This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security

pRide: Privacy-preserving Ride-matching over Road Networks for Online Ride Hailing Service

Yuchuan Luo, Xiaohua Jia, Shaojing Fu, and Ming Xu

Abstract—Online Ride Hailing (ORH) service such as Uber and Didi Chuxing can provide on-demand transportation service to users via mobile phones, which brings great convenience to people's daily life. Along with the convenience, high privacy concerns also raised when using ORH service since users and drivers must share their real-time locations with the ORH server, which results in the leakage of the mobility patterns and additional privacy of users and drivers. In this paper, we propose a privacy-preserving ride-matching scheme, called *pRide*, for ORH service. *pRide* allows an ORH server to efficiently match rider and drivers based on their distances in the road network without revealing the location privacy of riders and drivers. Specifically, we make use of road network embedding technique together with cryptographic primitives, and design a scheme to securely and efficiently estimate the shortest distances between riders and drivers in road networks approximately. Moreover, by incorporating garbled circuits, the proposed scheme is able to output the nearest driver around a rider. We implement the scheme and evaluate it on representative real-world datasets. The theoretical analysis and experimental results demonstrate that *pRide* achieves efficient, secure yet accurate ride-matching for ORH service.

Index Terms—Location-based service, Privacy-preserving, Ride hailing system, Ride-matching.

1 Introduction

As an emerging instantiation of sharing economy, Online Ride-Hailing (ORH) service such as Uber and Didi Chuxing provides on-demand transportation service to riders (i.e., passengers). Unlike traditional taxi service, ORH allows a rider to request a ride via its mobile phone through a few finger touches rather than standing in the street and waiting for an available taxi to come by. With its great convenience, ORH service has become increasingly popular and serves millions of people every day across the world. An expectation shows that the ride-hailing industry will grow eightfold in the next decade to nearly three hundred billion dollars [1].

Despite its convenience and promising market scale, ORH service raises high security concerns. The ORH servers collect a vast amount of sensitive information from the riders and drivers every day, which includes real-time locations of riders and drivers, pick-up and drop-off locations, and time duration of rides, etc. Utilizing these sensitive data, mobility patterns of riders and drivers can be constructed and further private information can be mined by the ORH servers.

Although there are many privacy-enhancing solutions designed for ride sharing services [2], [3], [4], [5], [6], few of them considered the privacy and security of ORH service. To protect privacy of users in ORH service, Phame *et al.* first proposed *PrivateRide* [7], a private ride requesting scheme, and later extended it to *ORide* [8], which enables the ORH server to match riders and drivers without knowing their location privacy. For efficiency reason, their scheme utilizes the Euclidean distance metric for ride-matching. However, in ORH service, distance between a rider and a driver actually depends on the roads connecting them because vehicles are constrained to move along the roads. Using Euclidean distance in ride-matching will produce bad ride-matching accuracy. Our experiment with real-world datasets shows that Euclidean distance metric incurs nearly 25% false hits for the nearest drivers. In addition, their scheme requires the drivers to encrypt their locations using ephemeral public key from potential riders and send the ciphertexts to the ORH server for all ride requests. Thus, drivers and riders in their scheme are involved in heavy cost in both computation and communication.

We are motivated to investigate the issue of privacy-preserving yet practical ORH systems. Our aim is to enable the ORH server to efficiently and accurately match riders and drivers while keeping their locations private against the server. It is not an easy job to realize such a secure system that achieves both ride-matching accuracy and efficiency. A trivial solution to achieve accuracy is to allow the riders and drivers to encrypt their locations together with the road network using fully homomorphic encryptions such as [9], and then let the server perform the ride-matching over the ciphertexts using shortest distance computation algorithms such as Dijkstra's algorithm. However, computing shortest distance homomorphically on encrypted data is extremely time consuming and will incur an unacceptable delay for the riders. As reported in [9], it takes a typical server several minutes to compute a shortest distance for a pair of nodes in a medium size graph that contains hundreds of thousands of nodes. To improve efficiency, Meng *et al.* [10] resorted to distance oracles [11] and proposed *GREC*, a graph encryption scheme that returns approximate shortest distances. But, their scheme produces bad accuracy for ride-matching over road networks as shown in our experiments.

To address these challenges, we propose *pRide*, a privacy-preserving ride-matching scheme over road networks for ORH service. *pRide* enables the ORH server to match riders

- *Y. Luo, S. Fu and M. Xu are with the college of Computer, National University of Defense Technology, Changsha, CN, 410073.*
- *X. Jia is with City University of Hong Kong, Hong Kong SAR.*
- *Corresponding author: Shaojing Fu (e-mail: fushaojing@nudt.edu.cn).*

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security

and drivers based on their distances in the road network without learning their locations. To improve the efficiency, instead of performing ride-matching directly over encrypted road network, we convert the original road networks into a higher dimensional space using a technique called road network embedding [12], so that ride-matching in the embedding space can be computed efficiently through our carefully tailored cryptographic primitives. As the result, *pRide* overcomes the limitations in the previous works, and outperforms these solutions in terms of accuracy and efficiency without sacrificing location privacy of riders and drivers. In summary, our contributions are as follows:

- We propose a privacy-preserving ride-matching scheme that enables the ORH server to efficiently match riders with drivers based on their road network distances without learning the locations of the riders and drivers.
- We implement a prototype of our proposed privacy-preserving ride-matching scheme. Our implementation utilizes homomorphic encryption as the encryption primitive and Yao's garbled circuit for the distance comparisons.
- We evaluate the scheme using real-world datasets. The experimental results demonstrate that the accuracy of our scheme significantly outperforms previous schemes such as Euclidean distance based *ORide* [8]. Moreover, the processing time of one ride request is at acceptable level, which takes less than 10 seconds for 99% ride-matching accuracy.

The rest of this paper is organized as follows: Section 2 formalizes of the problem and introduces necessary preliminaries. Section 3 describes the method of road network embedding that allows efficient approximate distances evaluation for riders and drivers in the road network. In section 4, we present the details of our *pRide*, followed by the security analysis and performance evaluation in Section 5 and 6, respectively. Finally, we review the related work in Section 7 and conclude the whole paper in Section 8.

