ORide: A Privacy-Preserving yet Accountable Ride-Hailing Service

Anh Pham[1], Italo Dacosta[1], Guillaume Endignoux[1], Juan Ramón Troncoso-Pastoriza[1]
Kévin Huguenin[2], Jean-Pierre Hubaux[1]
[1]*School of Computer and Communication Sciences (IC), EPFL, Lausanne, Switzerland*
[2]*Faculty of Business and Economics (HEC), UNIL, Lausanne, Switzerland*

Abstract

In recent years, ride-hailing services (RHSs) have become increasingly popular, serving millions of users per day. Such systems, however, raise significant privacy concerns, because service providers are able to track the precise mobility patterns of all riders and drivers. In this paper, we propose ORide (Oblivious Ride), a privacy-preserving RHS based on somewhat-homomorphic encryption with optimizations such as ciphertext packing and transformed processing. With ORide, a service provider can match riders and drivers without learning their identities or location information. ORide offers riders with fairly large anonymity sets (e.g., several thousands), even in sparsely populated areas. In addition, ORide supports key RHS features such as easy payment, reputation scores, accountability, and retrieval of lost items. Using real data-sets that consist of millions of rides, we show that the computational and network overhead introduced by ORide is acceptable. For example, ORide adds only several milliseconds to ride-hailing operations, and the extra driving distance for a driver is less than 0.5 km in more than 75% of the cases evaluated. In short, we show that a RHS can offer strong privacy guarantees to both riders and drivers while maintaining the convenience of its services.

1 Introduction

Ride-hailing services (RHSs), such as Uber and Lyft, enable millions of riders and drivers worldwide to set up rides via their smartphones. Their advantage over traditional taxi services is due to the convenience of their services, e.g., ride requests at the touch of a button, fare estimation, automatic payments, and reputation ratings. Moreover, the accountability provided by RHSs is a key feature for riders and drivers, as it make them feel safer [11, 15]. For instance, in case of a criminal investigation, the RHS provider can offer law-enforcement agencies with the location trace of a particular ride and the identities of the participants.

To offer such services, however, RHSs collect a vast amount of sensitive information that puts at risk the privacy of riders and drivers. First, for each ride, the location traces and rider's and driver's identities are known to the service provider (SP). As a result, the SP, or any entity with access to this data, can infer sensitive information about riders' activities (such as one-night stands [35]), monitor the locations of riders in real-time for entertainment [18], track the whereabouts of their ex-lovers [42], look up trip information of celebrities [25], and even mount revenge attacks against journalists critical of such services [46]. In the case of drivers, there are reports of SPs that track drivers to find if the drivers attended protests [1]. Second, due to the release of drivers' personal identifiable information (PII) early in the ride set-up procedure, an outsider adversary can massively collect drivers' PII [39]. Third, there is evidence that RHS drivers and riders are discriminated based on the racial and/or gender information specified in their profiles [20]. Hence, there is a strong need to provide privacy and anonymity for *both* riders and drivers w.r.t. the SP and each other.

To the best of our knowledge, the only privacy-friendly alternative to current RHSs is PrivateRide, proposed by us [39]. However, this work has some limitations, i.e., it does not provide strong privacy guarantees for riders, and offers less accountability and usability, compared to the current RHSs (see Section 2). Therefore, a mechanism with more robust privacy and accountability guarantees is needed.

We present ORide, a privacy-preserving RHS inspired by PrivateRide; it reuses only one operation from PrivateRide, i.e., the proximity check to prevent drivers' PII from being harvested (see Section 5.4). ORide enables the SP to efficiently match riders and drivers without leaking either their identities or their locations, while providing accountability to deter misbehavior. ORide provides strong privacy for both riders and drivers, i.e., all users in the system are part of large anonymity sets, even if they are in sparsely populated areas. Even in the extreme case of targeted attacks (i.e., a curious SP wants to know the destination of a specific rider given the time and location of her pick-up event [33]), the location privacy of the rider's destination is still guaranteed. For this purpose, ORide relies on state-of-the-art somewhat-homomorphic encryption system [16] (SHE), to which we apply optimizations for ciphertext packing and transformed processing [38], hence enabling a notable boost in performance and a reduction in overhead w.r.t. naive cryptographic solutions.

Accountability and usability are often considered as important as privacy in RHSs [11, 15]; this introduces challenges in resolving the uneasy tension between privacy, accountability and usability. To achieve accountable privacy, ORide enables the SP to revoke, when needed, the anonymity of misbehaving riders or drivers. However, the SP does not have full control over this re-identification operation, i.e., it is able to do it only with the support from the affected party. In addition, to preserve the convenience of the service, ORide supports automatic payment through credit cards and enables riders to contact drivers for lost items. ORide also preserves the reputation-rating operations of current RHSs.

The evaluation of ORide by using real data-sets from NYC taxi cabs [44] shows that, even with strong bit-security of more than 112 bits, ORide introduces acceptable computational and bandwidth costs for riders, drivers and the SP. For example, for each ride request, a rider needs to download only one ciphertext of size 186 KB with a computational overhead of less than ten milliseconds. ORide also provides large anonymity sets for riders at the cost of acceptable bandwidth requirements for the drivers: e.g., for rides in the boroughs of Queens and Bronx, a ride would have an anonymity set of about 26,000, and the drivers are only required to have a data-connection speed of less than 2 Mbps. Moreover, our results show that ORide is scalable, as we considered a request load that is significantly higher than the one in current RHSs, e.g., Uber accounts for only 15% of the ride pick-up requests in NYC [43].

In summary, we make the following contributions:

- *A novel, oblivious, and efficient ride-matching mechanism.* ORide includes a novel protocol based on quantum-resistant SHE to match riders and drivers, without revealing their identities and locations to the SP. We optimize our SHE-based protocol to considerably reduce the bandwidth requirements and the processing overhead, compared to a vanilla SHE-based protocol; and we propose an efficient extension to deal with malicious drivers.

- *The design and prototype of ORide.* ORide supports the matching of riders and drivers, different accountability mechanisms, and it reduces the amount of sensitive information revealed to the SP. In particular, ORide supports functionalities that are often considered also as important as privacy, such as credit-card payment, reputation rating, contacting drivers in case of lost items and traceability in case of criminal activity during a ride.

- *Thorough performance evaluation.* Using real data-sets and robust security parameters (i.e., 112 bits security), we show that ORide provides strong privacy guarantees for riders and drivers. In addition, the computational and network overhead introduced by ORide is practical for riders, drivers and SP. We also show that ORide has a negligible effect on the accuracy of matching riders and drivers compared with current RHSs. The source code of our evaluation is available at [36].

2 Related Work

Researchers have proposed different privacy-enhancing solutions for ride sharing (i.e., car pooling) services [6, 14, 21, 22, 40] and public transportation ticketing systems [8, 26, 31]. However, little work exists in the area of privacy and security for RHSs, probably due to their relative novelty. According to our literature review, the most relevant work in this area is PrivateRide [39].

PrivateRide is the first system to enhance location privacy for riders and protect drivers' information from harvesting attacks while maintaining the convenience of the service. However, it has several limitations that are addressed in this work. First, PrivateRide cannot guarantee the same level of privacy to all riders, because the size of the anonymity set in a particular cloaked area depends on the density of riders in that area. For instance, the anonymity set is smaller for ride requests in areas outside a city center. Also, the tradeoff between the size of a cloaked area and the accuracy of the ride-matching results prevents the use of larger cloaking areas (i.e., to achieve larger anonymity sets). Second, PrivateRide does not protect drivers's privacy, also important [1]. Third, PrivateRide provides limited accountability features to deal with relatively common scenarios such as drivers and riders physically attacking each other (i.e., safety concerns) or items being lost during a ride; for many users, such features can be as important as their privacy. Fourth, PrivateRide's usability is reduced w.r.t. current RHSs because the supported payment mechanism is less convenient (i.e., PrivateRide requires payments with e-cash bought in advance before a ride). Moreover, ride-matching is suboptimal, because the distance between rider and drivers is estimated using the centers of the cloaked areas, instead of exact locations, resulting in additional waiting time for riders.

3 System Model

Our goal is to design a RHS that provides stronger privacy guarantees to both riders and drivers, as well as better or equivalent usability and accountability compared with PrivateRide [39] and current RHSs (e.g., Uber, Lyft, and Easy Taxi). To do so, we assume a system consisting of three parties: riders, drivers and the service provider (SP). We now describe our adversarial and system assumptions.

3.1 Adversarial Assumptions

In our model, riders and drivers are active adversaries. The SP is a passive adversary (i.e., honest-but-curious). We assume that most riders and drivers do not collude with the SP, as drivers are independent contractors rather than SP's employees. The case of a covertly active SP is discussed in Section 7.2. In such a case, we assume that the SP does not provide riders and drivers with malicious apps. This is a reasonable assumption, because such attacks can be detected by third-parties via reverse-engineering or black-box analyses; the risk of public exposure and reputation loss is a strong deterrent against such attacks.

Given that they have been observed in current RHSs (i.e., higher chance of occurring), we focus on the following attacks:

- **(A1)** The riders and drivers might attempt to assault each other [48]; in extreme cases, a driver might attempt to kidnap and/or kill the rider, or vice versa [37,49].
- **(A2)** The SP uses its knowledge about side information about riders and drivers, including their home/work addresses, together with protocol transcripts, to perform *large-scale* inference attacks to profile riders' and drivers' activities [35].
- **(A3)** The SP might attempt to carry out *targeted attacks* on specific riders. That is, besides their home/work addresses, the SP knows the precise pick-up location and time of a specific rider and wants to know the drop-off location and time of this ride, or vice versa [25,33,46].

3.2 Design Goals

The goal of ORide is to defend against the attacks listed in Section 3.1, and to offer the same level of accountability and usability as current RHSs, as follows.

- Riders and drivers are held accountable for their behaviors during their rides, i.e., the SP is able to identify misbehaving riders or drivers when needed, e.g., if one party attacks the other. However, the SP is able to identify the misbehaving party only with support from the affected party (or her trusted contacts, see Section 6.)
- The system preserves the convenience and usability properties offered by current RHSs, such as payment through credit cards and reputation rating. In addition, once a rider is matched with a driver, she can track the location of the driver approaching the pick-up location, and they can contact each other to coordinate the pick-up. The system also enables riders to contact drivers of their past rides to find lost items.

3.3 System Assumptions

We assume that the metadata of the network and lower communication layers cannot be used to identify riders and drivers or to link their activities. Such an assumption is reasonable because, in most cases, the smartphones of drivers and riders do not have fixed public IP addresses; they access the Internet via a NAT gateway offered by their cellular provider. If needed, a VPN proxy or Tor could be used to hide network identifiers.

