applied sciences

MDPI

Article

pShare: Privacy-Preserving Ride-Sharing System with Minimum-Detouring Route

Junxin Huang, Yuchuan Luo *, Ming Xu, Bowen Hu and Jian Long

College of Computer Science, National University of Defense Technology, Changsha 410073, China; huang_jx14@nudt.edu.cn (J.H.); xuming@nudt.edu.cn (M.X.); hbw_14@nudt.edu.cn (B.H.); jian_long@nudt.edu.cn (J.L.)

* Correspondence: luoyuchuan09@nudt.edu.cn

Abstract: Online ride-hailing (ORH) services allow people to enjoy on-demand transportation services through their mobile devices in a short responding time. Despite the great convenience, users need to submit their location information to the ORH service provider, which may incur unexpected privacy problems. In this paper, we mainly study the privacy and utility of the ride-sharing system, which enables multiple riders to share one driver. To solve the privacy problem and reduce the ride-sharing detouring waste, we propose a privacy-preserving ride-sharing system named *pShare*. To hide users' precise locations from the service provider, we apply a zone-based travel time estimation approach to privately compute over sensitive data while cloaking each rider's location in a zone area. To compute the matching results along with the least-detouring route, the service provider first computes the shortest path for each eligible rider combination, then compares the additional traveling time (ATT) of all combinations, and finally selects the combination with minimum ATT. We designed a secure comparing protocol by utilizing the garbled circuit, which enables the ORH server to execute the protocol with a crypto server without privacy leakage. Moreover, we apply the data packing technique, by which multiple data can be packed as one to reduce the communication and computation overhead. Through the theoretical analysis and evaluation results, we prove that *pShare* is a practical ride-sharing scheme that can find out the sharing riders with minimum ATT in acceptable accuracy while protecting users' privacy.

Keywords: online ride-hailing; ride-sharing matching; privacy-preserving; location privacy; path planning

Citation: Huang, J.; Luo, Y.; Xu, M.; Hu, B.; Long, J. *pShare*: Privacy-Preserving Ride-Sharing System with Minimum-Detouring Route. *Appl. Sci.* **2022**, *12*, 842. https://doi.org/10.3390/app12020842

Academic Editors: Martin Lauer, Angel Llamazares and Javier Alonso Ruiz

Received: 24 October 2021
Accepted: 10 January 2022
Published: 14 January 2022

1. Introduction

With the advent of GPS-embedded mobile devices and digital maps, Online Ride-Hailing (ORH) services have gained much popularity and have seen remarkable development in the past decade. Unlike traditional taxi services, ORH companies such as Uber and Didi provide a platform for riders to request a ride using few operations through their mobile devices, which has had a revolutionary impact on the ride-hailing industry [1]. According to the statistics of Didi Chuxing, they have completed over 10 billion ride-hailing orders [2].

Ride-matching is the primary function of ORH service, and it usually matches the nearest driver for any rider who starts a ride-matching request. To request the ride-matching service, the rider needs to submit his location information to the ORH service provider (i.e., the ORH company) through the ride-hailing application on their smartphone. Then the service provider matches the nearby drivers who update their locations in real time.

In order to improve the traffic utilization and the matching success rate, some ORH services also provide an enhanced function, i.e., ride-sharing service, which enables several riders with similar traveling routes to share one driver. Ride-sharing is in the broad category of the sharing economy [3], which is environmentally friendly and beneficial to all participant entities of the service. For riders, it provides a financial and efficient alternative.

Moreover, it enables drivers to cut down their traveling costs and improve the revenue with payment from several passengers. Besides, it reduces the emissions of CO_2 [4].

However, despite the convenience the ORH service brings, the privacy concern is also highly raised. For example, the largest Chinese ORH company, Didi Chuxing, was recently demanded to take off all applications from app stores due to a serious violation of data and privacy laws [5]. To request a ride, the rider must submit his ride information, which usually includes some private information on riders such as pick-up and drop-off locations and times. The submitted data can be analyzed for one's mobility patterns or can even be exploited to deduce one's home or working address. The leakage of users' private information invades their rights and sometimes even threatens their security.

It is necessary to preserve the privacy of users, whether riders or drivers. To achieve that, many encryption techniques and privacy-enhanced solutions are designed for ride-hailing services [6–11], which provide private ride-matching schemes but do not support ride-sharing. There are also a few privacy-preserving ride-sharing schemes including SRide [12], PRIS [13], and PrivatePool [14], but they can only enable ride-sharing matches over fixed locations. PSRide is a flexible ride-sharing matching scheme, but it does not support ride-sharing between multiple riders. PGRide supports group ride-sharing matching. However, it cannot find the shortest path to pick up and drop off all riders.

Considering the issues stated above, we are motivated to research how to construct a private and practical ride-sharing mechanism, which supports flexible ride-sharing matching over multiple riders and meanwhile finds the shortest path with minimum detouring distance. However, we cannot compute the shortest path over plaintext precise locations of users due to the consideration of efficiency and privacy; thus, a zone-based method is applied to trade off between these factors. By hiding all users' locations in distinct zones, a user's privacy is well protected since enemies cannot distinguish from hundreds of points of interest in one zone. It is noteworthy that the matching is also computed over discretized zones, so the matching results are not necessarily optimal on a non-discretized map. In the following paper, expressions such as "minimum detouring" indicate the computing results over the zones. In Section 6, we implement *pShare* under zone-based estimation data and non-discretized ground-truth map data to evaluate the real accuracy of the scheme.

The ride-sharing service is an optional ride-hailing service since it can be replaced by ride-matching if there are no suitable sharing riders. Therefore, we apply the ride-sharing from the perspective of drivers and match several riders for each driver. For an available driver, the ORH server first searches his surrounding requested riders, then computes for the best matching riders with minimum detouring time. More specifically, our main contributions can be summarized as follows:

1. We propose a flexible, privacy-preserving online ride-sharing matching scheme *pShare*, which performs matching between a shared driver and group riders with minimum detouring distance, while protecting their location privacy against the ORH service provider.
2. We propose a zone-based riders-filtering approach, which can divide surrounding requested riders on the similarity of their traveling routes. Meanwhile, we have designed a secure comparing protocol, which computes users' encrypted data and outputs the ride-sharing results along with the minimum-detouring route.
3. We evaluate the scheme using generated user datasets which obey real-world distribution. The experimental results demonstrate that *pShare* can achieve the functionality with high accuracy and practical efficiency.

The rest of the paper is organized as follows: In Section 2, we formalize the problem and introduce the system model and the threat model. In Section 3, we present some preliminaries of the paper. Then we present the specific construction of *pShare* in Section 4 and the privacy analysis in Section 5, followed by performance evaluation in Section 6. Finally, we conclude our work in Section 7.

2. Model and Problem Definitions

2.1. Problem Definition

Given a driver d, supposing the passenger capacity of d is CAP_d and the current number of onboard riders is OBR_d, we define that d could be at one of three possible situations: idle, full, and not-full, which represent $OBR_d = 0$, $OBR_d = CAP_d$, and $0 < OBR_d < CAP_d$, respectively.

The problem of our concern is, for the driver d and the riders set $\mathcal{R}$, to find the best match between the driver and v riders along with the optimal route to pick up and drop off all riders while protecting users' privacy, in which $v \leqslant CAP_d - OBR_d$. The additional travel time (ATT) of the matching result should be minimal, and the schedule will miss no deadlines of the riders.

2.2. System Model and Threat Model

As shown in Figure 1, the system consists of four entities: an ORH server, a crypto server, the riders, and the drivers. We use the RS to represent the ORH server and the CS to represent the crypto server in the following paper. Drivers start ride-sharing queries and submit their current status when they are idle or not-full. Riders submit ride requests when they need a ride-sharing service. The RS finds the optimal riders for the driver with minimum average ATT. Besides, we introduce the CS to set up the cryptographic parameters and enable the matching computation. Based on the system model, we make the following assumptions:

1. In our system, all entities are semi-honest. That means they always honestly follow the scheme but are curious about the information of users. So the goal of privacy is to prevent the RS or the CS from learning about users' sensitive infomation, which includes precise locations and appointed time.
2. There is no collusion between the RS and the CS because they are managed by different parties. Nor will the RS collude with the drivers because the RS is always a big company and takes much attention to the reputation.
3. The communication channel between two entities is secure, which means no man-in-the-middle attack will be launched. Nobody can utilize channel information to locate users.