2 Problem Statement

2.1 System and Threat Model

In this paper, we consider an ORH system that allows the riders to ask for rides from the drivers through the ORH server. As shown in Fig. 1, the system consists of four entities: an ORH server, a Crypto Provider (CP), the riders and the drivers. The drivers periodically update their locations to the ORH server. The ORH server receives ride requests from the riders and always matches the riders and the drivers by finding the nearest driver for every incoming rider according to their road distances. Our goal is to preserve the location privacy of the riders and drivers during the matching process, namely ensuring that the ORH server performs the ride-matching without learning the locations of the riders and drivers. We introduce a third party, Crypto Provider (CP), to setup the cryptographic parameters and facilitate the ride-matching computation.

In our system, we assume that the ORH server is *honest-but-curious* in the sense that it always honestly follows the scheme but is curious on the locations of riders and drivers. Therefore, one of the most important goals in the design of *pRide* is to prevent the ORH server from learning anything about the locations of the riders and drivers except the ride-matching result. Similarly, we also assume that the CP is *honest-but-curious*, and hence the design of *pRide* should also guarantee that no private information will be leaked to the CP. In addition, we make the following system assumptions:

1) No collusion between the ORH server and the CP. This is a reasonable assumption in practice, because the ORH server and CP are managed by different parties.
2) The road network, i.e., the map, is public and all entities can access the road network without being known by others using techniques such as Private Information Retrieval (PIR) [13].
3) There exist secure channels between entities, which means that adversaries can neither use the channel information to locate the involved entities nor obtain the content transmitted over the channel.

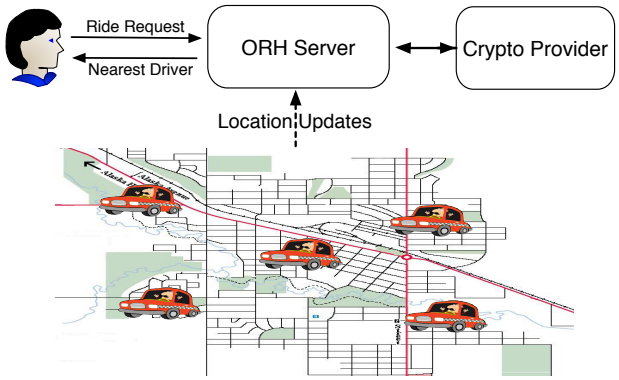


Fig. 1: System model with an ORH server, a Crypto Provider, the riders and drivers.

2.2 Design Goals

Under the system and threat model described above, a privacy-preserving ride-matching scheme over road networks for ORH service should satisfy the following design goals:

1) *Accuracy:* The ORH server should be able to produce accurate ride-matching results. That is, to find the nearest driver with a high probability for every rider with respect to their shortest distances in the road network.
2) *Efficiency:* The scheme should be efficient for practical use. The proposed scheme should perform ride-matching in real-time fashion and be scalable with the growing number of drivers and riders.
3) *Security:* Throughout the whole process, the ORH server and CP should learn nothing about the locations of the riders or drivers except the final ride-matching results, and a rider and a driver should not learn anything about the locations of each other unless a ride agreement is made between them.

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security

2.3 Cryptographic Tools

In this work, we make use of some cryptographic primitives, including homomorphic encryptions and garbled circuits.

Homomorphic encryption. Homomorphic encryption allows certain functions to be evaluated on ciphertexts to obtain an encrypted result, which matches the result of performing the functions on corresponding plaintexts after being decrypted. In a nutshell, existing homomorphic encryptions can be classified into three categories: i) Partially Homomorphically Encryption (PHE), such as Paillier cryptosystem [14], which supports only limited types of operations on ciphertexts with unlimited times; ii) Somewhat Homomorphic Encryption (SHE), such as BGN cryptosystem [15], which supports some types of operations on ciphertexts with limited times; iii) Fully Homomorphic Encryption (FHE), such as Gentry's cryptosystem [16], which supports unlimited operations with unlimited times. Generally, PHE schemes with limited functions are much more efficient than SHE and FHE schemes. Our first construction is built on SHE due to the need of performing multiplications over ciphertexts, while our optimized construction avoids that and thus adopts much more efficient PHE.

Garbled Circuit. Garbled Circuit was first proposed by Yao [17] to solve the famous Millionaires' problem. At a high level, Garbled Circuit allows two parties to jointly evaluate any function $f(x, y)$ without leaking any information about each other's inputs x and y beyond what is implied by the function. More precisely, one party, the generator, first converts the function f into a garbled version $\widehat{f}$. Then the generator sends the garbled circuit $\hat{f}$ together with the garbled input value $\hat{x}$ for his input x to the other party, the evaluator. To evaluate the function, the evaluator first runs a *1-out-2* Oblivious Transfer (OT) protocol [18] to obtain the garbled input value $\hat{y}$ for his private input y and then evaluates the garbled circuit $\hat{f}$ with $\hat{x}$ and $\hat{y}$ as its inputs to obtain the result of $f(x, y)$. However, directly applying Gabled Circuit to compute complex problem such as ride-matching over road networks is too costly to be practical. In this work, we only apply Garbled Circuit to lightweight non-linear computations such as comparisons.

3 Road Network Embedding

At a high-level, our approach to design a privacy-preserving ride-matching scheme for ORH service consists of secure shortest distance evaluation in road networks. However, directly computing shortest distances for massive moving objects (i.e., riders and drivers) in large-scale road networks can be very expensive. Moreover, such complex computation cannot be efficiently supported by existing cryptographic primitives. To solve this issue, we resort to the road network embedding technique, which allows efficient computation of approximate shortest distance.

Road Network Embedding (RNE), proposed by Shahabi *et al.* [12], is an approach to compute shortest path distance in road networks, which bases on the LLR embedding techniques [19]. RNE transforms a road network into a higher dimensional space by assigning a sketch (i.e., a vector) to every node such that the distance between any two nodes can be efficiently approximated using only their sketches.