In addition, we assume that, besides localization capabilities, the rider's and driver's smartphones support peer-to-peer wireless communication, e.g., Bluetooth and WiFi Direct. Also, for all location-based computations, the apps use a coordinate system such that the Euclidean distances correspond to the great-circle distances, e.g., by using map-projection systems for local areas such as UTM [47] to convert a pair of (latitude, longitude) to planar coordinates (x, y). Moreover, drivers use a navigation app that does not leak their locations to the SP. This can be done by using a third-party navigation/traffic app (e.g., Google Maps, TomTom, Garmin) or pre-fetching the map of their operating areas (e.g., a city) and using the navigation app in off-line mode.

3.4 Notation

Throughout the rest of this work, we denote polynomials and scalar values with lowercase letters, variables and rings with uppercase letters, and vectors with boldface letters. $\lfloor . \rceil$ denotes rounding to the nearest integer. A polynomial of degree $(d-1)$ will be interchangeably denoted as $a = \sum_{i=0}^{d-1} a_i X^i$ or in its vector form $\boldsymbol{a}$ when there is no ambiguity. The used symbols and terms are summarized in Table 1.

4 Oblivious Ride-Matching Protocol

One of the challenges in privacy-preserving RHSs is how to efficiently match ride requests to ride offers without revealing the riders' and drivers' locations to each other and to the SP. For this, ORide relies on somewhat-homomorphic encryption (see Section 4.1) where the riders and drivers send their encrypted locations to the SP, from which the SP computes the encrypted squared Euclidean distances between them. We detail this in the following sections. For details about other cryptographic primitives used in ORide, see Appendix A.3.

Notation	Description
k_s	Ephemeral private key
$\boldsymbol{k}_p$	Ephemeral public key
$cert_X$	Public-key certificate of X
loc_X	Planar coordinates of X, $loc_X = (x_X, y_X)$
n	Number of available drivers
d	Degree of the polynomial
dt	Deposit token
r_{dt}	A random number to create a deposit token
z	A geographical zone
$sig_X\{m\}$	Message m and signature of X on m
$Bsig_{SP}(m)$	Blind signature of the SP on message m
$sig_{R-D}\{m\}$	$sig_D\{sig_R\{m\}\}$

Table 1: Table of notations

4.1 Somewhat-Homomorphic Encryption

Somewhat-Homomorphic Encryption (SHE) is a special kind of malleable encryption that allows a certain number of operations (additions and multiplications) over ciphertexts, without the need to decrypt them first. All SHE cryptosystems present semantic security, i.e., it is not (computationally) possible to know if two different encryptions conceal the same plaintext. Therefore, it is possible for a party without the private key (in our case, the SP), to operate on the ciphertexts produced by riders and drivers, without obtaining any information about the plaintext values. Additionally, we choose one of the most recent and efficient SHE schemes based on ideal lattices, the FV scheme [16]. This scheme relies on the hardness of the Ring Learning with Errors (RLWE) problem [29]. Note that whenever working with cryptosystems based on finite rings, we usually work with integer numbers, hence, from here on, we will assume that all inputs are adequately quantized as integers. Here, we briefly describe the main functions of the FV scheme.

For plaintext elements in a polynomial quotient ring $m \in R_t = \mathbb{Z}_t[X]/(X^d+1)$ and ciphertext elements in $R_q = \mathbb{Z}_q[X]/(X^d+1)$, where q and t are positive integers $q > t$ defining the upperbound of the ciphertext and plaintext coefficients, respectively. Let $\Delta = \lfloor q/t \rfloor$ and χ_k, χ_n be two *short* noise random distributions in R_q, the FV encryption of a message $m \in R_t$ with secret key $k_s = s \sim \chi_k$ and public key $\boldsymbol{k}_p = [p_0, p_1] = [(-a \cdot s + e), a] \in R_q^2$, with e drawn from χ_n and a randomly chosen in R_q, generated by FV.GenKeys, results in a vector expressed as

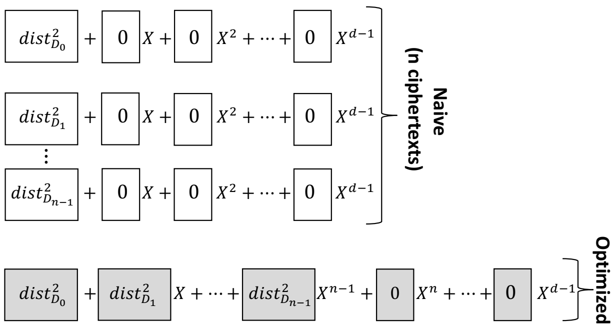
$$\boldsymbol{c} = \mathsf{FV.Enc}(\boldsymbol{k}_p, m) = [p_0 \cdot u + e_1 + \Delta \cdot m,\ p_1 \cdot u + e_2], \quad (1)$$

where u is drawn from χ_k, and e_1, e_2 are short random polynomials from the error distribution χ_n. All operations are in R_q.

Decryption of a ciphertext $\boldsymbol{c} = [c_0, c_1]$ works as

$$m = \mathsf{FV.Dec}(k_s, \boldsymbol{c}) = (\lfloor t \cdot [c_0 + c_1 \cdot s \bmod q]/q \rceil) \bmod t.$$

The scheme enables us to seamlessly add (FV.Add), subtract (FV.Sub) and multiply (FV.Mul) two encryp-



$dist^2_{D_0}$ + 0 X + 0 X^2 + ⋯ + 0 X^{d-1}
$dist^2_{D_1}$ + 0 X + 0 X^2 + ⋯ + 0 X^{d-1}
$dist^2_{D_{n-1}}$ + 0 X + 0 X^2 + ⋯ + 0 X^{d-1}
Naive (n ciphertexts)
$dist^2_{D_0}$ + $dist^2_{D_1}$ X + ⋯ + $dist^2_{D_{n-1}}$ X^{n-1} + 0 X^n + ⋯ + 0 X^{d-1}
Optimized

Figure 1: Our optimized ride-matching approach enables the SP to send to the rider a single ciphertext containing all the squared distances ($dist^2_{D_i}$) between the rider and available drivers as opposed to one ciphertext per driver (naive approach).

tions to obtain the encryption of the added, subtracted, and multiplied plaintexts respectively; multiplications consider the encryptions as polynomials in v: $[c_0, c_1] \rightarrow c_o + c_1 \cdot v$, such that the product between $\boldsymbol{c}$ and $\boldsymbol{c}'$ is evaluated as: $[c_0, c_1] \cdot [c'_0, c'_1] \rightarrow c_0 \cdot c'_0 + (c_0 \cdot c'_1 + c_1 \cdot c'_0)v + c_1 \cdot c'_1 \cdot v^2 \rightarrow [c''_0, c''_1, c''_2]$, which results in a ciphertext in R_q^3, with one extra polynomial. It is possible to recover a fresh-like encryption with two polynomials by employing a relinearization primitive, which requires the usage of a matrix (relinearization key) composed of encrypted pieces of the secret key (we refer the reader to [16] for further details).

4.2 Naive Approach

SHE can be applied to the ride-matching problem in RHSs as follows. When a rider wants to make a ride request, she generates an ephemeral FV public/private key-pair together with a relinearization key. She uses the public key to encrypt her planar coordinates and obtains their encrypted forms. She then informs the SP about the zone of her pick-up location, the public and relinearization keys and her encrypted planar coordinates. When this information arrives at the SP, the SP broadcasts the public key to all drivers available in that zone. Each driver uses the public key to encrypt their planar coordinates and sends them to the SP. The SP computes, based on their encrypted coordinates, the encrypted distances between the rider and the drivers, and it returns the encrypted distances to the rider, from which the rider can decrypt and select the best match, e.g., the driver who is the closest to her pick-up location.

However, due to the high ciphertext expansion, a naive use of SHE would incur impractical computational and bandwidth costs for the riders and the SP. Furthermore, for each ride request, the SP would need to separately compute the encrypted distances between the rider and each of the drivers: For n drivers, this would mean n distance calculations between encrypted polynomials of

d coefficients each, and n ciphertext distances returned to the rider. This would incur an unfeasible overhead in terms of computations for the SP, consequently delaying the ride-matching for the rider and a considerable bandwidth overhead at the rider-SP link, e.g., hundreds of MBs if the system has several thousand drivers (see Section 9.3).

4.3 Optimized Approach

We propose two optimizations: ciphertext packing and transform processing, to enable the SP to operate on d elements of $\mathbb{Z}_t$ packed as a polynomial in R_t in a *single* ciphertext, such that each encrypted operation affects all the coefficients in parallel (see Fig. 1). When the rider decrypts this ciphertext, she can recover these d values by looking at all the coefficients. From here on, we assume that $d \geq n$, which will usually be the case due to the security bounds on d (see Section 8); in other cases, $\lceil n/d \rceil$ encryptions can be used to pack the whole set of distances analogously.

First, ciphertext packing enables the SP to pack n ciphertext distances into one ciphertext, hence reducing the bandwidth overhead, but this is not enough for our goal. As we show in Section 5.4, we use all the n packed encrypted planar coordinates from the drivers independently of each other to calculate all the distances homomorphically in the same encrypted operation, so we need coefficient-wise homomorphic operations. While polynomial additions and subtractions are naturally coefficient-wise, polynomial multiplication in R_t (and its homomorphic counterpart in R_q) is a *convolution product* of the coefficients. A well-known method for transforming convolution products into coefficient-wise products (and vice-versa) in polynomial rings is the *Number-Theoretic Transform* (NTT) [38], a Fourier transform specialized for finite fields. This transform is commonly used in the ciphertext space to speed up polynomial multiplications that are then implemented as coefficient-wise products. More details about NTT can be found in Appendix A.3.

In our case, for the second optimization, in order that products in the encrypted domain be translated into coefficient-wise products in the plaintext domain, we apply an inverse-NTT to plaintexts before encryption and an NTT after decryption. The NTT does not affect additions and subtractions because it is linear. We note that the NTT exists only for certain values of d and t, in particular when t is a prime and d divides $t-1$. To make operations in $\mathbb{Z}_t$ simulate operations in $\mathbb{N}$ on our values, we choose $d = 2^l$ as a power of two and t as a sufficiently large Proth prime (of the form $k2^l + 1$, see [38]) such that all squared-Euclidean distances are less than t. As a result, we improve on both the bandwidth and the computation overhead.

Moreover, due to the low degree of the evaluated operations (squared Euclidean distances), we avoid the use of re-linearizations at the SP, which (a) reduces the need to generate and to send the relinearization key from the rider to the SP, (b) reduces the noise inside the encryptions, and (c) enables more efficient operations at the SP, at the cost of one extra polynomial to represent the encrypted distance returned to the rider.

5 ORide

In this section, we present our solution, called ORide (Oblivious Ride). We begin with an overview of the system and then detail ORide operations.