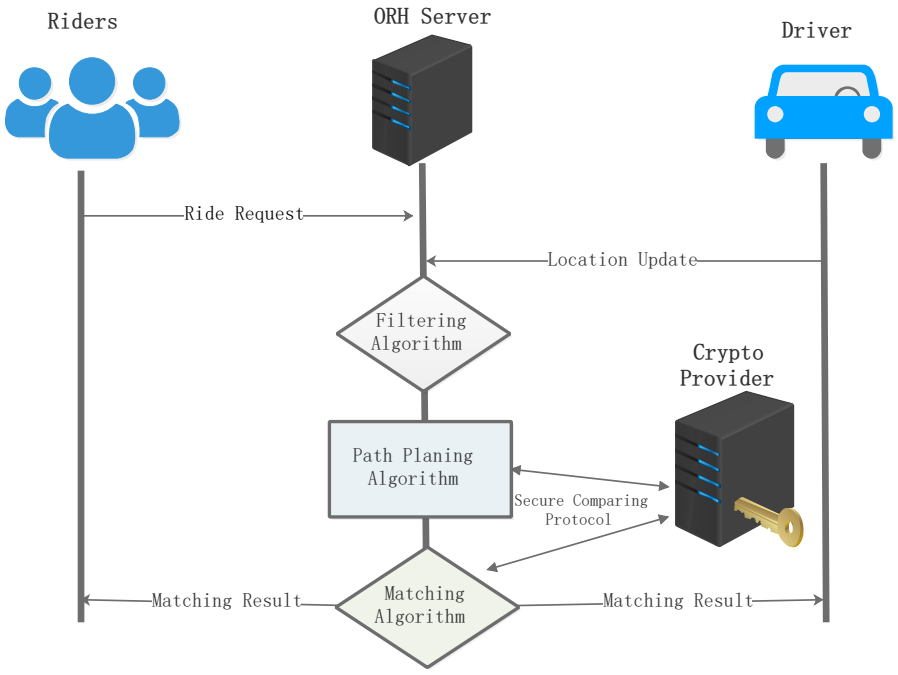

Figure 1. System model.

2.3. Design Goals

Under the proposed system model and threat model stated above, *pShare* system should satisfy the following design objectives:

1. Accuracy. The RS should perform matching over ride-requests and drivers' status accurately. However, there exists a trade-off between accuracy and privacy level with different zone scales. Thus, under a proper privacy level, our goal of accuracy is that the RS should be able to find riders with minimum ATT in a big probability.
2. Efficiency. The system should be efficient such that the responding time for riders and drivers is all acceptable in a real-time ride-matching scenario.
3. Privacy. During the whole process of the scheme, the RS and CS should be prevented from learning the precise location information of users. Riders and drivers should likewise not learn each other's information unless they are matched.

3. Preliminaries

3.1. Paillier Cryptosystem

The Paillier cryptosystem [15] is a public-key cryptosystem with the property of additive homomorphism based on the composite residuosity assumption [1]. We briefly describe it as follows:

KeyGen$() \rightarrow (pk, sk)$: Choose two large primes p and q, and let $n = pq$. Then compute the least common multiple $\lambda = lcm(p-1, q-1)$ and select $g \in Z^*_{n^2}$, ensuring the greatest common divisor $gcd\left(L\left(g^\lambda \left(mod\ n^2\right)\right), n\right) = 1$, in which $L(x) = (x-1)/n$. Then the public key is $pk = (n, g)$ and the secret key $sk = \lambda$.

Encryption: $E(m, pk) \rightarrow c$. Given a plaintext $m \in Z^*_n$, randomly select $r \in Z^*_n$ and compute the ciphertext as follows:

$$c = g^m r^n \left(mod\ n^2\right) \tag{1}$$

Decryption: $D(c, sk) \rightarrow m$. Given a ciphertext $c \in Z_{n^2}$, the corresponding plaintext can be computed as follows:

$$m = \frac{L\left(c^\lambda mod\ n^2\right)}{L\left(g^\lambda mod\ n^2\right)}\ mod\ n \tag{2}$$

The Paillier cryptosystem described above is semantically secure. Besides, it has the following homomorphism properties:

$$E(m_1)E(m_2) = E(m_1 + m_2) \tag{3}$$

$$E(m_1)g^{m_2} = E(m_1 + m_2) \tag{4}$$

3.2. Garbled Circuit

Garbled circuit was first introduced by Yao [16] to solve the millionaire's problem. It enables two parties to evaluate any function jointly such that nothing can be learned from the evaluation but their outputs. Specifically, one party first generates a garbled version of function $f(x, y)$ as the generator and obtains the garbled circuit $\hat{f}$. Then the generator sends $\hat{f}$ together with his garbled input $\hat{x}$ to the other party, which is the evaluator. Then they execute a 1-out-of-2 Oblivious Transfer (OT) protocol jointly, in which the evaluator can obtain his garbled input $\hat{y}$ without any information leakage to the generator. Then the evaluator can evaluate the garbled circuit to obtain the result of $f(x, y)$. Several optimizations have been proposed to construct Yao's GC, such as [17,18].

3.3. Data Packing

Let $(t_1, t_2, \cdots, t_n)$ be an n-dimensional vector. We can pack it in the following form:

$$S(t) = \sum_{j=1}^{n} t_1 x^{j-1} \tag{5}$$

where x^{j-1} is the parameter which is big enough to separate two values at the bit level. Similarly, for data encrypted in paillier cryptosystem $([\![t_1]\!], [\![t_2]\!], \cdots, [\![t_n]\!])$, we can pack it as:

$$S([\![t]\!]) = \prod_{j=1}^{n} [\![t_i]\!]^{x^{j-1}} \tag{6}$$

By applying the data packing technique, we can use just one decryption to the packed ciphertext or one encryption to the packed plaintext.

3.4. Road Travel Time Estimation

Our approach to compute the best matching results for ride-sharing service consists of shortest road traveling time computation. Traditional methods compute it directly in large-scale road networks, which will cause an expensive computation cost and cannot be applied with cryptographic primitives. To solve this, a zone-based method could be used to estimate the road traveling time [19]

First, the road network is partitioned into $m \times n$ zones $\{z(i,j)\}_{(i,j)=(1,1)}^{(m,n)}$. For each grid, we set an anchor point for travel time estimation between zones. All anchor points are selected considering the number of its adjacent edges (namely the degrees) and the geographical distance to the center of its located zone. We use $\{AP(i,j)\}_{(i,j)=(1,1)}^{(m,n)}$ to represent the anchor point of each zone.

Once the process of anchor setting is finished, the travel time between zones can be estimated and precomputed with the distance between anchor points as follows:

$$t_{z(i,j)\to z(x,y)} = t_{AP(i,j)\to AP(x,y)} \ (1 \leqslant i, x \leqslant m, 1 \leqslant j, y \leqslant n) \tag{7}$$

For any two points in the road network, we denote as s and d, respectively. The travel time between s and d can be estimated with the following expression:

$$t_{s\to d} \approx t_{s\to AP(s)} + t_{AP(s)\to AP(d)} + t_{AP(d)\to d} \tag{8}$$

where $AP(x)$ denotes the anchor point of the zone in which x is located. The information of road network and anchor points are public.

In our proposal scheme, for a ride request with start point s and destination d, the ride requester can easily compute $t_{s\to AP(s)}$ and $t_{AP(d)\to d}$ at the local mobile device. Then the rider can submit them after encryption to hide the precise locations s and d in two located zones.

4. Proposed Approach

In this section, we describe the architecture of *pShare* and the protocol details we apply to obtain the best matching results without privacy leakage of users.