Let $G = (V, E)$ denote a road network, where the node set V represents intersections of roads or road ends, and the edge set E represents the road segments connecting two nodes. Let $n = |V|$ be the size of the node set V. Define R as a set of $O(\log^2 n)$ reference sets, which are subsets of V: $R = \{V_{1,1}, \ldots, V_{1,\alpha}, \ldots, V_{\beta,1}, \ldots, V_{\beta,\alpha}\}$, where $\alpha = O(\log n)$ and $\beta = O(\log n)$. Each subset $V_{i,j}$ is defined as a random subset of V with 2^i nodes randomly chosen from V. The distance between node v and subset $V_{i,j}$ is defined as $dist(v, V_{i,j}) = \min_{w \in V_{i,j}} dist(v, w)$, that is, the smallest distance from v to the nodes in $V_{i,j}$, where $dist(v, w)$ denotes the shortest distance between nodes v and w in the road network. Based on this, each node $v \in V$ in the road network is mapped to a sketch as follows:

$$S(v) = (S_{V_{1,1}}(v), \ldots, S_{V_{1,\alpha}}(v), \ldots, S_{V_{\beta,1}}(v), \ldots, S_{V_{\beta,\alpha}}(v)),$$

where $S_{V_{i,j}}(v) = dist(v, V_{i,j})$. Note that the sketch $S(v)$ is a $O(\log^2 n)$-dimension vector in the high-dimensional embedding space defined by R. Finally, denote $\Omega = \{S(v) | v \in V\}$ as the embedded road network.

The riders and drivers are moving objects in the road network, which can be regarded as temporal nodes. They are dynamically mapped into the embedding space. Let u be a rider or driver located in the road segment connecting node s and node t, the distance between u and reference set $V_{i,j}$ can be efficiently estimated as:

$$\begin{aligned} &S_{V_{i,j}}(u) = dist(u, V_{i,j}) \\ &= \min\left\{dist(u, s) + S_{V_{i,j}}(s), dist(u, t) + S_{V_{i,j}}(t)\right\}. \end{aligned}$$

Note that $dist(u, s)$ and $dist(u, t)$ can be computed by u locally in plaintext using its GPS coordinates, and $S_{V_{i,j}}(s) \in S(s)$ and $S_{V_{i,j}}(t) \in S(t)$ are public information that can be obtained from Ω. Thus, a rider or a driver can dynamically map its current location to a sketch:

$$S(u) = (S_{V_{1,1}}(u), \ldots, S_{V_{1,\alpha}}(u), \ldots, S_{V_{\beta,1}}(u), \ldots, S_{V_{\beta,\alpha}}(u)).$$

$S(u)$ is a vector in the embedding space. For simplicity, we denote a sketch of a rider or driver u as:

$$S(u) = (S_1(u), \ldots, S_\kappa(u)), \tag{1}$$

where $\kappa = O(\log^2 n)$. Note that, we use the notations of *rider* and *location of a rider* interchangeably when there is no confusion. It is the same with the notations of *driver* and *location of a driver*.

With all the riders and drivers mapped into the embedding space, the shortest distance between a rider a and a driver b can be approximated through evaluating the chessboard distance of their corresponding sketches as:

$$dist(a, b) \approx \delta(a, b) = \max_{1 \leqslant j \leqslant \kappa} \left(|S_j(a) - S_j(b)|\right), \tag{2}$$

where $S_j(a)$ and $S_j(b)$ are the jth dimension of the sketch vectors $S(a)$ and $S(b)$, respectively.

Given a set of available drivers $D = \{b\}$ and a rider a, the ride-matching process is to find the nearest driver b^* such that:

$$b^* = \min_{b \in D} \delta(a, b). \tag{3}$$

In the following section, we will use RNE to design a privacy-preserving ride-matching scheme for ORH service.

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security

4 *pRide*: PRIVACY-PRESERVING RIDE-MATCHING SCHEME

Our objective is to design a privacy-preserving, efficient and accurate ride-matching scheme over road networks for ORH service, which allows the ORH server to efficiently match the riders and the drivers according to their road distances without learning the locations of riders and drivers. First, we give an overview of our scheme. Then, we present the details of our construction.

4.1 Overview

Given the locations of riders and drivers, the ride-matching is to find the nearest driver for every incoming rider based on the shortest roads distances between the riders and drivers. However, computing shortest distances on massive road networks can be very expensive, let alone find the nearest driver for a rider among numerous moving drivers in the ciphertext domain. To tackle this challenge, we convert the original road network into a higher embedding space using RNE. Thus, road distance between a rider and a driver can be efficiently estimated using their corresponding sketches through simple distance metric such as chessboard distance in the embedding space. Based on this, we leverage homomorphic encryptions and garbled circuits as building blocks and achieve a privacy-preserving, efficient yet accurate ride-matching scheme.

We assume that the road network G (i.e., the map), together with its embedded form Ω, is public. Every entity in the system can privately access them without being known by the others by maintaining a local copy or from a trusted server. The drivers periodically update the ciphertexts of the sketches corresponding to their current locations to the ORH server. To request a ride, a rider queries the ORH server with the ciphertext of the sketch corresponding his expected pickup location. With the encrypted sketches, the ORH server conducts the ride-matching operation without learning the underlying location information of the rider and drivers. To realize this idea, we propose two constructions. The first scheme relies on powerful SHE and garbled circuit, which can provide high ride-matching accuracy, but with a low efficiency. The second scheme leverages efficient PHE together with partitioning and data packing techniques, which achieves optimal efficiency and accuracy. The details of the two schemes are described in the following sections.

4.2 The Construction by Using SHE

We now describe the construction for privacy-preserving ride-matching over road networks, which is able to preserve the location privacy of riders and drivers. The construction *pRide*$_1$ = (*Setup, Location-update, Ride-request, Ride-matching*) makes use of a somewhat homomorphic encryption $SHE = (Gen, Enc, Dec)$ and a secure comparison circuit ξ_1. Let D be the set of all available drivers.

Setup. In this procedure, the server specifies the system parameters such as the security parameter λ, the number of bits representing a number. Besides, the server converts the road network G into a κ-dimensional embedding space Ω using RNE described above. With the system parameters, the CP generates a secure comparison circuit $\xi_1(\cdot)$, which takes as inputs four garbled input values $\widehat{d'_{b_1}}, \widehat{d'_{b_2}}, \widehat{\mu_{b_1}}, \widehat{\mu_{b_2}}$ and

1) unrandomizes $d_{b_1} = d'_{b_1} - \mu_{b_1}$ and $d_{b_2} = d'_{b_2} - \mu_{b_2}$;
2) outputs 1 if $d_{b_1} > d_{b_2}$, 0 otherwise.

Also, the CP generates a random key pair $(pk, sk) \leftarrow SHE.Gen(1^\lambda)$ and makes the public key pk available to every entity in the system.