5.1 ORide Overview

ORide provides strong location privacy and anonymity for riders and drivers while still guaranteeing service accountability, secure payment and reputation rating operations. For this purpose, the riders and the drivers must possess *ride prerequisites* (Section 5.2), including anonymous credentials (ACs), deposit tokens, and digital certificates issued by the SP. To participate in the system, both riders and drivers create anonymous sessions by *logging in to the service* (Section 5.3) with their respective ACs. Drivers periodically report to the SP the geographical zones where they are located. These zones are defined by the SP to balance the network load in the system and the size of the anonymity set of the zones (Section 9.4). Note that, in contrast to PrivateRide, expanding the size of a zone in ORide *does not* affect the performance of the ride-matching and fare-calculation operations (Section 5.4).

When a rider initiates a ride request, the SP, the rider and drivers are involved in a *ride set-up* procedure (Section 5.4) that matches the rider to a driver. In addition, as in current RHSs, the rider and the driver agree on the fare based on the estimated distance and duration of the ride [34,41]. Some random time after the fare agreement, they terminate their anonymous sessions. When the ride is completed, the driver creates a new anonymous session and notifies the SP that she is available again. Note that drop-off times and locations are not reported to the SP. Moreover, some time after the ride finishes, i.e., at the end of the day, the rider and driver perform *ride-payment and reputation-rating* operations (Section 5.5).

5.2 Ride Prerequisites

Digital certificates. We assume each rider and driver has a digital certificate denoted as $cert_R$ or $cert_D$, issued by the SP at registration time. Each certificate contains a public key and a randomly generated ID. The SP can use this random ID to find the real identity of the certificate holder. Note that the digital certificates are not used by the riders and drivers to log in to the service, and they are not revealed to the SP during a ride. They are used by the riders and drivers to identify each other during the ride as part of ORide's accountability mechanism (Section 6).

Anonymous credentials. ORide relies on Anonymous Credentials Light (ACL) [9], a linkable anonymous credential system, i.e., a user should use an AC only once to avoid her transactions from being linkable. To use the service anonymously, each user (rider or driver) requests ACs in advance from the SP, using their digital certificate. Hereafter, we denote the anonymous credential for a user X as AC_X, where X is R for riders or D for drivers. Each AC_X contains the average reputation score rep_X, an expiration date exp_X, and the secret key sk_X associated with the public key pub_X in the digital certificate of the AC holder. To differentiate between riders and drivers in the system, an AC also contains a role attribute $role_X$, i.e., $role_X = 1$ if $X = D$, and $role_X = 0$ if $X = R$.

Note that to prevent the SP from de-anonymizing users by correlating the time an AC is issued with the time it is used, or by relying on the AC's expiration date, the user's app could automatically request ACs from the SP at a certain time (e.g., at midnight), and the expiration date is coarse-grained, e.g., all ACs issued in a day expire at the end of that day. The reputation scores cannot be used by the SP to de-anonymize the users, as they are never shown to the SP during the rides. Furthermore, to prevent users from abusing the system, the SP defines a threshold on the number of ACs a rider or driver can acquire per day.

Deposit token. Each rider is required to possess a *deposit token* and give it to the SP at the beginning of a ride. In case of misbehavior, the token is not returned to the rider. A deposit token, denoted as dt, is worth a fixed amount of money defined by the SP. It is a random number generated by the rider, blindly signed by the SP (by using blind-signature schemes e.g., [13]) such that the SP is not able to link a token it issued and a token spent by a rider. A rider deposits a token to the SP in the beginning of the ride, and she is issued a new token by the SP after the ride payment is successfully completed. Note that the driver is not required to make a deposit because, during the ride set-up operation, the rider and driver exchange their digital certificates with each other. Consequently, if the driver misbehaves, the SP can identify the driver by collaborating with the rider. We discuss this in more detail in Section 5.6.

5.3 Log in to the Service

To use the service, the rider and the driver need to create anonymous sessions to the SP: to do so, they use their anonymous credentials AC_R and AC_D, respectively.

Rider. The rider sends to the SP the rider-role $role_R$ and the expiry date exp_R stated in her AC_R. In addition, she proves to the SP that the claimed values are correct and that, in a zero-knowledge fashion, she knows the secret key sk_R tied to the AC_R.

Driver. Similarly to the rider, by using her AC_D, the driver follows the same aforementioned procedure to anonymously log in to the service.

The SP assigns a one-time session ID to each anonymous session, to keep track of that session for coordination. For the sake of simple exposition, hereafter, we exclude this one-time session ID from messages exchanged between the rider/driver and the SP.

5.4 Ride Set-up

When a rider requests a ride, the operations performed by the rider, the drivers and the SP are as follows (see Fig. 2).

1. The rider generates an ephemeral FV public/private key pair, denoted as $(\boldsymbol{k}_p, k_s)$. She first computes the polynomial representations of the coordinates $p_{x_R} = \sum_{i=0}^{d-1} x_R X^i$ and $p_{y_R} = \sum_{i=0}^{d-1} y_R X^i$. She then applies the inverse-NTT on the polynomials and uses $\boldsymbol{k}_p$ to encrypt these values: $\boldsymbol{c}_{x_R} = \mathsf{FV.Enc}(\boldsymbol{k}_p, \mathtt{NTT}^{-1}(p_{x_R}))$ and similarly for $\boldsymbol{c}_{y_R}$. She then sends the zone of her pick-up location (denoted as z), deposit token dt, $\boldsymbol{k}_p$, $\boldsymbol{c}_{x_R}$ and $\boldsymbol{c}_{y_R}$ to the SP.

2. The SP checks the validity of the deposit token, i.e., it has not been used before. If the token is valid, the SP adds it to the list of used tokens. It then sends to each driver in zone z a different randomly permuted index $0 \leq i < n$ and the public key $\boldsymbol{k}_p$.

3. The i-th driver encodes her coordinates in the i-th coefficient: $q^i_{x_D} = x_{D_i} X^i$ and $q^i_{y_D} = y_{D_i} X^i$. Similarly to the rider, she applies the inverse-NTT, encrypts these values and sends them to the SP: $\boldsymbol{c}^i_{x_D} = \mathsf{FV.Enc}(\boldsymbol{k}_p, \mathtt{NTT}^{-1}(q^i_{x_D}))$ and analogously for $\boldsymbol{c}^i_{y_D}$.

4. The SP sums all drivers' ciphertexts by using the homomorphic property of the cryptosystem to pack them together: $\boldsymbol{c}_{x_D} = \sum_{i=0}^{n-1} \boldsymbol{c}^i_{x_D}$ and similarly for $\boldsymbol{c}_{y_D}$. It then homomorphically computes the n packed squared values of the Euclidean distances between the n drivers and the rider in parallel, due to the packing $\boldsymbol{c}_{dist} = (\boldsymbol{c}_{x_R} - \boldsymbol{c}_{x_D})^2 + (\boldsymbol{c}_{y_R} - \boldsymbol{c}_{y_D})^2$, and it sends the result to the rider (see Fig. 1).

5. The rider decrypts the ciphertext and applies the NTT to obtain a squared distance in each coefficient: $\boldsymbol{dist} = \mathtt{NTT}(\mathsf{FV.Dec}(\boldsymbol{k}_s, \boldsymbol{c}_{dist}))$. Then, she selects the driver with the smallest squared distance.

6. The SP notifies the selected driver. If she declines the offer, the SP asks the rider to select a different driver; it repeats this operation, until one driver accepts. The

