing location. The multimodal graph has been generated using data from OpenStreetMap[1] for the road network and GTFS files from *Tisseo*[2] for the public transportation network. The resulting graph has the following characteristics: $|V| = 75\ 837$, $|E| = 527\ 053$, $\Sigma = \{\text{Walk, Car, Bus, Subway, Tramway}\}$.

Instances. Overall, we randomly generate 200 instances of ridesharing problem. To generate a ridesharing instance, we randomly choose (in the transportation network) an origin and a destination for each pair of driver and rider. Origins and destinations locations are selected to geographically close in the transportation network to reflect a real-world situation. More precisely we create two groups each containing 100 instances of ridesharing. In Table 3.3, we summarize key features of each group. The main difference between the two datasets is

	Group I	Group II
Distance between the both origins (respectively destinations)	2000 m	800 m
Travel time for the rider alone	1h15min	2h
Travel time for the driver alone	19min	20min

Table 3.3: Instances characteristics.

the proximity between the driver and the rider. In the first group ridesharing candidates are

[1] http://www.openstreetmap.org
[2] https://developers.google.com/transit/gtfs

more distant to each other.

Implementations details. The multimodal routing algorithms have been implemented in C++. For our cryptographic operations, we rely on the *NFLlib* library [3], an optimized open-source C++ library which includes optimized subroutines to perform arithmetic operations over polynomials and allows to easily implement ideal lattice-based cryptography [79]. As explained in Section 3.3.4, to reduce the overhead the shared path computation subroutine, we compute the intersection of the isochrones by gradually growing their radius. In this experiment, we divide each isochrone into rings such that the difference between the two radii of each ring is 1 minutes. Then, we intersect rings and stop iterations as soon as there is 1 common meeting point.

Experiments were run in a virtual machine with 5GB RAM, on a 2.9 GHz Intel Core i7 host machine.

3.5.2 Experimental results

To evaluate our protocol, we run a `Priv-2SP-SP` protocol for each of our generated instances and compare our results to the optimal solutions obtained using the centralized method `2SP-SP` [21] (which does not take into account any privacy requirements).

Computation overhead. We summarize in Table 3.4 average values of the computation overhead and the ridesharing cost for instances in both Group I and Group II. Concerning the runtime, `Priv-2SP-SP` is more efficient than `2SP-SP`. This observation confirms our preliminary complexity analysis.

	`Ridesharing cost` [min]		CPU [s]	
	Group I	Group II	Group I	Group II
`2SP-SP`	52 ± 13	47 ± 11	1.09 ± 1.66	0.68 ± 0.55
`Priv-2SP-SP`	54 ± 14	49 ± 11	0.67 ± 0.37	0.49 ± 0.30

Table 3.4: Ridesharing costs and computational overhead per pair of driver and rider. Statistics (avg $\pm$ std) are computed over the 100 instances in both Group I and Group II.

Distribution of the computation overhead. `Priv-2SP-SP` is composed of 4 major phases, namely the computation of potential meeting points, the determination of common meeting points, the computation of shared paths, and the computation of ideal pick-up and drop-off locations. In Table 3.5, we summarize the distribution of the computation overhead between the most significant phases of `Priv-2SP-SP`. More precisely, we report on each phase, its runtime per user, and the ratio of this runtime to the overall runtime per user. Additionally, we recall the algorithm it uses.

The most computationally intensive subroutines are the computation of potential meeting

	Technique	Group I		Group II	
		Driver	Rider	Driver	Rider
Phase 1	`One-To-All SPP`	110 ms (16%)	190 ms (34%)	110 ms (22%)	170 ms (45%)
Phase 2	`PSI`	308 ms (45%)	308 ms (56%)	178 ms (36%)	178 ms (47%)
Phase 3	`Many-To-Many SPP`	203 ms (30%)	0 ms (0%)	180 ms (36%)	0 ms (0%)
Phase 4	`Scoring`	56 ms (8%)	56 ms (10%)	31 ms (6%)	31 ms (8%)

Table 3.5: Distribution of the computation between the major phases of `Priv-2SP-SP`.

points, the computation of shared paths, and the private set intersection protocol. As shown in Table 3.6, on average, there are $23 * 17 = 391$ (respectively $15 * 14 = 210$) shared paths per instance of Group I (respectively Group II). The higher overhead introduced by Phase 2 is because we compute common meeting points with several calls of the PSI protocol. In average, it takes 6 (respectively 3) iterations to compute common meeting points for instances of Group I (respectively Group II).

Ridesharing cost and quality of the solutions. Results in Table 3.4 show that `2SP-SP` performs slightly better than `Priv-2SP-SP` as it always returns the optimal solution.

We use the gap between the ridesharing cost of `Priv-2SP-SP` and that of `2SP-SP` to evaluate the quality of our protocol. More precisely, the gap is computed as the percentage difference between the two values. On average over the 200 instances, the gap between the ridesharing cost of `Priv-2SP-SP` and `2SP-SP` is small (4.40% in group I and 2.45% in group II). In Figure 3.3, we report on the distribution of the gap for both Group I and Group II instances. Overall, 75% of the instances in Group I have a gap less than 6.19% and 75% of the instances in Group II have a gap less than 3.36%. That is, the smaller the distance between drivers and riders the smaller the gap between the optimal solution and the solution of the privacy-preserving approach.

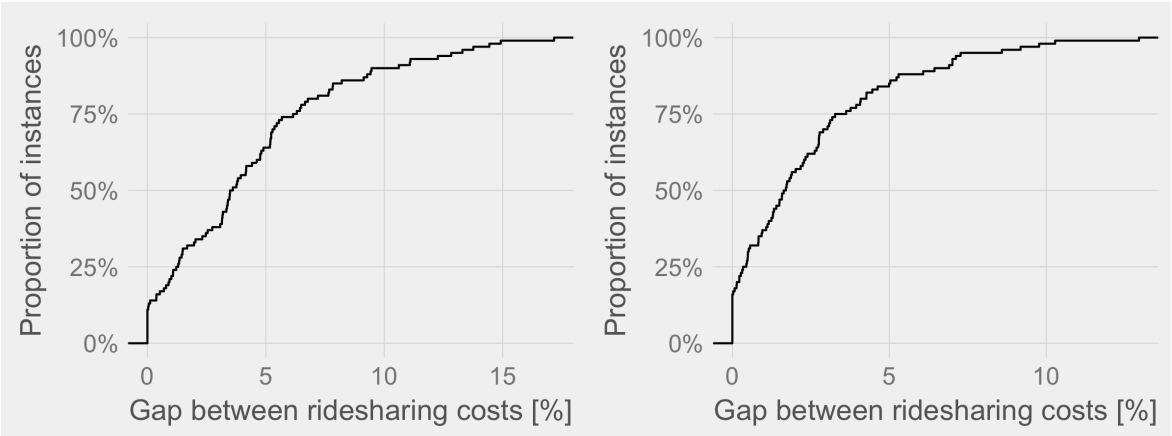

Figure 3.3: Distribution of gaps between ridesharing cost in Group I (left) and Group II (right).

In Table 3.6, we report on the average value of the size of common pick-up and drop-off locations for the `Priv-2SP-SP` approach, and the similarity between the solutions of `2SP-SP` and those of `Priv-2SP-SP` regarding the distance between their pick-up locations on the one hand and the proximity of the drop-off locations on the other.

The size of common pick-up and drop-off locations is small because we intersect subsets of isochrones in several iterations instead of intersecting entire isochrones. This strategy helps in reducing the number of shared paths.

The average distance between ridesharing locations is 230.06 (respectively 103.42) meters for pick-ups and 202.13 (respectively 99.23) meters for drop-offs in Group I (respectively Group II). More details are given by Figures 3.4 and 3.5. To summarize, for 75% of the instances in Group I, the distance between the pick-up location (respectively the drop-off location) of the optimal and that of the privacy-preserving protocol is less than 311 (respectively 328) meters. In Group II, the geographic proximity of pick-up locations (respectively drop-off locations) of both approaches is less than 154 (respectively 150) meters for 75% of the instances.

Observation	**Group I**	**Group II**
Pick-ups size	23	15
Drop-offs size	17	14
Proximity of pick-ups (m)	230.06	103.42
Proximity of drop-offs (m)	202.13	99.23

Table 3.6: Additional characteristics of `Priv-2SP-SP`'s ridesharing solutions.

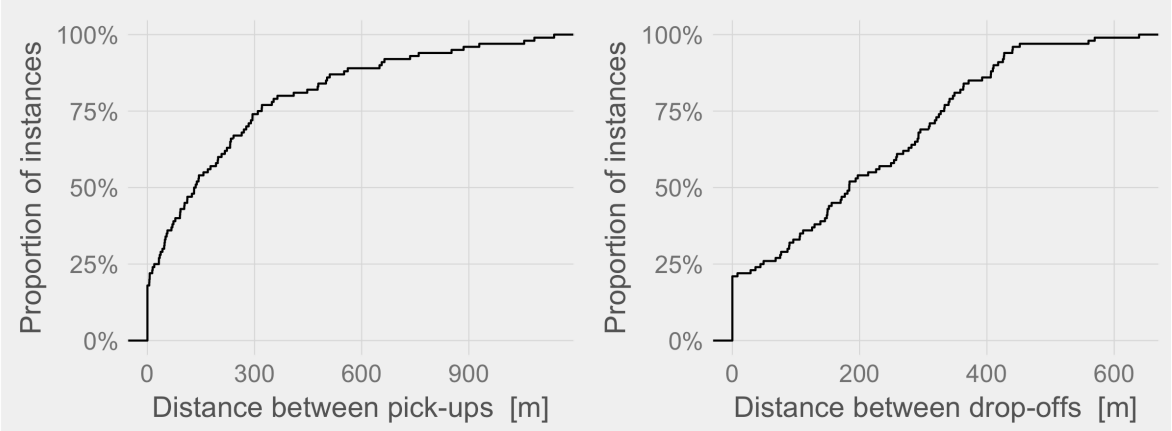

Figure 3.4: Distribution of the distance between pick-ups (left) and the distance between drop-offs (right) in Group I

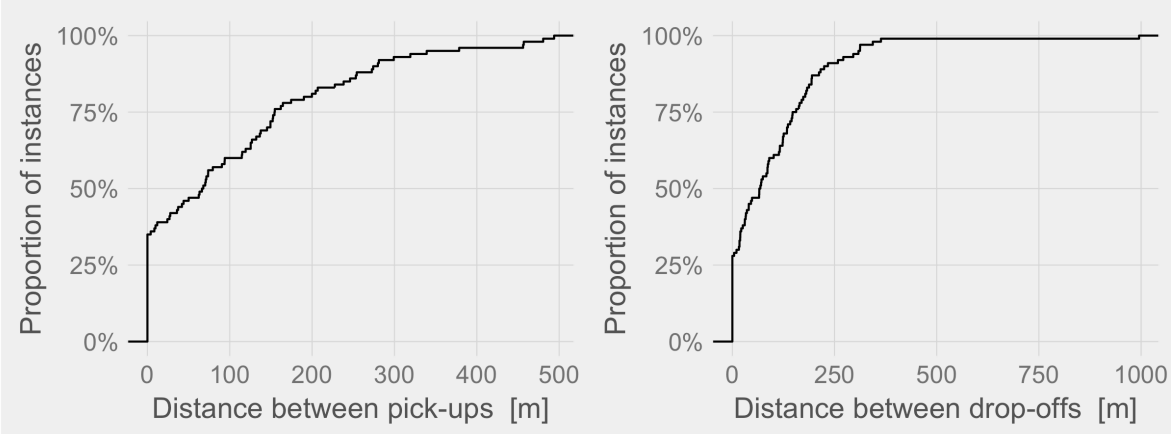

Figure 3.5: Distribution of the distance between pick-ups (left) and the distance between drop-offs (right) in Group II

3.6 PPLD4R — A privacy-preserving location determination protocol for ridesharing

So far, we propose Priv-2SP-SP, a privacy-preserving protocol to compute interesting ridesharing meeting points for drivers and riders while protecting their location privacy. As one can notice, in the current form of the proposed protocol, waiting times for both driver and rider at the pick-up location are not captured by the trip cost model. Furthermore, we consider every node in the transportation network as a ridesharing station. Because this assumption increases the computational overhead of the protocol, we use an iterative approach which considers subsets of isochrones to compute common meeting points and shared paths. Furthermore, a scoring algorithm is used to hide real trip costs. Consequently, the solution found by Priv-2SP-SP approach is not always the optimal one, though the gap is acceptable.

In this section, we present PPLD4R, a privacy-preserving location determination protocol for ridesharing.

PPLD4R is an improved version of Priv-2SP-SP which tackles the above-mentioned limitations. More precisely, our improvements are fourfold. First, we assume that the set of ridesharing stations is a predefined subset of vertices known by everyone. Next, we propose a speed-up technique based on the pre-computation of shared paths. Then, we rely on a secure comparison protocol to integrate the waiting time in the trip cost model. Finally, we use a secure shared sum protocol to consider real trip costs instead of scores when selecting the best meeting points.

3.6.1 Ridesharing stations

In Section 3.3.2, we make the assumption that any node in the transportation network can be a ridesharing stations ($\mathcal{S} = V$). As we show in our preliminary experiments, the pri-

vate set intersection protocol contributes significantly to the total computational overhead of `Priv-2SP-SP`, and its impact on the scalability will increase with the size of the transportation network.

To tackle this issue, we propose to consider only a predefined subset $\mathcal{S} \subset V$ of vertices known by every user. Furthermore, the number of ridesharing stations is expected to be very small as compared to the order of the transportation network; that is, $|\mathcal{S}| \ll |V|$. In fact, in a real-world scenario, ridesharing pick-up locations are usually public transport stops or parking of shopping malls. As a consequence, the driver and the rider do not need to run a private set intersection protocol to compute common meeting points. Instead of that, both users consider all the ridesharing stations while computing the best meeting points. For unreachable ridesharing stations, the trip cost is set infinity.

3.6.2 Pre-computation of shared paths

Since the potential ridesharing station are known in advance, we can compute and store the shared paths offline. Then, during the online phase, distances will be retrieved in constant time. More precisely, given the set $\mathcal{S}$ of ridesharing stations, we compute all-pairs shortest paths between the ridesharing stations $s \in \mathcal{S}$ and store them in a lookup table $\mathcal{B}$. This step requires $|\mathcal{S}|$ calls to the algorithm DRegLC.

3.6.3 Integration of waiting time in the trip cost

By definition

$$\begin{aligned}\texttt{Ride}_{i-j}(d,r) &= Tr_{i-j}(d) + Tr_{i-j}(r) + |\tau_i^d - \tau_i^r| \\ &= W_{\Sigma^d}(\pi_{O_d i}) + W^t_{\Sigma^r}(\pi_{O_r i}, \tau^r_{O_r}) + |\tau_i^d - \tau_i^r| \\ &\quad + 2 * W_{\Sigma^d}(\pi_{ij}) + W_{\Sigma^d}(\pi_{jD_d}) + W^t_{\Sigma^r}(\pi_{jD_r}, \tau_j^r)\end{aligned}$$

As arrival dates are positive values, we have

$$|\tau_i^d - \tau_i^r| = max(\tau_i^d, \tau_i^r) - min(\tau_i^d, \tau_i^r)$$

which implies

$$\begin{aligned}\texttt{Ride}_{i-j}(d,r) = W_{\Sigma^d}(\pi_{O_d i}) + W^t_{\Sigma^r}(\pi_{O_r i}, \tau^r_{O_r}) + max(\tau_i^d, \tau_i^r) - min(\tau_i^d, \tau_i^r) \\ + 2 * W_{\Sigma^d}(\pi_{ij}) + W_{\Sigma^d}(\pi_{jD_d}) + W^t_{\Sigma^r}(\pi_{jD_r}, \tau_j^r)\end{aligned}$$

Once both users securely determine arrival order at each common pick-up $i \in \mathcal{M}^{\uparrow}$, they can compute their trip costs with waiting time included. This step requires $|\mathcal{M}^{\uparrow}|$ calls to a secure comparison protocol. We introduce, a boolean flag $\varphi_u(i)$ to model the fact that a user u

arrives first at location i.

That is, for each shared path π_{ij} the driver's trip cost will be computed as

$$Tr_{i-j}(d) = \begin{cases} 2 * \mathcal{H}_d^{\uparrow}(i) + \mathcal{B}(i,j) + \mathcal{H}_d^{\downarrow}(j), & \text{if } \varphi_d(i) \\ \\ \mathcal{B}(i,j) + \mathcal{H}_d^{\downarrow}(j), & \text{otherwise} \end{cases}$$

Likewise, for each shared path π_{ij} the rider's trip cost will be computed as

$$Tr_{i-j}(r) = \begin{cases} 2 * \mathcal{H}_r^{\uparrow}(i) + \mathcal{B}(i,j) + \mathcal{H}_r^{\downarrow}(j), & \text{if } \varphi_r(i) \\ \\ \mathcal{B}(i,j) + \mathcal{H}_r^{\downarrow}(j), & \text{otherwise} \end{cases}$$

Notice that, the cost of each π_{ij} is directly retrieved from the lookup table $\mathcal{B}$. To implement the secure comparison protocol, we rely on the Goldreich-Micali-Wigderson (GMW) protocol [62] which provides security against passive adversaries. Furthermore, to reduce the communication overhead of the secure comparison subroutine, we encode all arrival times in a single circuit.

3.6.4 Integration of actual trip cost in the election phase

To select the ideal shared path, both users compute the ride cost as an arithmetic shared secret. More precisely, for each shared path π_{ij}, the driver (respectively the rider) holds a secret share $\mathcal{SS}_d(\pi_{ij})$ (respectively $\mathcal{SS}_r(\pi_{ij})$) such that $\mathcal{SS}_d(\pi_{ij}) + \mathcal{SS}_r(\pi_{ij}) = \texttt{Ride}_{i-j}(d, r)$. As explained in Section 1.7, one party (here the driver) uses a homomorphic cryptosystem to encrypt her trip cost and send the encrypted version as well as the public key to the rider. Upon receiving, the rider generates a random number and add it to her trip cost before computing (obliviously the sum) of both trip costs and sends it to the driver. The same operation is repeated for all shared path and the ideal pick-up and drop-off locations are computed as $(i^*, j^*) = \text{argmin}_{(i,j)}\ (\mathcal{SS}_d(\pi_{ij}))$. Notice that, the rider must use the same random number for each shared path to guarantee correctness.

We implement the homomorphic secret-sharing protocol by using the `Fan-Vercauteren (FV) homomorphic encryption scheme` [48], which offers semantic security. Furthermore, we represent all the trip costs as coefficients of a single $|\mathcal{S}|^2$th degree polynomial; that is, cryptographic operations are done coefficient-wise. This process allows us to reduce the communication overhead.

3.6.5 Putting It All Together

The PPLD4R protocol is summarized in Figure 3.6. First, the driver generates a pair of public and private keys of a homomorphic cryptosystem (*c.f.*, line 1). Then both users scan the set $\mathcal{S}$ of ridesharing stations to compute travel time (*c.f.*, lines 2,3). In order to compute the order of arrival at the potential pick-up points, both the driver and the rider run a secure comparison protocol to compute the boolean $\varphi_d(i)$ respectively $(\varphi_r(i))$ as $(\mathcal{H}_d^{\uparrow}(i) - \mathcal{H}_r^{\uparrow}(i)) > 0$ (respectively $(\mathcal{H}_r^{\uparrow}(i) - \mathcal{H}_d^{\uparrow}(i)) > 0$) for each ridesharing station $i \in \mathcal{S}$ (*c.f.*, line 4). Once arrival orders are determined, both users compute their trip costs. Afterward, for each shared path, the corresponding ridesharing cost is calculated as a shared secret (*c.f.*, lines 5–12). That is, for each shared path the driver holds a random $r_A = \texttt{Ride}(d, r) - r_B$ where r_B is a random number generated by the rider. Finally, the driver selects the shared path for which the corresponding shared secret is minimum (*c.f.*, lines 13, 14), and notifies the rider on the result (*c.f.*, line 15).

PPLD4R

	Driver: d		**Rider: r**
1 :	$(\texttt{pk}, \texttt{sk}) \leftarrow\!\!\$\ \texttt{Gen}(1^n)$		
2 :	$\mathcal{H}_d^{\uparrow} = \{W_{\Sigma^d}(\pi_{O_d i}) : i \in \mathcal{S}\}$		$\mathcal{H}_r^{\uparrow} = \{W_{\Sigma^r}^t(\pi_{O_r i}, \tau_{O_r}^r) : i \in \mathcal{S}\}$
3 :	$\mathcal{H}_d^{\downarrow} = \{W_{\Sigma^d}(\pi_{jD_d}) : j \in \mathcal{S}\}$		$\mathcal{H}_r^{\downarrow} = \{W_{\Sigma^r}^t(\pi_{jD_r}, \tau_j^r) : j \in \mathcal{S}\}$
4 :	$\varphi_d = \texttt{SecureGT}(\mathcal{H}_d^{\uparrow}, \mathcal{H}_r^{\uparrow})$		$\varphi_r = \texttt{SecureGT}(\mathcal{H}_r^{\uparrow}, \mathcal{H}_d^{\uparrow})$
5 :	$\texttt{Poly}_d = \Sigma_{i=1}^{\lvert\mathcal{S}\rvert}\Sigma_{j=1}^{\lvert\mathcal{S}\rvert} Tr_{i-j}(d) X^{ij}$		$\texttt{Poly}_r = \Sigma_{i=1}^{\lvert\mathcal{S}\rvert}\Sigma_{j=1}^{\lvert\mathcal{S}\rvert} Tr_{i-j}(r) X^{ij}$
6 :	$[[\texttt{Poly}_d]] = \texttt{Enc}(\texttt{Poly}_d, \texttt{pk})$		
7 :		$[[\texttt{Poly}_d]],\ \texttt{pk} \longrightarrow$	
8 :			$\mathcal{SS}_r = rand * \Sigma_{i=1}^{\lvert\mathcal{S}\rvert^2} X^i$
9 :			$[[\mathcal{SS}_r]] = \texttt{Enc}(\mathcal{SS}_r, \texttt{pk})$
10 :			$[[\texttt{Poly}_r]] = \texttt{Enc}(\texttt{Poly}_r, \texttt{pk})$
11 :			$[[\texttt{Poly}_{dr}]] = [[\texttt{Poly}_d]] + [[\texttt{Poly}_r]] - [[\mathcal{SS}_r]]$
12 :		$\longleftarrow [[\texttt{Poly}_{dr}]]$	
13 :	$\mathcal{SS}_d = \texttt{Dec}([[\texttt{Poly}_{dr}]], sk)$		
14 :	$(i^*, j^*) = \text{argmin}_{(i,j)}\ (\mathcal{SS}_d(\pi_{ij}))$		
15 :		$i^*,\ j^* \longrightarrow$	

Figure 3.6: PPLD4R — A privacy-preserving location determination protocol for ridesharing.

3.7 Analysis of PPLD4R

3.7.1 Security and privacy analysis

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 [62].

During the secret sharing subroutine, the rider receives encrypted inputs from the driver. Hence, she can not learn anything about the driver's inputs because of the semantic security of the FV scheme [48]. On the other hand, the driver receives random-looking numbers. Hence, she cannot learn anything about the rider's input.

3.7.2 Communication and computational complexities of PPLD4R

We respectively denote by $|V|$, $|E|$ and $|\mathcal{S}|$, the number of vertices, the number of edges, and the number of ridesharing stations in the transportation network. Furthermore, we denote by l and l_H the number of bits required to represent private inputs in the secure comparison, and in the homomorphic secret sharing subroutine of PPLD4R.