Location-update. For a driver $b \in D$, he first obtains the sketch of his location using Eq. 1 , and then encrypts every dimension of his sketch with pk as

$$[\![S_j(b)]\!] = SHE.Enc_{pk}\left(S_j(b)\right), 1 \leqslant j \leqslant \kappa.$$

The encrypted sketch $[\![S(b)]\!] = \left([\![S_1(b)]\!], \ldots, [\![S_\kappa(b)]\!]\right)$ is sent to the ORH server as a ride offer.

Ride-request. In a similar way, the rider a obtains his sketch $S(a)$ using Eq. 1 and encrypts the jth dimension of it as

$$[\![S_j(a)]\!] = SHE.Enc_{pk}\left(S_j(a)\right).$$

Finally, the rider sends $[\![S(a)]\!] = \left([\![S_1(a)]\!], \ldots, [\![S_\kappa(a)]\!]\right)$ to the ORH server as his ride request.

Ride-Matching. With the encrypted sketches $[\![S(a)]\!]$ from the rider and $\{[\![S(b)]\!]\}_{b\in D}$ from the drivers, the ORH server matches the rider with the drivers in two steps:

Step 1: the ORH server estimates the distance between the rider and every available driver. For driver $b \in D$, the ORH server first computes $[\![d_j(a,b)]\!] = ([\![S_j(a)]\!] - [\![S_j(b)]\!])^2$ for $1 \leqslant j \leqslant \kappa$. Then, the ORH server obtains their road distance $[\![\delta(a,b)]\!]$ by running a secure maximum algorithm (described in Algorithm 1) over $\{[\![d_j(a,b)]\!]\}_{1\leqslant j\leqslant\kappa}$ such that $\delta(a,b) = \max\limits_{1\leqslant j\leqslant\kappa}(d_j(a,b))$. Note that we compute the square of the chessboard distance to avoid absolution operation on ciphertext.

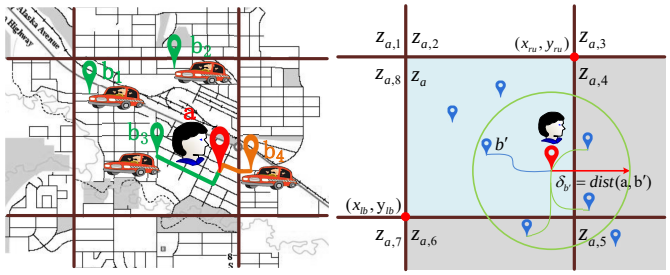
Algorithm 1 Secure Maximum Algorithm

Input: $[\![d_1(a,b)]\!], \ldots, [\![d_\kappa(a,b)]\!]$
Output: $j^* = \max_j(d_j(a,b))$
1: Initializes $j^* = 1$
2: **for** $j \in [2, \kappa]$ **do**
3: generate a random value μ_{j^*};
4: compute $\left[\!\left[d'_{j^*}(a,b)\right]\!\right] = [\![d_{j^*}(a,b)]\!] + [\![\mu_{j^*}]\!]$;
5: send $\left[\!\left[d'_{j^*}(a,b)\right]\!\right]$ to the CP, where the CP decrypts it and sends back its garbled form $\widehat{d'_{j^*}(a,b)}$;
6: run a *1-out-2* OT protocol with the CP to obtain the garbled form $\widehat{\mu_{j^*}}$ for μ_{j^*};
7: run step 3-6 again for $d_j(a,b)$ to obtain $\widehat{d'_j(a,b)}$ and $\widehat{\mu_j}$;
8: **if** $\xi_1(\widehat{d'_j(a,b)}, \widehat{d'_{j^*}(a,b)}, \widehat{\mu_j}, \widehat{\mu_{j^*}}) > 0$ **then**
9: Set $j^* = j$;
10: **end if**
11: **end for**
12: Return j^*

Step 2: the ORH server finds the nearest driver b^* by running a secure minimum algorithm over the encrypted chessboard distances $\{[\![\delta(a,b)]\!]\}_{b\in D}$ obtained in the last step such that $\delta(a,b^*) = \min_{b\in D}(\delta(a,b))$. The secure minimum

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security



(a) (b)

Fig. 2: Road network partitioning and refinement. (a) An example of road network partition. We emphasize that the local nearest driver (i.e, b_3) may not be the nearest driver at a global view, which is b_4 in the adjacent zone; (b) An example of refinement, which further checks adjacent zones that intersect with the circle that is centered at the rider and of a radius $\delta_{b'} = dist(a, b')$.

algorithm is similar to Alg. 1 that simply changes the "$>$" with "$<$" at line 8 of Alg. 1.

Finally, the nearest driver b^* is assigned to serve the requesting rider and a secure channel is established between them, where further communication happens.

Remark. Based on the properties of road network embedding, it is straightforward to demonstrate the correctness of the scheme. The scheme avoids complex decryption of encrypted inputs within the garbled circuit by leveraging the random masking technique, which reduces the computation cost for the ORH server. However, the communication overhead between the ORH server and the CP in the *Ride-matching* procedure is linear to both the sketch size and the number of available drivers. Though the sketch size $O(\log^2 n)$ is sub-linear to the number of nodes in the road network, it could still incur heavy communication overhead considering the number of ride requests and drivers in practice. Besides, this construction computes squares of sketch dimension differences to avoid absolution operations over ciphertexts, which requires at least SHE schemes that supports both additive and multiplicative homomorphism. However, the homomorphic operations of existing SHE schemes are expensive. As our experiments demonstrated, a typical SHE scheme, the BGN cryptosystem, takes almost 1 seconds to perform one operation on ciphertexts. To provide a practical solution, we present an optimized construction in the following section, which achieves privacy-preserving, accurate yet efficient ride-matching.

4.3 Optimized Construction

We now present an optimized construction that achieves both accuracy and efficiency by exploiting partitioning and data packing techniques. We have two important observations about the ride-matching operation in ORH systems.

Observation 1: The ride-matching operation cares about only drivers near the rider. It will significantly reduce the overall complexity by filtering out far away drivers before expensive distance computations and comparisons over ciphertexts. To do that, a straightforward approach is to partition the road network (i.e., the map) into small zones and perform the ride-matching only between riders and drivers within the same zone. However, the returned driver may not be the closest one at the global scale as shown in Fig. 2a.

Observation 2: The same operations in the distance computation are performed on all dimensions of the sketch vectors. Furthermore, the message space of road distances is much smaller than the space of homomorphic ciphertext. Thus, we can pack the dimensions of a sketch into a single ciphertext and manipulate them simultaneously in one operation.