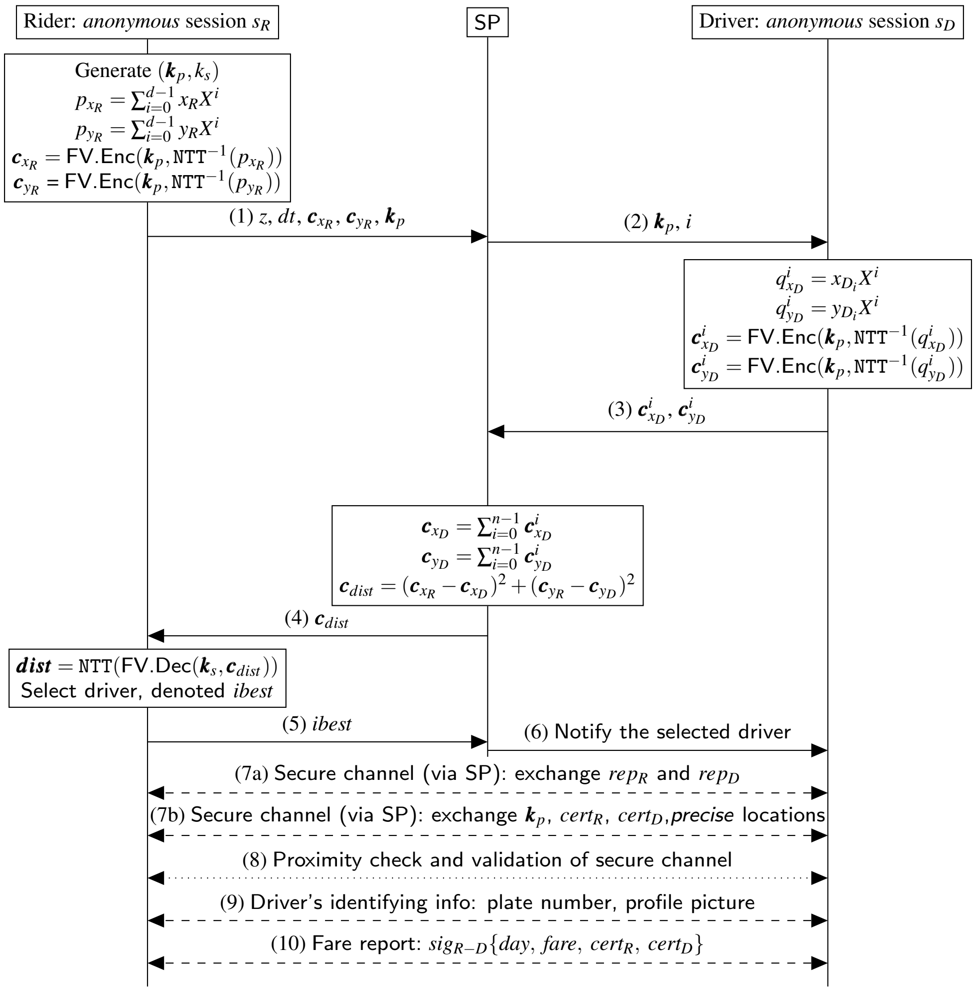

Rider: *anonymous* session s_R
SP
Driver: *anonymous* session s_D
Generate $(\boldsymbol{k}_p, k_s)$
$p_{x_R} = \sum_{i=0}^{d-1} x_R X^i$
$p_{y_R} = \sum_{i=0}^{d-1} y_R X^i$
$\boldsymbol{c}_{x_R} = \mathsf{FV.Enc}(\boldsymbol{k}_p, \mathtt{NTT}^{-1}(p_{x_R}))$
$\boldsymbol{c}_{y_R} = \mathsf{FV.Enc}(\boldsymbol{k}_p, \mathtt{NTT}^{-1}(p_{y_R}))$
(1) z, dt, $\boldsymbol{c}_{x_R}$, $\boldsymbol{c}_{y_R}$, $\boldsymbol{k}_p$
(2) $\boldsymbol{k}_p$, i
$q^i_{x_D} = x_{D_i} X^i$
$q^i_{y_D} = y_{D_i} X^i$
$\boldsymbol{c}^i_{x_D} = \mathsf{FV.Enc}(\boldsymbol{k}_p, \mathtt{NTT}^{-1}(q^i_{x_D}))$
$\boldsymbol{c}^i_{y_D} = \mathsf{FV.Enc}(\boldsymbol{k}_p, \mathtt{NTT}^{-1}(q^i_{y_D}))$
(3) $\boldsymbol{c}^i_{x_D}$, $\boldsymbol{c}^i_{y_D}$
$\boldsymbol{c}_{x_D} = \sum_{i=0}^{n-1} \boldsymbol{c}^i_{x_D}$
$\boldsymbol{c}_{y_D} = \sum_{i=0}^{n-1} \boldsymbol{c}^i_{y_D}$
$\boldsymbol{c}_{dist} = (\boldsymbol{c}_{x_R} - \boldsymbol{c}_{x_D})^2 + (\boldsymbol{c}_{y_R} - \boldsymbol{c}_{y_D})^2$
(4) $\boldsymbol{c}_{dist}$
$\boldsymbol{dist} = \mathtt{NTT}(\mathsf{FV.Dec}(\boldsymbol{k}_s, \boldsymbol{c}_{dist}))$
Select driver, denoted *ibest*
(5) *ibest*
(6) Notify the selected driver
(7a) Secure channel (via SP): exchange rep_R and rep_D
(7b) Secure channel (via SP): exchange $\boldsymbol{k}_p$, $cert_R$, $cert_D$,*precise* locations
(8) Proximity check and validation of secure channel
(9) Driver's identifying info: plate number, profile picture
(10) Fare report: $sig_{R-D}\{day, fare, cert_R, cert_D\}$

Figure 2: ORide ride setup protocol. The dashed arrows represent the secure channel (via the SP), and the dotted arrows represent the proximity channel.

SP confirms with the rider and the driver that they have been assigned to each other.

7a. The rider and the driver establish a secure channel via the SP, e.g., using the unauthenticated Diffie-Hellman protocol, to exchange data that should not be observed by the SP.[1] From the information used to derive the secret key of the secure channel, the rider and the driver compute a shared secret *pairing PIN*. This *pairing PIN* will be used for the proximity-check operation in Step 8.

With this secure channel, the rider and the driver reveal their reputation scores to each other. The trustworthiness of the revealed values is proved by showing that they are indeed the values in the rider's and driver's ACs. If the rider's reputation is too low, the driver can abort the protocol at this step. Likewise, the rider can select another driver, by using the list of cleartext squared Euclidean distances she obtained in Step 5.

7b. Via the secure channel, the rider and the driver exchange their precise locations (i.e., loc_R and loc_D, respectively). In addition, they exchange their digital certificates (i.e., $cert_R$ and $cert_D$) with each other. This provides accountability for the rider and driver (see Section 6). Also, the driver can reveal to the rider the public key $\boldsymbol{k}_p$ that she used to encrypt her locations; this helps to detect possible man-in-the-middle attacks at Step 2 of the protocol by the SP.

The driver drives from her current location loc_D to the pick-up location loc_R, using an off-line navigation app or a third-party navigation app (such as Google Maps or TomTom). She sends, in real time via the secure channel, her precise locations to the rider, thus the rider can track the movements of the car. Also, at this point, the rider and the driver can call or message each other through their ride-hailing apps, if needed.

[1]Detection of possible man-in-the-middle attacks by the SP is done in Step 8. Note that this check is needed only if the SP is an active adversary.

8. When the rider and the driver are in proximity, the driver performs a proximity check to verify the *physical presence* of the rider before releasing her identifying information: they use a short-range wireless technology (e.g., Bluetooth or WiFi Direct) to set up a proximity channel using the *pairing PIN*. If the channel is successfully established, the driver can verify that the rider is in her proximity. This is similar to the approach proposed in [39] to prevent drivers' PII from being harvested. If this step fails, the driver can decide to abort the protocol. Also, via the proximity channel, the rider and the driver can check whether the secure channel (established at Step 7a) was tampered with by the SP.

9. The driver releases her identifying information to the rider, including her vehicle's license plate number and her profile picture. This information helps the rider to identify the driver and her car and to prevent certain threats, e.g., fake drivers [45]. Therefore, it is needed when the rider is about to enter the car, i.e., the required communication distance between the phones of the rider and the driver is small (e.g., several meters).

10. The rider and the driver create a *fare report*. A fare report is a token generated by the rider and driver; and at the end of the day, the driver deposits it to the SP to get paid (Section 5.5). A fare report is created as follows. The rider sends her drop-off location to the driver via the secure channel, they agree on the path, and based on the estimated path, they compute the fare. The rider and driver then sign a message consisting of the day of the ride, the fare and their certificates, i.e., *fare report* = $sig_{R-D}\{day, fare, cert_R, cert_D\}$, using the private keys associated with their $cert_R$ and $cert_D$. Note that this upfront-fare method has been implemented in current RHSs, such as in Uber [34] and in Lyft [41]. Once the driver receives the fare report from the rider, the ride begins. The rider's and driver's app do not report any information to the SP at this step and during the ride. Also, to prevent the SP from inferring the starting time of the ride based on the interactions between the rider and the driver over the secure channel, the rider and driver can randomly send dummy information to each other through the secure channel. Also, some random time after the fare-report agreement, they terminate their anonymous sessions.

Intuitively, because the distances between the rider and drivers are computed based on their (encrypted) *precise* locations, expanding the size of the zone will not result in negative effects on the performance of the ride-matching and fare-calculation operations. In addition, with ciphertext packing, we reduce by a factor of n the communication between the SP and the rider. However, if the drivers are malicious, they could corrupt the inputs from other drivers. Furthermore, note that in Step 1 of the protocol, any valid rider can generate an ephemeral public/private key pair. Consequently, if the SP is an active attacker, it could track the locations of the drivers, thus indirectly track the locations of the riders. We discuss solutions to these potential issues in Section 7.

5.5 Ride Payment and Reputation Rating

When the car arrives at the drop-off location, the driver creates a new anonymous session to the SP. This enables her to receive ride-request broadcasts from the SP. Note that the driver *does not* report to the SP that the ride is completed.

At the end of the day, the driver sends to the SP the fare report $sig_{R-D}\{day, fare, cert_R, cert_D\}$ she received during the ride set-up operation (step 10, Section 5.4). The SP checks the correctness of the rider certificate $cert_R$ in the fare report and the correctness of the signature. If they are valid, the SP charges the rider according to her payment method, e.g., credit card. It then subtracts the service fee, and deposits the remainder to the driver. The SP then notifies the rider about the payment and that a new deposit token is available. The rider generates a random number r_{dt}, blinds it to r'_{dt}, and sends r'_{dt} to the SP. The SP signs r'_{dt} (i.e., $dt' = sig_{SP}\{r'_{dt}\}$, and it sends this blind signature to the rider's account. The rider unblinds the signature to obtain the deposit token which she can use for her next ride. Note that this procedure can be done automatically by the rider's app.

Once the payment is successfully completed, the rider and driver can rate the reputation of each other, similarly to current RHSs. They can log in to the service with their real credentials and provide the reputation score for the party whom they rode with.

Note that ORide preserves the payment and reputation-rating operations of the current RHSs. That is, unlike PrivateRide, it does not require the rider to purchase e-cash in advance, and it does not require the rider and the driver to generate and keep extra cryptographic tokens for the reputation-rating operation. Also, ORide does not require the rider and the driver to hide their identifying information to the SP during the payment and reputation-rating operations, because both the rider and driver are anonymous during the ride. However, it is important to note that, in order to prevent the SP from de-anonymizing the rider and the driver by correlating the time that a fare report is deposited with the drop-off event of the ride, the payment operation should *not* occur immediately after the ride, e.g., the drivers deposit the fare reports to the SP at the end of the day.

5.6 Ride Cancellation

As in current RHSs, a rider or a driver can cancel a ride at any time before or during the ride. This, however, is discouraged by the SP, because it can lead to malicious behaviors: For example, once a rider and a driver are assigned to each other by the SP, they meet at the pick-up location and start the ride as normal; but, to avoid the service fee, the rider or the driver can send a cancellation notification to the SP. Therefore, similarly to current RHSs, if a rider or a driver cancels a ride a certain amount of time after the ride request, they should be penalized by the SP, e.g., their reputation scores are lowered or fees are charged [12].

In ORide, when a rider cancels a ride, the SP can offer her two options: to lose her *deposit token* (i.e., pay a penalty) or to reveal her $cert_R$ and have her reputation score lowered. If a driver cancels a ride, the SP can ask the rider to reveal the $cert_D$, from which the SP can identify and penalize the driver according to its policy.

6 Accountability

In this section, we discuss the accountability goals (mentioned in Section 3.2) of ORide. This includes audit trail mechanisms against the attack **A1** in Section 3.1 and additional features such as retrieval of lost items, assurance of payment, and integrity of the reputation-rating operation. Attacks **A2** and **A3** are discussed in Section 8.

(A1) Accountability. ORide enables the rider and the driver to exchange, during the ride set-up procedure, their digital certificates, i.e., $cert_R$ and $cert_D$, respectively, and the fare report. This provides accountability for riders and drivers, i.e., an affected party can report to the SP the digital certificate of the attacker and the fare report, from which the SP can identify the attack to charge her a fee, lower her reputation and/or support legal action. However, the SP is only able to identify the attacker with support from the affected party. Likewise, the affected party cannot obtain the real identity of the attacker without support from the SP, because the certificates $cert_R$ and $cert_D$ contain only the pseudonyms and only the SP knows the mapping between the pseudonyms and the real identities of the certificate owners.

ORide enables the rider to share with her trusted peers the driver's certificate $cert_D$ and the fare report, via out-of-band channels such as messaging apps, or a plug-in in her rider app. Similarly, during the ride, via out-of-band channels, she can share her GPS trace with her friends using (k,l) threshold secret sharing [17], i.e., each GPS location point is split into l parts so that any k out of l parts reconstruct the original coordinate. Likewise, the driver can follow the same mechanism. Such information can be shared with law enforcement in case riders or drivers disappear (e.g., kidnapping), as in current services. This is similar to the approach used in personal safety apps, such as Google Trusted Contacts [23].

ORide guarantees assurance of payment. A rider cannot avoid paying the fare of a ride, because the fare report contains her digital certificate $cert_R$ and the day of the ride. As the rider and driver agree on the fare and both sign it before the ride, they cannot subsequently increase or decrease this fare. However, they might collude to underpay the service fee to the SP, by agreeing on a small fare and paying the difference in cash. Yet in this case, ORide offers the same guarantees as current RHSs, because riders can already request a small ride through the application and then pay in cash for a longer ride once they have met the driver. In future work, we will explore mechanisms to protect against such attacks.