4.1. System Overview

Given the locations of drivers and riders, the ride-sharing matching is to find the best combination of riders with minimum detouring time. The RS applies the matching between a driver and several riders who accept to share their ride with other passengers. Before the RS applies the ride-sharing matching, the system needs to finish the initialization. With the coarse locations of riders and drivers, the RS first executes a filtering algorithm and obtains several sets of nearby riders, which are divided in terms of the similarity of their trips.

Then for each set, the RS traverses all feasible rider combinations of v riders and computes the encrypted ATT of each rider combination. To compare the ciphertexts without leaking privacy, the RS and the CS execute comparing protocols using Yao's GC together to compute the results securely. The combination with minimum ATT will be selected and matched to the driver. Finally, a secure communication channel will be established between the driver and selected riders for pick-up details.

It is noteworthy that no precise locations are exposed to other entities except the user himself. All sensitive information such as location and time deadline are submitted in ciphertext form.

4.2. Protocol Details

We now describe the construction details of *pShare* scheme, which consists of the following five steps:

1. **Setup.** In this procedure, the system needs to be initialized. First, the map is partitioned into zones, and the anchor point of each zone needs to be set up as described in III.D; the travel time between any two anchor points is precomputed. Then the CS generates the key pair (pk, sk) and broadcasts pk to other parties in the system.
2. **Ride Request Submit.** When a rider needs the ride-sharing service, he will first compute the estimated travel time $t_{s\to AP(s)}$ and $t_{d\to AP(d)}$ using the public road network and navigation system in his mobile device, where s and d are the locations of his start point and destination point. Then he encrypts them with pk and obtains $[\![t_{s\to AP(s)}]\!]$ and $[\![t_{d\to AP(d)}]\!]$, respectively, in which $[\![]\!]$ represents the ciphertext form. After that, the rider selects the deadline of pick-up time t_s and the deadline of drop-off time t_d and encrypts them. Finally, the rider submits the ride-sharing request $R = \left(z(s), z(d), [\![t_{s\to AP(s)}]\!], [\![t_{d\to AP(d)}]\!], [\![t_s]\!], [\![t_d]\!]\right)$.
3. **Driver Location Update.** Similarly, the driver needs to submit his current status to finish the matching process. First, he computes $[\![t_{l\to AP(l)}]\!]$ with his end device and pk, then submits his status $D = \left(CAP_d, OBR_d, z(l), [\![t_{l\to AP(l)}]\!]\right)$, where l is the driver's current location and v is the current vehicle load. If $OBR_d > 0$, namely that the driver has existing riders, the schedule of driver $\mathcal{S}_d = \left(\{R_i\}_{i=1}^{OBR_d}, \mathcal{P}\right)$ should also be submitted to the RS, which include onboard riders' encrypted requests and the existing path. The definition of path is described in the following paper.
4. **Riders Filtering**. To mitigate the computation overhead, the RS needs to filter the riders on their coarse locations of start point and destination point (i.e., the corresponding zones) as Algorithm 1.
   On receiving from a driver's status D, the RS first checks if there are any existing riders. If $OBR_d = 0$, the RS searches the surrounding riders of $z(l)$ within a fixed-size region SR and outputs the set $\mathcal{R}_{nearby}$. For each rider r of $\mathcal{R}_{nearby}$, the RS obtains the shortest route between $AP(r_i.z(s))$ and $AP(r_i.z(d))$ from precomputed data. Then the RS gets $r_i.Zpath = \{r_i.z(s), \cdots, r_i.z(d)\}$, which represents the zones sequence the route passes through. The RS selects M riders $\{\ddot{r}_i\}_{i=1}^{M}$ with the longest length of $Zpath$ from $\mathcal{R}_{nearby}$ as the root riders, meanwhile M corresponding rider sets $\{\mathcal{R}_i\}_{i=1}^{M}$ are constructed. In addition, the RS computes the expanded path for each root rider of $\{\ddot{r}_i\}_{i=1}^{M}$. As Figure 2 indicates, $r.Zpath$ indicates the black area, and $r.EZpath$ is computed as the grey area and the black area in order to include more potential riders. For the rest of the riders, the RS compares their paths and the directions with the root riders as follows, and riders with similar traveling routes will be added to the corresponding set.

Algorithm 1 Request Filtering Algorithm (*RFA*).

Input: D, $\mathcal{R}$
Output: $\{\mathcal{R}_i\}_{i=0}^{M}$

for $r \in \mathcal{R}$ **do**
  if $r.z(s) \in SR$ **then**
    Add r into $\mathcal{R}_{nearby}$;
  end if
end for
Select M riders $\{\ddot{r}_i\}_{i=1}^{M}$ with max length of $Zpath$;
for 1 to M **do**
  Compute $R_i.EZpath$;
  Construct $\{\mathcal{R}_i\}_{i=1}^{M}$;
  Add $\ddot{r}_i$ to $\mathcal{R}_i$;
end for
for every $r \in \mathcal{R}_{near}$ **do**
  for 1 to M **do**
    if $r.z(s), r.z(d) \in \ddot{r}_i.EZpath$ **and** $Dir(r, \mathcal{R}_i) < 0$ **then**
      Add r to $\mathcal{R}_i$;
    end if
  end for
end for
return $\{\mathcal{R}_i\}_{i=0}^{M}$

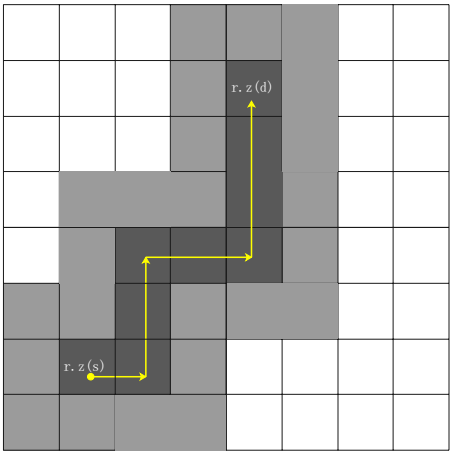

Figure 2. Expanded path of r.

For each rider $r_j \in \mathcal{R}_{nearby}$ except root riders, the RS judges if the following conditions are met:

$$r_j.z(s), r_j.z(d) \in \ddot{r}_i.EZpath \; i \in [1, M] \tag{9}$$

$$Dir(r_j, \mathcal{R}_i) = \ddot{r}_i.EZpath.indexof(r_j.z(s)) - \ddot{r}_i.EZpath.indexof(r_j.z(d)) \tag{10}$$

where (9) indicates r_j is in the path of r_i; when (9) is met, the RS judges if they have similar directions with the appearing order of the start point and the destination point in (10), if r_j. Once qualified, r_j will be added to the set $\mathcal{R}_i$. Note that a rider could be added to more than one set.

If $OBR_d > 0$ and the driver has an existing schedule, the RS will refer the existing path $\mathcal{P}$ to add potential riders to just one set as the Algorithm 2 does.

Algorithm 2 Filtering Algorithm with Existing Schedule (*FAES*).

Input: D, R
Output: $\mathcal{R}$

```
Construct the rider set R
Compute R.EZpath with P
for every r ∈ R_near do
    if r.z(s), r.z(d) ∈ R.EZpath and Dir(r, R) < 0 then
        Add r to R;
    end if
end for
return R
```

5. **Ride-Sharing Matching.** In this module, the RS performs the ride-sharing matching with the CS in two phases. In the first phase, the RS and CS jointly compute the shortest path for all rider combinations. Each combination consists of v riders, where we usually choose $v = CAP_d - OBR_d$. If there is no eligible combination of $CAP_d - OBR_d$ riders, the RS will try with smaller combinations. After that, a comparing phase is conducted to pick out the combination with minimum ATT along with the shortest path.
Phase 1 (Path Planning) For a rider combination C with v riders, the RS finds the shortest path as follows:
For each rider r_i $i \in [1, v]$ in C, we regard $\mathcal{V} = \{V_i\}_{i=0}^{2v} = \{d.x, r_i.s, r_i.d\}_{i=1}^{v}$ as vertices of a graph, the travel time between V_i and V_j is the corresponding edge $t_{V_i \to V_j}$. Therefore, the path-planning problem is to find the shortest path on ciphertexts that passes through all vertices.