In PPLD4R, each participant makes two calls to the DRegLC algorithm (which has a computation complexity of $\mathcal{O}((|E|+|V|)\times \log|V|)$) to computes her trip costs. The secure comparison requires 4 interactions per AND gates, and for the GreaterThan circuit, there are $(3l - \lceil 2\log_2 l\rceil - 2)|S|$ AND gates [106]. Overall, the secure comparison protocol has a computation complexity of $\mathcal{O}(|\mathcal{S}|)$ and a communication complexity proportional to $|\mathcal{S}| * l$. The secret-sharing protocol has a computation complexity of $\mathcal{O}(|\mathcal{S}|^2)$ and a communication complexity of proportional to $|\mathcal{S}|^2 * l_H$. As in practice $|\mathcal{S}|^2 \ll |V|$, the overall computation complexity of PPLD4R is $\mathcal{O}((|E|+|V|)\times \log|V|)$. The overall communication complexity of PPLD4R is $\mathcal{O}(|\mathcal{S}|^2)$.

The respective communication and computational complexities of 2SP-SP and PPLD4R are summarized in Table 3.7.

	Communication cost	**Computational cost**
2SP-SP	$\mathcal{O}(1)$	$\mathcal{O}(\|E\|\times\|V\|^2)$
PPLD4R	$\mathcal{O}(\|\mathcal{S}\|^2)$	$\mathcal{O}((\|E\|+\|V\|)\times \log\|V\|)$

Table 3.7: Communication and computational complexities of 2SP-SP and PPLD4R.

As the 2SP-SP protocol is centralized, it has a constant communication cost while the PPLD4R protocol has a quadratic (in the number of ridesharing stations) communication cost. On the other hand, the PPLD4R is faster than 2SP-SP mainly because of the pre-computation of shared paths (as the ridesharing stations are predefined).

3.8 Evaluation of PPLD4R

In this section, we evaluate PPLD4R in term of solution quality as well as computational and bandwidth overhead. We compare its performance to 2SP-SP-V2, a modified version of 2SP-SP in which we force the ridesharing solutions' space to the set $\mathcal{S}$ of pre-defined ridesharing stations.

3.8.1 Experimental settings

For our experiments, we consider two cases, namely an intracity scenario and an intercity scenario. We run the intracity scenario in the city of *Toulouse*, and for the intercity scenario, we consider *Nantes* as the pick-up city and *Rennes* as the drop-off city.

Transportation network. We generate the multimodal transportation network by using data from *Openstreetmap*[3] for the road network and *GTFS* data from *Navitia*[4] for the public transportation network. the main characteristics of the transportation networks for both scenarios are given in Table 3.8.

Scenario	$\|V\|$	$\|E\|$	Transport modes
Toulouse	75837	527567	walk, bike, bus, subway, tramway, car
Nantes - Rennes	297664	2108820	walk, bike, bus, tramway, car

Table 3.8: Main characteristics of the transportation networks

Instances. We partition each city according to their districts as shown in Figure 4.5. In both scenarios, we generate 1000 instances of the ridesharing meeting point problem. More precisely, we first randomly select a district and generate the driver's origin within the selected district. Then the rider's departure location is generated within a radius of 5 kilometers around driver's origin. The same operation is conducted to produce destination locations. For simplicity, we consider the same departure time for both users. Finally, we generate $|\mathcal{S}| = 64$ ridesharing stations for each scenarios.

Figure 3.7: City of Toulouse [119] with its 6 districts (left), city of Nantes [84] with its 11 districts (middle) and city of Rennes [99] with its 12 districts (right).

Implementation details.

For the experiments, we use a computer (Intel Xeon CPU E3-1271 v3, 3.60GHz, 32GB RAM) running on Linux 3.13.

[3]http://www.openstreetmap.org
[4]https://www.navitia.io/datasets

The PPLD4R protocol is implemented in *C++*.

For the homomorphic secret sharing protocol, we use the *FV-NFLlib* library [5], which implements the *FV* scheme [48]. As the degree of the polynomial has to be a power of 2 (see [3]), for the 64 ridesharing stations we consider in our experiment, we use a $64 * 64 = 4096$th degree polynomial to integrate all the trip costs of each user in a single polynomial. The coefficients of the polynomial are 64-bit length and the modulus p has 124 bits. Therefore, a public key or a ciphertext takes up 124 KB, while a plaintext takes up 32 KB.

For the secure comparison protocol, we use the *ABY* framework [38], which implements both the Goldreich-Micali-Wigderson (GMW) protocol [62], with security against passive adversaries. Each of the $n = 64$ private shares used for the secure two-party comparison test are $l = 10$-bit length.

3.8.2 Isochrones' computation overhead

In addition to the pre-computation of shared path, the offline phase of the protocol includes the local computation of arrival times by each user. More precisely each user u uses two shortest path algorithms to compute a forward bucket $\mathcal{H}_u^{\uparrow}$ (respectively backward bucket $\mathcal{H}_u^{\downarrow}$) to store traveling times of the transit phase of the ridesharing. In this Section, we analyze the computation overhead introduced by this subroutine. Table 3.9 summarizes the runtime for each user in both scenarios.

	Sequential [ms]		Parallel [ms]	
	Driver	Rider	Driver	Rider
Toulouse	605.6 ± 1.68	663.9 ± 2.87	302.0 ± 1.54	331.0 ± 2.00
Nantes - Rennes	2691.8 ± 4.48	2875.6 ± 3.90	1338.8 ± 2.10	1426.3 ± 2.14

Table 3.9: Performances of isochrones' computation. Statistics (avg $\pm$ std) for computational overhead are computed over the 1000 instances.

Overall, when isochrones are computed sequentially, the runtime is less than 0.7 seconds (respectively 2.9 seconds) for both users in the intercity (respectively intracity) scenario, and performances increase by a factor of two with parallelization. Furthermore, since the ridesharing stations are known in advance, each participant can compute his travel time in advance, so that only the secure comparison protocol and the secret sharing protocol are run while a driver and a rider engage in a PPLD4R protocol. That is, the computational overhead of the travel time computation will not impact the overall performance of the PPLD4R protocol.

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

3.8.3 Secure comparison overhead

The performance of the secure comparison only depends on the number of ridesharing stations. That is, the communication overhead is the same in both scenarios and computation overheads are of the same order.

Communication overhead. We convert arrival times in minutes, which reduce the number of bits to represent them. In the secure comparison protocol, the two parties run a secure equality on their private shares obtained with the secret sharing protocol. Overall each party has 64 private shares, each of which is 10-bit length.

The ABY framework sends per *AND* gates 4 bits (2 bits per party) in the online phase. In the online phase, each party sends 1 bit per input and 1 bit per output. Hence, we have $1152/8 + 1536 * 2/8 + 64/8 = 536$ bytes in the online phase.

Computation overhead. In the setup phase, the secure comparison protocol takes about 0.854 ms to complete. The online phase takes only 0.461 ms to complete.

In Table 3.10, we report the communication and computation overheads of the secure comparison subroutine. The small memory footprint and computation time of the secure comparison subroutine are due to the use of the *SIMD* (Single Instruction Multiple Data) features of the *ABY* framework which allow encoding all the arrival times in a single circuit.

Communication [KB]				Runtime [ms]
Driver		Rider		
Upload	Download	Upload	Download	
0.536	0.536	0.536	0.536	0.461 ± 0

Table 3.10: Secure comparison performances. Statistics (avg $\pm$ std) for computational overhead are computed over the 1000 instances of the intracity scenario.

3.8.4 Secure shared sum overhead

Communication overhead. The 64 ridesharing stations lead to a total of 4096 different trip costs, which are later on encoded in a single 4096th degree polynomial to compute ridesharing costs as a shared sums. First, the driver sends to the rider a public key and a ciphertext for her encrypted trip costs. This process requires has a payload size of 248 KB. Then, the rider sends a ciphertext of random shares to the driver. This step has a payload size of 124 KB. A detailed summary of the communication overhead for both users is given in Table 3.11.

Driver		Rider	
Upload	Download	Upload	Download
248 KB	124 KB	124 KB	248 KB

Table 3.11: Communication overhead of the secure shared sum subroutine

Computation overhead. In Table 3.12 (respectively Table 3.13), we report the runtime of cryptographic operations performed by the driver (respectively the rider) during the secure shared sum sub-protocol.

Parameters	$(n = 4096, p = 124$ bits$)$
Keys generation [ms]	141 ± 0.0
Encrypt [ms]	17 ± 0.0
Decrypt [ms]	7 ± 0.0

Table 3.12: Secure shared sum performances for the driver. Statistics (avg $\pm$ std) for computational overhead are computed over the 1000 instances.

Parameters	$(n = 4096, p = 124$ bits$)$
Encrypt [ms]	17 ± 0.0
Hom. Add. [ms]	1 ± 0.0

Table 3.13: Secure shared sum performances for the rider. Statistics (avg $\pm$ std) for computational overhead are computed over the 1000 instances.

Overall, the footprint of cryptographic operations is small. In fact, it takes about 183 ms to complete all the cryptographic operations.

3.8.5 `PPLD4R` vs `2SP-SP-V2`

In this Section, we compare `PPLD4R` and `2SP-SP-V2` protocols regarding ridesharing cost, communication overhead, and computational overhead. In Table 3.14, we summarize the average ridesharing cost, the communication and the average runtime of both `PPLD4R` and `2SP-SP-V2` for both the intracity scenario and the intercity scenario.

Computational overhead. Regarding the computational overhead, the average runtime of `2SP-SP-V2` is 1.3 seconds (respectively 6.1 seconds) for the intracity (respectively intercity) scenario. On the other hand, the overall runtime of `PPLD4R` is very low (less than 200 ms). By relying on the pre-computation of shared paths and the use adequate cryptographic tools, `PPLD4R` achieves better computational performance than `2SP-SP-V2`.

Communication overhead. As `2SP-SP-V2` is centralized, it requires only a few bits (repre-

senting origin and destination locations) to compute the meeting points. On the other hand, PPLD4R is a *P2P* protocol in which users exchange encrypted data in other to compute their meeting points. Overall, PPLD4R requires about 249 KB of communication to complete. Even if this communication is higher than that of 2SP-SP-V2, it remains very low and acceptable. For instance, assuming a user of PPLD4R makes two intercity ridesharing every day, the monthly data usage induced will be about 14.6 MB. To compare, a regular daily use of 15 minutes of *turn by turn directions* navigation services like *Google Maps*, lead to a monthly data usage of 38 MB and visiting 5 web pages per day consumes 59 MB per month [34].

Solution quality. Overall, in both the intercity and intracity scenario, PPLD4R finds the same solution as 2SP-SP-V2 because integrating the waiting time and considering actual trip costs allow PPLD4R to have the same objective function as 2SP-SP-V2.

	PPLD4R			2SP-SP-V2		
	Ridesharing cost [min]	Communication [KB]	CPU [ms]	Ridesharing cost [min]	Communication [KB]	CPU [ms]
Toulouse	42	248.68	183.461	42	-	1300
Nantes - Rennes	201	248.68	183.461	201	-	6100

Table 3.14: Overall performance of PPLD4R and 2SP-SP-V2.

3.9 Discussion

3.9.1 Priv-2SP-SP Vs PPLD4R

We respectively denote by $|V|$, $|E|$ and $|\mathcal{S}|$, the number of vertices, the number of edges, and the number of ridesharing stations in the transportation network.

In Table 3.15, we summarize the overall communication and computational complexities of Priv-2SP-SP and PPLD4R protocols.

	Communication cost	**Computational cost**
Priv-2SP-SP	$\mathcal{O}(\|V\|\times\log\|V\|)$	$\mathcal{O}(\|E\|\times\|V\|\times\log\|V\|)$
PPLD4R	$\mathcal{O}(\|\mathcal{S}\|^2)$	$\mathcal{O}((\|E\|+\|V\|)\times\log\|V\|)$

Table 3.15: Computational and communication complexities of Priv-2SP-SP and PPLD4R.

Overall, PPLD4R has the best computational overhead. Moreover, it scales better than Priv-2SP-SP. In fact, its communication cost is independent of the size of the transportation network. Priv-2SP-SP is suitable for dynamic scenarios, in which the ridesharing stations are not known in advance. On the other hand, PPLD4R is suitable for scenarios in which the ridesharing stations are known in advance.

3.9.2 The case of malicious adversary

Although the fact we have considered in our system the honest but curious adversary model, it is interesting to see how one can deal with malicious adversary's behavior. At the opposite of honest but curious adversary, the malicious adversary does not intend to respect the protocol and will try to inject manipulated inputs like fake isochrone to leak information about users or to turn the ridesharing solution to her favor. To thwart such kind of attack, one can rely on a privacy-preserving location proof system [130] which, by allowing proof of location, will serve as a mean to provide certificates on isochrones. Furthermore, the secure two-party computation subroutines can be implemented to provide security against malicious adversaries (e.g., using [88, 87]), with additional computational and communication costs.

3.9.3 Real world implementation

All our experiments have been done in simulation mode on computer. In contrast, in the real world, we can rely on a *publish/subscribe* model to match drivers and riders in the first place and then a peer-to-peer communication will start between each couple of rider and driver to run the proposed scheme and obtain meeting and arriving points. In Figure 3.8, we present an architecture of the system. Our solution can be implemented with the existing network infrastructure and the current mobile technologies.

For the rider, *wearable* like *smart watch* can serve as a user interface, *smartphone* will be used for both local computations and *peer-to-peer* communications. Finally, the most computationally expensive subroutines will be delegated to a remote personal cloud.

For the driver, computations can be balanced between an embedded calculator and a *smartphone*, which will also take care of *peer-to-peer* communications.

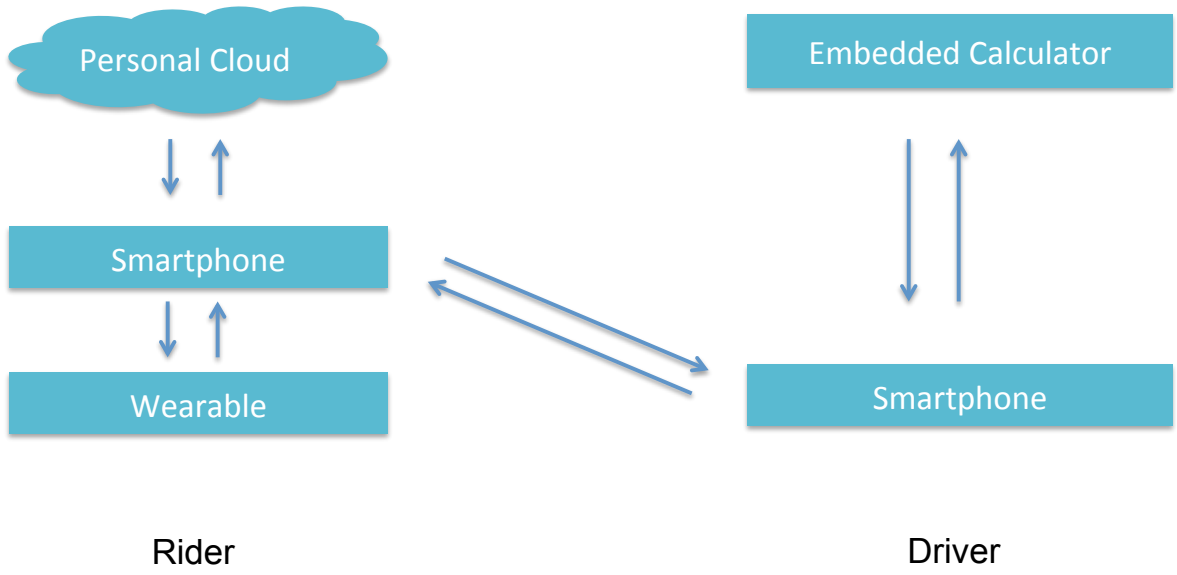

Figure 3.8: Architecture of the system.

3.10 Summary

Ridesharing services have the potential to reduce local air pollution, greenhouse gas emissions, and congestion. However, the existing users' privacy concerns may disrupt the adoption of such service.

In this chapter, we have addressed the secure meeting points for ridesharing (`SMP4R`) problem by providing practical and effective solutions based on homomorphic encryption, private set intersection, secret sharing, and secure two-party computation schemes. Moreover, our solutions use multimodal routing algorithms to take into account mobility constraints of both the rider and the driver. We have designed and implemented our privacy-preserving protocols and evaluated their performances. Our solutions are proved to be effective regarding privacy and have acceptable performance compared to traditional systems that do not protect the privacy of users.

Chapter 4

SRide: A Privacy-Preserving Ridesharing System

In this chapter, 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 secure protocol based on several existing well-established techniques, combined in an adequate way 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 time and space generalizations to their private inputs. During the second stage, the set of potential drivers/rider pairs is reduced with a secure filtering protocol to those whose preferences and constraints have a reasonable match. In the third stage, a two-party protocol is executed between feasible pairs, to determine the meeting times and locations, as well as the overall quality of the match (i.e., ridesharing cost), in a privacy-preserving way (i.e., *without revealing origin and destination location*). 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 we 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 would compute the secure meeting points protocol `PPLD4R` (cf. Chapter 3) on the *complete* bipartite graph formed by the divers and the riders.

The rest of this chapter is organized as follows:

In Section 4.1, we describe the system model, and we formalize the privacy-preserving ride matching problem. We detail the proposed approach `SRide` in Section 4.2 and analyze its security, privacy and performance in Section 4.3. In Section 4.4, we describe our experimental setup and methodology, including the datasets used, and we report on our experimental results. In Section 4.5, we discuss how *SRide* can be extended to provide all the features for a real-world application. In Section 4.6, we survey academic works related to ridesharing and privacy enhancing technologies in transportation. We conclude this chapter in Section 4.7.

4.1 System model

Our objective is to design a ridesharing system that provides stronger privacy guarantees to both the drivers and the riders, without sacrificing the usability of the system. At a high level, our system is composed of three parties: drivers, riders, and the service provider (SP).

4.1.1 Notation

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 and m^r respectively denote the number of drivers and riders. We assume that each user has 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 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:

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

The maximum number of intermediate locations on a driver's trajectory 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 corresponding to the distance traveled by using public transport.

4.1.2 Adversarial model

We consider the following privacy threats defined in [93]:

- **T1: SP $\rightarrow$ D/R location tracking.** The SP might try to learn location data of drivers and riders, to improve its service quality or to monetize harvested data.

- **T2: D → R *PII* harvesting.** Drivers might try to learn personally identifiable information (*PII*) of riders (e.g. for stalking or blackmailing purposes).
- **T3: R → D *PII* harvesting.** Riders might try to learn *PII* of drivers (e.g. for stalking or blackmailing purposes).

In our model, the drivers, the riders, and the SP are passive adversaries (semi-honest). A semi-honest adversary is a computationally bounded adversary who tries to learn additional information from the messages seen during the protocol execution. In contrast to the stronger malicious (active) adversary, the semi-honest adversary is not allowed to deviate from the protocol. As the SP is semi-honest, we assume that it does not collude with the drivers (respectively the riders) in their attempt to collect private data of riders (respectively drivers). Finally, we assume that the SP cannot observe the IP addresses of drivers and riders, thanks to the usage of anonymous network systems, such as Tor [41].

4.1.3 Design goals

The goal of SRide is to protect ridesharing users from threats mentioned in Section 4.1.2 while offering similar usability as traditional ridesharing systems, which do not provide solutions to protect users' privacy. A profile is said to be **protected** if none of its spatiotemporal components (location and departure time) is leaked to the service provider or the other participants. While the origin and the destination locations can be easily considered as private information, as they may correspond to home or work address, the sensitivity of departure time is more ambiguous even though they can allow an adversary to learn origin and destination locations. For instance, the knowledge of the departure time can be used to perform a triangulation attack. In a triangulation attack, one can infer the location of a user based on the time she takes to reach a set of locations.

The desired properties of *SRide* are the following:

- **P1: Rider anonymity.** It is computationally difficult for the SP to infer the identities and the location data of a rider. It is also computationally difficult for a driver to learn the location data of a rider without explicit consent. **P1** addresses the location tracking threat **T1** and the *PII* harvesting threat **T2**.
- **P2: Driver anonymity.** It is computationally difficult for the SP to infer the identities and the location data of a driver. It is also computationally difficult for a rider to learn the location data of drivers with whom she does not match. **P2** addresses the location tracking threat **T1** and the *PII* harvesting threat **T3**.

4.1.4 Types of Ridesharing Systems

Our approach is designed to support common types of ridesharing implemented by state-of-the-art matching agencies, namely *identical ridesharing* and *inclusive ridesharing* [52].

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 Euclidean distance in a projection coordinates system).

In the *inclusive ridesharing* setting, riders may be picked-up and dropped on 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 distance of the rider must be less or equal to her maximum transit distance. The third constraint captures consistency between rider's arrival time at the pick-up and 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.

4.1.5 Problem statement

Definition 4.1. Privacy-preserving ride matching problem. Given a set $\mathcal{D}$ of drivers d with profile $\mathcal{P}^d = \{T^d, wt^d\}$, and a set $\mathcal{R}$ of riders r with profile $\mathcal{P}^r = \{(O^r, \tau^r_{O^r}), D^r, \delta^r\}$, find the subset of drivers and riders that satisfy the inclusive ridesharing constraint, while protecting the profile of each user.

4.2 `SRide` — A privacy-preserving ridesharing system

4.2.1 Naive approach

A naive solution to solve privacy concerns related to the matching in ridesharing is to consider the *complete* bipartite graph formed by all the drivers and riders, and run the secure meeting

point protocol for ridesharing PPLD4R (cf. Chapter 3) between every pair. The corresponding bipartite graph has $m^d \times m^r$ arcs (one arc per pair of driver and rider), each weighted by the ridesharing cost for the corresponding pair. Finally, by using a minimum cost bipartite matching algorithm, one can compute the optimal assignment for drivers and riders.

However, running PPLD4R for the complete bipartite graph can be expensive. In fact, it will require m^d peer-to-peer communication for each rider to interact with the drivers, which can introduce important communication and computation overheads. Our idea is to reduce 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 filtering protocol to filter out pairs of drivers and riders that are unlikely to match. Then we run the PPLD4R protocol on the *feasible* bipartite graph before computing optimal assignments based on ridesharing costs.

4.2.2 General overview

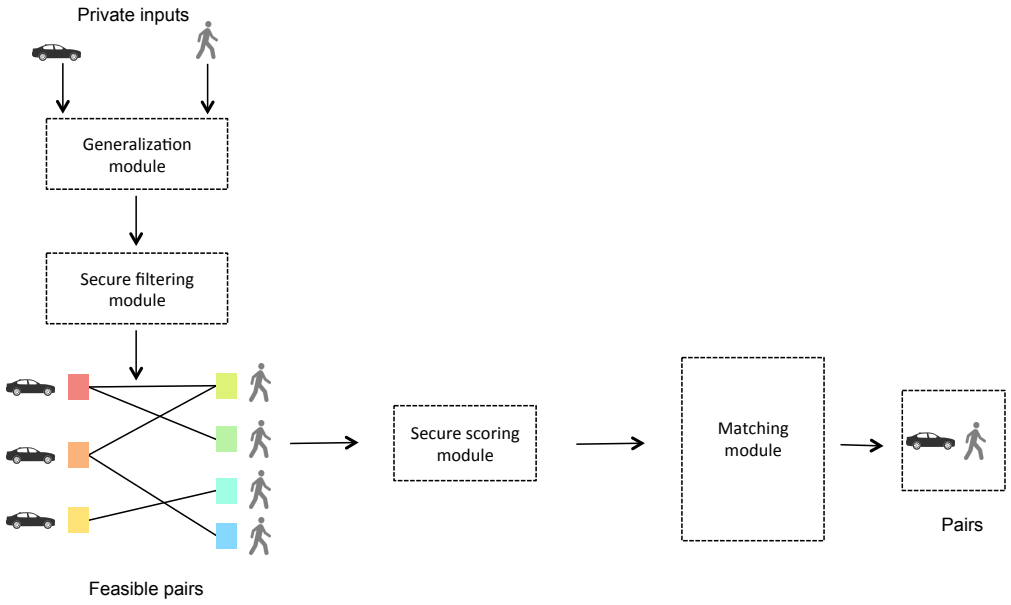

Figure 4.1: **SRide Overview:** Participants use their generalized inputs and the `Secure filtering` module to build a bipartite graph of feasible pairs. The `Secure scoring` module is used to securely compute ridesharing costs as well as meeting points. The `Matching` module takes the bipartite graph as input and computes optimal matches based on ridesharing cost. Finally, users are notified of their assignments.