Based on these two observations, we proposed an optimized scheme *pRide*$_2$ = (*Setup, Location-update, Ride-request, Ride-matching*) by exploiting graph partitioning and data packing techniques.

Partitioning with refinement: To ensure the ride-matching accuracy, we propose a refinement procedure to refine the ride-matching result. As shown in Fig. 2b, our intuition is to first find the local nearest driver b' within the zone where the rider a is located and then search the adjacent zones that intersect with the circle centered at the rider with a radius $\delta(a, b')$. The methodology behind this is that the Euclidean distance of two nodes lower bounds their network distance. Thus, drivers closer to the rider a than b' should be within Euclidean distance $\delta(a, b')$ from the rider, i.e, they should lie in the shaded zones in Fig. 2b. The details of the refinement algorithm are presented in Alg. 3 and the garbled circuit implementation is described in Fig. 4.

Data packing: let $S(u) = (S_1(u), \ldots, S_\kappa(u))$ be a κ-dimensional sketch vector, we can pack it through a polynomial given by

$$P(u) = \sum_{j=1}^{\kappa} S_j(u) x^{j-1},$$

where the base $x = 2^\Phi$ and Φ is a parameter that is large enough to separate two dimension values at the bit level. Then, we can encrypt the sketch $S(u)$ as a single ciphertext $[\![P(u)]\!] \leftarrow HE.Enc_{pk}(P(u))$. For two κ-dimensional sketches $S(a)$ and $S(b)$, the homomorphic addition of $[\![P(a)]\!]$ and $[\![P(b)]\!]$ gives the ciphertext of the polynomial addition $P(a) + P(b) = \sum_{j=1}^{\kappa} (S_j(a) + S_j(b)) x^{j-1}$.

With the partitioning and data packing techniques, we achieve *pRide*$_2$ by leveraging an efficient partial homomorphic encryption $PHE = (Gen, Enc, Dec)$ and two garbled circuits $\xi_2(\cdot)$ and $\xi_3(\cdot)$. The details of *pRide*$_2$ are as follows.

Setup. As before, the ORH server sets up the system and the CP initializes the key pair $\langle pk, sk \rangle \leftarrow PHE.Gen(1^\lambda)$. Besides, the ORH server partitions the road network into zones. For simplicity, we assume the network is partitioned vertically and horizontally as shown in Fig. 2. In addition, the CP prepares two garbled circuits $\xi_2(\cdot)$ and $\xi_3(\cdot)$ for the ride-matching procedure.

Alg. 2 describes the distance comparison circuit $\xi_2(\cdot)$, which takes as inputs the garbled values of two randomized packed distance vectors as well as their corresponding random values, and outputs the index of the one with the minimum distance. As shown in Fig. 3, we construct the circuit $\xi_2(\cdot)$ from bottom to top. First, we construct a subtraction and absolution circuit *SUB_ABS*, which consists of a subtractor, two multiplexers and a comparator. After that, we compose the *SUB_ABS* circuits and a maximum circuit *MAX* to obtain a *Chessboard* circuit, which outputs the

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security

chessboard distance of two vectors. At the top, a comparator circuit is used to obtain the index of the one with minimum chessboard distance.

Alg. 3 presents the refinement circuit $\xi_3(\cdot)$, which takes as inputs the garbled values of a rider's location, the randomized packed distance vector from the local nearest driver b' to the rider, the corresponding packed random value, and the zone information, and outputs an array that indicates the adjacent zones where a nearer driver may exist. Note that $z_{a,1}$, $z_{a,2}$, $z_{a,3}$, $z_{a,4}$, $z_{a,5}$, $z_{a,6}$, $z_{a,7}$ and $z_{a,8}$ denote the adjacent zones around the zone z_a where the rider a is located as shown in Fig. 2b. The garbled circuit implementations of $\xi_3(\cdot)$ are shown in Fig. 4. The output of $\xi_3(\cdot)$ is a 8-bit value $\Gamma = l_1|l_2|\cdots|l_8$ with each bit indicates whether a nearer driver may exist in its corresponding adjacent zone.

Location-update. Similar to the construction by SHE, the driver $b \in D$ obtains the sketch $S(b)$ of his current location according to Eq. 1. Instead of encrypting $S(b)$ dimension-wisely, the driver packs it $P(b) = \sum_{j=1}^{\kappa} S_j(b)x^{j-1}$ and encrypts it into a single ciphertext

$$[\![P(b)]\!] = PHE.Enc_{pk}\left(P(b)\right)$$

Then, the driver sends $[\![P(b)]\!]$ and the zone z_b where it is located to the ORH server.

On receiving the encrypted location information $([\![P(b)]\!], z_b)$, the ORH server generates a packed random vector $P(\mu_b) = \sum_{j=1}^{\kappa} \mu_{b,j}x^{j-1}$ and randomizes the encrypted sketch as $[\![P(b)']\!] = [\![P(b)]\!] - [\![P(\mu_b)]\!]$, where $[\![P(\mu_b)]\!] = PHE.Enc_{pk}\left(P(\mu_b)\right)$. Note that the values $\mu_{b,j}$ in the random vector should be large enough to ensure that $S_j(a) - S_j(b) + \mu_{b,j} > 0$ for any rider a.

Ride-request. To request a ride at location (x_a, y_a), the rider a generates a secure ride request as follows:

1) encrypts the expected pickup location as $[\![x_q]\!] \leftarrow PHE.Enc_{pk}\left(x_q\right)$ and $[\![y_q]\!] \leftarrow PHE.Enc_{pk}\left(y_q\right)$;
2) obtains its sketch vector $S(a)$ using Eq. 1 and encrypts it as a single ciphertext

$$[\![P(a)]\!] = PHE.Enc_{pk}\left(P(a)\right),$$

where $P\left(a\right) = \sum_{j=1}^{\kappa} S_j(a)x^{j-1}$.

Finally, the rider sends $([\![P(a)]\!], [\![x_a]\!], [\![y_a]\!], z_a)$ to the ORH server as a ride request, where z_a denotes the zone where the rider is located.