Definition 1. *Given a rider combination C, the ride-sharing path* $\mathcal{P} = [V_i]_{i=0}^{2v}$ *is a vector of vertices that indicates the pick-up or drop-off order. Each vertice represents a location point of* $\{d.l, r.z(s), r.z(d)\}$ *together with a time constraint* t_{V_i}*. The path should cover all nodes of the graph. A legal path* $\mathcal{P}$ *has to meet the following constraints:*

$$\mathcal{P}[0] = d.l \tag{11}$$

$$\mathcal{P}.indexof(r_i.s) < \mathcal{P}.indexof(r_i.d) \tag{12}$$

$$\begin{cases} t_1 \leqslant t_{V_1} \\ t_1 + t_2 \leqslant t_{V_2} \\ \cdots \end{cases} \tag{13}$$

To find the shortest path, the RS needs to compare all practical paths with their total traveling time. As depicted in Figure 3, for all paths that meet (11) and (12), the RS checks and selects the driving path that would not fail any deadline in (13). To do this, the RS computes the encrypted travel time for all edges using (3) and (8) and gets $[[\![t_i]\!]]_{i=1}^{2v}$. Then the RS computes $[\![\mathcal{T}(\mathcal{P})]\!]$ for each path $\mathcal{P}$ as follows:

$$[\![\mathcal{T}(\mathcal{P})]\!] = \left[\prod_{j=1}^{i} [\![t_i]\!]\right]_{i=1}^{2v} = [[\![T_i]\!]]_{i=1}^{2v} \tag{14}$$

in which T_i represents the total time consumption to reach the node v_i.

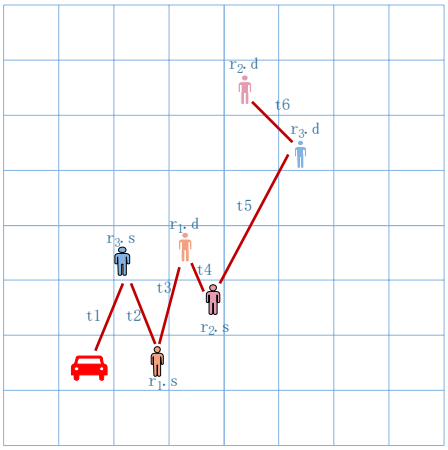

Figure 3. A pratical path of the rider combination.

For each path, the RS compares each point with its deadline time. Suppose $T_1, T_2, \cdots T_{2v}$ are p-bit integers. Then the RS randomly chooses $2v$ $(k-1)$-bit length integers $\mathcal{A} = \{a_1, a_2, \cdots a_{2v}\}$ $(k-1 > p)$ and applies the following operations:

$$S([\![\mathcal{T}(\mathcal{P})]\!]) = \prod_{i=1}^{2v} [\![T_i]\!]^{2^{k(2v-i)}} \tag{15}$$

$$S(\mathcal{A}) = S(a_1, a_{2,} \cdots a_{2v}) = \sum_{i=1}^{2v} a_i 2^{k(2v-i)} \tag{16}$$

$$S\left([\![\widetilde{\mathcal{T}}(\mathcal{P})]\!]\right) = E(S(\mathcal{A}))S([\![\mathcal{T}(\mathcal{P})]\!]) \tag{17}$$

The RS first packs $[\![T_1]\!], [\![T_{2,}]\!] \cdots [\![T_{2v}]\!]$ and $a_1, a_{2,} \cdots a_{2v}$ in (15), (16) and then adds the encrypted integers to each item of $\{[\![T_i]\!]\}_{i=1}^{2v}$ in (17) and obtains $[\![\widetilde{\mathcal{T}}(\mathcal{P})]\!]$, which is the packing ciphertexts after blinding operation. Similarly, for the deadline time of each point $[\![T_V(\mathcal{P})]\!] = [[\![t_{V_i}]\!]]_{i=1}^{2V}$, the RS generates random integers $\mathcal{A}' = \left\{a'_1, a'_{2,} \cdots a'_{2V}\right\}$ and executes the same blinding operations in (15)–(17).

Then the RS sends $[\![\widetilde{\mathcal{T}}(\mathcal{P})]\!]$ and $[\![\widetilde{\mathcal{T}_V}(\mathcal{P})]\!]$ to the CS, and they jointly finish the judging for each path. On receiving $[\![\widetilde{\mathcal{T}}(\mathcal{P})]\!]$ and $[\![\widetilde{\mathcal{T}_V}(\mathcal{P})]\!]$, the CS decrypts the ciphertext and unpacks the data. For each path $\mathcal{P}_j$, the RS holds $\mathcal{A}$ and $\mathcal{A}'$; the CS holds $\widetilde{\mathcal{T}}(\mathcal{P}_j)$ and $\widetilde{\mathcal{T}_V}(\mathcal{P}_j)$; and they execute the garbled circuit TC as shown in Figure 4, in which the *SCMP* circuit is shown in Figure 5. CMP, MUX, SUB, and MIN denote a comparator circuit, a multiplexer circuit, a subtractor circuit, and a minimum circuit, respectively. We present the details of the circuit TC in Algorithm 3. The circuit first restores the real values from blind values (1) and then compares the arriving time and the deadline time for each point (2–8). Only if all deadlines are satisfied will the circuit output the total time consumption of the ride-sharing process (9–10), or the circuit outputs a big enough integer $\{1\}^p$ (12).

Then for all paths $\mathcal{P}_j$, the RS and CS jointly execute a path-planning circuit C_{pp} to select the path with minimum traveling time. We present the circuit C_{pp} in Figure 6.

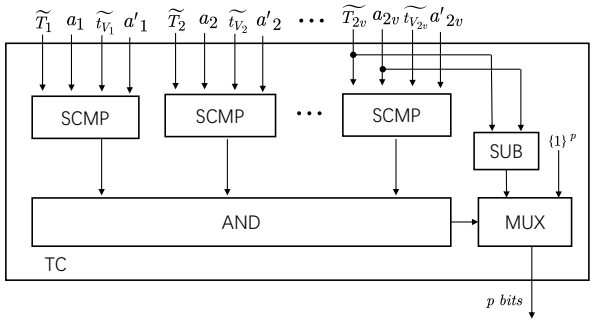

Figure 4. The time-comparing circuit TC.

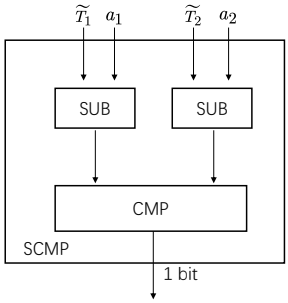

Figure 5. The $SCMP$ circuit.

Algorithm 3 The time-comparing circuit TC.

Input: $\widetilde{\mathcal{T}}(\mathcal{P}_1)$, $\widetilde{\mathcal{T}_V}(\mathcal{P}_1)$, $\mathcal{A}$, $\mathcal{A}'$
Output: $\{1\}^p$ or T_{2v}

Compute $T_i = \widetilde{T_i} - a_i$ and $t_{V_i} = \widetilde{t_{V_i}} - a'_i$
for $1 \leqslant i \leqslant 2v$ **do**
  if $T_i <= t_{V-i}$ **then**
    $\gamma_i = 1$
  else
    $\gamma_i = 0$
  end if
end for
if $AND\left(\{\gamma_i\}_i^{2v}\right) = 1$ **then**
  return T_{2v}
else
  return $\{1\}^p$
end if

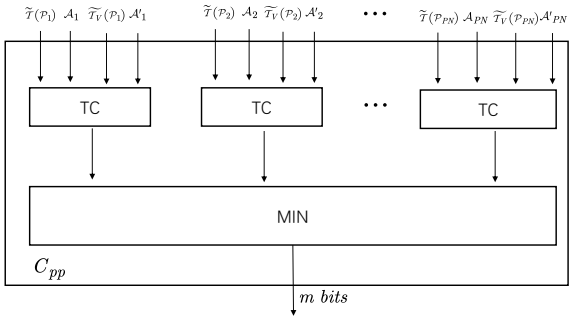

Figure 6. The path-planning circuit C_{pp}.