At a high level, SRide is composed of four modules (see Figure 4.1), namely the *generalization* module, the *secure filtering* module, the *secure scoring* module and the *matching* module. Each of them will be detailed in the following sections.

To use SRide, whenever a rider is looking for a ride, she uses the *generalization* module to generalize her private inputs (see Section 4.2.3). Then, she uses the *secure filtering* module to

initiate a *secure filtering protocol* (see Section 4.2.4) 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 feasible matches, she relies on the *secure scoring* module to launch the `PPLD4R` protocol with each potential driver, and optimal assignments of drivers and riders are computed by the `SP` (see Section 4.2.5).

4.2.3 Generalizations of inputs

4.2.3.1 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 $\mathcal{E}$ of n_E epochs of the same size ω: $\mathcal{E} = \{e_t\}, \forall t \in 1, \ldots, n_E$. 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 $\mathcal{C} = \{c_s\}$ of n_C polygons or cells of the same area θ. The parameter θ can also represent a level of granularity (*i.e.*, country, city, borough, ...). We denote $\phi_\theta(l)$ the function that converts a location point $l \in \mathcal{L}$ to its corresponding cell $c_s \in \mathcal{C}$.

4.2.3.2 Computation of generalized inputs

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 $\mathcal{P}^r$. More precisely, for every location l_k on his trajectory T^d, each driver d enumerates all the combination $(l_k, \tau^d_{l_k}, l_{k'})\ \forall\ \tau^d_{l_{k'}} > \tau^d_{l_k}$ of pick-up, departure time and drop-off. Overall, each driver d generates an input vector with $|T^d| \times (|T^d| - 1)/2$ 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 (see Algorithm 1). The maximal size of drivers' generalized input vector is $n_T \times (n_T - 1)/2$ where n_T is the maximal number of locations in a driver's trajectory. For each rider, the corresponding generalized input vector $\mathcal{I}^r$ is composed of a single generalized spatiotemporal triplet which contains the generalization of her origin, departure time and destination (see Algorithm 2).

In these generalizations, we consider that the driver 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.

Data: $\mathcal{P}^d = \{T^d, wt^d, F^d\}, \omega, \theta$
Result: $\mathcal{I}^d$: Generalized Inputs
$\mathcal{I}^d = \emptyset$
for $k \in 1 \dots |T^d| - 1$ **do**
  for $k' \in k+1 \dots |T^d|$ **do**
    $(c_k, e_k, c_{k'}) = (\varphi_\theta(l_k), \varphi_\omega(\tau^d_{l_k}), \varphi_\theta(l_{k'}))$
    $\mathcal{I}^d = \mathcal{I}^d \cup \{(c_k, e_k, c_{k'})\}$
  end
end

Algorithm 1: `DriverMacroInputs`

Data: $\mathcal{P}^r = \{(O^r, \tau^r_{O^r}), D^r, \delta^r\}, \omega, \theta$
Result: $\mathcal{I}^r$: Generalized Inputs
$\mathcal{I}^r = \{(\phi_\theta(O^r), \phi_\omega(\tau^r_{O^r}), \phi_\theta(D^r))\}$

Algorithm 2: `RiderMacroInputs`

4.2.3.3 Illustrative example

Let us consider the scenario of Figure 4.2 with one rider r and two drivers d_1 and d_2 having the following profiles:

- $\mathcal{P}^{d_1} = \{\{(O_1, t_1), (l_1, t_2), (D_1, t_3)\}, wt^{d_1}\}$
- $\mathcal{P}^{d_2} = \{\{(O_2, t_4), (D_2, t_5)\}, wt^{d_2}\}$
- $\mathcal{P}^r = \{(0_3, t_6), D_3, \delta^r\}$

In this example, each arrival time is generalized to its corresponding 30-minute length epoch of the time horizon starting at time 00:00 and ending at time 23:59. That is, $\varphi_{\omega=30}(h : m) = \lceil \frac{60*h+mm}{30} \rceil$. In additions, locations are generalized to the number of the grid in which they are located.

The computation of generalized inputs for each user produces the following generalized input vectors:

- $\mathcal{I}^{d_1} = \{(2, 17, 6), (2, 17, 12), (6, 18, 12)\}$
- $\mathcal{I}^{d_2} = \{(6, 17, 4)\}$
- $\mathcal{I}^r = \{(6, 18, 12)\}$

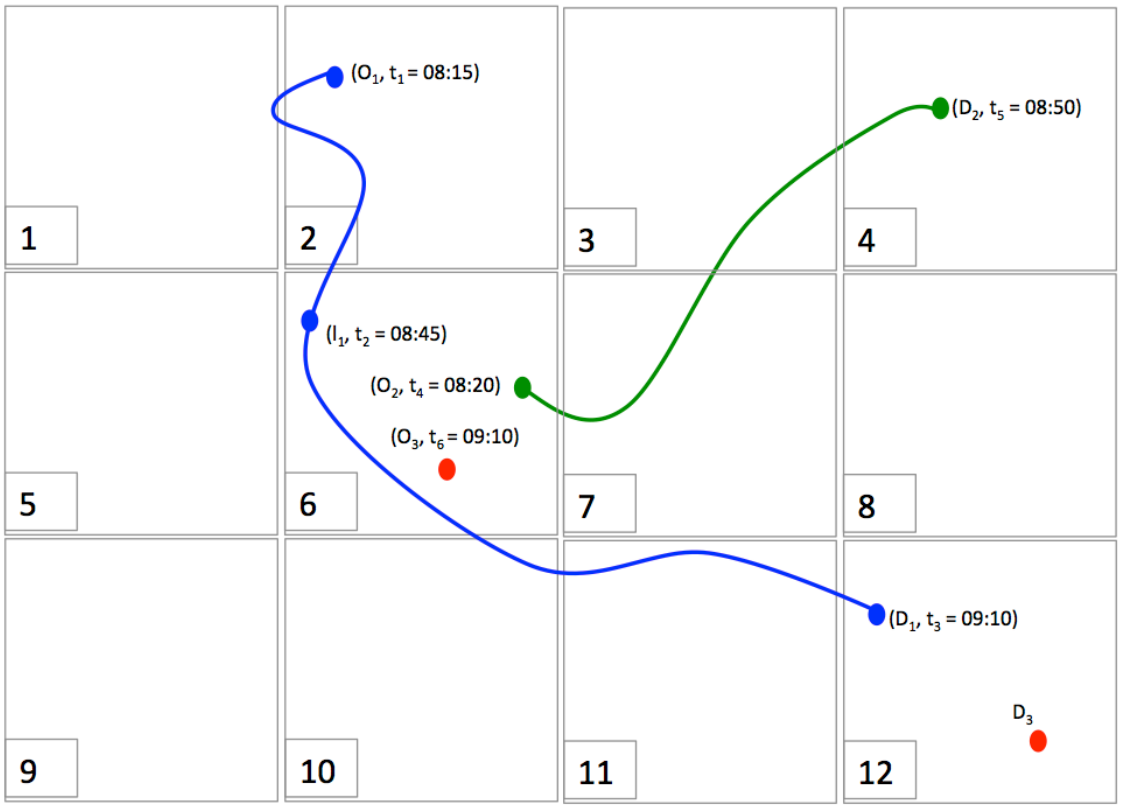

Figure 4.2: **Illustration of the generalization approach:** A scenario with one rider (red color) and two drivers (blue and green colors).

4.2.4 Secure computation of feasible matches

4.2.4.1 Characterization of feasible matches

During the secure filtering phase, the aim is to obtain pairs of riders and drivers that can travel together considering inclusive ridesharing hypothesis.

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

$\exists\ (c_s^d, e_t^d, c_{s'}^d) \in \mathcal{I}^d$ and $\exists\ (c_s^r, e_t^r, c_{s'}^r) \in \mathcal{I}^r$ such that:

1. $c_s^d = c_s^r$
2. $e_t^d = e_t^r$
3. $c_{s'}^d = c_{s'}^r$

The single triplet of the rider's generalized input $\mathcal{I}^r = \{(c_s^r, e_t^r, c_{s'}^r)\}$ corresponds to the generalization her origin, the generalization of her departure time at the origin and the generalization cell of her destination.

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).

In the example presented in 4.2.3.3, there is a feasible match between the rider r and 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.

4.2.4.2 `SF4R` — A secure filtering protocol for ridesharing

In this section, we present `SF4R`, a secure filtering protocol for ridesharing. `SF4R` is summarized in Figure 4.3. It 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, $\texttt{encode}(c_i, e_i, c_j) = [[c_i]_2 \mid [e_i]_2 \mid [c_j]_2]_{10}$.

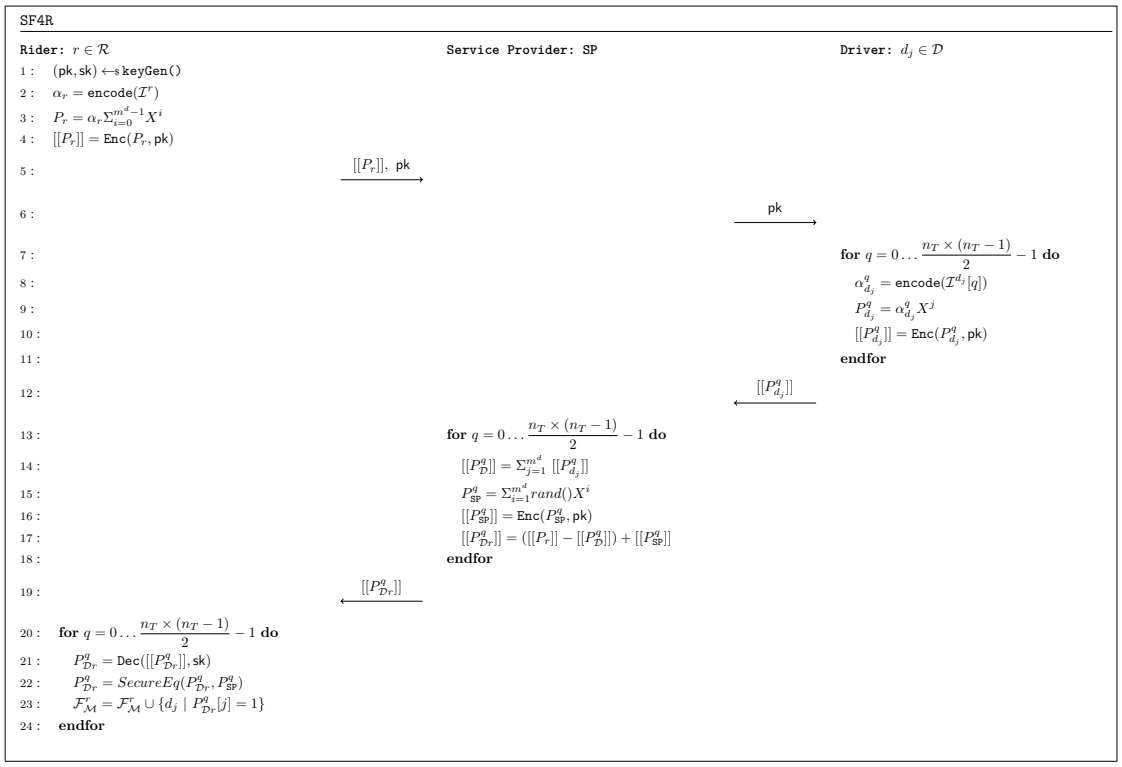

Figure 4.3: SF4R — A secure filtering protocol for ridesharing.

The details of the protocol are summarized as follows:

- The rider r generates a public/private key pair $(\mathsf{pk}, \mathsf{sk})$ of a somewhat homomorphic cryptosystem. Then, she creates a $(m^d - 1)$th degree polynomial P_r (the degree equals the number of drivers minus 1) whose coefficients are identical and correspond to the encoding of her generalized input. That is, $P_r[i] = \mathtt{encode}(\mathcal{I}^r)$, $\forall i = 0 \dots m^d - 1$. 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 jth driver creates, for her qth generalized input, a polynomial $P_{d_j}^q$ such that $P_{d_j}^q = \mathtt{encode}(\mathcal{I}^{d_j}[q])X^j$. Overall, $n_T \times (n_T - 1)/2$ of such polynomial 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 polynomial of each driver into a single encrypted $(m^d - 1)$th degree polynomial $[[P_{\mathcal{D}}^q]]$ whose jth coefficient corresponds to the qth generalized input of the jth driver. Next, it computes the feasible matches of rider r on the qth 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 $P^q_{\texttt{SP}}$ corresponding to its share, and compute the share $[[P^q_{\mathcal{D}r}]]$ of the rider r as $[[P^q_{\mathcal{D}r}]] = [[P_r]] - [[P^q_{\mathcal{D}}]] + [[P^q_{\texttt{SP}}]]$. The SP sends $[[P^q_{\mathcal{D}r}]]$ to the rider and keeps $P^q_{\texttt{SP}}$. The same operation is repeated for all the $n_T \times (n_T - 1)/2$ encrypted polynomials $[[P^q_{\mathcal{D}}]]$ of the drivers.

- The rider decrypts each of her share $P^q_{\mathcal{D}r}$ and engages in a secure equality test with the SP to learn her feasible matches. Notice that, whenever $P^q_{\mathcal{D}r}[j] = P^q_{\texttt{SP}}[j]$, $P_r[j] = P^q_{\mathcal{D}}[j]$, and there is a match between the rider r and the jth driver d_j on the qth 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, SF4R is composed of two major phases. In the first phase, hereafter referred to as the *secret-sharing of feasible matches*, the rider and the SP use an homomorphic arithmetic secret sharing protocol to secret-share feasible matches of the rider. In the second phase, hereafter referred to as the *secure computation of feasible matches*, the rider and the SP use a secure equality test and their private shares to compute the feasible matches of the rider.

4.2.4.3 Illustrative example

Let us consider the same scenario of Figure 4.2 with one rider r and two drivers d_1 and d_2. We summarize in Table 4.1, the different steps of the secure filtering protocol that will engage the service provider (SP), the rider (r_1), and the drivers (d_1, d_2), and will lead to the computation of feasible matches of r_1.

In Step 0, the rider and each driver encode their generalized input vectors. In this example, to encode generalized inputs, we convert each of the three components c_i, e_i, and c_j into a 5-bit length binary numbers, concatenate them into a single binary number, which is then reconverted into a decimal number. Then each party encodes the generalized inputs thus computed in a polynomial. More precisely, the rider r produces the 1st degree polynomial (there are 2 drivers) $P_r = 6732X^0 + 6732X^1$, and each driver produces 3 1st degree polynomials (there are at most 3 locations on the trajectories of Figure 4.2)

In Step 1, the rider sends her public key and her encrypted polynomial to the SP and in Step 2 it receives the encrypted polynomial of the drivers.

Finally, in the Step 3 (composed of 3 rounds, one round per each existing spatiotemporal triplet in drivers' trajectories), the rider and the SP compute the secret shares of the feasible matches and the feasible matches of the rider for each of the 3 private inputs of the drivers. First the SP computes 3 1st degree polynomials (for each private inputs of the driver) combining the first encrypted polynomial of each driver (Step 3.1), the second encrypted polynomial of each driver (Step 3.2) and the third encrypted polynomial of each driver (Step 3.3). Secondly, the SP generates a random 1st -degree polynomial and combines the encrypted polynomial of the rider, the random polynomial and the encrypted polynomial

of the drivers to generate a new 1st degree polynomial: $[[P^1_{\mathcal{D}r}]]$ (`Step 3.1`), $[[P^2_{\mathcal{D}r}]]$ (`Step 3.2`), and $[[P^3_{\mathcal{D}r}]]$ (`Step 3.3`). These polynomials are sent to the rider who can decrypt them, engage a secure equality test with the `SP`, and find her feasible matches.

	Rider: r	Service provider: SP	Driver: d_1	Driver: d_2
Step 0	$P_r = 6732X^0 + 6732X^1$		$P^1_{d_1} = 2598X^0 + 0X^1$ $P^2_{d_1} = 2604X^0 + 0X^1$ $P^3_{d_1} = 6732X^0 + 0X^1$	$P^1_{d_2} = 0X^0 + 6692X^1$ $P^2_{d_2} = 0X^0 + 0X^1$ $P^3_{d_2} = 0X^0 + 0X^1$
Step 1		pk $[[P_r]] = [[6732X^0 + 6732X^1]]$	pk	pk
Step 2		$[[P^1_{d_1}]] = [[2598X^0 + 0X^1]]$ $[[P^2_{d_1}]] = [[2604X^0 + 0X^1]]]$ $[[P^3_{d_1}]] = [[6732X^0 + 0X^1]]$ $[[P^1_{d_2}]] = [[0X^0 + 6692X^1]]$ $[[P^2_{d_2}]] = [[0X^0 + 0X^1]]$ $[[P^3_{d_2}]] = [[0X^0 + 0X^1]]$		
Step 3.1	$P^1_{\mathcal{D}r} = 4680X^0 + 255X^1$ $P^1_{\mathcal{D}r} = SecureEQ(P^1_{\mathcal{D}r}, P^1_{\mathtt{SP}}) = (0,0)$ $\mathcal{F}^r_{\mathcal{M}} = \emptyset$	$[[P^1_{\mathcal{D}}]] = [[P^1_{d_1}]] + [[P^1_{d_2}]] = [[2598X^0 + 6692X^1]]$ $P^1_{\mathtt{SP}} = 546X^0 + 215X^1$ $[[P^1_{\mathcal{D}r}]] = [[P_r]] - [[P^1_{\mathcal{D}}]] + [[P^1_{\mathtt{SP}}]] = [[4680X^0 + 255X^1]]$		
Step 3.2	$P^2_{\mathcal{D}r} = 7484X^0 + 8879X^1$ $P^2_{\mathcal{D}r} = SecureEQ(P^2_{\mathcal{D}r}, P^2_{\mathtt{SP}}) = (0,0)$ $\mathcal{F}^r_{\mathcal{M}} = \emptyset$	$[[P^2_{\mathcal{D}}]] = [[P^2_{d_1}]] + [[P^2_{d_2}]] = [[2604X^0 + 0X^1]]$ $P^2_{\mathtt{SP}} = 3356X^0 + 2147X^1$ $[[P^2_{\mathcal{D}r}]] = [[P_r]] - [[P^2_{\mathcal{D}}]] + [[P^2_{\mathtt{SP}}]] = [[7484X^0 + 8879X^1]]$		
Step 3.3	$P^3_{\mathcal{D}r} = 1239X^0 + 8838X^1$ $P^3_{\mathcal{D}r} = SecureEQ(P^3_{\mathcal{D}r}, P^3_{\mathtt{SP}}) = (1,0)$ $\mathcal{F}^r_{\mathcal{M}} = \{d_1\}$	$[[P^3_{\mathcal{D}}]] = [[P^3_{d_1}]] + [[P^3_{d_2}]] = [[6732X^0 + 0X^1]]$ $P^3_{\mathtt{SP}} = 1239X^0 + 2106X^1$ $[[P^3_{\mathcal{D}r}]] = [[P_r]] - [[P^3_{\mathcal{D}}]] + [[P^3_{\mathtt{SP}}]] = [[1239X^0 + 8838X^1]]$		

Table 4.1: Illustration of `SF4R` solving the ridesharing matching scenario of Figure 4.2.

4.2.5 Scoring feasible matches and matching

Once feasible matches are computed, the `PPLD4R` protocol is run by each feasible pair of driver d and rider r to compute ideal meeting points, as well as their ridesharing costs w_{dr}. To compute optimal assignments of drivers and riders, the `SP` runs a matching algorithm on the bipartite graph $\mathcal{G}_{\mathcal{DR}}$ formed by feasible pairs of drivers and riders, and weighted with ridesharing costs w_{dr} computed using the `PPLD4R` protocol.

4.2.6 Putting it all together

SRide integrates the four previously presented modules. A summary of the complete protocol is detailed in Algorithm 3.

First, the service provider `SP` initiates the system by setting the time and space granularity and generates an empty bipartite graph $\mathcal{G}_{\mathcal{DR}}$ (lines $1-2$). Then, (lines $3-8$), each participant generates its generalized inputs. Lines 9 to 12 correspond to the secure computation of feasible matches in which each rider make one call to the `SF4R` protocol. Then, each feasible pair securely computes its ridesharing cost with the `PPLD4R` protocol (lines 13 to 16) after which both users learn the pick-up and drop-off locations. Finally, the `SP` centrally solves an assignment problem based on the bipartite graph of feasible pairs, and notifies users of their matches (lines 17 to 18).

SP: Generate parameters for generalization : ω, θ
SP: Generate $\mathcal{G}_{\mathcal{DR}} = \emptyset$
foreach *driver* $d \in \mathcal{D}$ **do**
  $\mathcal{I}^d = \texttt{DriverMacroInputs}(\mathcal{P}^d, \omega, \theta)$
end
foreach *rider* $r \in \mathcal{R}$ **do**
  $\mathcal{I}^r = \texttt{RiderMacroInputs}(\mathcal{P}^r, \omega, \theta)$
end
foreach $r \in \mathcal{R}$ **do**
  $\mathcal{F}^r_{\mathcal{M}} = \texttt{SF4R}(r, \mathcal{D})$
end
$\mathcal{F}_{\mathcal{M}} = \{\mathcal{F}^r_{\mathcal{M}}, \ \forall \ r \in \mathcal{R}\}$
foreach $(r, d) \in \mathcal{F}_{\mathcal{M}}$ **do**
  $w_{dr} = \texttt{PPLD4R}(r, d)$
  $\mathcal{G}_{\mathcal{DR}} = \mathcal{G}_{\mathcal{DR}} \cup (d, r, w_{dr})$
end
SP: solves the matching problem on $\mathcal{G}_{\mathcal{DR}}$
SP: Notifies users on their matches

Algorithm 3: Privacy Preserving Ride-Matching

4.3 Security and Privacy Analysis

The security and privacy guarantees of *SRide* are directly derived by the security a privacy guarantees of SF4R and PPLD4R protocols.

We implement the homomorphic secret-sharing scheme of PPLD4R by using the *Fan-Vercauteren* (FV) homomorphic encryption scheme [48]. 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 subroutine of PPLD4R, we rely on the Goldreich-Micali-Wigderson (GMW) protocol [62] which provides security against passive adversaries. Finally, to compute ridesharing cost in a privacy-preserving manner, we rely on the PPLD4R protocol which security and privacy against passive adversaries have been proved in Chapter 3.

Anonymity of the rider
During the secret sharing subroutine, the SP receives encrypted generalized inputs of the rider. Hence, it can not learn anything about the rider's generalized inputs because of the semantic security of the *FV* scheme. Similarly, the drivers can not 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 [62]. Finally, during the PPLD4R protocol, the driver can not learn the rider's location data thanks to the privacy guarantee against passive adversaries of the PPLD4R protocol (cf. Chapter 3).

Anonymity of the driver
During the secret sharing subroutine, the SP receives encrypted generalized inputs of the drivers. However, as the FV scheme has semantic security, the SP can not 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 PPLD4R protocol, the rider can not learn the driver's location data thanks to the privacy guarantee against passive adversaries of the PPLD4R protocol (cf. Chapter 3).

Overall, *SRide* achieves its privacy goals **P1** and **P2** related to the rider and driver anonymity.