Moreover, the bilateral rating system enables the SP to ban abusive riders and drivers from the service. A rider or driver cannot claim a better reputation for herself, because the proof for attributes in her AC will not be correct w.r.t. her falsely claimed reputation. They also cannot arbitrarily rate the reputation of each other, because a payment record is needed (the deposit of a *fare report*). In addition, as discussed in Section 5.6, similarly to current RHSs, ORide enables the SP to hold riders and drivers accountable for ride cancellations.

SP incentives. From an economic perspective, ride-hailing service SPs would have incentives to deploy ORide because it provides privacy and security for the riders and still preserves their business models (i.e., the SP can still charge a commission for each ride). In order to monetize ride data, the SPs can provide a discount for riders if they reveal (part of) their GPS traces. In addition, privacy and security for RHSs could be required by law and legislation, and ORide shows that it is technically possible to achieve a strong level of protection. As such, this work lays the foundation for the design of a privacy-preserving and secure RHSs.

Additional features. Similarly to current RHSs, ORide enables the riders to retrieve lost items (i.e., items forgotten in the car), as drivers' certificates $cert_D$ and car information are provided during the ride set-up procedure. As discussed earlier, the riders can share $cert_D$ with their friends, hence even if the riders lose their phones, they can still be able to retrieve the $cert_D$ from their friends and to then contact the driver (as in current RHSs). Moreover, due to the secure channel established between the rider and the driver, the rider can still track the driver trajectory while waiting at her pick-up location, or they can contact each other (e.g., messaging or calling).

7 Protecting against Malicious Behaviors

In this section, we describe how the protocol presented in Section 5 can be extended to defend against malicious drivers and a covertly active SP.

7.1 Malicious Drivers: Masking

As mentioned in Section 5.4, if a driver behaves maliciously, she could encrypt non-zero values in the slots other than her allotted one, thus corrupting the inputs from other drivers. Our protocol can cope with this malicious behavior by adding one extra step in which the SP homomorphically multiplies each driver ciphertext by a mask $m_i = \mathtt{NTT}^{-1}(X^i)$ for the driver's index i (see notations from Section 5.4), which preserves only the contents in the allocated slot. However, because the mask does not hold any sensitive information and it is known by the SP, a naive homomorphic multiplication with an encrypted mask would incur an unjustified overhead. Therefore, we propose, instead, a more efficient multiplication operation, denoted $\star$, as follows.

Given a ciphertext $\boldsymbol{c} = [c_0, c_1] \in R_q^2$ corresponding to a plaintext $m \in R_t$, and a mask $m_i \in R_t$, we want to obtain a ciphertext $\boldsymbol{c}' = \boldsymbol{c} \star m_i$ corresponding to the masked plaintext $\mathsf{FV.Dec}(k_s, \boldsymbol{c}) \cdot m_i$. Here, m_i can be thought of as its own noiseless and unscaled encryption (Equation (1) on page 4, evaluated for $u, e_1, e_2 = 0$, and no scale Δ), being a vector in R_q^2 with only one non-zero component ($[m_i, 0] \in R_q^2$). Therefore, the product results

$$\boldsymbol{c} \star m_i = [c_0 \cdot m_i, c_1 \cdot m_i].$$

The $\star$ operation consists of two polynomial multiplications, it avoids encryption of m_i, halves the number of products w.r.t. an encrypted homomorphic multiplication, and keeps the cipher size from growing after the product, thus considerably improving the performance of this operation.

In any case, this precaution is only needed in case the drivers are malicious; and random checks on their locations can be implemented instead if the drivers are just covertly active (i.e., they refrain from cheating if there is a negligible chance of being caught in the act).

7.2 Covertly Active SP

If the SP is an active attacker, it might attempt to perform a man-in-the-middle (MITM) attack at Step 2 of the ride set-up protocol (Section 5.4) by replacing the public key $\boldsymbol{k}_p$. However, this can be detected because the driver can share the key she received with the rider through the secure channel. If the rider detects that this key is different from the one she originally sent to the SP, then a MITM attack must have happened. The SP might also attempt to tamper with the set-up of the secure channel (Step 7a, Fig. 2). However, this can be detected because via the proximity channel, the rider and the driver can compare with each other the inputs that they received from the SP during the set-up protocol.

As mentioned in Section 5.4, any valid rider can generate an ephemeral key to make a ride request. As the SP issues credentials for riders and drivers, it can impersonate a rider or a driver in its own system. If the SP *continuously* impersonates a rider, it could learn the drivers' locations from which it could learn the coarse-grained pick-up locations of the riders. In other words, if a rider chooses the driver who is the closest to her pick-up location, the SP would know that she is in the Voronoi cell of her selected driver [2]. Next, we present a mechanism for deterring this attack. We note that the attack is not trivial, due to the high dynamics of the system, i.e., drivers can arbitrarily go on-line and off-line anytime. The SP would not have strong incentives to perform this attack, because it would add computational and bandwidth overhead to the service, thus negatively affecting the productivity of the service itself. To deter this attack, we introduce the notion of *Proof-of-Ride* (PoR), defined and used as explained below. An illustration of the protocol with PoR is shown in Appendix A.1.

A PoR is a random number *rand* generated by the rider, signed by the driver by using the secret key associated with her $cert_D$, and then *blindly* signed by the SP by using blind-signature schemes such as [13], i.e., PoR $= Bsig_{SP}\{sig_D\{rand\}\}$. It is used to prove to the drivers that the rider is real, i.e., she did a ride in the past. When a rider makes a ride request, she has to provide in her ride request a PoR, the $cert_D$ and the random number *rand* used in the PoR. A PoR can be used only once. For the first ride, PoR $= cert_R$.

To prevent the SP from creating its own $cert_R$ and $cert_D$ in order to create its own fake PoR, the SP has to provide a public bulletin board such as certificate transparency [27]: the SP maintains and publishes a publicly auditable and append-only log of all rider and driver certificates it has issued and revoked. Whenever a driver receives a PoR, she can check whether the rider's certificate $cert_R$ (in the case of the first ride), or the driver's certificate indicated in the PoR, is in the list of certificates published by the SP. In this way, if the SP internally creates fake accounts, it can be detected by auditing authorities, similarly to the cases of companies opening fake user accounts [10]. Similarly, to prevent a rider from double-spending a PoR, and the SP from reusing a valid PoR to perform the aforementioned active attack, the SP maintains and publishes an append-only logs of PoRs that have been spent, or cancelled (due to ride cancellation).

Note that PoR could create a point of linkability, i.e., the SP is able to know that the rider is in the set of identities indicated in the fare reports deposited by a specific driver. This can be easily prevented by using anonymous-reputation and anonymous-payment systems (e.g., e-cash), as used in the PrivateRide system [39].

	Identities	Pick-up loc.	Pick-up time	Drop-off loc.	Drop-off time	Loc. trace	Fare
Current RHSs	Rider, Driver	Precise	Precise	Precise	Precise	Full	Yes
PrivateRide	Driver	Zone	Obfuscated	Zone	Obfuscated	Partial	Yes
ORide	N/A	Zone	Obfuscated	N/A	N/A	N/A	N/A

Table 2: Information observed by the SP during ride set-up procedure w.r.t. different RHS designs. Note that the zone in ORide is larger than the zone in PrivateRide without affecting the ride-matching optimality (see Section 9.4). Also note that, the payment operation in ORide reveals some information about the riders, but it cannot be used to break the anonymity of the rides (see Section 8).

8 Privacy and Security Analysis

In this section, we present an analysis of ORide to show that it effectively addresses against the privacy attacks described in Section 3.1.

The SP cannot de-anonymize a rider or driver through their anonymous logins by using their ACs. This is guaranteed due to the anonymity and unlinkability properties of the ACL anonymous credential system [9]. Additionally, the SP cannot obtain extra information from the riders' and drivers' encrypted locations and their encrypted distances; this is due to the semantic security property of the FV encryption scheme [16].

In ORide, the information observed by the SP from ORide operations can be put in two databases, as follows (see Table 2).

- Ride DB, in which each entry contains the role and expiration date of the AC, the pick-up zone and obfuscated pick-up time. The role and expiration date are coarse-grained, i.e., all ACs issued on the same day expire at the end of the day they were issued.

- Payment DB, in which each entry contains a rider's ID, a driver's ID, a fare, and the day the fare report is deposited to the SP. Note that this database does not exist if payment is done through e-cash or regular cash, as the fare and payment are done without the SP knowledge.

(A2) Large-scale inference attacks by the SP. To profile riders' and drivers' activities, the SP needs to learn the identities, the locations, and the times associated with their rides.

By using the Payment DB, the SP would know which specific rider took a ride with a specific driver on which day and what its fare was. Since RHS drivers are often licensed to operate in a city or state, knowing that a rider took a ride with a specific driver, the SP might be able to know the city where the rider took a ride, but it does not know the specific location in the city. Note that, in most cases, as the city could be inferred from the zones reported by the riders in their ride requests, this is not an additional leakage of information. In addition, knowing the home/work addresses and the fares of the rides, the SP might be able to infer if a rider went from home to work. However, note that even for frequent rides between home and work of the same rider, the fares would not be the same due to different routes and traffic conditions. Therefore, the inference of rides between home and work of a rider is error-prone. Moreover, such rides are not sensitive, compared to others, such as one-night stands, going to abortion clinics or political-party meetings. For improved anonymity, anonymous-payment methods, such as e-cash or regular cash, could be used to decouple the riders' identities from the fares, thus preventing the SP from learning about rides between home and work of the riders.

By using the Ride DB, the SP might be able to guess the identities of the riders, only if the pick-up zone has a limited number of ride activities *and* riders, e.g., a zone where only one rider lives. This case, however, is unlikely to happen in ORide, because the zones are defined in such a way that each zone has *at least* a large minimum number of ride requests per day, while balancing the bandwidth requirements for the drivers. We illustrate this in Section 9. Note that the SP would be detected if it lied about the activity densities in the zones, because these densities are public knowledge [43], and the drivers would notice if they received very few ride requests from a certain zone.

In the case where the SP knows that a rider makes ride requests from a specific zone (e.g., the zone that contains her home/work addresses) and it wants to know the pick-up times of these rides, the anonymity set of a ride is the number of rides that occurred on the same day from that zone. As this requires the SP to have precise knowledge about the pick-up zones, the anonymity-set size in this case is the *lower-bound* estimation of the anonymity set for the general case of large-scale profiling attacks by the SP. This lower bound is used in the evaluation of the anonymity set achieved by ORide, presented in Section 9.

(A3) Targeted attacks by the SP. In the case where the SP knows the precise pick-up location and time of a ride, it still cannot know the drop-off location and time of the ride, because, in ORide, the drop-off event is not reported to the SP. Knowing the fare from the Payment DB, the SP might be able to guess whether the target went home or to work, but it could not know about other destinations. However, note that similar to the aforementioned case, the inference of rides between home and work of a rider is error-prone. Also such rides are not very sensitive. Anonymous-payment methods, such as e-cash could be used to prevent these attacks.