Phase 2. (Riders Selection) In this phase, for all rider combinations $\mathbb{C} = \left\{C_j\right\}_{j=1}^{CN}$, the system compares their ATT and outputs the best combination, in which CN is the

total number of combinations, and ATT is the total detouring time of all riders. First, the RS computes the $[\![ATT]\!]$ of each combination. For each rider r_i, the detouring time is the traveling time difference between the time consumption of driving in the current path and directly to the destination. The ATT of C_j is presented as the following equation:

$$ATT_j = \sum_{i=1}^{v} \left(T_i^{real} - T_i^{direct} \right) \tag{18}$$

For example, the ATT of the path in Figure 3 can be computed as:

$$\begin{aligned} ATT = \sum_{i=1}^{3} \left(T_i^{real} - T_i^{direct} \right) = \\ (t_3 - t_3) + \left(t_5 + t_6 - t_{r_2.s \to r_2.d} \right) + \\ \left(t_2 + t_3 + t_4 + t_5 - t_{r_3.s \to r_3.d} \right) \end{aligned} \tag{19}$$

Moreover, we can obtain the corresponding ciphertext as:

$$[\![ATT_j]\!] = \prod_{i=1}^{v} \left([\![T_i^{real}]\!] [\![T_i^{direct}]\!]^{-1} \right) \tag{20}$$

For each combination C_j, the RS computes the $[\![ATT_j]\!]$, then packs them and obtains $S\left(\left[[\![ATT_j]\!] \right]_{j=1}^{CN} \right)$. As with the previous blinding operations, the RS generates CN random integers $[a_1, a_2, \cdots a_{CN}]$, then computes the blinding ciphertexts $S\left(\left[[\![\widetilde{ATT_j}]\!] \right]_{j=1}^{CN} \right)$ and sends it to the CS. After decryption and unpacking operations, the RS and the CS execute the *RCS* algorithm to select out the best rider combination as Algorithm 4. Finally, the RS obtains the result and establishes a secure communication channel between the driver and the riders in combination so that they can communicate for pick-up details.

Algorithm 4 Rider Combination Selection (*RCS*).

Input: RS: $[a_i]_{i=1}^{CN}$; CS: $\left[\widetilde{ATT_j} \right]_{j=1}^{CN}$
Output: The combination with minimum ATT

```
k* = −1
for 1 ⩽ k ⩽ CN do
  if k* = −1 then
    k* = k
  else
    if SCMP(ATT̃_k, a_k, ATT̃_k*, a_k*) == 0 then
      k* = k
    end if
  end if
end for
```

5. Privacy and Security Analysis

In this section, we present the analysis of security. Our goal is to ensure no entity could learn the precise location of users, but for the utility and efficiency, we allow the RS to access the zone-based location. We make more discussion about the balance of privacy and utility in the following section. In *pShare*, we mainly consider the security against the RS and the CS. We adopt and modify the notion of adaptive security from [8,19].

Leakage function L_{access}: Given the map of the city $\mathcal{M}$, a set of riders $\mathcal{R} = (r_1, \cdots, r_n)$, and a set of drivers $\mathcal{D} = (d_1, \cdots d_n)$, we define the leakage function $\mathcal{L}_{access} = AP_{access}(\mathcal{M}, d, \mathcal{R})$, where $AP(\mathcal{M}, d, \mathcal{R})$ is the access pattern.

Definition 2 (Access Pattern)**.** *Given the map $\mathcal{M}$, a series of drivers $\mathcal{D}$ and a series of riders $\mathcal{R}$, we define the access pattern as $AP(\mathcal{M}, \mathcal{D}, \mathcal{R}) = \Big(z(\mathcal{R}), z(d), \{p*_x\}_{C_x \in \mathbb{C}}, C* \Big)$, where $z(\mathcal{R})$ and $z(d)$ are the zone information corresponding to $\mathcal{R}$ and d, respectively; $p*_x$ is the best path of the combination C_x; and $C*$ is the final selected combination with the minimum ATT.*

Definition 3 (Adaptive Semantic Security)**.** *Let pShare = (RRS, DLU, RFA, PPA, RCS) be our privacy-preserving ride-sharing scheme, representing the process of Ride Request Submit, Driver Location Update, Request Filtering Algorithm, Path-Planning Algorithm, and Rider Combination Selection, respectively. We consider the following experiments with probabilistic polynomial time (PPT) adversary $\mathcal{A}$, a challenger $\mathcal{C}$, a simulator $\mathcal{S}$, and the leakage function $\mathcal{L}_{access}$.*

$\text{Real}_{\mathcal{A},\mathcal{C}}(1^\lambda)$: Firstly, $\mathcal{C}$ runs the initialization to set the system parameters. Then $\mathcal{A}$ outputs a set of ride requests $\mathcal{R}$. For each rider $r_i \in \mathcal{R}$, $\mathcal{C}$ and $\mathcal{A}$ execute a simulation of *RRS*, with $\mathcal{C}$ acting as the rider and $\mathcal{A}$ acting as the RS. Then $\mathcal{A}$ generates a polynomial number of adaptively chosen drivers $\mathcal{D}$. For each driver $d_j \in \mathcal{D}$, $\mathcal{C}$ and $\mathcal{A}$ execute a simulation of *DLU*, in which $\mathcal{C}$ and $\mathcal{A}$ play the role of the driver and the RS, respectively. Afterward, $\mathcal{A}$ acts the *RS* and runs *RFA*, *PPA*, and *RCS*, and finally outputs γ as the result of the experiments.

$Ideal_{\mathcal{A},\mathcal{S}}(1^\lambda)$: Firstly, $\mathcal{S}$ generates system parameters in the process initialization. Then $\mathcal{A}$ outputs a set of riders $\mathcal{R}$ for each rider $r_i \in \mathcal{R}$, and $\mathcal{A}$ executes a simulation of *RRS*, with $\mathcal{S}$ acting as the rider and $\mathcal{A}$ acting as the RS. Then $\mathcal{A}$ generates a polynomial number of adaptively chosen drivers $\mathcal{D}$. For each driver d_j, $\mathcal{S}$ is given the leakage function $\mathcal{L}_{access} = AP_{access}(\mathcal{M}, d, \mathcal{R})$ and executes a simulation of *DLU* with $\mathcal{A}$, in which $\mathcal{C}$ and $\mathcal{A}$ plays the role of the driver and the RS, respectively. Afterward, $\mathcal{A}$ acts as the RS and runs *RFA*, *PPA*, and *RCS*, and outputs γ' as the result of the experiment.

We define the advantage of $\mathcal{A}$ and in the experiment as $Adv(\lambda) = |Pr(\gamma = \gamma') - 1/2|$. Then we say *pShare* is adaptively L_{access}-semantic secure if for all PPT adversaries $\mathcal{A}$, there exists a PPT simulator s.t. $Adv(\lambda)$ is negligible.

According to the above definitions, we analyze the user privacy of *pShare* as follows:

Theorem 1. *If Paillier cryptosystem is semantically secure, pShare has rider privacy in RRS algorithm.*

Proof. Supposing *pShare* cannot satisfy user privacy in *RRS* and there is a PPT adversary who has a non-negligible advantage to distinguish the outputs of the $\text{Real}_{\mathcal{A},\mathcal{C}}(1^\lambda)$ from the outputs of the $Ideal_{\mathcal{A},\mathcal{S}}(1^\lambda)$. We use $\mathcal{A}$ to break user privacy in *RRS* algorithm. First, we generate the public key *pk* and randomly generate two plaintext ride-sharing requests

$$\Big(z(s_1), z(d_1), t_{s_1 \to AP(s_1)}, t_{d_1 \to AP(d_1)}, t_{s_1}, t_{d_1} \Big)$$

and

$$\Big(z(s_2), z(d_2), t_{s_2 \to AP(s_2)}, t_{d_2 \to AP(d_2)}, t_{s_2}, t_{d_2} \Big)$$

in which $z(s_1) = z(s_2), z(d_1) = z(d_2)$. Then we send the requests to the challenger, and the challenger randomly chooses $b \in \{0, 1\}$ and returns

$$R_b \;=\; \Big(z(s_b), z(d_b), [\![t_{s_b \to AP(s_b)}]\!], [\![t_{d_b \to AP(d_b)}]\!], [\![t_{s_b}]\!], [\![t_{d_b}]\!] \Big)$$

with the RRS algorithm. We send R_b to the adversary $\mathcal{A}$, who gives his guess b'. So the advantage for $\mathcal{A}$ to distinguish two requests is $Adv(\lambda) = |Pr(b = b') - 1/2|$, which is also non-negligible. Based on the non-negligible advantage to distinguish the ciphertext requests encrypted by Paillier cryptosystem, we break the semantic security of Paillier cryptosystem. It is contradictory to the assumption of the theorem. This completes the proof. □

Theorem 2. *If Paillier cryptosystem is semantically secure, pShare has user privacy in the DLU algorithm.*

Similar to the proof of Theorem 1, we can prove Theorem 2 that there is no PPT adversary who has a non-negligible advantage to distinguish two outputs of *DLU* algorithm with drivers from the same zone.