4.4 Experimental Evaluation

In this section, we evaluate *SRide* in terms of communication and computational cost and compare the performance of the proposed method to a brute-force approach that produces a matching using the complete bipartite graph obtained by running the Priv-2SP-SP algorithm between every pair of (driver, rider).

4.4.1 Experimental Settings

Dataset. Our experimental dataset has been generated from data collected over a 19-month period on the website *Covoiturage-libre.fr*,[1] a popular, openly available ridesharing web service operating in France (cf. Chapter 5). The collected data includes pick-up and drop-off cities and schedules. Figure 4.4 (left) represents a ridesharing network of *France*, where all shared rides have been taken into account. It has 3986 cities and 24459 weighted and directed edges (representing shared rides). In this network, cities are sized according to their weighted degree. According to our results, *Paris* is not the most important node in this network. Indeed, the network shows that *Rennes* has the highest frequency of shared rides. As a whole, *Rennes* appears in 46457 notices (roughly 23% of all shared rides), whereas *Paris* appears in 38243 notices. *Rennes* is pick-up towards 466 distinct cities against 389 for *Paris*, and drop-off from 472 distinct cities against 435 for *Paris*. The most shared trips are between *Rennes* and *Paris* (roughly 5% of all shared rides), followed by trips between *Rennes* and *Nantes*. Figure 4.4 (right) shows the distribution of rides per day. To summarize, over the 19-month period, 75% of the days have less 600 rides. In additions, on average there are 468 rides per day and the busiest day has 1309 rides.

Instances. We simulate 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 frequencies of rides observed between them in the dataset. In these experiments, we consider that all the users want to travel from a given city to the other one and that there

[1] http://covoiturage-libre.fr

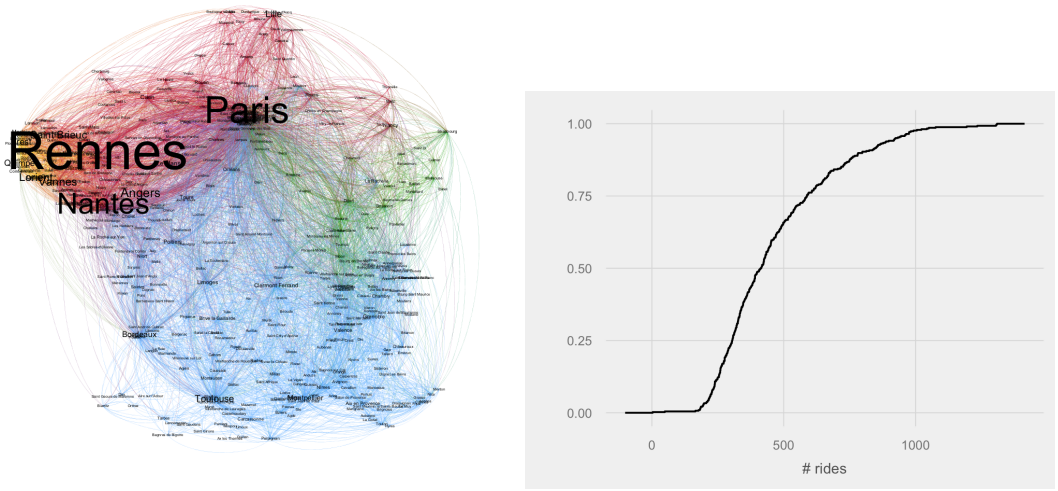

Figure 4.4: French ridesharing network (left) and Nation wide distribution of rides per day (right).

are no intermediate points between those two cities. We generate overall $m^d = 1000$ drivers' offers and $m^r = 1000$ riders' requests. Departure locations are generated in the city of *Nantes* and arrival locations in the city of *Rennes*. Each driver's 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.

Time and Space generalizations. We consider the interval $[5p.m\ 8p.m]$ within a weekday as the time horizon. This range is then divided into equal length epochs with granularity values in $\{15, 30\}$ minutes. We generalized locations to the district in which they are located. 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 (cf. Figure 4.5).

Implementation details. In the generated instances, there is no intermediate locations in drivers' trajectory. Thus, there is a single triplet for the drivers generalized input vectors: $|\mathcal{I}^d| = 1,\ \forall d \in \mathcal{D}$, and by definition $|\mathcal{I}^r| = 1,\ \forall r \in \mathcal{R}$. The `SRide` protocol is implemented in *C++*.

For the homomorphic secret sharing protocol, we use the *FV-NFLlib* library [2], which implements the *FV* scheme. As the degree of the polynomial has to be a power of 2 (cf. [3]), for the 1000 drivers we consider in our experiment, we use a 1024th degree polynomial to integrate generalized inputs of all the drivers. The coefficients of the polynomial are 64-bit length and the modulus p has 124 bits. Therefore, a public key or a ciphertext takes up 31

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

Figure 4.5: City of Nantes [84] with its 11 districts (left) and city of Rennes [99] with its 12 districts (right).

KB, while a plaintext takes up 8 KB.

For the secure computation of feasible matches protocol, we use the *ABY* framework [38], which implements both the Goldreich-Micali-Wigderson (GMW) protocol [62] and the Yao's garbled circuit protocol [129], with security against passive adversaries. As suggested in [106, 10], 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 length.

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

4.4.2 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.

4.4.2.1 Performances of the secure filtering protocol `SF4R`

In this part, we report the communication overhead and the computational overhead of both the secret-sharing and the secure computation subroutines used in `SF4R`.

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 forward 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 send 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 of SF4R
A detailed summary of the communication overhead for a ride request is given in Table 4.2. 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.

The communication complexity of the secure filtering protocol is given in Table 4.3. The only parameters impacting the bandwidth are the number of drivers m^d, the number of bits l_H used to represent private inputs in the homomorphic arithmetic secret sharing scheme, the number of bits l used to represent the private shares in the secure two-party computation of feasible matches, and the maximum number n_T of locations on drivers' trajectories.

Time generalization (min)	Rider		Driver	
	Upload (KB)	Download (KB)	Upload (KB)	Download (KB)
15	67.4	36.4	31	31
30	67.4	36.4	31	31

Table 4.2: Communication overhead of SF4R for a rider and for a driver in one ride request. SF4R engages a rider, the SP, and the 1000 drivers at the same time.

Computational overhead of the secret-sharing subroutine.
A summary of the different cryptographic operations of each party is given in Table 4.4. 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). Note that encryption and decryption operations have a small footprint (less than 5 ms). The relatively high value of the key generation runtime (33 ms)

Rider	
Upload	Download
$2^{2+\lceil log_2 m^d \rceil} * l_H + \frac{m^d n_T (n_T-1)(3l-1)}{2}$	$n_T(n_T-1)(2^{\lceil log_2 m^d \rceil} * l_H + m^d(3l-1))$

Driver	
Upload	Download
$n_T(n_T-1)(2^{\lceil log_2 m^d \rceil} * l_H)$	$2^{1+\lceil log_2 m^d \rceil} * l_H$

Table 4.3: Communication complexity for the driver and the rider in SF4R.

is due to the high-security setting. The overhead introduced by the oblivious computation of shared secret of feasible matches is due to the fact that the SP needs to add all the encrypted input of the drivers. 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 of SF4R.
A summary of all the subroutines of SF4R is given in Table 4.4. Overall, the computational overhead of the secure filtering protocol is very small (520 ms).

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 4.4: Computational overhead of SF4R. SF4R engages a rider, the SP, and the 1000 drivers at the same time. Statistics (avg ± std) for the runtime are computed over the 1000 riders.

4.4.2.2 Feasible Matches

In this section, we report the number $|\mathcal{F}^r_{\mathcal{M}}|$ of feasible matches per rider obtained after the secure filtering protocol for time generalizations of 15 and 30 minutes. Statistics in Table 4.5 present 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 have been 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.

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

Table 4.5: Number of feasible matches per rider. Statistics (avg $\pm$ std) for number of matches are computed over the 1000 ride requests, for the two values of time generalization.

4.4.2.3 Secure scoring

In this section, we discuss the bandwidth overhead, and the computational footprint of the `Priv-2SP-SP` protocol used to find optimal pick-up and drop-off locations and ridesharing costs.

We run the `PPLD4R` protocol with 64 ridesharing stations. Overall, it takes less than 200 ms and total communication cost of 249 KB for each pair of rider and driver to run the protocol.

4.4.2.4 Comparison with the naive approach

Overall, it takes about 9 seconds (respectively 5 seconds) to securely compute feasible matches and their corresponding ridesharing cost, for a time generalization of 30 (respectively 15) minutes. This include one execution of the `SF4R` protocol and 22 (respectively 43) P2P executions of the `PPLD4R` protocol for a time generalization of 30 (respectively 15) minutes. The communication overhead is about 11 MB (respectively 5 MB) for a time generalization of 30 (respectively 15) minutes. To compare, the naive approach (running a `PPLD4R` protocol for every pair of rider and driver) requires about 3 minutes and 244 MB. That is, our approach is an order of magnitude faster than the brute force approach. Concerning the bandwidth requirement, the data footprint of our protocol is an order of magnitude smaller than that of the naive approach.

4.5 Discussion

4.5.1 Malicious adversaries

SRide has been proven secure against semi-honest adversaries. However, it can be extended to provide security against malicious adversaries. Let assume a model in which both the drivers and riders are malicious adversaries and the `SP` is passive.

A malicious driver could corrupt the input of other drivers by encrypting non-zero values in coefficients of his polynomial at the indexes that are not allocated to him. To prevent such attack, the `SP` can multiply (coefficient-wise) the polynomial of each driver i by a masking

polynomial $P_{M_i} = X^i$, which preserves only the content of the allocated index.

During the secure two-party equality test between the rider and the SP, a malicious rider could try to learn the inputs of drivers with whom she does not match. It is possible to build a secure two-party computation scheme that is secure against a malicious adversary (e.g., using [88, 87]), with additional computational and communication costs.

Assuming a semi-honest SP is reasonable. In fact, since there is a non-negligible chance of the SP being caught while acting differently, the risk of public exposure and reputation loss is a strong economic deterrent against malicious behavior. Nonetheless, *SRide* can be implemented in a pure decentralized way without the SP by delegating its tasks to a small set of drivers (randomly selected).

4.5.2 Toward a fully-blown privacy preserving ridesharing system

In this chapter, we focus on the privacy-preserving matching problem in ridesharing with routing considerations. However, to be more realistic, a privacy-preserving ridesharing system need to cover secure reputation management as well as payment. Our solution can be easily composed with the privacy-preserving reputation management system proposed in [102]. More precisely, before launching PPLD4R, each rider can select, in the set of feasible matches, only drivers having a good reputation. As for payment, one can rely on e-cash [72] as suggested in [93].

4.6 Related Work

In this section, we survey related works in transportation in general and in ridesharing and ride-hailing in particular.

4.6.1 Privacy research in Transportation

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

Sun et al. [114] 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. [58] devised an advanced anonymization techniques 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* [116] but considers that the adversary has a 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. [127] proposed a privacy-preserving shortest path algorithm based on the use of a cryptographic primitive known as *Private Information Retrieval* (PIR) [30]. 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. [126] have applied a graph compression algorithm on road networks to improve PIR-based navigation services' performances. 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. [127], the `SP` 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 even when the drivers do not trust the `SP`.

4.6.2 Privacy in Ridesharing and Ride-hailing

Hallgren et al. [65] propose a privacy-preserving protocol allowing peers of drivers and riders to check their feasible matches. The authors propose two approaches, namely the *proximity-based ridesharing* and the *intersection-based ridesharing*. The former relies on an homomorphic encryption scheme to compute euclidean distances between a driver and a rider, while the later use a threshold private set intersection [50] protocol to compute the similarities between trajectories of the rider and the driver. Unlike *SRide*, the proposed approaches do not consider time constraints of both the rider and the driver. Furthermore, with the proposed approaches, to find her feasible matches, a rider has to make as many $P2P$ executions of the protocols as there are drivers in the systems, while in *SRide* the rider only makes one execution to check its similarity with all the drivers.

Sanchez et al. [102] propose a fully decentralized approach to solve the matching between riders and drivers and also a privacy-preserving distributed protocol for reputation management in ridesharing. For the matching phase, space and time generalizations are used and a *publish-subscribe* [47] subroutine allows drivers to subscribe to topics corresponding to their generalized inputs, and to receive a notification when a rider publishes on the corresponding topics. By contrast, *SRide* is partially decentralized and also relies on generalizations to compute feasible matches. It relies on the `SP` to reduce communication costs. Nevertheless, the `SP` does not learn anything about the location data of both the riders and the riders. In additions, *SRide* allows the secure computation of the meeting points (in the generalized area on which users match). The ridesharing meeting points thus computed are used to find optimal assignments of drivers and riders.

Tong et al. [118] propose a jointly differentially private approach to solve the dynamic ridesharing problem while preserving users' location privacy. They formalize the ridesharing problem as a linear program. First, drivers are grouped according to their proximities at origin and destination. Next, for each cluster of drivers, an anchor driver is computed as the cluster centroid, such that its fare (respectively its capacity) corresponds to the minimum fare (respectively capacity) within the cluster. Then, riders are assigned to clusters thanks to a jointly differentially mechanism. When the number of riders assigned to a cluster is greater

than its size, the winning riders are computed differentially privately by using the exponential mechanism [81]. This work is very interesting since it provides a formal way to quantify the privacy and the utility of the riders. However, it only protects riders' privacy. By contrast, *SRide* protects the privacy of both riders and drivers.

Pham et al. [93] analyze the privacy threats for a Ride-Hailing system and propose `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 [92], a ride-hailing system based on somewhat-homomorphic encryption, which addresses most limitations of PrivateRide and provides stronger privacy and accountability guarantees. In this two privacy-preserving implementations of Ride-Hailing systems, the temporal constraints of users are not considered. By contrast, *SRide* integrate both geographic and temporal constraints. Furthermore, 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 term of use cases and properties of rides. Ride-hailing essentially replaces taxi cabs (short trips, potentially frequent–several times a week) while ridesharing replaces trains and planes (medium long trips, typically for weekend excursions and commutes or vacations).

4.7 Summary

In this chapter, we have proposed `SRide`, a practical solution to implement matching in ridesharing systems while protecting the privacy of users *w.r.t.* 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 `PPLD4R` 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.

Part III

Data analysis and prototype applications

Chapter 5

Empirical Network Analysis of a Ridesharing Service

Towards a better understanding of social networking interactions on ridesharing services, this report analyses data collected from *Covoiturage-libre.fr* [1], a popular, openly available ridesharing web service in France which allows users to publish short-notices to share rides. The service went online in 2011, and roughly 688000 ridesharing notices have been published since then.

The publication process of a short-notice is straightforward, fast, and does not require any account creation. To publish a short-notice, each user must specify his/her *role*, which is either a *driver* or *rider*, and provide a pseudonym, gender, email, pick-up and drop-off cities, schedule time for pick-up as well as optional information like age, phone number, and message. In case the user is a driver, comfort level, price per passenger, number of available seats are required, and intermediate points could be provided. After registering a short-notice, a confirmation message is sent to the mailbox of the user to confirm the registration request. Once the recipient confirms the reception of this message, the notice is published on the main page of the website along with the last ten ridesharing notices and is indexed in the site database for future ride searches.

During 19 months, we have crawled and extracted data from short-notices of *Covoiturage-libre.fr*. Collected data have key features of ridesharing users and their trip request, including age, gender, pick-up and drop-off cities, ride price and schedule. From this data, we perform an exploratory analysis, and we construct a network where cities are nodes and shared rides are edges. The resulting empirical network analysis provides some insights on the cost, distance, and most popular cities of shared rides.

The remainder of this chapter is organized as follows:

Section 5.1 shortly describes the French ridesharing website, *Covoiturage-libre.fr*, from which we collected data. In Section 5.2, we present the exploratory analysis of the dataset. In Section 5.3, we highlight the network construction, visualization, and the main findings of empirical network analysis. We conclude in Section 5.4.

[1] http://covoiturage-libre.fr

5.1 Dataset

The dataset used for this study is built on the information extracted from short-notices on *Covoiturage-libre.fr* website. Figure 5.1 describes tools we use for the study. First, data are extracted from the website with a crawler implemented using *scrapy* [2], an open source framework for extracting data from websites. Every five minutes, it crawls the 10 most recent short-notices that appear on the main page of the website and extracts the following information: pseudonym, age, gender, schedule, available seats, ride price, pick-up and drop-off cities, car comfort, the entire message left by user, and the notice URL. At this step, geographic coordinates pick-up and drop-off cities are inferred, and the record is inserted into the *SQL* database. To infer geographic information we use *GeoPy*[3], a popular client for geocoding web services. Then, *NetworkX*[4], a high-productivity software for complex networks, is used to construct a network from the raw data. The network is saved into *graphml* format. Finally, we use *Gephi* [5], an open-source network analysis software, to visualize and analyze the network.

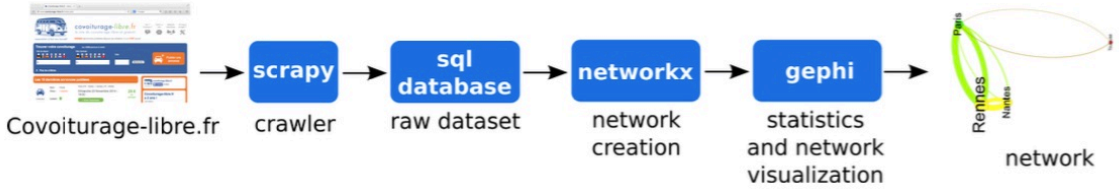

Figure 5.1: Main building blocks used to collect data from *Covoiturage-libre.fr* website, and to construct the ridesharing networks.

Roughly, 221000 distinct ridesharing short-notices were collected during a crawling period of 19 months, from November 2014 to June 2016. The vast majority of ride announcements are proposed by drivers, accounting for almost 215000 notices. Overall, notices were created by 44664 users, including 27861, men, 16803 women. Most of the drivers sharing a ride are men, roughly 69%, whereas the majority of passengers publishing a request to find a ride are women, about 57%. Among the 80% of users who filled their ages, the average age is 35. Passengers looking for a ride are slightly older than ridesharing drivers on average, 40 and 34 years old respectively. Among ridesharing drivers, female drivers are younger than male drivers on average, 31 and 36 years old respectively.

In the rest of this study, we only consider 14 months out of the 19 months. In fact, the study begins in the middle of November 2014 and ends at the beginning of June 2016. Furthermore, the server used for the crawling went down a few times during the data collection campaign.

[2] http://scrapy.org/
[3] https://github.com/geopy/geopy
[4] https://networkx.github.io
[5] http://gephi.github.io/

5.2 Exploratory analysis

Rides distribution nation wide. Overall, 198646 rides offers have been proposed within the period considered. Figure 5.2 presents the distributions of ridesharing per month in France. There are 14189 ± 2892 rides in average per month.

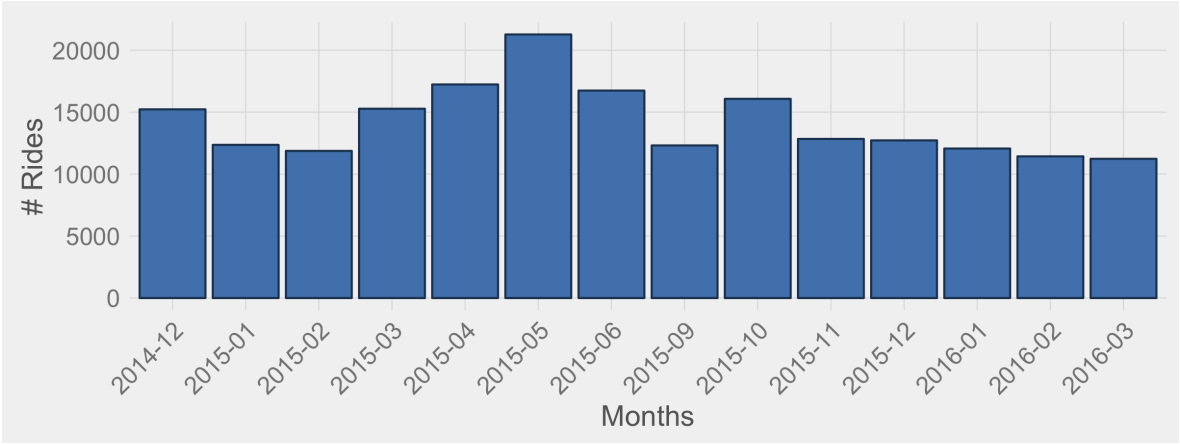

Figure 5.2: Nation wide distribution of rides per month.

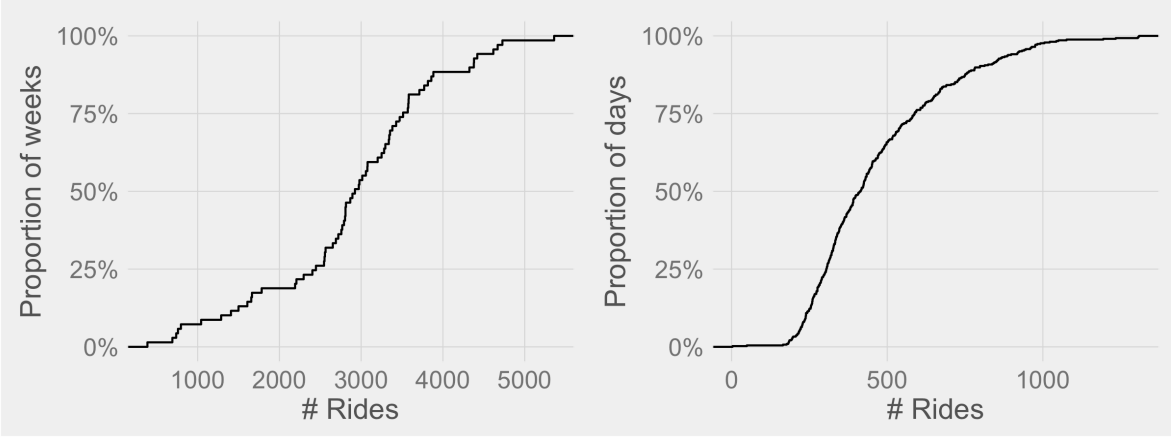

Figure 5.3: Nation wide distribution of rides per week (left) and per day (right).

Figure 5.3 depicts the nationwide distribution of rides per week and per day. On average 2878 ± 1058.264 rides occur each week and the busiest week has a total of 5360 rides against 386 rides for the less busiest week. The average number of rides per day is 468 ± 225 with a pick at 1309 against 3 for the less busiest day. Furthermore, half of the 69 weeks considered record less than 2924 rides and half of the 424 days considered record less than 468 rides.

Ride distances. Figure 5.4 shows the empirical cumulative distribution functions of ride's distances and fares over the entire dataset. It highlights that for 75% of drop-off cities, the distance to pick-up cities is less than 264 kilometers. This suggests that the majority of the

rides are likely to last less than 2 hours, transporting people across nearby cities.

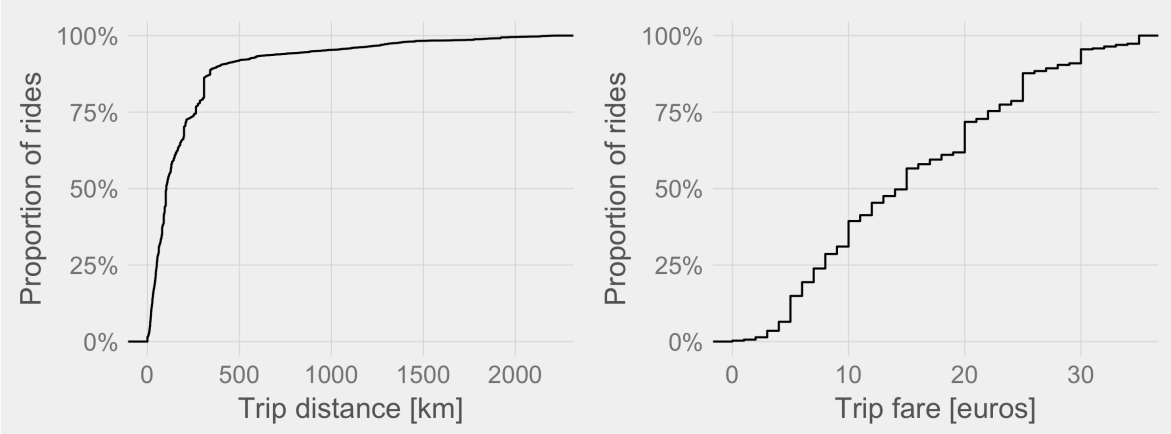

Figure 5.4: Nation wide distribution of rides distances (left) and fares (right).