PII- and location-harvesting attacks by outsiders. ORide relies on a similar proximity-check mechanism as PrivateRide, hence it provides the same guarantees for harvesting-attacks against drivers' PII. However, a malicious outsider might attempt to triangulate drivers, to obtain a snapshot of the locations of all drivers in a zone: It could make three fake ride requests from different locations at the same time to obtain the distances, and cancels these requests immediately. ORide mitigates this attack by applying two measures: (1) requiring a deposit token from each rider per request, thus making the attack more financially expensive and enabling the SP to identify riders who make many requests and cancel (as discussed in Section 5.6), and (2) permuting the list of drivers' indices for each ride request (Step 2 in Section 5.4). Also, the SP can define a smaller threshold on the number of ACs each rider account can obtain per day, if the threat of such an attack is high.

9 Evaluation

In this section, we evaluate our protocols by using a real data-set of taxi rides. We first evaluate the performance of the ride-matching operation in terms of computational and bandwidth requirements for the riders and drivers. We then evaluate the effect of Euclidean distances on the optimality of ride-matching operations.

9.1 Data-Sets

Our data-set consists of over 1.1 billion taxi rides in New York from January 2009 to June 2015 [44]. We extracted data for the month of October in 2013, one of the busiest months in the data-set, which resulted in a subset of over 15 million rides. In this subset, the average duration of the rides is 13 minutes. The GPS traces of the rides are not given; however, the precise pick-up and drop-off locations and times, and pseudo-IDs of the taxi drivers associated with the rides are provided. In addition, the data-set provides mapping between latitude/longitude coordinates to NYC census tracts (CTs), neighborhood tabulation areas (NTAs) and boroughs in NYC.

We make the following assumptions. First, the drop-off location of a driver is her waiting location for new ride requests. Second, a ride-request event is a pick-up event (i.e., consisting of a pick-up location and pick-up time) in our data-set. Third, for each ride-request event, the set of drivers available for that request consists of drivers who have at least one drop-off event in the last 30 minutes since the ride-request timestamp. The 30-minute interval is chosen, because the data-set shows that 99th percentile of the time gap between the drop-off event of a driver and her next pick-up event is approximately 30 minutes.

Setting	Rider		Driver	
Algorithm	Upload (KB)	Download (KB)	Download (KB)	Upload (KB)
S1	372	761856	124	248
S2	372	186	124	248
S3	372	186	124	248

Table 3: Per-ride bandwidth requirements of ORide, with $d = 4096$, $\log_2(q) = 124$, and there are 4096 drivers available for a ride request ($n = 4096$). Compared to the naive SHE approach **S1**, optimized approaches (**S2** and **S3**) significantly reduce the bandwidth requirements for the riders

9.2 Implementation Details

Our ORide prototype features the main cryptographic operations for the ride matching in the ride set-up procedure (Section 5.4). Other cryptographic operations needed for requesting a ride, i.e., AC operations and blind signatures, and for setting up the proximity channel between the rider's app and the driver's app, can be found in the evaluation of PrivateRide [39].

To measure the cryptographic overhead of ride-matching operations, we implemented a proof-of-concept ORide in C++, by relying on the NFLlib library [30]. In our experiments, the SP, the rider, and the driver are located on the same computer, hence network delays are not considered. However, the network delay would not impose a considerable overhead, because a ride-matching operation requires only one round-trip message between the rider and the SP, and one round-trip message between the SP and each driver. Also, the amount of data exchanged between the rider and the SP, and the SP and the drivers, is small (as discussed in Section 9.4). Note that, similarly to current RHSs, the SP can implement a timeout for responses from the drivers such that the latency is reasonable for the service. Due to the dependency requirements of the NFLlib, it is not trivial to port the implementation to mobile platforms. However, to make our experiments close to the performance of smartphones, in all of our evaluations, we *did not* use SSE or AVX optimizations for Intel processors. The source code is made available at [36]. The ORide proof-of-concept implementation on smartphones is work in progress.

9.3 Per-Ride Overhead

In this section, we describe our experimental setup, and presents the bandwidth and computational overhead per ride request for a rider and a driver.

We used ORide's prototype to estimate the overhead added for ride-matching operations in three settings: **(S1)** the naive SHE approach (Section 4.2) without using re-linearizations at the SP, **(S2)** ciphertext-packing optimizations and honest-but-curious drivers (i.e., drivers

Setting	Rider			Driver		SP	
Algorithm	Gen. keys (ms)	Encrypt (ms)	Decrypt (ms)	Load key (ms)	Encrypt (ms)	Load key (ms)	Compute Dist. (ms)
S1	1.51±0.06	2.6±0.2	7823.4±573.4	0.53±0.01	2.6±0.2	0.53±0.01	113868.8±6553
S2	1.51±0.06	2.6±0.2	2.2±0.1	0.53±0.01	2.6±0.2	0.53±0.01	208.9±4
S3	1.51±0.06	2.6±0.2	2.2±0.1	0.53±0.01	2.6±0.2	0.53±0.01	745.5±24.5

Table 4: Per-ride computational overhead of ORide (without AVX/SSE optimizations), for $d = 4096$, $\log_2(q) = 124$, and there are 4096 drivers available for a request. Statistics ($avg \pm std.dev.$) were computed from 1000 experiments. Compared to the naive SHE approach (**S1**), optimized approaches (**S2** and **S3**) significantly reduces the computation time for the SP and the decryption time for the riders.

follow the protocols correctly) (Section 5.4), and **(S3)** ciphertext-packing optimizations and malicious drivers (Section 7.1).

Experimental Setup. To measure the performance of our system, we used a computer (Intel i5-4200U CPU, 2.6 GHz, 6 GB RAM) with Debian Jessie (Linux kernel 3.16). The security parameters used in our experiments are tuned to achieve an equivalent bit-security of more than 112 bits, therefore exceeding current NIST standards for 2016-2030 [5]. With this security target, and a plaintext size of 20 bits the needed polynomial dimension is $d = 4096$, with coefficients of size 124 bits (each polynomial has a size of 62 KB). These parameters guarantee both 112-bits of security and correct operations for homomorphically adding up to 4096 encrypted locations in the same ciphertext and calculating the corresponding Euclidean distances.[2] We refer the reader to Appendix A.2 for more details about the possible granularity a geographical area can have.

Assuming that a rider makes a ride request to the SP and that there are 4096 drivers available for the request ($n = 4096$), with the aforementioned security parameters, the bandwidth requirements and computational overhead per ride request, for a rider and a driver, are shown in Table 3 and Table 4, and explained below.

- *Bandwidth overhead for a rider:* In all three settings, for each ride request, a rider sends to the SP a public key and two ciphertexts for her encrypted planar coordinates. This totals 6 polynomials, a payload size of 372 KB.

  Regarding the number of distance ciphertexts a rider receives from the SP, in setting **S1**, it is equal to n, i.e., the number of responding drivers. In settings **S2** and **S3**, it is significantly reduced to $\lceil n/d \rceil$, due to ciphertext packing. A ciphertext distance, when avoiding relinearizations (see Section 5.4), consists of 3 polynomials, thus having a total size of 186 KB. Assuming 4096 drivers respond to a ride request, setting **S1** would require the SP to send 4096 distance ciphertexts (744 MB) to the rider, whereas **S2** would require only one distance ciphertext (186 KB).

- *Bandwidth overhead for a driver:* In all three settings, for each request: (1) on the downlink, the SP forwards to each driver a public key, 2 polynomials of size 124 KB, and (2) on the uplink, each driver sends back to the SP her encrypted planar coordinates, totaling 4 polynomials of size 248 KB.

- *Computational overhead:* As shown in Table 4, for both riders and drivers, in all three settings, the computational overhead introduced by key generation and encryption operations are small, i.e., 1.5 ms and 2.6 ms, respectively. Due to masking, setting **S3** introduces a small computational overhead for the SP in homomorphic squared-Euclidean-distance computation, compared to setting **S2** (745 ms vs. 208.9 ms). However, noticeably, due to ciphertext packing, settings **S2** and **S3** significantly reduce the computational cost for the SP (208.9 and 745 ms compared to 113868.8 ms required by **S1**). It also significantly reduces the decryption overhead for the rider, from 7823 ms in setting **S1** to 2.2 ms in settings **S2** and **S3**.

Note that the results for the rider and driver are optimistic, as we used a laptop instead of a smartphone (however, as stated before, CPU optimizations were not used to reduce the difference). While such comparisons are not straightforward, we can do a rough estimation of the expected performance of ORide in smartphones. For instance, comparing the performance scores of top multicore CPUs in smartphones [3] with top multicore CPUs in desktops [4], we can see that the difference is less than an order of magnitude. Assuming such difference, then we can see that the computational overheads for key generation, encryption and decryption are still acceptable in smartphones, around 15 ms, 26 ms, and 22 ms, respectively. The overhead is still acceptable even if we consider two orders of magnitude difference, as the total time to hail a ride is in the order of minutes [19].

[2]We refer the reader to Section 6 in [16] for more details on the choice of cryptographic parameters for FV. It is worth noting that we have considered pessimistic bounds in order to cope with recently published attacks that reevaluate the security of lattice-based cryptosystems [7].

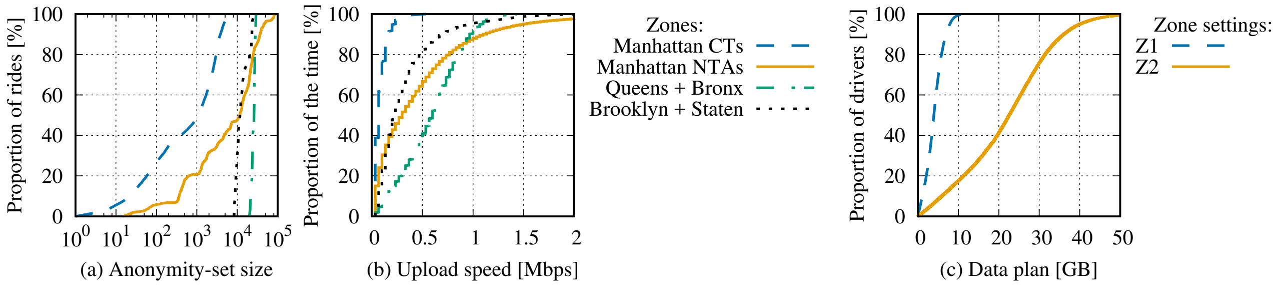


Figure 3: System performance. (a) Anonymity-set size, (b) Upload speed requirement for the drivers, (c) Monthly-data plan requirement for the drivers. Our results show that ORide is scalable while providing good anonymity-set for riders.

9.4 Riders' Anonymity and Drivers' Bandwidth Requirements

In this section, we present the trade-off between the ride-anonymity set vs. bandwidth requirements for the riders and drivers, by using the real data-set presented in Section 9.1.