Theorem 3. *If the garbled circuit C_{pp} is secure under the semi-honest adversary model, then the PPA algorithm is secure under the semi-honest model.*

Proof. During the PPA algorithm, we simulate the view of the RS and the CS, respectively. Let S_{CS} simulate the view of the CS. The RS first blinds the ciphertexts by adding them with a set of randomly generated integers. Then the CS receives the blinded ciphertexts and decrypts them. Since the random integers are big enough, the CS cannot distinguish between two blinded values. That is to say, if S_{CS} randomly picks an integer from the range $\left(2^{k-1}+1, 2^k+2^{l+1}-2\right)$, encrypts it, and sends it to the CS, there is no PPT adversary who can distinguish the blinded plaintext from random integer with non-negligible probability. □

Then C_{pp} outputs the path-planning results, which is an m-bits integer that indicates which path to choose. We use S_{RS} to simulate the RS's view. For each combination, S_{RS} randomly chooses a path that meets Equations (11) and (12) and sends the number of the path as the result of *PPA* algorithm to the RS. There is no PPT adversary who can distinguish the best path, which is a sequence of encrypted location information, from the randomly chosen one.

The process of circuit execution is also secure. The security proof of C_{pp} can be found in [20]. We complete the proof of the theorem.

Theorem 4. *If the garbled circuit SCMP is secure under the semi-honest adversary model, then the RCS algorithm is secure under the semi-honest model.*

Proof. Similar to the proof of Theorem 3, from the view of the CS, the adversary $\mathcal{A}$ cannot distinguish the blinded plaintext from the random integer. Then S_{RS} randomly selects a number from all eligible combinations as the result of combination selection. The randomly selected number is also indistinguishable from the real result. □

Theorem 5. *pShare is adaptively semantically secure against semi-honest adversaries.*

Proof. We first describe a PPT simulator $\mathcal{S}$ to simulate the whole process of *pShare* such that any PPT adversary $\mathcal{A}$ cannot distinguish the results of $\text{Real}_{\mathcal{A},\mathcal{C}}(1^\lambda)$ and $Ideal_{\mathcal{A},\mathcal{S}}(1^\lambda)$ with non-negligible advantages. Given $\mathcal{L}_{access} = AP_{access}(\mathcal{M}, d, \mathcal{R})$, $\mathcal{S}$ randomly generates a set of drivers $\hat{\mathcal{D}}$, and for each $\hat{d} \in \hat{\mathcal{D}}$, $\mathcal{S}$ randomly generates a set of riders around the driver. Then $\mathcal{S}$ simulates riders to run *RRS* algorithm and drivers to run *DLU* algorithm with $\mathcal{A}$ acting as the RS, and then running *RFA*, *PPA*, and *RCS* algorithms. □

According to Theorems 1 and 2, $\mathcal{A}$ cannot distinguish $\hat{d} \in \hat{\mathcal{D}}$ from $d \in \mathcal{D}$ and $\hat{r} \in \hat{\mathcal{R}}$ from $r \in \mathcal{R}$ during *RRS* and *DLU* algorithm execution. Since there is no data transmission in *RFA*, $\mathcal{A}$ cannot obtain any useful information. According to Theorems 3 and 4, $\mathcal{A}$ still cannot distinguish $\hat{d} \in \hat{\mathcal{D}}$ from $d \in \mathcal{D}$ and $\hat{r} \in \hat{\mathcal{R}}$ from $r \in \mathcal{R}$ in *PPA* and *RCS* algorithms.

In summary, *pShare* is adaptively $\mathcal{L}_{access}$-semantic secure against semi-honest adversaries. We complete the proof.

6. Evaluation

We compare *pShare* with schemes [7,8,14,19,21] under the same threat model and summarize the differences between them in the Table 1. Compared to the other privacy-preserving ORH schemes, *pShare* is able to match the driver with a group of riders and

computes for the least-detouring path while protecting users' privacy. In this section, we design an experiment to show the utility and efficiency of *pShare*. We present the experimental setup in Section 6.1 and experiment results in Section 6.2.

Table 1. Comparison with existing schemes.

Description	*pShare*	*ORide* [7]	*pRide* [8]	*PSRide* [19]	*PrivatePool* [14]	*PGRide* [21]
Ride-sharing	Yes	No	No	Yes	Yes	Yes
Traveling cost	Road traveling time	Euclidean distance	Road traveling time	Road traveling time	Euclidean distance	Euclidean distance
Group matching	Yes	No	No	No	No	Yes
Minimum ATT	Yes	\	\	Yes	Yes	No

6.1. Experimental Setup

In this section, we evaluate our experiments on the road network of an about 50×50 km map of Beijing, which is generated by OpenStreetMap [22]. Since the distribution of riders and drivers is always dynamic, we generate the ride-sharing requests and driver locations using the tool TaxiQueryGenerator [23], whose outputs conform to Beijing's real spatio-temporal distribution. We implement Paillier cryptosystem by an open-source library, jpaillier [24], and Yao's garbled circuits by Faster-GC [25]. During the experiment, we set the modulus of the Paillier cryptosystem to 1024 bits, the bit-length of a blinded value k to 32 bits, the bit-length of a domain value l to 16 bits, and the bit-length of a wire label to 80 bits. All our experiments are run on a PC with 8 GB memory and an Intel-8265U CPU of 1.8 GHz.

We divide the selected area of the map into grids and set up all the parameters of the *pShare* ride-sharing system. Then we randomly initialize the locations of the drivers and the riders in the selected area, meanwhile generating the corresponding pick-up time and drop-off time of the riders and the available time of the drivers. Then the pick-up deadline and drop-off deadline are computed as $t_s = t_{cur} + 600$ s, $t_d = t_s + t_{s \to d} + 600$ s.

The default patience time of a rider is 30 min, i.e., the rider will turn to other options for traveling rather than ride-sharing beyond this amount of time. We choose the generated requesting datasets of period 10:00–11:00 and the drivers' datasets of period 10:30–11:00 as the experimental datasets. We first read all rider requesting data, and for each record of driver datasets, we read it sequentially and apply the ride-sharing matching between the driver and riders within the patience time. The result of the ride-sharing matching may be a failure or success. To evaluate the accuracy of *pShare*, we record the matched riders if matching successfully and compare the results to another experiment *pShare-GT*, which takes ground truth to replace the estimated traveling time. Besides, we also measure the overhead of the experiments and record it in the table.