Ride fares. The average fare for a ride is 15 ± 9 euros. The vast majority of rides cost less than 22 euros, which is relatively cheap. Explanations for low prices for short distance may stem from (i) the shorter the ride distance the higher the probability of the driver shares his/her car, and (ii) that is more likely that a driver avoid taking roads where the toll must be paid, so that reducing the price that becomes more attractive for passengers compared to other transportation modes like taxi, train or airplane.

Ride schedule Figure 5.5 depicts the frequency of rides according to the days of the week. As expected, drivers commonly share rides from Friday to Sunday, accounting for 75% of all shared rides. This suggests that people use ridesharing for weekends. To shed some light on the scheduling of rides, Figure 5.6 shows the frequency of rides by hours of the day. It highlights that almost 64% of rides are shared in the afternoon.

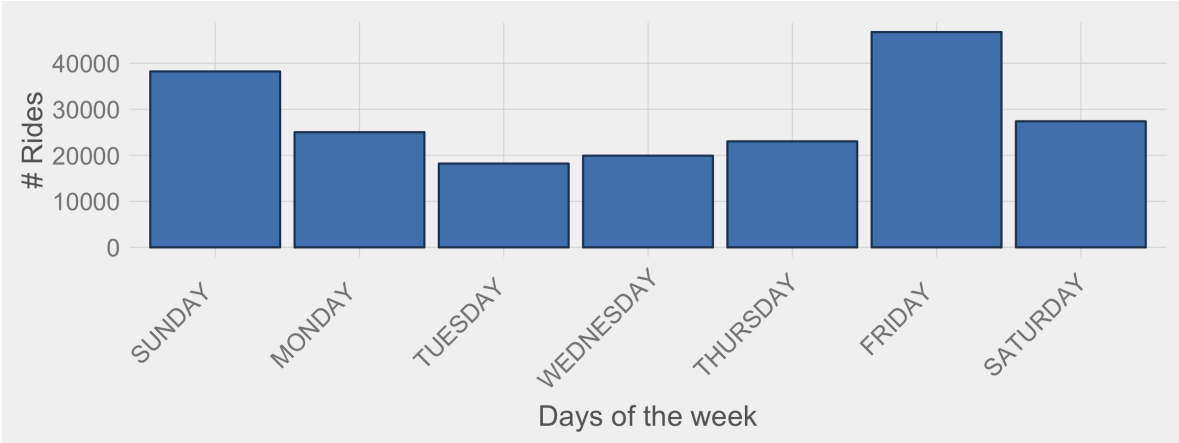

Figure 5.5: Drivers commonly share rides for weekends.

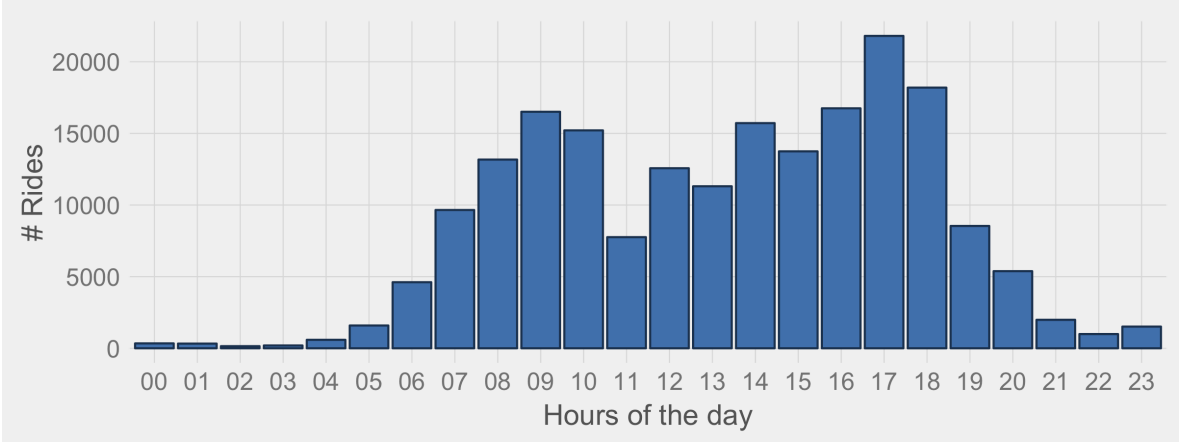

Figure 5.6: It is more common to share a ride in the afternoon.

5.3 Network analysis

This Section describes how our French ridesharing network is constructed and visualized. As depicted in Figure 5.1, network construction, and visualization are based on raw datasets stored in the database. The main insights of the network data analysis are highlighted and discussed in turn.

5.3.1 Network construction

A French ridesharing network is represented as a directed network where nodes are cities and edges are shared rides. The directed network is created from raw datasets using *NetworkX*. In this network, nodes, edges, and edge weights are defined as follows.

Nodes: It represents a city and comprises four attributes, a numerical identifier, a label representing the name of the city, the latitude and the longitude values.

Edges: They are derived from rides between pairs of cities. Concerning the frequency of rides between pairs of cities, there are two variants of edge definition:

1. All shared rides: There exists a directed edge from city a to b whenever a driver shares at least one ride where a is the pick-up city and b the drop-off city.

2. More frequent rides: An edge from city a and to city b is created only if drivers share at least n rides from a to b, where $n > 1$.

Edge weights: Edges were weighted according to the normalized frequency of shared rides between the pair of cities, hence the higher the frequency, the heavier the weight of

a directed edge. Finally, the network could be exported to the *GraphML*[6] file format using *NetworkX* to be used as input for the visualization tool.

5.3.2 Network data visualization

The French ridesharing network analyzed in this report is visualized and analyzed using *Gephi*. Once the *GraphML* file containing network is loaded in *Gephi*, the following steps are performed:

Computing statistical metrics: A bunch of statistical metrics is computed to help in understanding the structure of the network. These metrics for the network overview are: *average degree*, *average weighted degree*, *network diameter*, average clustering coefficient, average path length, *modularity* (based on a community detection algorithm), and *connected components*. A summary of these metrics is provided in Section 5.3.3.

Fitting to the geographic layout: We use *GeoLayout*, a *Gephi* plug-in, to display the network of cities according to their geographic coordinates. This plug-in seems to provide an intuitive way to display nodes and helped us in quickly figuring out interesting insights from data, such as assessing the relationship between the frequency of rides and the geographic density of France.

Adjusting nodes and edges visualization: Color and sizes of nodes and edges are adjusted according to the weighted degree for nodes and weights for edges.

Identifying communities of closer cities: We color cities according to communities or groups identified by the modularity metric available in *Gephi*, based on the algorithm proposed by [23]. Therefore, cities that seem to be closer to each other in the network to each other have the same color. This allows us to have some insights on the transportation pattern of French ridesharing users, highlighting rides within a group of cities that are more frequently shared.

5.3.3 Empirical analysis of the ridesharing network

This section provides the empirical analysis of the network that was constructed from the collected data. According to the way edges were created, ridesharing networks can be divided into two following groups: networks with all rides and network with more frequent rides.

Ridesharing network with all rides. Figure 5.7 represents a ridesharing network where all shared rides were taken into account (see Section 5.3.1 for details about edge definition). It has 3986 cities and 24459 weighted, directed edges (representing shared rides), with a low average path length (3.49) and a relatively high clustering coefficient (scoring 0.33). In the

[6] http://graphml.graphdrawing.org

network, cities are sized according to the weighted degree, and the length of edges are drawn according to their weights.

Surprisingly, *Paris* is not the most important node in this network. Indeed, the network shows that *Rennes* has the highest frequency of shared rides. As a whole, *Rennes* appears in 46457 notices (roughly 23% of all shared rides), whereas *Paris* appears in 38243 notices. *Rennes* is pick-up towards 466 distinct cities against 389 for *Paris*, and drop-off from 472 distinct cities against 435 for *Paris*. The most shared trips are between *Rennes* and *Paris* (roughly 5% of all shared rides), followed by trips between *Rennes* and *Nantes*.

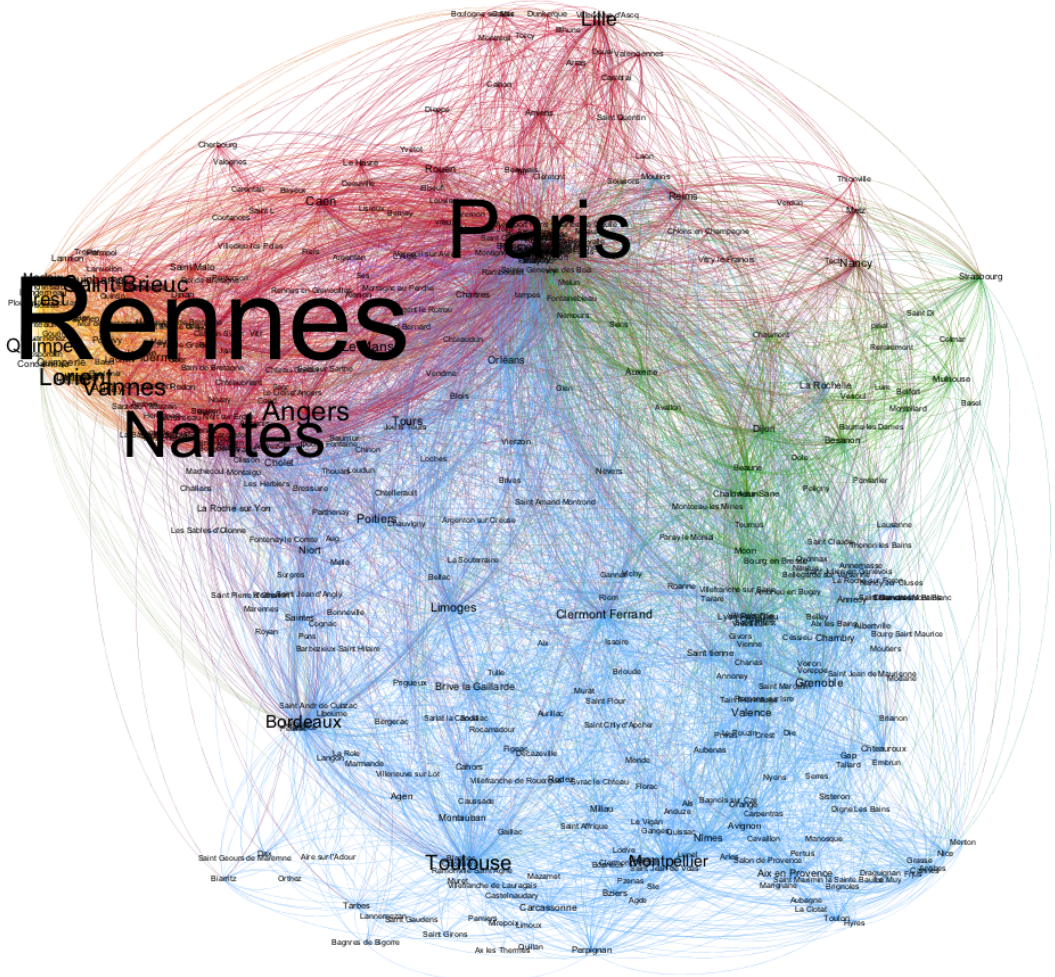

Figure 5.7: French ridesharing network: Rennes has more frequent shared rides than Paris.

Figure 5.7 also highlights with different colors the communities of cities that are closer to each other in the network (within rides are most frequently shared). The biggest community

City	Weighted degree	Degree	Community color
Rennes	7.9235	938	Red
Paris	5.7128	847	Red
Nantes	4.6036	737	Red
Angers	1.7697	379	Red
Vannes	1.5924	324	Red
Lorient	1.4655	217	Yellow
Saint Brieuc	1.4137	277	Red
Toulouse	1.2900	612	Cyan
Quimper	1.1514	204	Yellow
Lille	1.1329	236	Red
Brest	1.1123	221	Yellow
Bordeaux	0.9815	481	Cyan
Montpellier	0.8013	445	Cyan
Le Mans	0.5629	325	Red
Caen	0.5197	327	Red
Guingamp	0.4824	142	Cyan
Clermont Ferrand	0.4751	470	Cyan
Limoges	0.4549	338	Cyan
Morlaix	0.4491	150	Yellow
Poitiers	0.4392	331	Cyan
Grenoble	0.4334	424	Cyan
Auray	0.4330	103	Yellow
Ploërmel	0.4186	124	Red
Tours	0.4115	327	Cyan
Nancy	0.3695	197	Red
Lanester	0.3381	98	Yellow
Valence	0.3374	311	Cyan
Lamballe	0.2966	106	Red
Orléans	0.2831	320	Cyan
Quimperlé	0.2756	81	Yellow

Table 5.1: Top 30 cities ranked by the weighted degree.

(the cyan-colored one) gathers many cities from the South of France, accounting for 52.95% of all cities. The second largest community (the red-colored one), with 28.64%, includes most cities from the North of France. Since most of the shared rides have short distances in kilometers, it makes sense that the majority of cities of the same group are geographically close to each other. For example, *Rennes* and *Vannes*, which belong to the same community, are located 121 kilometers far from each other. Table 5.1 summarizes some statistics for the thirty cities with higher weighted degree scores.

To study cities according to their centrality in the network, we rank them concerning the measure of their betweenness centrality. This well-known centrality measure allows us to identify nodes that behave as hubs in the network. As we expected, the reshaped network shows that Paris is the most important hub in this network. While *Rennes* scores 1367448, *Paris* scores 1855894 in betweenness centrality. This shows that rides from involving a larger number of cities are connected to Paris, even if individually their frequencies are lower than those involving *Rennes*. We summarized some statistics metrics of top ten cities ordered by the betweenness centrality in the Table 5.2.

City	Betweenness centrality	Closeness	Community color
Paris	1855894	0.4697	Red
Rennes	1367448	0.4408	Red
Toulouse	1278559	0.4429	Cyan
Nantes	1004196	0.4451	Red
Clermont Ferrand	955129	0.4526	Cyan
Bordeaux	913353	0.4393	Cyan
Montpellier	742930	0.4196	Cyan
Grenoble	678636	0.4127	Cyan
Limoges	503129	0.4286	Cyan
Angers	439762	0.4257	Red

Table 5.2: Top 10 cities ranked by betweenness centrality.

Ridesharing network with most shared rides. We construct a second network with the most shared rides. The new network has two main advantages, (i) it allows us to shed some light on the characteristics of frequent rides and (ii) it makes the ridesharing network easier to visualize. Therefore, we represent pairs of cities only if they shared at least 1 rides per month on average. According to the edge definition in Section 5.3.1, we construct a edge between cities that shared at least fourteen rides, *i.e.*, $n \geq 14$. Figure 5.8 depicts the reconstructed ridesharing network where nodes are ranked according to the weighted degree.

Overall, the new directed network has 9074 edges and 2011 nodes. We observe that *Rennes* remains the most important node of the network with the highest frequency of shared rides, followed by *Paris* and *Nantes*. Table 5.3 highlights some information of this network.

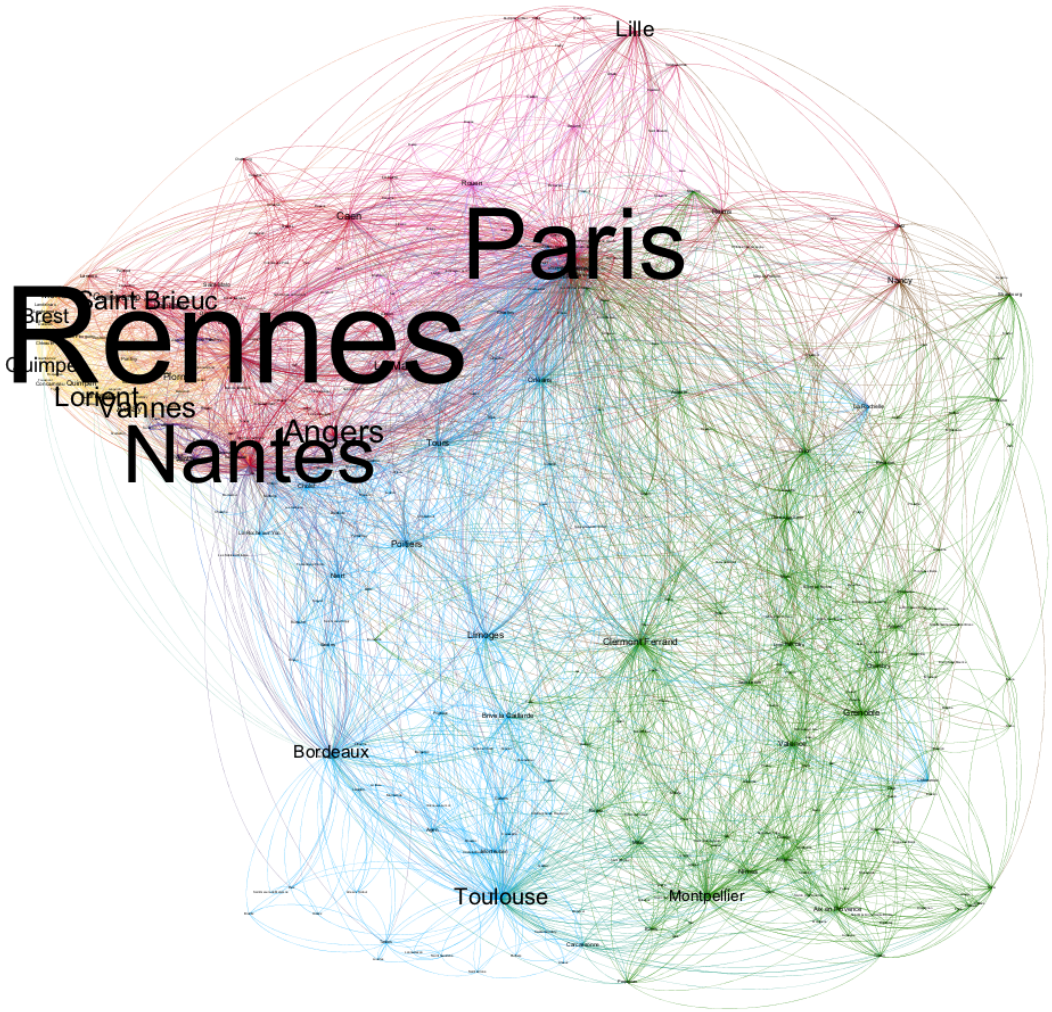

Figure 5.8: French ridesharing network with most shared rides.

City	Weighted degree	Degree	Community color
Rennes	7.58	194	Red
Paris	5.43	133	Red
Nantes	4.34	116	Red
Angers	1.62	39	Red
Vannes	1.48	52	Yellow
Lorient	1.39	43	Yellow
Saint Brieuc	1.32	47	Red
Toulouse	1.08	74	Cyan
Quimper	1.07	42	Yellow
Lille	1.04	20	Red

Table 5.3: Top 10 cities ranked by the weighted degree.

5.4 Summary

The ridesharing network from data of *Covoiturage-libre.fr* that we describe and analyze in this chapter showed some interesting insights about ridesharing in France. We observe that drivers share their cars for rides within nearby cities, whose distances are more likely to be shorter than 264 kilometers and prices are lower than 22 euros. We notice that rides are more frequently shared for weekends, Friday and Sunday being the days with most shared offers. Our empirical network analysis provides some insights into the ridesharing service in France. Although *Paris* is the best hub in the overall network, *Rennes* is the city with the highest frequency of shared rides. Our findings suggest that rides among cities East France are more frequent, consisting a relatively small subset of geographically close cities. These results also suggest that drivers from a subset of East France cities (geographically close) use the *Covoiturage-libre.fr* highly frequently.

Chapter 6

PlayMob — A platform for mobility problem

During this thesis, we devoted considerable time and effort to designing and developing a platform named `PlayMob` (Privacy Layer for Mobility). We implement all our privacy protocols for ridesharing in this platform. Furthermore, we implement a web interface to visualize transportation networks and to simulate the execution of ridesharing algorithms. In addition, `PlayMob` integrates other transportation algorithms developed in our laboratory *LAAS-CNRS*.

In this chapter, we present the main features of `PlayMob`. The remainder of this chapter is organized as follows:

Section 6.1 gives a general description of `PlayMob`. Section 6.2 present the `IHM` modules, a web platform which allows the visualization of algorithms developed during this thesis. Finally, Section 6.3 shows an example of integration of other mobility related applications into `PlayMob`.

6.1 General description

The platform `PlayMob` integrates implementations of all privacy-preserving ridesharing protocols and multimodal routing algorithms proposed in this thesis. For instance it can be used to compare `2SP-SP` and `Priv-2SP-SP`. It also include tools to build multimodal transportation networks. `PlayMob` is written in `C++` and `Python` and is available online[1]. It has been presented in several events including **Fête de la Science 2016 - LAAS-CNRS**[2] and **CPS 2017**[3].

At a high level, `PlayMob` is composed of a `Core` module and three other modules, namely the `NetworkBuilder`, the `Ridesharing`, and the `IHM` modules.

The `Core` module contains graph data structures used to model transportation network. It also contains implementations of the DRegLC algorithm [15] used for multimodal routing and several building blocks including *goal directed search*, *cost pruning*, *Landmark* ...

[1] **https://redmine.laas.fr/projects/playmob**.
[2] **https://goo.gl/nNchoq**.
[3] **https://cps2017.sciencesconf.org/**.

The `NetworkBuilder` module is used to build transportation networks. An example of a transportation network is shown in Figure 6.1. A transportation network is composed of two major sub-networks, namely the road network and the public transportation network. A road network is composed of all roads accessible by car (car network), by foot (foot network), and by bicycle (bicycle network). A public transportation network is a network formed by public transit services (Bus, Subways, Tramways, Trains, Ferries, Gondolas ...). To create a transportation network, the `NetworkBuilder` module builds separately a layer for the road network by using OpenStreetMap (`OSM`)[4] data and a layer for the public transportation by using General Transit Feed Specification (`GTFS`)[5] data. Building a road network from a `OSM` dataset is straightforward. To each **node** in the `OSM` dataset, we create a **vertex** in the road network. Likewise, to each **way** in the `OSM` dataset, the corresponding **edges** are created in the road network. Each edge is weighted according to the distance between its two vertices and labeled (*Foot*, *Car*, or *Bike*) according to the metadata of the way. On the other hand, building a public transportation network is a little bit more complex. First, we extract information related to the transit stations (stations, routes, trips). Then the transportation network is built by using the *time-dependent* approach described in Section 2.2.3. Once both layers are built, the `NetworkBuilder` merges them into a single transportation network by linking each public transit stations to its closest vertex in the road network with the so-called *transfer edge*. Each transfer edge weights 1 minute, which models the time required to move from the public transport network to the road network (e.g., moving from a subway station to the land).