On receiving the ride request, the ORH server first generates two random values η_x, η_y, and a packed value $P(\eta_a) = \sum_{j=1}^{\kappa} \eta_{a,j}x^{j-1}$, where $\eta_a = (\eta_{a,1}, \ldots, \eta_{a,\kappa})$ is a random vector. Then, the ORH server precomputes $[\![x'_a]\!] = [\![x_a]\!] + [\![\eta_x]\!]$, $[\![y'_a]\!] = [\![y_a]\!] + [\![\eta_y]\!]$ and$[\![P(a)']\!] = [\![P(a)]\!] + [\![P(\eta_a)]\!]$, where $[\![\eta_x]\!]$, $[\![\eta_y]\!]$ and $[\![P(\eta_q)]\!]$ are the ciphertexts of η_x, η_y and $P(\eta_q)$ under pk, respectively.

Ride-matching. The ORH server performs ride-matching in two phases. In the first phase, the ORH server finds the local nearest driver b' within the zone where the rider is located. After that, a refinement phase is conducted to refine the ride-matching result, namely, finding the nearest driver at global scale.

Phase 1 (Local Zone matching). Denote D_{z_a} as the set of drivers who are located in the same zone as the requesting rider a. The ORH server finds the local nearest driver as follows.

1) For driver $b \in D_{z_a}$, the ORH server computes $[\![P(d_b)']\!]$ as below and sends the result to the CP, where the CP decrypts it $P(d_b)' = PHE.Dec_{sk}\left([\![P(d_b)']\!]\right)$ and sends back its garbled input value $\widehat{P(d_b)'}$. Besides, the ORH server runs a *1-out-of-2* OT protocol with the CP to obtain the

Algorithm 2 Distance Comparison Circuit

Input: $\widehat{P(d_{b_1})'}$, $\widehat{P(d_{b_2})'}$, $\widehat{P(\mu_{b_1})}$ and $\widehat{P(\mu_{b_2})}$
1: Unpack $P(d_{b_1})'$ and $P(\mu_{b_1})$ into $d'_{b_1,1}, \ldots, d'_{b_1,\kappa}$ and $\mu_{b_1,1}, \ldots, \mu_{b_1,\kappa}$, respectively
2: Set $\delta_{b_1} = \max_{1\leqslant j\leqslant\kappa}\left(\left|d'_{b_1,j} - \mu_{b_1,j}\right|\right)$
3: Similar, unpack $P(d_{b_2})'$ and $P(\mu_{b_2})$, and obtain $d'_{b_2,1}, \ldots, d'_{b_2,\kappa}$ and $\mu_{b_2,1}, \ldots, \mu_{b_2,\kappa}$
4: Set $\delta_{b_2} = \max_{1\leqslant j\leqslant\kappa}(|d'_{b_2,j} - \mu_{b_2,j}|)$
5: Return $\operatorname{argmin}\left(\delta_{b_1}, \delta_{b_2}\right)$

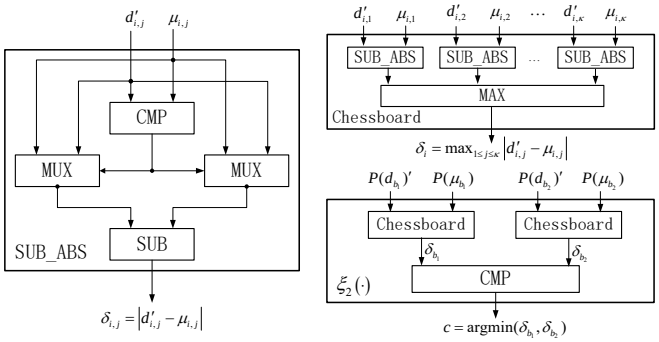


Fig. 3: The construction details of Distance Comparison Circuit $\xi_2(\cdot)$, where *CMP, MUX, SUB* and *MAX* denote a comparator circuit, a multiplexer circuit, a subtractor circuit and a maximum circuit, respectively.

Algorithm 3 Refinement Circuit

Input:
1: 1) The location: $\widehat{x'_a}$, $\widehat{y'_a}$, $\widehat{\eta_x}$, $\widehat{\eta_y}$
2: 2) The distance: $\widehat{P(d_{b'})'}$, $\widehat{P(\eta_a)}$
3: 2) The zone: x_{lb}, y_{lb}, x_{ru}, and y_{ru}
Output: An array of zones Γ that a nearer driver may exist.
4: Compute $x_a = x'_a - \eta_x$ and $y_a = y'_a - \eta_y$
5: Unpack $P(d_{b'})'$ and $P(\eta_a)$ into $d'_{b',1}, \ldots, d'_{b',\kappa}$ and $\eta_{a,1}, \ldots, \eta_{a,\kappa}$, respectively.
6: Set $\delta_{b'} = \max_{1\leqslant j\leqslant\kappa}(|d'_{b',j} - \eta_{a,j}|)$
7: Initialize an empty array Γ
8: If $x_a + \delta_{b'} > x_{ru}$, then add $z_{a,4}$ into Γ
9: If $x_a - \delta_{b'} < x_{lb}$, then add $z_{a,8}$ into Γ
10: If $y_a + \delta_{b'} > y_{ru}$, then add $z_{a,2}$ into Γ
11: If $y_a - \delta_{b'} < y_{lb}$, then add $z_{a,6}$ into Γ
12: If $(x_a - x_{lb})^2 + (y_a - y_{lb})^2 < \delta_{b'}^2$, then add $z_{a,7}$ into Γ
13: If $(x_a - x_{lb})^2 + (y_a - y_{ru})^2 < \delta_{b'}^2$, then add $z_{a,1}$ into Γ
14: If $(x_a - x_{ru})^2 + (y_a - y_{ru})^2 < \delta_{b'}^2$, then add $z_{a,3}$ into Γ
15: If $(x_a - x_{ru})^2 + (y_a - y_{lb})^2 < \delta_{b'}^2$, then add $z_{a,5}$ into Γ
16: Return Γ

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security

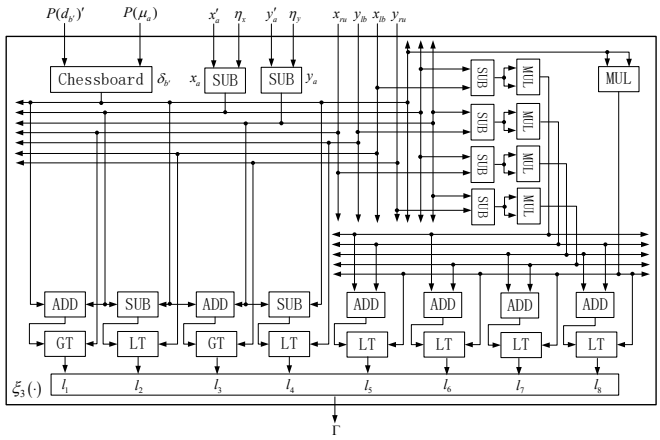


Fig. 4: The construction details of Refinement Circuit $\xi_3(\cdot)$, which outputs a bit array that indicates the indexes of the adjacent zones that a nearer driver may exist. The *ADD*, *GT*, *LT* and *MUL* denote *addition* circuit, *greater than* circuit, *less than* circuit and *multiplication* circuit, respectively.