6.2. Experiment Results

Accuracy: In the experiments, we use two metrics to evaluate the accuracy of the system, which are Successful Matching Rate (SMR) and Accurate Matching Rate (AMR), respectively. We compute SMR as $SMR = N_{suc}/N_{tot}$, in which N_{suc} indicates the number of successful matching and N_{tot} indicates the total number of matching. We compute AMR as $AMR = N_{acc}/N_{tot}$; N_{acc} indicates the number of matches where the results of *pShare* are the same as the results of *pShare-GT*. We use the metric AMR to evaluate the accuracy of the scheme to find the minimum detouring route. Since there is a trade-off between the matching accuracy and the user privacy, we introduce the Average Number of POI (ANP) in one zone to represent the privacy level of current zone size, in which POI is point of

interest, corresponding to the location points of the riders' probable destinations. Therefore, the users' precise locations are disclosed in the probability of $1/ANP$. We compute and compare the variants of SMR, AMR, and ANP by varying the zone scale and the number of riders to be searched $\mathcal{R}_{nearby}$. As shown in Figure 7, SMR has little association with the zone size, but AMR increases remarkably with zone getting smaller, which indicates the accuracy of the zone-based time estimation method is highly relevant to the zone scale. Besides, there is a drastic reduction in APN with smaller zone size, so it is necessary to properly set the zone scale considering protecting users' privacy at an acceptable level. Furthermore, the rider number to be searched has an influence on SMR but has no evident impact on AMR/SMR.

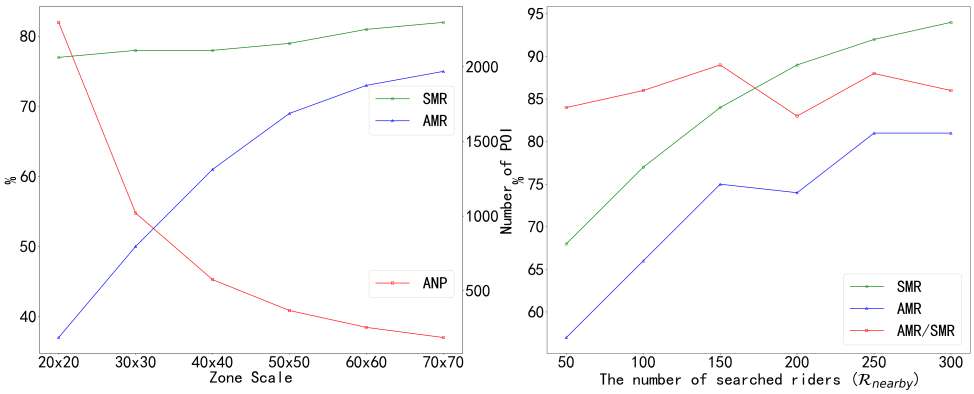

Figure 7. SMR, AMR, and ANP under different zone scales when the number of searched riders is 100 (**left**) and SMR, AMR, and AMR/SMR under different sizes of $\mathcal{R}_{nearby}$ when the zone scale is 50×50 (**right**).

Efficiency: We take communication and computation cost to evaluate the efficiency of *pShare* scheme. During the ride-sharing, the computation cost mainly includes the encryption of riders and drivers in the process of Request Submit and Location Submit, the packing/unpacking operations and the garbled circuit execution in the process of ride-sharing matching.

As shown in Table 2, for the rider, the communication cost is always 0.51 KB, and the computation cost is 26 ms. For a driver with no onboard rider, the communication cost is 0.13 KB. For a driver with onboard riders, the submitted information should also contain the requests of riders onboard. As for the server (includes both the ride-sharing server and the crypto server), the communication cost and the computation cost are subject to the size of searched riders. The communication cost of the server is mainly from the packed ciphertexts the RS sends to the CS, and the computation cost mainly contains the cost of ciphertexts packing and computation of the RS, the ciphertexts unpacking of the CS, and the execution time of the garbled circuit. From the table, we can see that the total execution time of the server is about 30 s if the number of riders to be searched is 100. The responding time is acceptable for users with an appropriate searching scale. Moreover, we especially evaluate the computation cost of the garbled circuit with different searched riders and the number of available seats. From Figure 8, we can find the execution time of the garbled circuit takes up most of the total computation cost, and the cost rises faster as the size of $\mathcal{R}_{nearby}$ and v become bigger.

Table 2. System overhead for per ride-sharing matching under different sizes of $\mathcal{R}_{nearby}$.

Size of ($\mathcal{R}_{nearby}$)	*pShare* Scheme					
	Rider		Driver		Server	
	Comm. (KBs)	Comp. (ms)	Comm. (KBs)	Comp. (ms)	Comm. (MBs)	Comp. (s)
25	0.51	26	$0.13 + 0.52 * OBR_d$	11	0.48	3.2
50	0.51	26	$0.13 + 0.52 * OBR_d$	11	1.64	10.1
75	0.51	26	$0.13 + 0.52 * OBR_d$	11	3.66	15.8
100	0.51	26	$0.13 + 0.52 * OBR_d$	11	6.25	23.3
125	0.51	26	$0.13 + 0.52 * OBR_d$	11	10.76	30.3
150	0.51	26	$0.13 + 0.52 * OBR_d$	11	18.48	39.4

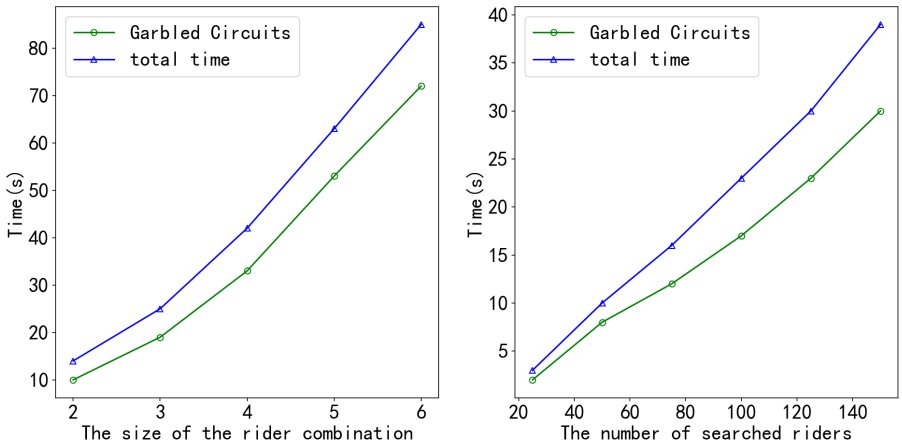

Figure 8. Per matching computation cost of servers under different size of rider combination and number of searched riders.

7. Related Work

7.1. Privacy-Preserving Ride-Matching

Many approaches have been proposed to preserve users' privacy in location-based services. For the cryptographic primitives, there is homomorphic encryption, secure multi-party computation, and private set intersection. Besides, some non-encrypted techniques include location perturbation [26], dummy location generation [27], and spatial cloaking [28]. Based on these methods, a few privacy-preserving ride-matching schemes have been proposed. A privacy-enhanced scheme, PrivateRide [6], was proposed by Pham et al., which provides anonymity for users' locations. Later, they extended it to ORide [7], which is based on somewhat homomorphic encryption and enables the matching more efficiently while protecting users' privacy. Besides, Luo et al. proposed a privacy-preserving ride-matching scheme based on road networks [8], which can perform matching more accurately. Considering side-channel attacks, they also presented a private and efficient scheme with Intel SGX enclave, which provides a hardware-enforced Trusted Execution Environment [29]. However, all these schemes are ride-matching schemes, which do not support ride-sharing matching.

7.2. Privacy-Preserving Ride-Sharing

There are also some relevant schemes for ride-sharing. SRide is a privacy-preserving dynamic ride-sharing system, which computes the ride-sharing results based on users' spatiotemporal data. PRIS [13] utilizes PHE [15] to find eligible matching between riders and drivers. Sherif et al. proposed a scheme [30] that computes the similarity of riders' trips on ciphertexts to match riders. The FICA scheme [31] ensures data privacy and reliability using edge computation and blockchain techniques. PrivatePool [14] constructs a

distributed and privacy-enhanced ride-sharing system with PSI [32] protocol and SHE [33] technique. PSRide [19] utilizes PHE [15] and the Garbled Circuit technique to build a privacy-preserving scheme that supports flexible ride-sharing matching. Yu et al. proposed PGRide [21], in which they designed an encrypted aggregate distance computation approach by using SHE and ciphertexts packing technique. Through the approach, the aggregate distances can be computed in encrypted form. Howerver, none of these schemes can support group matching and meanwhile find minimum detouring matching results over road traveling time.