The `Ridesharing` module contains implementations of `2SP-SP`, `2SP-SP-V2`, `Priv-2SP-SP`, `PPLD4R` (cf. Chapter 3) and *SRide* (cf. Chapter 4). For the privacy-preserving applications `Priv-2SP-SP`, `PPLD4R` and *SRide*, in addition to routing algorithms of the `Core` modules, we rely on some external libraries to implement our cryptographic and secure multiparty computation subroutine. More precisely, we use *NFLlib* [3] and *FV-NFLlib*[6] libraries to implement the private set intersection protocol in `Priv-2SP-SP`, the homomorphic additive secret sharing schemes in `PPLD4R` and *SRide*. *FV-NFLlib* is an implementation of the *Fan-Vercauteren* (FV) homomorphic encryption scheme [48] built on top of *NFLlib*. Finally, we use the *ABY* framework [38] to implement secure two-party computation subroutine in `PPLD4R` and *SRide*.

The `IHM` module is a web application designed with the `Django`[7] framework. It uses the software `SWIG`[8] to connect our `C++` programs to the web application, which is written in `Python`. It converts the outputs of transportation problems into the `GeoJSON`[9] format and shows them on a interactive maps by using the `Leaflet`[10] library.

[4] **http://www.openstreetmap.org/**.
[5] **https://developers.google.com/transit/gtfs/**.
[6] **https://github.com/CryptoExperts/FV-NFLlib**.
[7] **https://www.djangoproject.com/**.
[8] **http://swig.org/**.
[9] **http://geojson.org/**.
[10] **http://leafletjs.com/**.

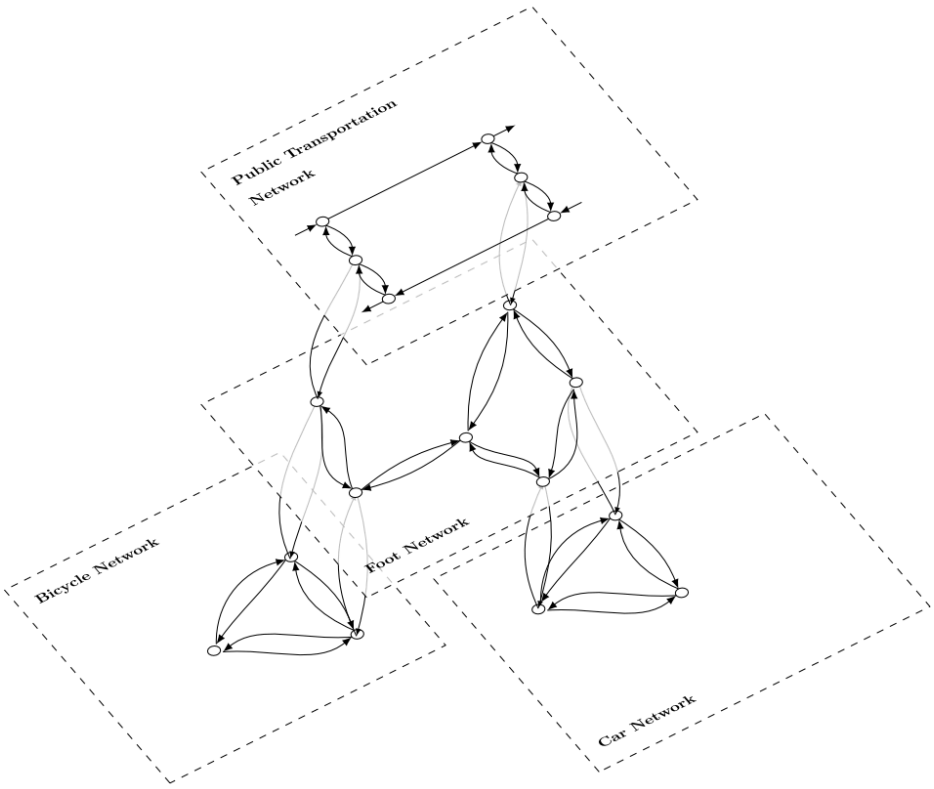

Figure 6.1: Example of transportation network.

6.2 Presentation of the IHM module

In this section, we present the most important features of the IHM module, namely the graph analytic application, the multimodal routing application, the accessibility analysis application, and the ridesharing application.

6.2.1 Graph analytic

The graph analytic application allows the user to visualize the properties of a transportation network. More precisely, for each transportation network, it gives its order, its size, the number of edges for each transportation mode (e.g., Car, Foot, Bus, Subway . . .). Figure 6.2 shows a screenshot of the graph analytic application.

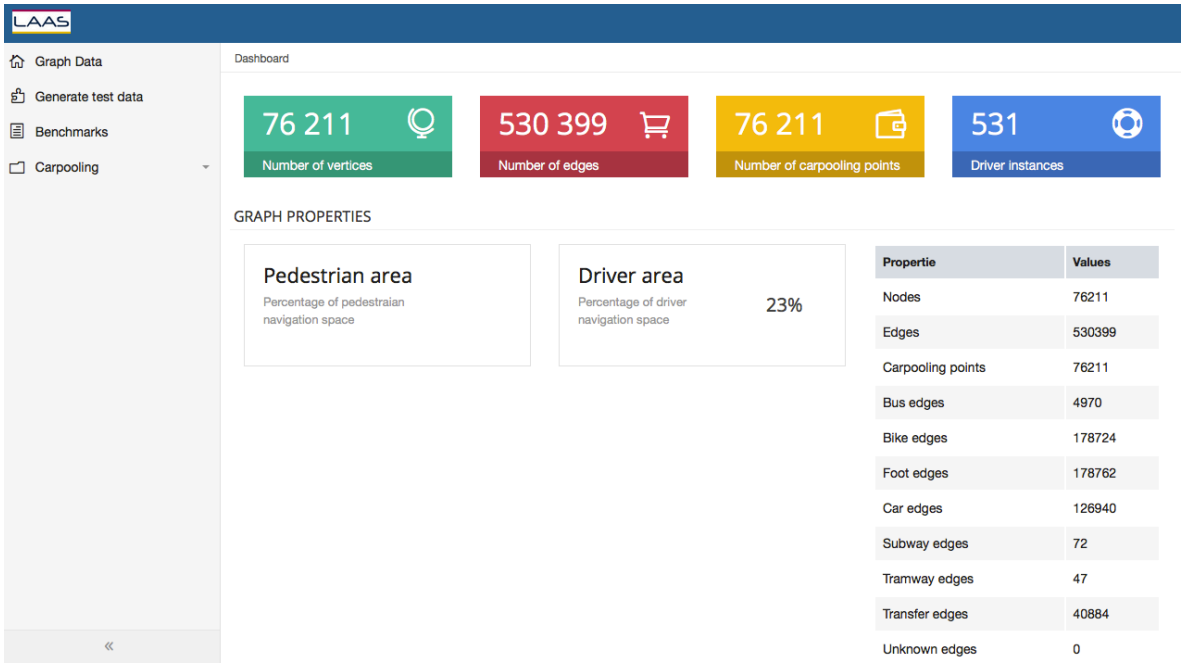

Figure 6.2: Screenshot of the graph analytic application.

6.2.2 Multimodal routing

The multimodal routing application allows a user to compute his shortest path given an origin location, a destination location and mobility constraints including departure time, transport mode. . . Figure 6.3 show an example of shortest path query. The output is the shortest path from the origin location to the destination location, with a different color per transportation mode.

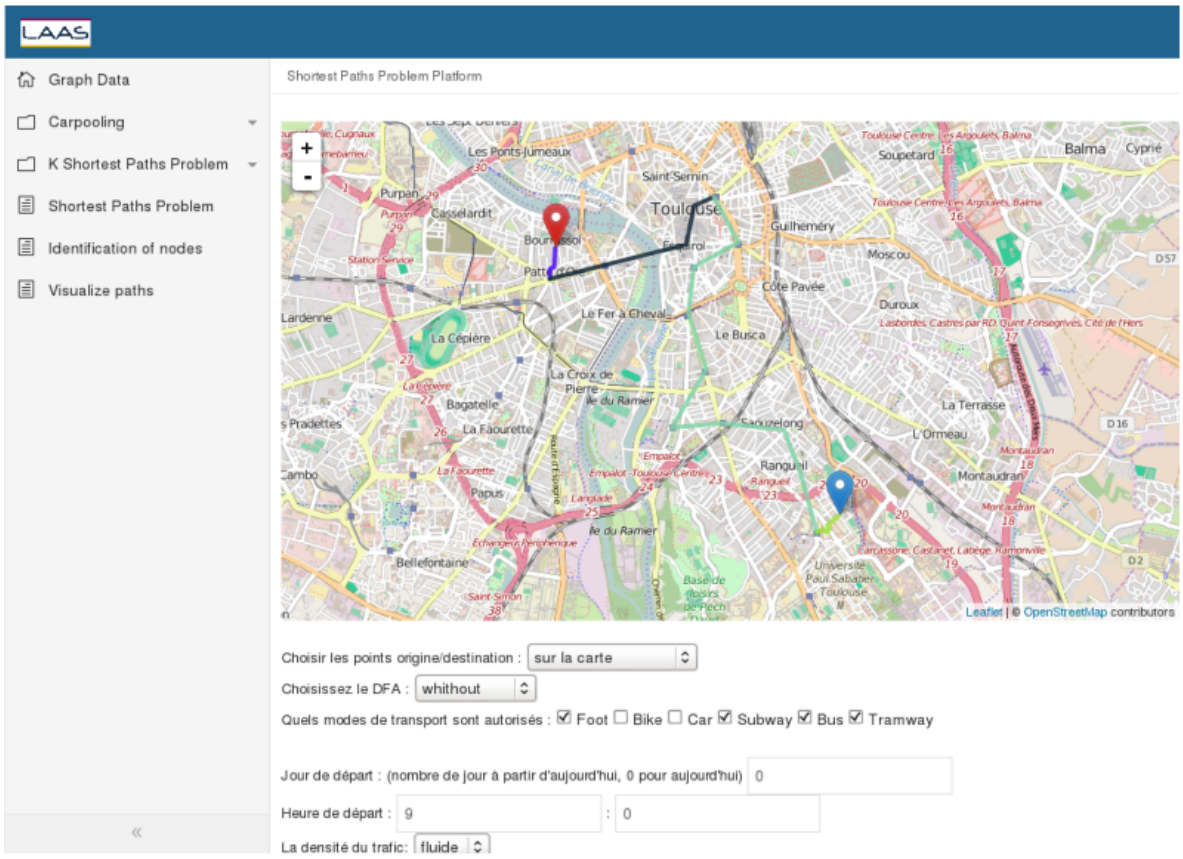

Figure 6.3: Screenshot of the multimodal routing application.

6.2.3 Accessibility analysis

The `IHM` module can also be used for accessibility analysis. The accessibility analysis application allows the user to obtain travel-time isochrone. A travel time isochrone is the shortest path tree rooted at an origin location, which shows the time to reach any location in the transportation network while the origin location as the starting point. For instance, it can be used to determine all the points of interest within a given commute time of the current location of a user. It can also help measuring network changes effects on travel time. Figures 6.4 and 6.5 are examples of travel time isochrones with our laboratory as origin location and departure time of `9:00am`. Figure 6.4 shows travel time isochrones while using walking and public transportation, while Figure 6.5 shows travel time isochrones for walking mode only. In both cases, each color band represents locations accessible within the same range of time. For instance, the red band represents locations accessible within 10 minutes, while the green band contains location that are between 30 and 40 minutes away.

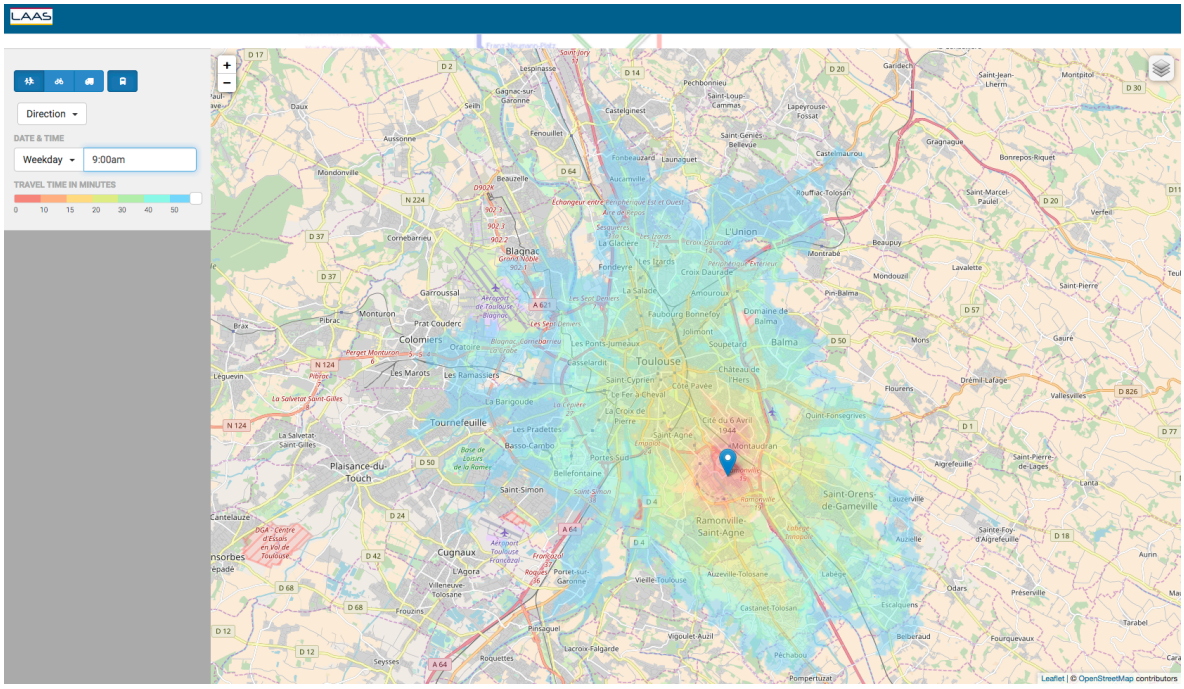

Figure 6.4: Example of travel time isochrone for walking and public transportation modes.

6.2.4 Ridesharing

The ridesharing application allows a user to solve ridesharing instances. The user inputs origin and destination locations of both the rider and the driver. The output of each query is the pick-up and the drop-off locations as well as the shortest paths that link the meeting points to the origin and destinations locations of both users. In additions, the application shows the corresponding travel time of each path as well as the ridesharing cost. Figures 6.6 and 6.7 show a scenario in which we compare `2SP-SP` and `Priv-2SP-SP` using the ridesharing application.

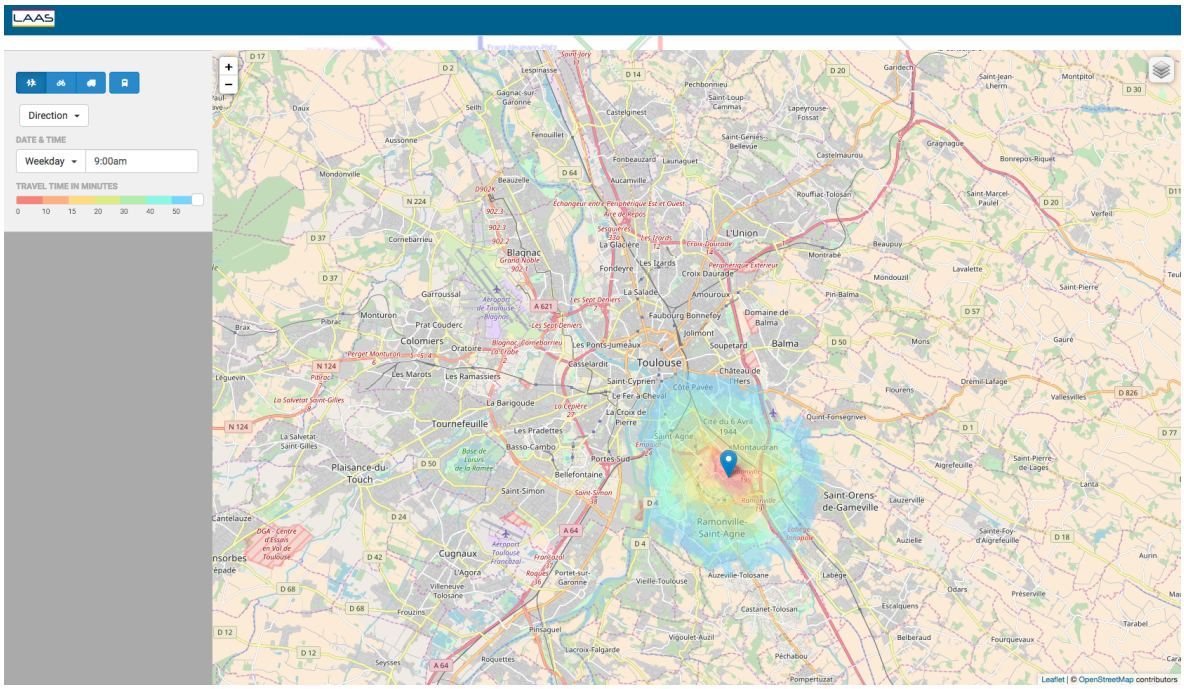

Figure 6.5: Example of travel time isochrone for walking mode

Figure 6.6 shows the inputs of our scenarios. Both scenarios have the same origin for the driver and the same destination for the rider. Origin locations are in green, and destination locations are in red. For the `Priv-2SP-SP`'s scenario, some additional parameter (isochrone radius and ring radius) are required. Figure 6.7 shows the outputs of both scenarios. The pick-up and the drop-off locations are yellow markers. In these scenarios, the ridesharing costs are the same. They have the same drop-off locations (as shown by the details related to travel time) but they differ in the pick-up locations.

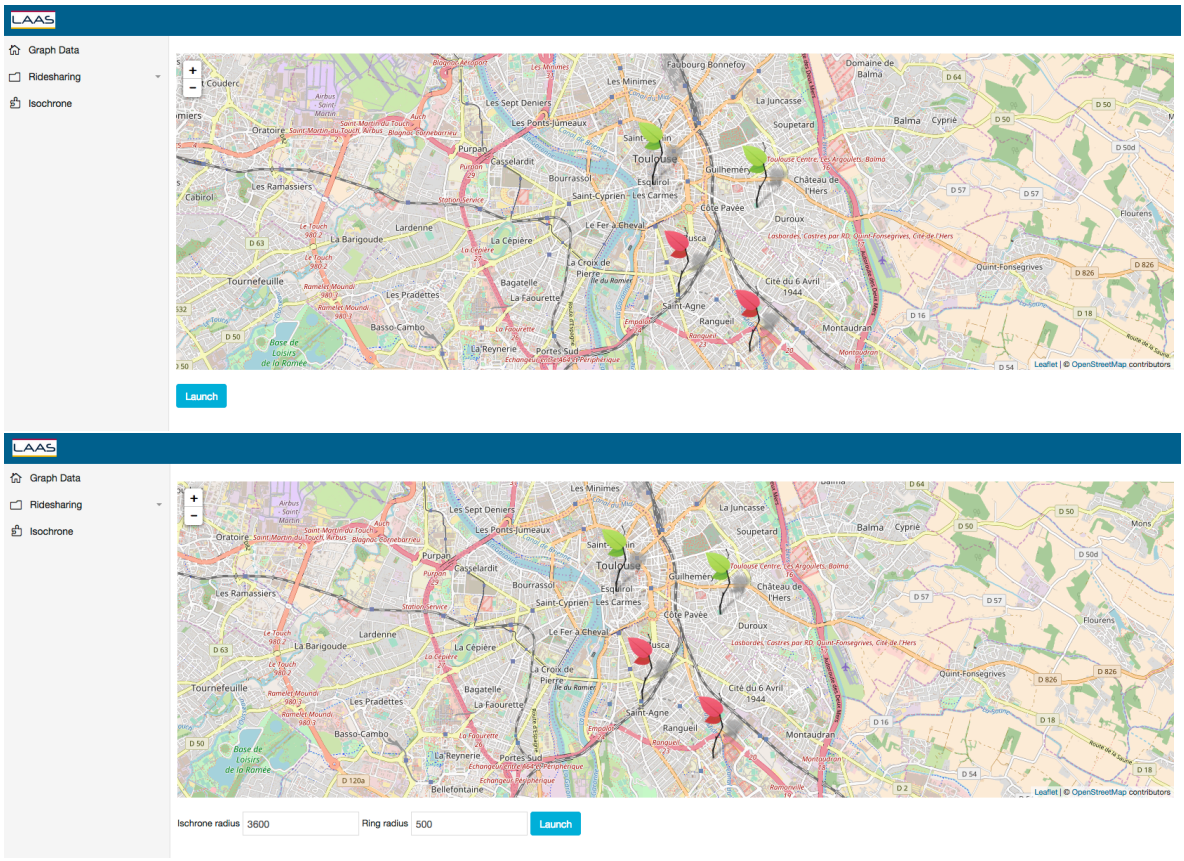

Figure 6.6: Example of ridesharing scenarios: inputs of `2SP-SP` (up) and `Priv-2SP-SP` (down)

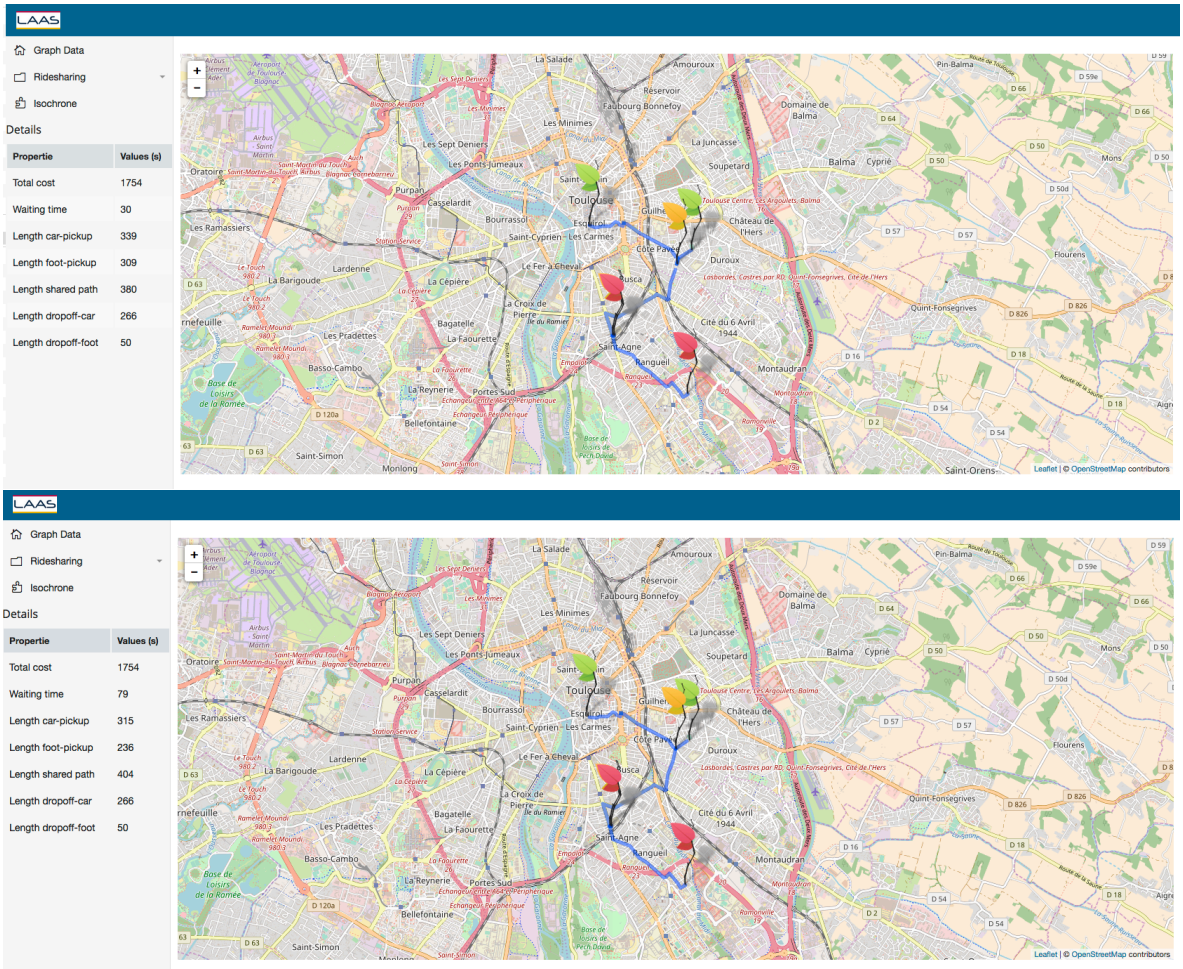

Figure 6.7: Example of ridesharing scenario: outputs of `2SP-SP` (up) and `Priv-2SP-SP` (down)

6.3 Example of integration

`PlayMob` has also been used by other researchers in our laboratory for other transportation problems. For example, authors in [105] propose adaptations of some k-shortest paths algorithms to consider the multimodal aspect of transportation networks. These algorithms have been implemented in the `IHM` module by an intern. The developed application, called `k-SP` allows the user to find alternative itineraries given his origin and destination locations, his departure time, the number of alternative itineraries, and her transportation modes and constraint (modeled as an automaton). Figure 6.8 shows an example of 50-shortest paths query.