Due to the high demand of taxi rides in Manhattan w.r.t. lower activity in other boroughs in NYC (from our data-set, Manhattan accounts for 90% of ride requests), we define two zone settings as follows.

- Setting one (Z1): Manhattan is divided into census tracts (CTs). Each CT is one zone. The boroughs of Queens and Bronx are merged into one zone, and the boroughs of Brooklyn and Staten Island are merged into one zone.

- Setting two (Z2): Manhattan is divided into neighborhood tabulation areas (NTAs). Each NTA is one zone. Similarly to setting one, the boroughs of Queens and Bronx are merged into one zone, and the boroughs of Brooklyn and Staten Island are merged into one zone.

Estimation of the anonymity set. As explained in Section 8, the number of rides in a day from a zone is a *lower-bound* estimation of the anonymity set for a ride. Fig. 3a shows the experimental cumulative distribution function (CDF) of the lower-bound anonymity-set size. It can be observed that, for Manhattan with the zone granularity of census tracts, 81.7% of the rides have an anonymity set of size *at least* 50, and for a zone consisting of Queens and Bronx, all of the rides have an anonymity-set size of *at least* approximately 26,000.

Bandwidth requirements for riders. The bandwidth requirements for a rider, per ride request, depends on the number of available drivers. Our experiments show that for both zone settings, for all ride requests, the number of available drivers is less than 3,500. This means that with the security parameter chosen (as presented in Section 9.3) and when proposed optimized packing approaches are used, a rider needs to download only one ciphertext distance, i.e., 186 KB, which is negligible.

Bandwidth requirements for drivers. Fig. 3b shows the CDF of the upload speed required for the drivers; the upload speed is computed by multiplying the number of requests a driver receives per second with the size of the ciphertexts she has to upload per request. Note that the required downlink speed is half of the uplink speed, because the downlink payload is half the size of the uplink payload (Section 9.3). It shows that for Manhattan with the zone granularity of census tracts, the required upload speed is less than 0.5 Mbps, and for other zones, the required upload speed is less than 2 Mbps, which is provided by 3G or 3.5G networks.

Monthly-data plan required for the drivers. Fig. 3c shows the CDF of a data plan required for the drivers for two aforementioned zone settings; this is calculated by multiplying the total number of requests a driver would receive during her waiting time with the uplink- and downlink-payloads per request. The result shows that with the zone setting Z1, a driver needs *at most* 10 GB of data per month, and with the zone setting Z2, 60% of drivers need less than 25 GB of data per month. This requirement is reasonable: For example, in the U. S. , an unlimited data plan typically offers 20-26 GB of high-speed data for less than \$100 [32]. In addition, the drivers can reduce their data-plan consumption by using free WiFi networks, such as LinkNYC [28]. Also, note that the results presented also show that ORide can scale, because current RHSs (e.g., Uber) accounts for only 15% of the ride pick-up requests in NYC [43].

The requirements on bandwidth for the drivers and the anonymity-set sizes for riders enables the SP to define the zones that balance the trade-off between the two aforementioned requirements. For example, for areas that have a high density of ride activities such as Manhattan, the SP could discretize the borough into zones of CTs or NTAs, or combinations of CTs and NTAs. Note that, as shown earlier, at the granularity level of CTs (Z1), the anonymity set provided by ORide for the

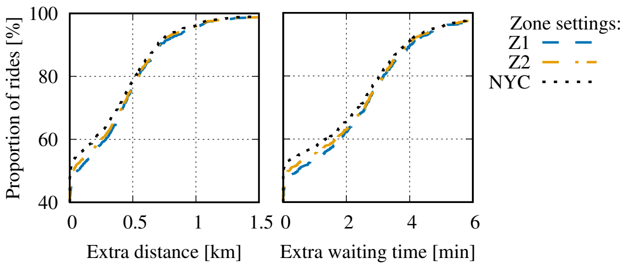


Figure 4: Effect of Euclidean distance on the extra distances for the drivers (left) and on the waiting time for the riders (right) w.r.t. different zone settings. Our results show that the overhead added by ORide is reasonable.

case of a very strong adversarial SP is already large. In special cases, such as concerts and sport events, the SP can split a crowded zone into sub-zones, in order to find a balance for the aforementioned trade-off. For areas that have fewer ride activities, such as other boroughs in NYC or other cities, an entire borough or city can be a zone. For example, a zone consisting of the boroughs of Queens and Bronx would guarantee an anonymity set of at least 26,000 for a ride, while requiring the drivers to have an Internet connection of only at most 2 Mbps.

9.5 Effect on Ride Matching

To minimize the extra costs for both the drivers (e.g., gas and driving time to pickup) and the riders (e.g., waiting times at pick-up locations), ideally, the ride-matching algorithm should take into account the road networks and real-time traffic conditions. Due to the limited operations supported by SHE, ORide uses a simpler matching metric, i.e., Euclidean distances between the riders' and drivers' locations. In addition, due to the bandwidth constraints, the ORide ride-matching algorithm matches a rider to drivers in the same zone, hence suboptimality, e.g., if a rider is close to the border of a zone, the closest driver might be in one of the neighboring zones.

Fig. 4 shows the CDFs of the relative extra costs due to the suboptimality of ORide, compared to the ideal solution, w.r.t. thee different zone settings: Z1, Z2, and the entire city of New York (NYC). The experiment was done on a set of 1,000 randomly selected ride requests. For the ideal matching, we used the Google Maps Distance Matrix APIs [24] to compute the times and distances between a pick-up request and the available drivers (see assumptions in Section 9.1). To reduce the number of requests made to the Google APIs[3], from the set of available drivers, we selected 100 drivers who were closest to the pick-up location as the potential candidates for the ideal matching.

It can be observed that the median extra costs are small: when Z1 is used, in more than 45% of the cases, the driver selected by ORide and the ideal solution is the same, and, in nearly 80% of the cases, the extra driving distance is less than 0.5 km. In addition, the size of the zone has only negligible effects on the optimality of the matching algorithm: If the set of all the drivers available in NYC was used for the ORide matching algorithm, compared to the ideal solution, 78.7% of the cases would have an extra distance of less than 0.5 km, compared to 76.2 % and 76.8 % of the cases when Z1 and Z2 were used, respectively.

[3]The number of requests per day is limited.

10 Conclusion

In this paper, we have proposed ORide, a practical solution that efficiently matches riders and drivers in a privacy-preserving way while still offering key RHS features such as easy payment, reputation scores, accountability, and retrieval of lost items. ORide enables the SP to choose a balanced trade-off between anonymity sets for riders vs. bandwidth requirements for the drivers. For example, for a lower-bound anonymity-set size of 26,000 for rides from the boroughs of Queens and Bronx, drivers only need to have an Internet connection of at most 2 Mbps. The trade-off enables the SP to define the zones such that all users in the system are guaranteed large anonymity sets, even if they are in sparsely populated residential areas with sparse ride activities (by expanding the zones). We have also shown that, even in the extreme case of targeted attacks, i.e., a curious SP wants to know the destination of a rider given the time and location of a rider's pick-up event, the location privacy of the rider's destination is still guaranteed.

For part of our future work, we plan to implement a full prototype of the system on mobile platforms and to design more advanced distance estimation algorithms, instead of the Euclidean distance.

Acknowledgements

The authors are grateful to Chris Soghoian, Zhicong Huang and Christian Mouchet for their insightful discussions and comments about the work.

References

[1] `http://www.theverge.com/2015/6/14/8778111/uber-threatens-to-fire-drivers-attending-protests-in-china`. Last visited: Jan. 2017.

[2] `http://mathworld.wolfram.com/VoronoiDiagram.html`.

[3] `http://browser.primatelabs.com/android-benchmarks`. Last visited: Jan. 2017.

[4] `http://browser.primatelabs.com/processor-benchmarks`. Last visited: Jan. 2017.

[5] Recommendation for Key Management, Part 1: General, SP 800-57 Part 1 Rev. 4. Online: `http://dx.doi.org/10.6028/NIST.SP.800-57pt1r4`, January 2016. Last visited: Feb. 2017.

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

[7] Albrecht, M. R. On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL. Cryptology ePrint Archive, Report 2017/047, 2017. http://eprint.iacr.org/2017/047.

[8] Arfaoui, G., Lalande, J.-F., Traoré, J., Desmoulins, N., Berthomé, P., and Gharout, S. A Practical Set-Membership Proof for Privacy-Preserving NFC Mobile Ticketing. *Proc. of the 15th Privacy Enhancing Technologies Symposium* (2015).

[9] Baldimtsi, F., and Lysyanskaya, A. Anonymous credentials light. In *Proc. of Conference on Computer & communications security* (2013).

[10] https://www.bloomberg.com/view/articles/2016-09-09/wells-fargo-opened-a-couple-million-fake-accounts. Last visited: Jan 2017.

[11] http://uk.businessinsider.com/despite-its-problems-uber-is-still-the-safest-way-to-order-a-taxi-2014-12?r=US&IR=T. Last visited: Feb. 2017.

[12] https://newsroom.uber.com/updated-cancellation-policy/. Last visited: Jan. 2017.

[13] Chaum, D. Blind signatures for untraceable payments. In *Proc. of CRYPTO* (1983).

[14] Dai, C., Yuan, X., and Wang, C. Privacy-preserving ridesharing recommendation in geosocial networks. In *Proc. of Conference on Computational Social Networks* (2016).

[15] http://www.dailydot.com/via/uber-lyft-safety-background-checks/. Last visited: Feb. 2017.

[16] Fan, J., and Vercauteren, F. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/2012/144.

[17] Feldman, P. A practical scheme for non-interactive verifiable secret sharing. In *Proc. of Symposium on Foundations of Computer Science* (1987).

[18] http://www.forbes.com/sites/kashmirhill/2014/10/03/god-view-uber-allegedly-stalked-users-for-party-goers-viewing-pleasure/. Last visited: Jan. 2017.

[19] www.forbes.com/sites/ellenhuet/2014/09/08/uber-lyft-cars-arrive-faster-than-taxis/#3f819c3c5f73. Last visited: May 2017.

[20] Ge, Y., Knittel, C. R., MacKenzie, D., and Zoepf, S. Racial and gender discrimination in transportation network companies. Tech. rep., National Bureau of Economic Research, 2016. http://www.nber.org/papers/w22776. Last visited: Jan. 2017.

[21] Goel, P., Kulik, L., and Ramamohanarao, K. Optimal pick up point selection for effective ride sharing. *IEEE Transactions on Big Data* (2016).

[22] Goel, P., Kulik, L., and Ramamohanarao, K. Privacy-aware dynamic ride sharing. *Trans. on Spatial Algorithms and Systems* (2016).

[23] https://support.google.com/trustedcontacts/?hl=en. Last visited: Jan. 2017.

[24] https://developers.google.com/maps/documentation/distance-matrix/. Last visited: Jan. 2017.