garbled input value $\widehat{P(\mu_b)}$ for $P(\mu_b)$.

$$[\![P(d_b)']\!] = [\![P(a)]\!] - [\![P(b)']\!] = [\![P(a)]\!] - [\![P(b)]\!] + [\![P(\mu_b)]\!]$$

2) With all the garbled input values for all drivers, the ORH server runs a minimum algorithm to obtain the local nearest driver b' such that $\delta(a, b') = \min_{b \in D_{z_a}}(\delta(a, b))$. During the process, distances from two drivers to the rider are compared in ciphertexts by evaluating the distance comparison circuit $\xi_2(\cdot)$, which is presented in Algorithm 2.

Phase 2 (Refinement). In this phase, the ORH server refines the ride-matching result in the following steps. An example is shown in Fig. 2b.

1) First, the ORH server computes $[\![P(d_{b'})']\!] = [\![P(a)']\!] - [\![P(b')]\!]$ and sends it to the CP together with $[\![x'_a]\!]$ and $[\![y'_a]\!]$, where the CP obtains their underlying values $P(d_{b'})', x'_a, y'_a$, and sends back their garbled input values $\widehat{P(d_{b'})'}$, $\widehat{x'_a}$, $\widehat{y'_a}$. Besides, the ORH server runs a *1-out-of-2* OT protocol with the CP to obtain the garbled input values $\widehat{\eta_x}$,$\widehat{\eta_y}$, $\widehat{P(\eta_a)}$ for η_x, η_y and $P(\eta_a)$.
2) Then, taking as inputs the garbled input values $\widehat{P(d_{b'})'}$, $\widehat{x'_a}$, $\widehat{y'_a}$, $\widehat{P(\eta_a)}$, $\widehat{\eta_x}$, $\widehat{\eta_y}$, the ORH server evaluates the refinement circuit $\xi_3(\cdot)$, which outputs the zones where a nearer driver may exist (details described in Algorithm 3). Note that the zone information x_{lb}, y_{lb}, x_{ru}, and y_{ru} are public. They are constants in the garbled circuit $\xi_3(\cdot)$.
3) After that, the ORH server runs iteratively the first phase over the drivers in every suspicious zone plus the current nearest driver b'.

Finally, the ORH server obtains the globally nearest driver b^* to the rider and assign b^* to serve the rider.

5 Security Analysis

At a high-level, our scheme is to ensure that the ORH server performs the ride-matching without learning the location information of riders and drivers. To formally analyze the security strength of our scheme, we modify the notion of adaptive semantic security from [20] for our setting of privacy-preserving ride-matching over road networks. Before diving into details, we first define the leakage functions:

- *Leakage function* $\mathcal{L}_1$: Given an embedded road network Ω, a set of drivers $D = (b_1, \ldots, b_d)$, $\mathcal{L}_1(\Omega, D) = DP(\Omega, D)$, where $DP(\Omega, D)$ is the driver pattern. We define it in the following paragraph.
- *Leakage function* $\mathcal{L}_2$: Given an embedded road network Ω, a series of riders $Q = (a_1, \ldots, a_\tau)$, $\mathcal{L}_2(\Omega, Q) = \{RP(\Omega, Q), AP(\Omega, Q)\}$, where $RP(\Omega, Q)$ and $AP(\Omega, Q)$ denote the rider pattern and the access pattern, respectively. The definitions of the two patterns are as follows.

Definition 1 (Drivers Pattern). *For two drivers with sketches $S(b_1) = (S_1(b_1), \ldots, S_\kappa(b_1))$ and $S(b_2) = (S_1(b_2), \ldots, S_\kappa(b_2))$, define $\pi_D(b_1, b_2) = 1$ if $S_j(b_1) = S_j(b_2)$ holds for all $1 \leqslant j \leqslant \kappa$, 0 otherwise. $DP(\Omega, D)$ defines a symmetric matrix Π_D such that $\Pi_D(b_1, b_2) = \pi_D(b_1, b_2)$, that is, the driver pattern reveals whether two drivers are from the same location.*

Definition 2 (Request Pattern). *Similar to the driver pattern, the rider pattern reveals whether two ride requests are from the same location. Specifically, $RP(\Omega, Q)$ defines a symmetric matrix Π_Q such that $\Pi_Q(a_1, a_2) = \pi_Q(a_1, a_2)$, where $\pi_Q(a_1, a_2)$ indicates whether each dimension of $S(a_1)$ equals that of $S(a_2)$.*

Definition 3 (Access Pattern). *Given a series of rider Q, the access pattern is defined as $AP(\Omega, Q) = \{b^*\}$, where $\{b^*\}$ are the ride-matching results for the ride requests. Besides the ride-matching results, the access pattern for the optimized construction also contains the set of zones searched during the ride matching, which reveals the zone where each b^* driver is located.*

With these leakage functions, we present the security definition as follows:

Definition 4 (Adaptive semantic security). *Let pRide = (**Setup**, **Location-update**, **Ride-request**, **Ride-matching**) be a privacy-preserving ride-matching scheme and consider the following probabilistic experiments with a semi-honest adversary $\mathcal{A}$, a challenger $\mathcal{C}$, a simulator $\mathcal{S}$ and two leakage functions $\mathcal{L}_1$ and $\mathcal{L}_2$:*

$\mathbf{Real}_{\mathcal{A},\mathcal{C}}(1^\lambda)$: *First, $\mathcal{C}$ runs **Setup** to obtain system parameters including the key pair (sk, pk). Then, $\mathcal{A}$ outputs a set of drivers $b_1, \ldots, b_d$. For each driver b_i, $\mathcal{C}$ and $\mathcal{A}$ execute a simulation of **Location-update** with $\mathcal{C}$ playing the role of driver b_i and $\mathcal{A}$ playing the role of ORH server. After that, $\mathcal{A}$ generates a polynomial number of adaptively chosen riders $a_1, \ldots, a_\tau$. For each rider a_i, $\mathcal{C}$ acts as a rider and runs **Ride-request** with $\mathcal{A}$ acting as the ORH server and running **Ride-matching**. Finally, the $\mathcal{A}$ outputs a bit γ as the result of the experiment.*