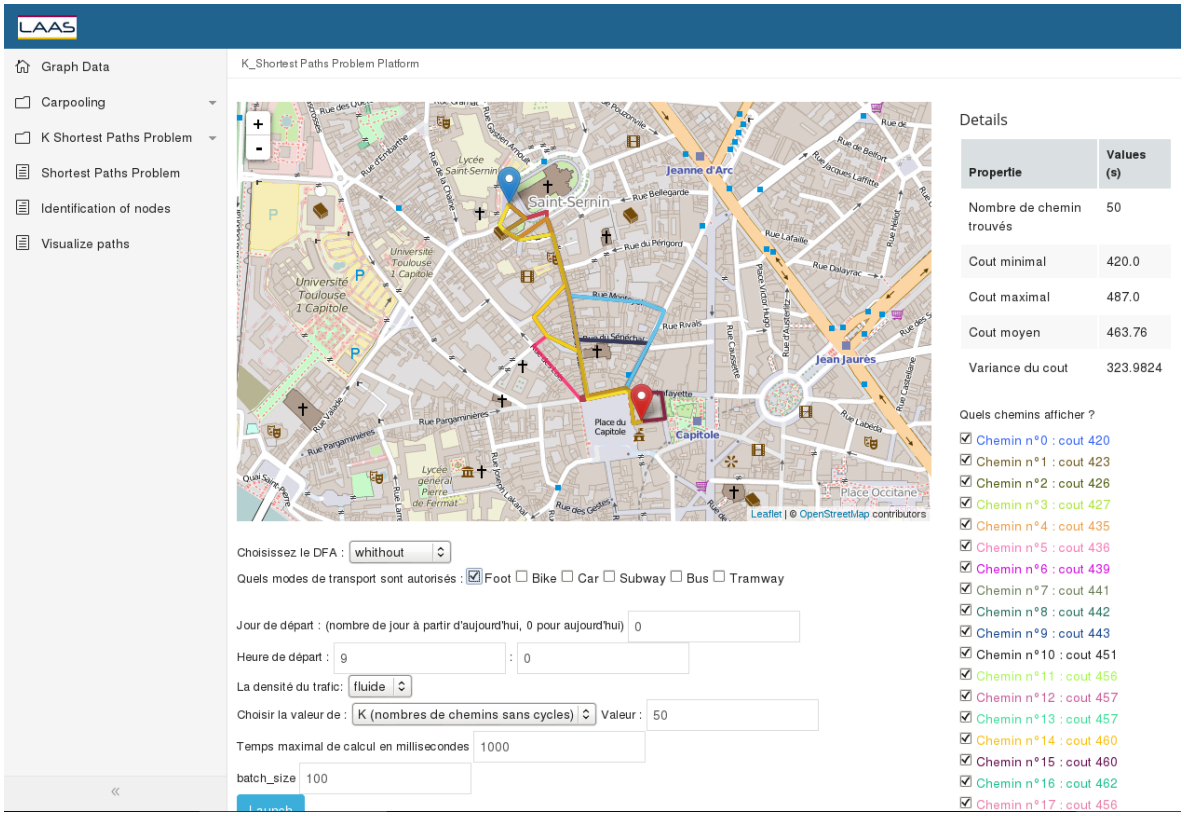

Figure 6.8: Homepages of PlayMob.

Conclusion

Summary

In this thesis, we have investigated the potential of using privacy enhancing technologies to design privacy-preserving ridesharing services. More precisely, this thesis has addressed the research question *"Can ridesharing services be implemented in a privacy-preserving fashion?"*. We have answered this question in the positive by proposing privacy-preserving protocols to solve two main features of ridesharing systems, namely the computation of meeting points, and the matching of drivers and riders. Our results are summarized as follows:

Privacy-preserving meeting points determination for ridesharing (Chapter 3). We have investigated the problem of meeting points determination in ridesharing. We have designed and evaluated privacy-preserving protocols that allow users to determine pick-up and drop-off locations for ridesharing, without revealing their origin and destination location. By using synthetic data and prototype applications, we have shown that our protocols have reasonable performances regarding computational and communication overheads and that they can compete with existing (non-privacy-preserving) ridesharing systems.

Privacy-preserving ride matching for ridesharing (Chapter 4). We have studied the problem of ride-matching in ridesharing. We have designed and evaluated a privacy-preserving ride matching protocol which allows riders and drivers to compute their matches without revealing their location data. By using a synthetic data, generated according to a real-world dataset, and a prototype application, we have demonstrated the computational and communication efficiency of our protocol. In particular, it has low communication overhead (under 11 MB) and low computational overhead (under 10 seconds) in realistic scenarios of ridesharing.

Overall, following the privacy-by-design principle, we have integrated existing privacy enhancing technologies to devise privacy-preserving protocols for ridesharing. We have shown that privacy requirements can be taken into account in ridesharing services without degrading their performance.

Future work

This thesis gives rise to several research directions.

In Chapter 3, to solve the secure meeting point determination problem, we have considered a case where the ridesharing stations are known in advance. We have analyzed the complexity of this approach and experimented on French transportation networks in both intercity and intracity scenarios. As future work, experiments on very large transportation networks

(e.g., New-York, Londres) can be run to see how this variant scales in practice. Additionally, differential privacy [44, 45] can be used in our privacy-preserving protocols for ridesharing. In particular, the exponential mechanism [81] can be integrated into the shared path election subroutine of our privacy-preserving meeting point determination protocols. Using differential privacy mechanisms will allow obtaining quantifiable privacy guarantees regardless the knowledge of the adversary.

In Chapter 4, we have used spatial and temporal generalization to compute feasible matches for each rider. More precisely, we have generalized each location to the district in which they are located, and each arrival to an epoch. As future work, other spatiotemporal generalizations can be considered. Furthermore, it is worth looking into how to solve the matching problem without using spatiotemporal generalizations. We would also like to extend our approach to consider the driver's vehicle capacity and the sharing of this capacity with several riders during the trip. The ridesharing's fare could also be integrated into the secure filtering protocol.

Another interesting research line is related to adapting our privacy-preserving protocols to shared mobility — the shared use of transportation mode, including bike sharing, ridesharing, and public transit — and allow to design privacy-preserving navigation services (services that cannot learn the origin-destination queries of users). Current implementations of privacy-preserving navigation services [127, 126] consider only road network and rely on the *private information retrieval* [30] protocol and the pre-computation of shortest paths. However, due to the dynamic nature of transit services, such techniques cannot be used. Hence, a solution to address this concern while not sacrificing efficiency can be designed.

In a very broad sense, the potential of the so-called *federated learning* [24] — a privacy-preserving machine learning approach that allows the training data to be distributed on the mobile devices, and learns a shared model by aggregating locally computed updates — for crowd-sourcing in the mobility context can be investigated.

As mobile devices are becoming more and more powerful, and the global awareness on privacy issues related to the use of these technologies are increasing, the integration of privacy-preserving technologies in mobility services could become more and more popular in the future. This thesis is a step towards designing privacy-preserving mobility services.

Appendix A

Résumé des Travaux de thèse

A.1 Contexte et Motivations

Le développement des technologies de l'information et de la communication (*TIC*) a contribué à l'émergence des services de covoiturage qui permettent aux conducteurs de partager les sièges libres de leurs véhicules avec des passagers ayant des trajets similaires.

Les services de covoiturage présentent des avantages économiques et écologiques. Du point de vue économique, ces services permettent la réduction des coûts de trajet grâce au partage des frais entre conducteurs et passagers. Sur le plan écologique, ils permettent de réduire la congestion et de facto l'émission des gaz à effet de serre.

Cependant, pour leur bon fonctionnement, ces services collectent des données personnelles telles que les données de localisation et les données financières des utilisateurs. La nature sensible des données de localisation peut conduire à des bris de vie privée. D'une part, l'analyse des données de localisation peut permettre l'identification des points d'intérêts (domicile, lieu de travail, loisirs) d'un individu ainsi que l'établissement de son modèle de mobilité [54]. D'autre part, ces données ont un fort pouvoir d'identification. En effet, il ne suffit que de la connaissance d'un quelques données de localisation pour identifier de façon unique un individu dans la population [82].

L'existence de ces risques pourrait porter un frein à l'adoption des systèmes de covoiturage. Dans le cadre de cette thèse, on s'intéresse au développement de technologies d'amélioration de la confidentialité (*TAC*) pour le covoiturage. Nous avons pour cela considéré deux problèmes inhérents au covoiturage : à savoir la synchronisation d'itinéraires permettant de déterminer les points de rencontre et de séparation utilisés pour faire du covoiturage et le problème d'appariement permettant de trouver un couplage des conducteurs et des passagers de sorte à réduire les coûts de covoiturage. Pour ces deux problèmes, nous avons proposé des algorithmes utilisant les TAC afin de garantir la protection de la vie privée des utilisateurs.

A.2 Nos contributions

Le **Chapitre 1** donne des définitions générales de la vie privée, de ses menaces, de ses propriétés ainsi que des dispositions de protection juridique et technique existantes.

Le **Chapitre 2** introduit des notions élémentaires de théorie des graphes et de routage multimodal. Il introduit également le covoiturage et analyse ses aspects de routage et d'appariement.

Dans le **Chapitre 3**, nous formalisons le problème de *synchronisation sécurisée pour le covoiturage* (`SMP4R`) et fournissons deux protocoles (`Priv-2SP-SP` et `PPLD4R`) préservant la confidentialité pour le résoudre. Le problème de synchronisation des itinéraires de covoiturage est illustré dans la figure A.1.

Nous considérons un conducteur d et un passager r. Chaque utilisateur $u \in \{d, r\}$ a un profil $P(u) = [O_u, D_u, \tau^u_{O_u}, \Sigma^u]$ constitué de son origine O_u, sa destination D_u, sa date de départ à l'origine $\tau^u_{O_u}$, ainsi que ses modes de transport Σ^u. Pour chaque utilisateur u et pour un doublet (i, j) de points de ramassage (*pick-up*) et de débarquement (*drop-off*), on définit le coût $Tr_{i-j}(u)$ de son itinéraire comme étant le coût du trajet $O_u \rightarrow i \rightarrow j \rightarrow D_u$. Ensuite, on définit le coût du covoiturage $\texttt{Ride}_{i-j}(d, r)$, pour un couple (d, r) de conducteur et de passager et un couple (i, j) de pick-up et drop-off, comme étant la somme des coûts des itinéraires du passager et de conducteur ainsi que du temps d'attente au pick-up: $\texttt{Ride}_{i-j}(d, r) = Tr_{i-j}(d) + Tr_{i-j}(r) + |\tau^d_i - \tau^r_i|$.

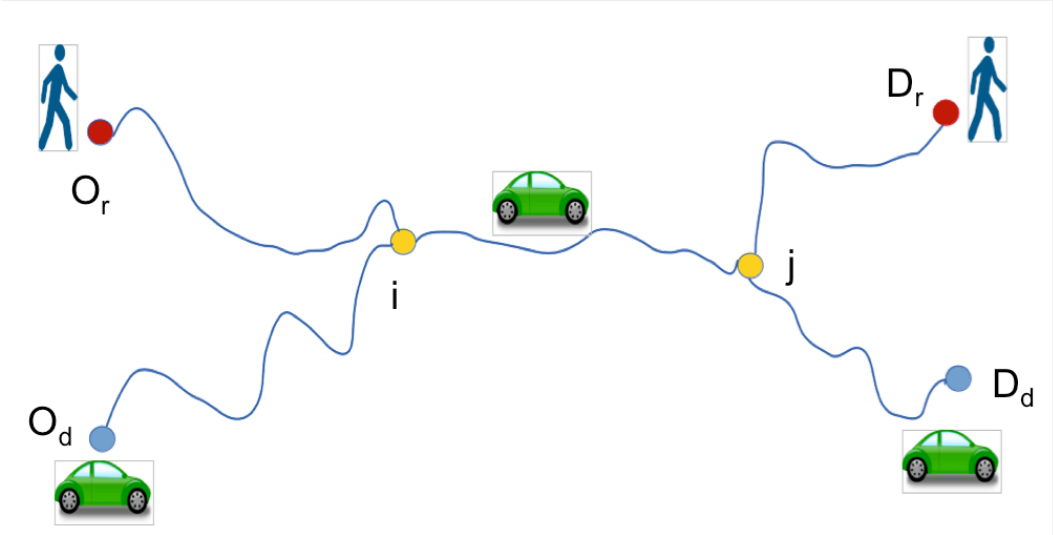

Figure A.1: Instance d'un problème de synchronisation pour le covoiturage

L'objectif du problème de synchronisation sécurisée est de trouver, étant donné un conducteur d, un passager r et un ensemble $\mathcal{S}$ de points de covoiturage potentiels, le couple $(i^*, j^*) \in \mathcal{S} \times \mathcal{S}$ tel que $\texttt{Ride}_{i^*-j^*}(d, r)$ est minimisé et $P(u)$ protégé $\forall u \in \{d, r\}$. Plus précisément, un profil $P(u)$ sera dit protégé si les trois éléments O_u, D_u et $\tau^u_{O_u}$ sont protégés (ne sont révélés à aucun autre utilisateur durant le calcul de (i^*, j^*)).

Le protocole `Priv-2SP-SP` est une version decentralisée et sécurisée de l'algorithme `2SP-SP` introduit dans [21]. Contrairement à `2SP-SP`, `Priv-2SP-SP` permet la détermination des points de pick-up et drop-off pour le covoiturage tout en garantissant la protection des données de localisation des utilisateurs. À l'instar de `2SP-SP`, `Priv-2SP-SP` modélise le réseau

de transport sous forme de graphe multimodal; il se déroule comme suit: dans un premier temps, le passager et le conducteur calculent chacun en local leurs ensemble de points de covoiturage (pick-up et drop-off) potentiels. Ensuite les deux utilisateurs engagent un protocole d'intersection sécurisée pour déterminer les points de pick-up et de drop-off communs. Puis, pour chaque couple de point (i, j), chaque utilisateur calcul un score inversement proportionnel au coût de l'itinéraire induit par (i, j). Enfin, le couple (i, j) ayant le score cumulé le plus grand est choisie comme solution.

Le protocole `PPLD4R` permet également de déterminer de manière sécurisée les points de synchronisation pour le covoiturage sans utiliser un protocole d'intersection sécurisée. Il est utilisé dans le cas statique où l'ensemble $\mathcal{S}$ des points de covoiturage potentiels est connu à l'avance, permettant ainsi le pré-calcul des trajets partagés $i \to j$. Il permet également d'intégrer le temps d'attente au point de pick-up grâce à une sous routine de comparaison sécurisée. Enfin, il utilise un protocole de secret partagé pour prendre en compte de façon explicite le coût du covoiturage dans la phase de sélection de la meilleur solution.

D'une part, on compare `Priv-2SP-SP` à `2SP-SP` sur deux groupes de 100 instances chacun. En termes de temps de calcul, `Priv-2SP-SP` est 1.6× (resp. 1.4×) plus rapide que `2SP-SP` sur le groupe I (resp. sur le groupe II). Cela est dû d'une part au fait que `Priv-2SP-SP` explore moins l'espace des solutions que `2SP-SP`, et d'autres part au fait que les routines cryptographiques ont un temps de calcul quasi-linéaire au nombre de nuds dans le réseau de transport. En terme de coût du covoiturage, on observe des écarts de 4.4% (resp. 2.5%) dans le groupe I (resp. le groupe II) entre les solutions du `Priv-2SP-SP` et les solutions optimales du `2SP-SP`. La qualité des solutions du `Priv-2SP-SP` est également évaluée en termes de proximité géographique par rapport aux solutions du `2SP-SP`. On observe des écarts de 230m (resp. 103 m) entres les pick-ups dans le groupe I (resp. le groupe II), et des écarts de 202m (resp. 99 m) entres les drop-offs dans le groupe I (resp. le groupe II).

D'autre part on compare `PPLD4R` et `2SP-SP` sur deux scénarios (intra-cité et inter-cité), de 1000 instances chacun, en terme de qualité de la solution et de temps de calcul, en supposant que le trajet du passager est indépendant du temps entre le drop-off et sa destination finale. Ici, on obtient les mêmes coûts de covoiturage pour `PPLD4R` et `2SP-SP`, tandis que `PPLD4R` est 7× (resp. 33×) plus rapide que `2SP-SP` sur le scénario intra-cité (resp. le scénario inter-cité), grâce au pré-calcul des chemins partagés. On observe également des coûts de communication raisonnables pour `PPLD4R`: de l'ordre de 250 ko.

Les résultats de nos expériences montrent que l'on peut concevoir des protocoles de synchronisation d'itinéraires préservant la vie privée avec une qualité similaire aux protocoles de covoiturage existants sans introduire de coûts de calcul et de communication importants.

Dans le **Chapitre 4**, nous considérons une version généralisée du problème `SMP4R`, dans laquelle nous considérons plusieurs conducteurs et passagers et souhaitons assigner les conducteurs aux passagers tout en préservant les données de localisation de chaque utilisateur. Nous proposons `SRide`, un système d'appariement préservant la vie privée qui fonctionne en trois étapes. Tout d'abord, il utilise un protocole de filtrage sécurisé pour construire le graphe bipartite des correspondances possibles. Ensuite, il s'appuie sur les protocoles sécurisés de

détermination de points de synchronisation proposés dans le Chapitre 3 pour obtenir le coût de chaque paire possible. Enfin, il détermine les affectations optimales des conducteurs et des passagers. Nous évaluons notre protocole et démontrons son efficacité en temps de calcul et en communication. En particulier, on obtient des temps de calcul inférieurs à 10 secondes et des coûts de communication de l'ordre de 11 MO sur des scénarios réalistes de 1000 offres et 1000 demandes de covoiturage sur une plage horaire de 1 heure.

Le **Chapitre 5** est dédié à l'analyse des données collectées sur *Covoiturage-libre.fr* [1]: un service web de covoiturage populaire français. Cette analyse nous permet d'avoir plus d'informations sur le covoiturage en France et d'utiliser cette information pour générer des données synthétiques réalistes pour nos expériences.

Dans le **Chapitre 6**, nous présentons les principales fonctionnalités de `PlayMob`: une plateforme web qui implémente les différents algorithmes proposés dans cette thèse.

[1] http://covoiturage-libre.fr

Bibliography

[1] Niels Agatz, Alan L Erera, Martin WP Savelsbergh, and Xing Wang. "Dynamic ride-sharing: A simulation study in metro Atlanta". In: *Procedia-Social and Behavioral Sciences* 17 (2011), pp. 532–550 (cit. on p. 28).

[2] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. "Optimization for dynamic ride-sharing: A review". In: *European Journal of Operational Research* 223.2 (2012), pp. 295–303 (cit. on pp. 1, 26, 28, 37).

[3] Carlos Aguilar-Melchor, Joris Barrier, Serge Guelton, Adrien Guinet, Marc-Olivier Killijian, and Tancrede Lepoint. "NFLlib: NTT-based Fast Lattice Library". In: *RSA Conference Cryptographers' Track*. 2016 (cit. on pp. 50, 59, 79, 104).

[4] Kamel Aissat and Ammar Oulamara. "Dynamic ridesharing with intermediate locations". In: *Computational Intelligence in Vehicles and Transportation Systems (CIVTS), 2014 IEEE Symposium on*. IEEE. 2014, pp. 36–42 (cit. on p. 27).

[5] Ulrich Matchi Aïvodji. "Privacy Enhancing Technologies for Ridesharing". In: *Student Forum of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. 2016 (cit. on p. 2).

[6] Ulrich Matchi Aïvodji, Sébastien Gambs, Marie-José Huguet, and Marc-Olivier Killijian. "Meeting points in ridesharing: A privacy-preserving approach". In: *Transportation Research Part C: Emerging Technologies* 72 (2016), pp. 239–253 (cit. on p. 2).

[7] Ulrich Matchi Aïvodji, Sébastien Gambs, Marie-José Huguet, and Marc-Olivier Killijian. "Privacy-preserving carpooling". In: *Odysseus 2015-6th International Workshop on Freight Transportation and Logistics (Odysseus)*. 2015 (cit. on p. 2).

[8] Ulrich Matchi Aïvodji, Kévin Huguenin, Marie-José Huguet, and Marc-Olivier Killijian. "SRide: A Privacy-Preserving Ridesharing System". In: *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*. (Submitted) (cit. on p. 2).

[9] Christian Artigues, Yves Deswarte, Jérémie Guiochet, Marie-José Huguet, Marc-Olivier Killijian, David Powell, Matthieu Roy, Christophe Bidan, Nicolas Prigent, Emmanuelle Anceaume, Sébastien Gambs, Gilles Guette, Michel Hurfin, and Frédéric Schettini. "AMORES: An Architecture for Mobiquitous Resilient Systems". In: *Proceedings of the 1st European Workshop on AppRoaches to MObiquiTous Resilience*. ARMOR '12. Sibiu, Romania: ACM, 2012, 7:1–7:6 (cit. on p. 37).

[10] Gilad Asharov, Daniel Demmler, Michael Schapira, Thomas Schneider, Gil Segev, Scott Shenker, and Michael Zohner. "Privacy-Preserving Interdomain Routing at Internet Scale". In: *Proceedings on Privacy Enhancing Technologies* 3 (2017), pp. 1–21 (cit. on p. 80).

[11] Küpper Axel et al. *Location-based services: fundamentals and operation*. John Wiley & Sons, 2005 (cit. on p. 7).

[12] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. “Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography”. In: *Proceedings of the 16th international conference on World Wide Web*. ACM. 2007, pp. 181–190 (cit. on p. 11).

[13] Roberto Baldacci, Vittorio Maniezzo, and Aristide Mingozzi. “An exact method for the car pooling problem based on lagrangean column generation”. In: *Operations Research* 52.3 (2004), pp. 422–439 (cit. on p. 28).

[14] Michael Barbaro, Tom Zeller, and Saul Hansell. “A face is exposed for AOL searcher no. 4417749”. In: *New York Times* 9.2008 (2006), 8For (cit. on p. 11).

[15] Chris Barrett, Riko Jacob, and Madhav Marathe. “Formal-language-constrained path problems”. In: *SIAM Journal on Computing* 30.3 (2000), pp. 809–837 (cit. on pp. 25, 103).

[16] Russell Belk. “You are what you can access: Sharing and collaborative consumption online”. In: *Journal of Business Research* 67.8 (2014), pp. 1595–1600 (cit. on p. 1).

[17] Richard Bellman. “On a routing problem”. In: *Quarterly of applied mathematics* 16.1 (1958), pp. 87–90 (cit. on p. 23).

[18] Alastair R. Beresford and Franck Stajano. “Location privacy in pervasive computing”. In: *Pervasive Computing* 2.1 (2003), pp. 46–55 (cit. on p. 17).