[25] http://www.ibtimes.co.uk/former-uber-employee-reveals-drivers-used-tracking-technology-stalk-celebrities-politicians-1596263. Last visited: Jan 2017.

[26] Isern-Deyà, A. P., Vives-Guasch, A., Mut-Puigserver, M., Payeras-Capellà, M., and Castellà-Roca, J. A secure automatic fare collection system for time-based or distance-based services with revocable anonymity for users. *The Computer Journal* (2013).

[27] Laurie, B., Langley, A., and Kasper, E. Certificate transparency. https://tools.ietf.org/html/rfc6962, 2013.

[28] https://www.link.nyc. Last visited: Feb. 2017.

[29] Lyubashevsky, V., Peikert, C., and Regev, O. On Ideal Lattices and Learning with Errors Over Rings. Cryptology ePrint Archive, Report 2012/230, 2012. http://eprint.iacr.org/2012/230.

[30] Melchor, C. A., Barrier, J., Guelton, S., Guinet, A., Killijian, M., and Lepoint, T. NFLlib: NTT-Based Fast Lattice Library. In *Proc. of the RSA conference - The Cryptographers' Track* (2016).

[31] Milutinovic, M., Decroix, K., Naessens, V., and De Decker, B. Privacy-preserving public transport ticketing system. In *Proc. of Conference on Data and Applications Security and Privacy* (2015).

[32] https://www.nerdwallet.com/blog/utilities/best-unlimited-data-plans/. Last visited: Feb. 2017.

[33] https://research.neustar.biz/2014/09/15/riding-with-the-stars-passenger-privacy-in-the-nyc-taxicab-dataset/. Last visited: Jan. 2017.

[34] https://newsroom.uber.com/philippines/new-upfront-fares-on-uberx/. Last visited: Jan. 2017.

[35] http://www.oregonlive.com/today/index.ssf/2014/11/sex_the_single_girl_and_ubers.html. Last visited: Jan. 2017.

[36] http://oride.epfl.ch.

[37] http://www.orlandosentinel.com/news/breaking-news/os-lyft-driver-arrest-sex-battery-20160929-story.html. Last visited: Jun. 2017.

[38] Pedrouzo-Ulloa, A., Troncoso-Pastoriza, J. R., and Perez-Gonzalez, F. Number theoretic transforms for secure signal processing. *Trans. on Information Forensics and Security* (2017).

[39] Pham, T. V. A., Dacosta Petrocelli, I. I., Jacot-Guillarmod, B., Huguenin, K., Hajar, T., Tramèr, F., Gligor, V., and Hubaux, J.-P. PrivateRide: A Privacy-Enhanced Ride-Hailing Service. In *Proc. of Privacy Enhancing Technologies Symposium* (2017).

[40] Sánchez, D., Martínez, S., and Domingo-Ferrer, J. Co-utile P2P ridesharing via decentralization and reputation management. *Transportation Research Part C: Emerging Technologies* (2016).

[41] https://techcrunch.com/2016/11/29/just-like-uber-lyft-launches-upfront-fares/. Last visited: Jan. 2017.

[42] http://thenextweb.com/insider/2016/12/13/uber-tracks-customers-long-after-their-ride-is-over/. Last visited: Jan. 2017.

[43] http://toddwschneider.com/posts/analyzing-1-1-billion-nyc-taxi-and-uber-trips-with-a-vengeance/. Last visited: Feb. 2017.

[44] https://github.com/toddwschneider/nyc-taxi-data. Last visited: Jan. 2017.

[45] http://www.usatoday.com/story/tech/columnist/stevenpetrow/2016/10/12/fake-uber-drivers-dont-become-next-victim/91903508/. Last visited: Jan 2017.

[46] http://www.usatoday.com/story/tech/2014/11/19/uber-privacy-tracking/19285481/. Last visited: Jan. 2017.

[47] https://www.uwgb.edu/dutchs/FieldMethods/UTMSystem.htm. Last visited: Feb. 2017.

[48] http://www.wcnc.com/news/crime/uber-driver-attacked-rider-over-politics-man-says/339458660. Last visited: Jan 2017.

[49] http://wfla.com/2016/12/27/uber-and-lyft-drivers-worry-about-passenger-attacks/. Last visited: Jun. 2017.

A Appendix

A.1 Covertly Active SP

Fig. 5 illustrates the changes introduced to the original ride set-up procedure (Section 5.4) to handle a *covertly active* SP. In this protocol, we introduce the notion of *Proof-of-Ride* (PoR), a token that is used to prove to the drivers that the rider is real, i.e., she did a ride in the past.

A.2 Plaintext Space

Assume a geographical area of size $s \times s$ and a plaintext space of b bits to represent the squared-Euclidean distances between points in the area. The area can be quantized into a grid with cells of size $s/2^{(b-1)/2} \times s/2^{(b-1)/2}$, with the explanation as follows. Assuming the area is discretized into a grid of $v \times v$ cells, the largest possible squared-Euclidean distance between any two points on the grid is $2 \times v^2$, and this has to be at most 2^b. Therefore, $v \leq 2^{(b-1)/2}$. In other words, each edge of size s can be discretized into v points, and the distance between any pair of two consecutive points is $s/2^{(b-1)/2}$. Therefore, the area can be represented by a grid with cells of size $s/2^{(b-1)/2} \times s/2^{(b-1)/2}$.

For example, with 20-bit plaintext space, a geographical area of size 60 km^2, such as the borough of Manhattan in NYC, would be quantized into a grid of resolution approximately 10 m $\times$ 10 m.

A.3 Cryptographic Primitives

In this section, we briefly describe the cryptographic building blocks used in ORide.

Blind signatures. A blind-signature scheme [13] is a form of digital-signature schemes in which the signer does not know the content of the message that she is signing. This is achieved by enabling the signature requester to 'blind' (i.e., randomize) the message before sending it to the signer. When the signature requester receives the signature on her blinded message, she 'unblinds' it to obtain a valid signature for the original message. The signer, when is asked to verify the signature of an unblinded message, is not able to link this message back to the blinded version she signed.

Anonymous credentials. An anonymous credential (AC) is a cryptographic token with which the credential owner can prove to another party that she satisfies certain properties without revealing her real identity. In ORide, a user is identified when she obtains ACs from the SP. However, when she wants to start an anonymous session, she reveals to the SP only the expiration date and the role specified in the AC (i.e., rider or driver), and she proves to the SP, in a zero-knowledge fashion, that she knows the private key associated with the AC. To prove her reputation to a driver, a rider reveals to the driver the reputation score specified in her AC together with the proof to show that the revealed value is trustworthy. ORide relies on the Anonymous Credentials Light (ACL) [9]. However, note that ACL is a linkable anonymous credential scheme, i.e., a user can only use a credential once to avoid her transactions from being linkable.

Number-Theoretic Transform (NTT). An NTT is the finite ring version of a Discrete Fourier Transform; an n-point NTT of a vector $\boldsymbol{x} \in \mathbb{Z}_t^n$ and its inverse operation $\mathtt{NTT}^{-1}$ have the form

$$\boldsymbol{X} = [\mathtt{NTT}(\boldsymbol{x})_k]_{k=0}^{n-1} = \left[\sum_{i=1}^{n-1} x_i \alpha^{ki}\right]_{k=0}^{n-1},$$

$$\boldsymbol{x} = \left[\mathtt{NTT}^{-1}(\boldsymbol{X})_k\right]_{k=0}^{n-1} = \left[n^{-1}\sum_{k=1}^{n-1} X_k \alpha^{-ki}\right]_{i=0}^{n-1},$$

where n^{-1} is the modulo inverse of n in $\mathbb{Z}_t$, and α is a principal n-th root of unity in $\mathbb{Z}_t$, whose existence is a necessary condition for the transform. The NTT can be implemented with fast algorithms with complexity $O(n \log n)$, especially when n is a power of 2. The NTT presents a convolution property, that relates the circular convolution ($\circledast$) of two vectors with the component-wise product ($\cdot$) of their transformed versions, such that

$$\boldsymbol{x} \circledast \boldsymbol{y} = \mathtt{NTT}^{-1}(\mathtt{NTT}(\boldsymbol{x}) \cdot \mathtt{NTT}(\boldsymbol{y})).$$

Therefore, an $O(n^2)$ operation like the convolution gets reduced to the complexity of the transforms ($O(n \log n)$) and the component-wise product ($O(n)$). For the used cryptographic application, the product operation in the polynomial ring is a nega-cyclic convolution instead of a cyclic one; this slightly changes its formulation, and it imposes the requirement of a $2n$-th root of unity in $\mathbb{Z}_t$ (we refer the reader to [38] for further details).

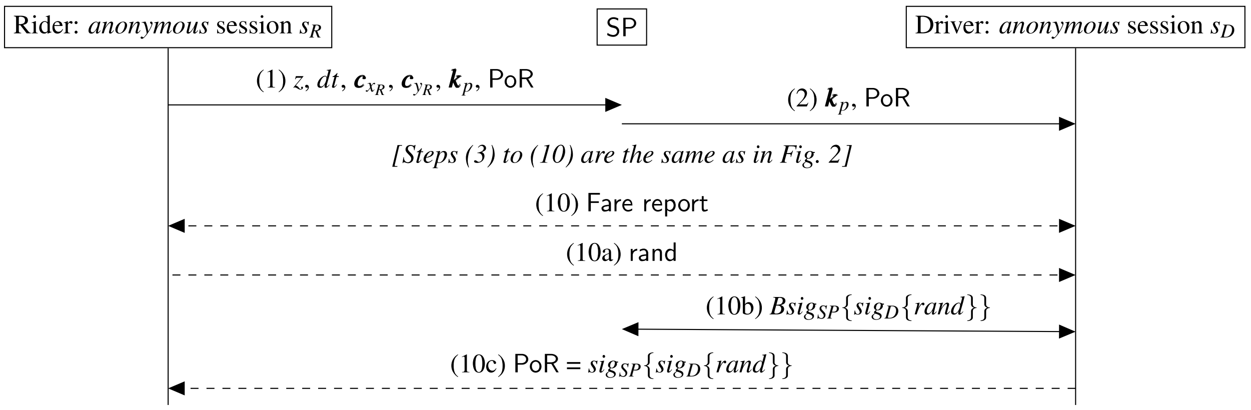

Rider: *anonymous* session s_R
SP
Driver: *anonymous* session s_D
(1) z, dt, $\boldsymbol{c}_{x_R}$, $\boldsymbol{c}_{y_R}$, $\boldsymbol{k}_p$, PoR
(2) $\boldsymbol{k}_p$, PoR
[Steps (3) to (10) are the same as in Fig. 2]
(10) Fare report
(10a) rand
(10b) $Bsig_{SP}\{sig_D\{rand\}\}$
(10c) PoR = $sig_{SP}\{sig_D\{rand\}\}$

Figure 5: Changes introduced to the original ride set-up protocols (Fig. 2) to handle *covertly active* SP.