8. Conclusions

In this paper, we propose a privacy-preserving ride-sharing scheme *pShare* for ORH service. *pShare* supports privacy-preserving ride-sharing between multiple riders, the matching results of which have a minimum ATT. To achieve this, the riders and drivers need to submit their encrypted location information to the ORH server, and the ORH server applies secure computing and compares matching protocols with the crypto server based on the garbled circuit and the data packing technique. They first compute the minimum path for each eligible rider combination, and finally, they compare and select the best combination with minimum ATT. We prove that *pShare* scheme achieves user privacy and is adaptively semantically secure under the random oracle model. The experimental results have shown that *pShare* achieves both accuracy and practical efficiency.

Author Contributions: Conceptualization, Y.L.; methodology, J.H.; software, J.H.; validation, Y.L., B.H. and J.L.; formal analysis, J.H.; writing—original draft preparation, J.H.; writing—review and editing, J.H. and Y.L.; visualization, J.H.; supervision, M.X. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Nature Science Foundation of China (No.62072466, 61872372, 62102429, 62102422) and the NUDT Research Grant (No. ZK19-38).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data provided in the article is public data.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Cramer, J.; Krueger, A.B. *Disruptive Change in the Taxi Business: The Case of Uber*; Working Paper 22083; American Economic Review: Pittsburgh, PA, USA, 2016. [CrossRef]
2. Available online: https://www.didiglobal.com/ (accessed on 23 October 2021).
3. Belk, R. You are what you can access: Sharing and collaborative consumption online. *J. Bus. Res.* **2014**, *67*, 1595–1600. [CrossRef]
4. Caulfield, B. Estimating the environmental benefits of ride-sharing: A case study of Dublin. *Transp. Res. Part D Transp. Environ.* **2009**, *14*, 527–531. [CrossRef]
5. Available online: https://www.scmp.com/tech/big-tech/article/3139786/china-takes-didi-app-stores-two-days-after-beijing-announces (accessed on 23 October 2021).
6. Pham, A.; Dacosta, I.; Jacot-Guillarmod, B.; Huguenin, K.; Hajar, T.; Tramèr, F.; Gligor, V.; Hubaux, J.P. Privateride: A privacy-enhanced ride-hailing service. *Proc. Priv. Enhancing Technol.* **2017**, *2017*, 38–56. [CrossRef]
7. Pham, A.; Dacosta, I.; Endignoux, G.; Pastoriza, J.R.T.; Huguenin, K.; Hubaux, J.P. ORide: A Privacy-Preserving yet Accountable Ride-Hailing Service. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, Canada, 16–18 August 2017; USENIX Association: Vancouver, BC, USA, 2017; pp. 1235–1252.
8. Luo, Y.; Jia, X.; Fu, S.; Xu, M. pRide: Privacy-Preserving Ride Matching Over Road Networks for Online Ride-Hailing Service. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 1791–1802. [CrossRef]
9. Yu, H.; Shu, J.; Jia, X.; Zhang, H.; Yu, X. lpRide: Lightweight and Privacy-Preserving Ride Matching Over Road Networks in Online Ride Hailing Systems. *IEEE Trans. Veh. Technol.* **2019**, *68*, 10418–10428. [CrossRef]
10. Wang, X.; Mu, Y.; Chen, R. One-round privacy-preserving meeting location determination for smartphone applications. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 1712–1721. [CrossRef]
11. Su, J.S.; Cao, D.; Wang, X.F.; Sun, Y.P.; Hu, Q.L. Attribute-based encryption schemes. *J. Softw.* **2011**, *22*, 1299–1315. [CrossRef]

12. Aïvodji, U.M.; Huguenin, K.; Huguet, M.J.; Killijian, M.O. Sride: A privacy-preserving ridesharing system. In Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks, Stockholm, Sweden, 18–20 June 2018; pp. 40–50.
13. He, Y.; Ni, J.; Wang, X.; Niu, B.; Li, F.; Shen, X. Privacy-preserving partner selection for ride-sharing services. *IEEE Trans. Veh. Technol.* **2018**, *67*, 5994–6005. [CrossRef]
14. Hallgren, P.; Orlandi, C.; Sabelfeld, A. PrivatePool: Privacy-preserving ridesharing. In Proceedings of the 2017 IEEE 30th Computer Security Foundations Symposium (CSF), Santa Barbara, CA, USA, 21–25 August 2017; pp. 276–291.
15. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1999; pp. 223–238.
16. Yao, C.C. How to generate and exchange secrets. In Proceedings of the 27th Annual Symposium on Foundations of Computer Science, Toronto, ON, Canada, 27–29 October 1986.
17. Huang, Y.; Evans, D.; Katz, J.; Malka, L. Faster secure two-party computation using garbled circuits. In Proceedings of the USENIX Security Symposium, San Francisco, CA, USA, 8–12 August 2011; Volume 201, pp. 331–335.
18. Kolesnikov, V.; Schneider, T. Improved garbled circuit: Free XOR gates and applications. In *International Colloquium on Automata, Languages, and Programming*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 486–498.
19. Yu, H.; Jia, X.; Zhang, H.; Yu, X.; Shu, J. PSRide: Privacy-preserving shared ride matching for online ride hailing systems. *IEEE Trans. Dependable Secur. Comput.* **2019**, *18*, 1425–1440. [CrossRef]
20. Lindell, Y.; Pinkas, B. A proof of security of Yao's protocol for two-party computation. *J. Cryptol.* **2009**, *22*, 161–188. [CrossRef]
21. Yu, H.; Zhang, H.; Yu, X.; Du, X.; Guizani, M. PGRide: Privacy-Preserving Group Ridesharing Matching in Online Ride Hailing Services. *IEEE Internet Things J.* **2020**, *8*, 5722–5735. [CrossRef]
22. Available online: www.openstreetmap.org (accessed on 23 October 2021).
23. Available online: www.cs.uic.edu (accessed on 23 October 2021).
24. Available online: https://github.com/kunerd/jpaillier (accessed on 23 October 2021).
25. Available online: https://github.com/yhuang912/FastGC (accessed on 23 October 2021).
26. Shokri, R.; Theodorakopoulos, G.; Le Boudec, J.Y.; Hubaux, J.P. Quantifying location privacy. In Proceedings of the 2011 IEEE Symposium on Security and Privacy, Oakland, CA, USA, 22–25 May 2011; pp. 247–262.
27. Niu, B.; Li, Q.; Zhu, X.; Cao, G.; Hui, L. Achieving k-anonymity in privacy-aware location-based services. In Proceedings of the IEEE INFOCOM 2014—IEEE Conference on Computer Communications, Toronto, ON, Canada, 27 April–2 May 2014.
28. Damiani, M.L.; Bertino, E.; Silvestri, C. The PROBE Framework for the Personalized Cloaking of Private Locations. *Trans. Data Priv.* **2010**, *3*, 123–148.
29. Luo, Y.; Jia, X.; Duan, H.; Wang, C.; Xu, M.; Fu, S. pRide: Private Ride Request for Online Ride Hailing Service with Secure Hardware Enclave. In Proceedings of the 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), Phoenix, AZ, USA, 24–25 June 2019; pp. 1–10. [CrossRef]
30. Sherif, A.B.; Rabieh, K.; Mahmoud, M.M.; Liang, X. Privacy-preserving ride sharing scheme for autonomous vehicles in big data era. *IEEE Internet Things J.* **2016**, *4*, 611–618. [CrossRef]
31. Li, M.; Zhu, L.; Lin, X. Efficient and privacy-preserving carpooling using blockchain-assisted vehicular fog computing. *IEEE Internet Things J.* **2018**, *6*, 4573–4584. [CrossRef]
32. Freedman, M.J.; Nissim, K.; Pinkas, B. *Efficient Private Matching and Set Intersection*; Springer: Berlin/Heidelberg, Germany, 2004.
33. Kerschbaum, F. Outsourced private set intersection using homomorphic encryption. In Proceedings of the ASIACCS '12—7th ACM Symposium on Information, Computer and Communications Securit, Seoul, Korea, 2–4 May 2012.