[19] Alastair R Beresford and Frank Stajano. “Location privacy in pervasive computing”. In: *IEEE Pervasive computing* 2.1 (2003), pp. 46–55 (cit. on p. 8).

[20] Igor Bilogrevic, Murtuza Jadliwala, Vishal Joneja, Kubra Kalkan, Jean-Pierre Hubaux, and Imad Aad. “Privacy-preserving optimal meeting location determination on mobile devices”. In: *Information Forensics and Security, IEEE Transactions on* 9.7 (2014), pp. 1141–1156 (cit. on pp. 39, 40).

[21] Arthur Bit-Monnot, Christian Artigues, Marie-José Huguet, and Marc-Olivier Killijian. “Carpooling: the 2 synchronization points shortest paths problem”. In: *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS)*. 2013, 12–p (cit. on pp. 1, 27, 37, 39, 50, 116).

[22] George Robert Blakley et al. “Safeguarding cryptographic keys”. In: *Proceedings of the national computer conference*. Vol. 48. 1979, pp. 313–317 (cit. on p. 17).

[23] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. “Fast unfolding of communities in large networks”. In: *Journal of statistical mechanics: theory and experiment* 2008.10 (2008), P10008 (cit. on p. 96).

[24] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. “Practical Secure Aggregation for Privacy Preserving Machine Learning.” In: *IACR Cryptology ePrint Archive* 2017 (2017), p. 281 (cit. on p. 114).

[25] L Burnett, K Barlow-Stewart, AL Proos, and H Aizenberg. “The" GeneTrustee": a universal identification system that ensures privacy and confidentiality for human genetic databases.” In: *Journal of Law and Medicine* 10.4 (2003), pp. 506–513 (cit. on p. 7).

[26] Roberto Wolfler Calvo, Fabio de Luigi, Palle Haastrup, and Vittorio Maniezzo. “A distributed geographic information system for the daily car pooling problem”. In: *Computers & Operations Research* 31.13 (2004), pp. 2263–2278 (cit. on p. 29).

[27] Brian Caulfield. “Estimating the environmental benefits of ride-sharing: A case study of Dublin”. In: *Transportation Research Part D: Transport and Environment* 14.7 (2009), pp. 527–531 (cit. on p. 1).

[28] Ann Cavoukian. “Privacy by design [leading edge]”. In: *IEEE Technology and Society Magazine* 31.4 (2012), pp. 18–19 (cit. on p. 13).

[29] Ann Cavoukian. *Privacy by design... take the challenge. Information and Privacy Commissioner of Ontario (Canada)*. 2009 (cit. on pp. 13, 45).

[30] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. “Private information retrieval”. In: *Journal of the ACM (JACM)* 45.6 (1998), pp. 965–981 (cit. on pp. 85, 114).

[31] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20152020*. 2016 (cit. on p. 1).

[32] Caitlin D. Cottrill. “Approaches to privacy preservation in intelligent transportation systems and vehicle-infrastructure integration initiative”. In: *Transportation Research Record: Journal of the Transportation Research Board* 2129 (2009), pp. 9–15 (cit. on p. 14).

[33] James Covert. *Uber says it got hacked over nine months ago*. `https://goo.gl/ip5oS7`. Accessed: 2016-11-11. 2015 (cit. on p. 1).

[34] *Data usage statistics*. `https://www.verizonwireless.com/data-calculator/`. Accessed: 2017-08-17 (cit. on p. 62).

[35] Emiliano De Cristofaro and Gene Tsudik. “Practical private set intersection protocols with linear complexity”. In: *Financial Cryptography and Data Security*. Springer, 2010, pp. 143–159 (cit. on pp. 18, 19).

[36] Yves-Alexandre De Montjoye, Laura Radaelli, Vivek Kumar Singh, et al. “Unique in the shopping mall: On the reidentifiability of credit card metadata”. In: *Science* 347.6221 (2015), pp. 536–539 (cit. on p. 11).

[37] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. “Unique in the crowd: The privacy bounds of human mobility”. In: *Scientific reports* 3 (2013), p. 1376 (cit. on p. 11).

[38] Daniel Demmler, Thomas Schneider, and Michael Zohner. “ABY-A Framework for Efficient Mixed-Protocol Secure Two-Party Computation.” In: *NDSS*. 2015 (cit. on pp. 59, 80, 104).

[39] Mina Deng, Kim Wuyts, Riccardo Scandariato, Bart Preneel, and Wouter Joosen. “A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements”. In: *Requirements Engineering* 16.1 (2011), pp. 3–32 (cit. on pp. 8–10).

[40] Edsger W. Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische mathematik* 1.1 (1959), pp. 269–271 (cit. on p. 23).

[41] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The second-generation onion router*. Tech. rep. Naval Research Lab Washington DC, 2004 (cit. on p. 67).

[42] Changyu Dong, Liqun Chen, and Zikai Wen. "When private set intersection meets big data: an efficient and scalable protocol". In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM. 2013, pp. 789–800 (cit. on pp. 18, 19, 39).

[43] Florian Drews and Dennis Luxen. "Multi-hop ride sharing". In: *Sixth annual symposium on combinatorial search*. 2013 (cit. on p. 28).

[44] Cynthia Dwork. "Differential Privacy". In: *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*. Ed. by Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener. Springer Berlin Heidelberg, 2006, pp. 1–12 (cit. on pp. 15, 114).

[45] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis". In: *TCC*. Vol. 3876. Springer. 2006, pp. 265–284 (cit. on pp. 15, 114).

[46] Taher ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE transactions on information theory* 31.4 (1985), pp. 469–472 (cit. on p. 17).

[47] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. "The many faces of publish/subscribe". In: *ACM Computing Surveys (CSUR)* 35.2 (2003), pp. 114–131 (cit. on p. 85).

[48] Junfeng Fan and Frederik Vercauteren. "Somewhat Practical Fully Homomorphic Encryption." In: *IACR Cryptology ePrint Archive* 2012 (2012), p. 144 (cit. on pp. 17, 55, 57, 59, 77, 104).

[49] Robert W. Floyd. "Algorithm 97: shortest path". In: *Communications of the ACM* 5.6 (1962), p. 345 (cit. on p. 24).

[50] Michael J Freedman, Kobbi Nissim, and Benny Pinkas. "Efficient private matching and set intersection". In: *Advances in Cryptology-EUROCRYPT 2004*. Springer. 2004, pp. 1–19 (cit. on pp. 18–20, 39, 47, 48, 85).

[51] Benjamin Fung, Ke Wang, Rui Chen, and Philip S Yu. "Privacy-preserving data publishing: A survey of recent developments". In: *ACM Computing Surveys (CSUR)* 42.4 (2010), p. 14 (cit. on p. 10).

[52] Masabumi Furuhata, Maged Dessouky, Fernando Ordóñez, Marc-Etienne Brunet, Xiaoqing Wang, and Sven Koenig. "Ridesharing: The state-of-the-art and future directions". In: *Transportation Research Part B: Methodological* 57 (2013), pp. 28–46 (cit. on pp. 27, 37, 67).

[53] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. "Show Me How You Move and I Will Tell You Who You Are". In: *Transactions on Data Privacy* 4.2 (2011), pp. 103–126 (cit. on p. 1).

[54] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. "Show Me How You Move and I Will Tell You Who You Are". In: *Transactions on Data Privacy* 4.2 (2011), pp. 103–126 (cit. on p. 115).

[55] Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. "Exact routing in large road networks using contraction hierarchies". In: *Transportation Science* 46.3 (2012), pp. 388–404 (cit. on p. 24).

[56] Robert Geisberger, Dennis Luxen, Sabine Neubauer, Peter Sanders, and Lars Volker. "Fast detour computation for ride sharing". In: *arXiv preprint arXiv:0907.5269* (2009) (cit. on p. 28).

[57] Craig Gentry et al. "Fully homomorphic encryption using ideal lattices." In: *STOC*. Vol. 9. 2009. 2009, pp. 169–178 (cit. on p. 17).

[58] Moein Ghasemzadeh, Benjamin CM Fung, Rui Chen, and Anjali Awasthi. "Anonymizing trajectory data for passenger flow analysis". In: *Transportation Research Part C: Emerging Technologies* 39 (2014), pp. 63–79 (cit. on p. 84).

[59] *God View: Uber Allegedly Stalked Users For Party-Goers Viewing Pleasure*. `http://www.huffingtonpost.com/entry/uber-settlement-god-view_us_568da2a6e4b0c8beacf5a46a`. Accessed: 2016-11-11 (cit. on p. 1).

[60] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. "On private scalar product computation for privacy-preserving data mining." In: *ICISC*. Vol. 3506. Springer. 2004, pp. 104–120 (cit. on p. 17).

[61] Andrew V Goldberg and Chris Harrelson. "Computing the shortest path: A search meets graph theory". In: *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics. 2005, pp. 156–165 (cit. on p. 24).

[62] Oded Goldreich, Silvio Micali, and Avi Wigderson. "How to play any mental game". In: *Proceedings of the nineteenth annual ACM symposium on Theory of computing*. ACM. 1987, pp. 218–229 (cit. on pp. 18, 55, 56, 59, 77, 80).

[63] P Gruebele. "Interactive system for real time dynamic multi-hop carpooling". In: *Global Transport Knowledge Partnership* (2008) (cit. on p. 28).

[64] Marco Gruteser and Dirk Grunwald. "Anonymous usage of location-based services through spatial and temporal cloaking". In: *Proceedings of the 1st international conference on Mobile systems, applications and services*. ACM. 2003, pp. 31–42 (cit. on p. 16).

[65] Per Hallgren, Claudio Orlandi, and Andrei Sabelfeld. "PrivatePool: Privacy-Preserving Ridesharing". In: *Computer Security Foundations Symposium (CSF), 2017 IEEE 30th*. IEEE. 2017, pp. 276–291 (cit. on p. 85).

[66] Peter E Hart, Nils J Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107 (cit. on p. 24).

[67] Carmit Hazay and Yehuda Lindell. “Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries”. In: *Theory of Cryptography*. Springer, 2008, pp. 155–175 (cit. on pp. 18, 19).

[68] Carmit Hazay and Yehuda Lindell. *Efficient secure two-party protocols: Techniques and constructions*. Springer Science & Business Media, 2010 (cit. on p. 11).

[69] Wesam Herbawi and Michael Weber. “Evolutionary Multiobjective Route Planning in Dynamic Multi-hop Ridesharing.” In: *EvoCOP*. Vol. 11. Springer. 2011, pp. 84–95 (cit. on p. 28).

[70] Wesam Herbawi and Michael Weber. “The ridematching problem with time windows in dynamic ridesharing: A model and a genetic algorithm”. In: *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE. 2012, pp. 1–8 (cit. on p. 29).

[71] Michael Herrmann, Alfredo Rial, Claudia Diaz, and Bart Preneel. “Practical privacy-preserving location-sharing based services with aggregate statistics”. In: *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*. ACM. 2014, pp. 87–98 (cit. on p. 39).

[72] Gesine Hinterwälder, Christian T Zenger, Foteini Baldimtsi, Anna Lysyanskaya, Christof Paar, and Wayne P Burleson. “Efficient e-cash in practice: NFC-based payments for public transportation systems”. In: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. 2013, pp. 40–59 (cit. on p. 84).

[73] Baik Hoh, Marco Gruteser, Hui Xiong, and Ansaf Alrabady. “Enhancing security and privacy in traffic-monitoring systems”. In: *IEEE Pervasive Computing* 5.4 (2006), pp. 38–46 (cit. on p. 12).

[74] Stanisław Jarecki and Xiaomin Liu. “Efficient oblivious pseudorandom function with applications to adaptive ot and secure computation of set intersection”. In: *Theory of Cryptography*. Springer, 2009, pp. 577–594 (cit. on pp. 18, 19).

[75] Hidetoshi Kido, Yutaka Yanagisawa, and Tetsuji Satoh. “Protection of location privacy using dummies for location-based services”. In: *Data Engineering Workshops, 2005. 21st International Conference on*. IEEE. 2005, pp. 1248–1248 (cit. on p. 15).

[76] Lea Kissner and Dawn Song. “Privacy-preserving set operations”. In: *Advances in Cryptology–CRYPTO 2005*. Springer. 2005, pp. 241–257 (cit. on pp. 18, 19).

[77] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. “t-closeness: Privacy beyond k-anonymity and l-diversity”. In: *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*. IEEE. 2007, pp. 106–115 (cit. on p. 15).

[78] *LIL: Données à caractère personel*. `https://www.legifrance.gouv.fr/affichTexte.do?cidTexte=JORFTEXT000000886460`. Accessed: 2017-05-15. 2017 (cit. on p. 7).

[79] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. “On ideal lattices and learning with errors over rings”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2010, pp. 1–23 (cit. on p. 50).

[80] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. “l-diversity: Privacy beyond k-anonymity”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1.1 (2007), p. 3 (cit. on p. 15).

[81] Frank McSherry and Kunal Talwar. “Mechanism design via differential privacy”. In: *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*. IEEE. 2007, pp. 94–103 (cit. on pp. 86, 114).

[82] Yves-Alexandre de Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. “Unique in the crowd: The privacy bounds of human mobility”. In: *Scientific reports* 3 (2013) (cit. on p. 115).

[83] Giacomo Nannicini, Daniel Delling, Dominik Schultes, and Leo Liberti. “Bidirectional A* search on time-dependent road networks”. In: *Networks* 59.2 (2012), pp. 240–251 (cit. on p. 26).

[84] *Nantes partitioning*. `http://www.nantes.fr/files/live/sites/nantesfr/files/cartequartier.jpg`. Accessed: 2017-08-09 (cit. on pp. 58, 80).

[85] Moni Naor and Benny Pinkas. “Oblivious transfer and polynomial evaluation”. In: *Proceedings of the thirty-first annual ACM symposium on Theory of computing*. ACM. 1999, pp. 245–254 (cit. on p. 18).

[86] Arvind Narayanan and Vitaly Shmatikov. “How to break anonymity of the netflix prize dataset”. In: *arXiv preprint cs/0610105* (2006) (cit. on p. 11).

[87] Jesper Buus Nielsen, Thomas Schneider, and Roberto Trifiletti. “Constant Round Maliciously Secure 2PC with Function-independent Preprocessing using LEGO.” In: *IACR Cryptology ePrint Archive* 2016 (2016), p. 1069 (cit. on pp. 63, 84).

[88] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. “A New Approach to Practical Active-Secure Two-Party Computation.” In: *CRYPTO*. Vol. 7417. Springer. 2012, pp. 681–700 (cit. on pp. 63, 84).

[89] *NIST Special Publication 800-122: Personally Identifiable Information*. `http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-122.pdf`. Accessed: 2017-05-15. 2017 (cit. on p. 7).

[90] Miguel Nuñez Del Prado Cortez. “Inference attacks on geolocated data”. PhD thesis. 2013 (cit. on p. 8).

[91] Pascal Paillier et al. “Public-key cryptosystems based on composite degree residuosity classes”. In: *Eurocrypt*. Vol. 99. Springer. 1999, pp. 223–238 (cit. on p. 17).

[92] Anh Pham, Italo Dacosta, Guillaume Endignoux, Juan Ramón Troncoso-Pastoriza, Kévin Huguenin, and Jean-Pierre Hubaux. “ORide: A Privacy-Preserving yet Accountable Ride-Hailing Service”. In: *Proc. of the 26th USENIX Security Symposium*. 2017, pp. 1235–1252 (cit. on p. 86).

[93] Anh Pham, Italo Dacosta, Bastien Jacot-Guillarmod, Kévin Huguenin, and Jean-Pierre Hubaux. “PrivateRide: A Privacy-Preserving and Secure Ride-Hailing Service”. In: *PoPETs* (2017). To appear (cit. on pp. 66, 84, 86).

[94] Ira Pohl. “Bi-directional and heuristic search in path problems”. PhD thesis. Department of Computer Science, Stanford University, 1969 (cit. on p. 24).

[95] *Privacy*. `https://en.oxforddictionaries.com/definition/privacy`. Accessed: 2017-05-11. 2017 (cit. on p. 7).

[96] Evangelia Pyrga, Frank Schulz, Dorothea Wagner, and Christos Zaroliagis. “Efficient models for timetable information in public transportation systems”. In: *Journal of Experimental Algorithmics (JEA)* 12 (2008), pp. 2–4 (cit. on p. 26).

[97] Daniele Quercia, Neal Lathia, Francesco Calabrese, Giusy Di Lorenzo, and Jon Crowcroft. “Recommending social events from mobile phone location data”. In: *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE. 2010, pp. 971–976 (cit. on p. 37).

[98] Michael O Rabin. “How To Exchange Secrets with Oblivious Transfer.” In: *IACR Cryptology ePrint Archive* 2005 (2005), p. 187 (cit. on p. 18).

[99] *Rennes partitioning*. `https://upload.wikimedia.org/wikipedia/commons/thumb/a/a6/Plan_quartiers_Rennes.svg/477px-Plan_quartiers_Rennes.svg.png`. Accessed: 2017-08-09 (cit. on pp. 58, 80).

[100] Ronald L Rivest, Adi Shamir, and Leonard Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. In: *Communications of the ACM* 21.2 (1978), pp. 120–126 (cit. on p. 17).

[101] Ryuichi Sakai and Jun Furukawa. “Identity-Based Broadcast Encryption.” In: *IACR Cryptology ePrint Archive* 2007 (2007), p. 217 (cit. on p. 39).

[102] David Sánchez, Sergio Martínez, and Josep Domingo-Ferrer. “Co-utile P2P ridesharing via decentralization and reputation management”. In: *Transportation Research Part C: Emerging Technologies* 73 (2016), pp. 147–166 (cit. on pp. 84, 85).

[103] Peter Sanders and Dominik Schultes. “Engineering highway hierarchies”. In: *ESA*. Vol. 6. Springer. 2006, pp. 804–816 (cit. on p. 24).

[104] Peter Sanders and Dominik Schultes. “Robust, Almost Constant Time Shortest-Path Queries in Road Networks.” In: *The Shortest Path Problem*. 2006, pp. 193–218 (cit. on p. 25).

[105] Grégoire Scano, Marie-José Huguet, and Sandra Ulrich Ngueveu. “Adaptations of k-shortest path algorithms for transportation networks”. In: *Industrial Engineering and Systems Management (IESM), 2015 International Conference on*. IEEE. 2015, pp. 663–669 (cit. on p. 112).

[106] Thomas Schneider and Michael Zohner. “GMW vs. Yao? Efficient secure two-party computation with low depth circuits”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2013, pp. 275–292 (cit. on pp. 57, 80).

[107] Adi Shamir. “How to share a secret”. In: *Communications of the ACM* 22.11 (1979), pp. 612–613 (cit. on p. 17).

[108] Reza Shokri. “Quantifying and protecting location privacy”. In: *it-Information Technology* 57.4 (2015), pp. 257–263 (cit. on p. 39).

[109] Reza Shokri, George Theodorakopoulos, Jean-Yves Le Boudec, and Jean-Pierre Hubaux. “Quantifying location privacy”. In: *Security and privacy (sp), 2011 ieee symposium on*. IEEE. 2011, pp. 247–262 (cit. on p. 8).

[110] Mitja Stiglic, Niels Agatz, Martin Savelsbergh, and Mirko Gradisar. “Making dynamic ride-sharing work: The impact of driver and rider flexibility”. In: *Transportation Research Part E: Logistics and Transportation Review* 91 (2016), pp. 190–207 (cit. on p. 1).

[111] Mitja Stiglic, Niels Agatz, Martin Savelsbergh, and Mirko Gradisar. “The benefits of meeting points in ride-sharing systems”. In: *Transportation Research Part B: Methodological* 82 (2015), pp. 36–53 (cit. on pp. 1, 28, 37).

[112] Amarnag Subramanya, Alvin Raj, Jeff A Bilmes, and Dieter Fox. “Recognizing activities and spatial context using wearable sensors”. In: *arXiv preprint arXiv:1206.6869* (2012) (cit. on p. 13).

[113] Jingchao Sun, Rui Zhang, and Yanchao Zhang. “Privacy-preserving spatiotemporal matching”. In: *INFOCOM, 2013 Proceedings IEEE*. IEEE. 2013, pp. 800–808 (cit. on p. 39).

[114] Zhanbo Sun, Bin Zan, Xuegang Jeff Ban, and Marco Gruteser. “Privacy protection method for fine-grained urban traffic modeling using mobile sensors”. In: *Transportation Research Part B: Methodological* 56 (2013), pp. 50–69 (cit. on p. 84).

[115] Latanya Sweeney. “Achieving k-anonymity privacy protection using generalization and suppression”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 571–588 (cit. on p. 15).

[116] Latanya Sweeney. “k-anonymity: A model for protecting privacy”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 557–570 (cit. on pp. 11, 14, 84).

[117] Latanya Sweeney. “Simple demographics often identify people uniquely”. In: *Health (San Francisco)* 671 (2000), pp. 1–34 (cit. on p. 11).

[118] Wei Tong, Jingyu Hua, and Sheng Zhong. “A Jointly Differentially Private Scheduling Protocol for Ridesharing Services”. In: *IEEE Transactions on Information Forensics and Security* (2017) (cit. on p. 85).

[119] *Toulouse partitioning*. `https://www.loi-duflot-toulouse.info/wp/wp-content/themes/loi_toulouse_duflot/img/carte_toulouse.png`. Accessed: 2017-08-09 (cit. on p. 58).

[120] *Universal Declaration of Human Rights*. `http://www.ohchr.org/EN/UDHR/Documents/UDHR_Translations/eng.pdf`. Accessed: 2017-05-29. 2017 (cit. on p. 13).

[121] GW Van Blarkom, JJ Borking, and JGE Olk. “Handbook of privacy and privacy-enhancing technologies”. In: *Privacy Incorporated Software Agent (PISA) Consortium, The Hague* (2003) (cit. on p. 14).

[122] Sacha Varone and Kamel Aissat. “Routing with public transport and ride-sharing”. In: *Sixth International Workshop on Freight Transportation and Logistics (Odysseus)*. 2015 (cit. on pp. 27, 37).

[123] Elizabeth Weise and Jessica Guynn. *Uber tracking raises privacy concerns*. `https://goo.gl/EmVjWU`. Accessed: 2016-11-11. 2014 (cit. on p. 1).

[124] Marius Wernke, Pavel Skvortsov, Frank Dürr, and Kurt Rothermel. “A classification of location privacy attacks and approaches”. In: *Personal and Ubiquitous Computing* 18.1 (2014), pp. 163–175 (cit. on p. 12).

[125] Alan F Westin. “Privacy and freedom”. In: *Washington and Lee Law Review* 25.1 (1968), p. 166 (cit. on p. 7).

[126] David J. Wu, Joe Zimmerman, Jérémy Planul, and John C. Mitchell. “Privacy-Preserving Shortest Path Computation”. In: *arXiv preprint arXiv:1601.02281* (2016) (cit. on pp. 85, 114).

[127] Yong Xi, Loren Schwiebert, and Weisong Shi. “Privacy preserving shortest path routing with an application to navigation”. In: *Pervasive and Mobile Computing* 13 (2014), pp. 142 –149 (cit. on pp. 85, 114).

[128] Hai Yang and Hai-Jun Huang. “Carpooling and congestion pricing in a multilane highway with high-occupancy-vehicle lanes”. In: *Transportation Research Part A: Policy and Practice* 33.2 (1999), pp. 139–155 (cit. on p. 1).

[129] Andrew Yao. “How to generate and exchange secrets”. In: *Foundations of Computer Science, 1986., 27th Annual Symposium on*. IEEE. 1986, pp. 162–167 (cit. on pp. 18, 80).

[130] Zhichao Zhu and Guohong Cao. “Applaus: A privacy-preserving location proof updating system for location-based services”. In: *INFOCOM, 2011 Proceedings IEEE*. IEEE. 2011, pp. 1889–1897 (cit. on p. 63).