$\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(1^\lambda)$: *First, given the 1^λ, $\mathcal{S}$ generates the system parameters including the key pair (sk, pk). Then, $\mathcal{A}$ outputs a set of drivers $D = b_1, \ldots, b_d$ of polynomial number of drivers. For each driver b_i, $\mathcal{S}$ is given $\mathcal{L}_1(\Omega, D)$, and $\mathcal{A}$ and $\mathcal{S}$ runs **Location-update** with $\mathcal{A}$ playing the ORH server and $\mathcal{S}$ playing the driver. After that, $\mathcal{A}$ generates a polynomial number of adaptively chosen riders $Q = a_1, \ldots, a_\tau$. For each rider a_i, given $\mathcal{L}_2(\Omega, Q)$, $\mathcal{S}$ acts as a rider and runs **Ride-request** with $\mathcal{A}$ acting as the ORH server and running **Ride-matching**. Finally, the $\mathcal{A}$ outputs a bit γ as the result of the experiment.*

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security

We say such a privacy-reserving ride-matching scheme is adaptively $(\mathcal{L}_1, \mathcal{L}_2)$-semantically secure if for all Probabilistic Polynomial Time (PPT) adversaries $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that

$$\left|\Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(1^\lambda) = 1] - \Pr[\mathbf{Real}_{\mathcal{A},\mathcal{C}}(1^\lambda) = 1]\right| \leqslant negl(\lambda),$$

where $negl(\cdot)$ is a negligible function.

Theorem 1. *If the HE (SHE or PHE) is CPA-secure, then our privacy-preserving ride-matching scheme, as described above, is adaptively $(\mathcal{L}_1, \mathcal{L}_2)$-semantically secure in the random oracle model.*

Proof: To proof the theorem, we describe a PPT simulator $\mathcal{S}$ such that no PPT adversary $\mathcal{A}$ is able to distinguish the outputs of $\mathbf{Real}_{\mathcal{A},\mathcal{C}}(1^\lambda)$ and $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(1^\lambda)$ computationally. The simulator $\mathcal{S}$ works as follows.

Setup: Given the security parameter λ, $\mathcal{S}$ starts the simulation by generating a key pair $(sk, pk) \leftarrow HE.Gen(1^\lambda)$.

*Simulating **Location-update**:* For each driver b_i, given leakage $\mathcal{L}_1(\Omega, D) = DP(\Omega, D)$, $\mathcal{S}$ first checks whether the location b_i has appeared before. If it appeared, $\mathcal{S}$ sets $\widetilde{[\![b_i]\!]}$ to previously used homomorphic encryption values; Otherwise, $\mathcal{S}$ outputs κ homomorphic encryptions $\widetilde{[\![S(b_i)]\!]} = (\widetilde{[\![S_1(b_i)]\!]}, \ldots, \widetilde{[\![S_\kappa(b_i)]\!]})$, where $\widetilde{[\![S_j(b_i)]\!]} \leftarrow HE.Enc_{pk}(r_{i,j})$, and $(r_{i,1}, \ldots, r_{i,\kappa})$ is random point in the embedding space Ω.

*Simulating **Ride-request**:* For each rider a_i, given $\mathcal{L}_2(\Omega, Q) = \{RP(\Omega, Q), AP(\Omega, Q)\}$, $\mathcal{S}$ checks whether a_i appeared in any previous ride request. If a_i appeared previously, $\mathcal{S}$ sets $\widetilde{[\![S(a_i)]\!]}$ to the homomorphic encryptions that were previously used. If not, it sets $\widetilde{[\![S(a_i)]\!]}$ for some previously unused $\widetilde{[\![S(a_i)]\!]}$.

The correctness of the simulation is easy to demonstrate by performing the ***Ride-matching*** operation over the simulated drivers $\left\{\widetilde{[\![S(b_i)]\!]}\right\}_{1\leqslant i\leqslant d}$ and the riders $\left\{\widetilde{[\![S(a_i)]\!]}\right\}_{1\leqslant i\leqslant \tau}$. In summary, the indistinguishability of simulation follows the CPA-security of HE. More precisely, $HE = SHE$ for the first construction *pRide*$_1$ and $HE = PHE$ for the optimized construction *pRide*$_2$.

6 Evaluation

In this section, we evaluate the practicality of our scheme on real-world datasets. We implement the Shahabi *et al.* road network embedding (RNE) and both of our privacy-preserving ride-matching schemes, the first construction *pRide*$_1$ and the optimized construction *pRide*$_2$. As a contrast, we provide another implementation *pRide*$_3$ of the optimized construction without the refinement phase ***Refinement***.

We built the garbled circuits on top of FastGC [21], an open-source framework that allows developers to compose circuits from AND, OR and XOR gates. It provides many circuit optimization techniques such as free XOR [22] and half AND [23]. In the experiments, we use the default label size 80 for all wires. We instantiate the SHE in our first construction *pRide*$_1$ and the PHE in the optimized construction *pRide*$_2$ with the Boneh-Goh-Nissim (BGN) scheme [15] and the Paillier's cryptosystem [14], respectively. We implement the BGN scheme with the Java Pairing-based Library (JPBC)[1] and adopt an open-source library[2] for Paillier's cryptosystem.

During the experiments, we set the modulus of Paillier's cryptosystem to 1024 bits, which corresponds to an 80-bit security level. For the BGN encryption, we also use an 80-bit security, which corresponds to a subgroup size of 160 bits. The experiments were conducted on a system equipped with 8 GB memory and Intel i7-6700 CPU of 3.4GHz. All the experimental results are mean of 10 trials.

6.1 Datasets

We evaluate our schemes on real-world datasets that come from [24]. The datasets present the road network of California that contains 21048 nodes and 21693 edges (roads). Besides, the datasets provide coordinates for every node in the road network. For ease to control experiment parameters, we created a sequence of synthetic riders and drivers at random. Since riders and drivers are constrained to move along the roads, we locate all the locations of riders and drivers on the edges of the road network. For simplicity, we partition the city into zones vertically and horizontally as shown in Fig. 2a. In the experiments, we have tested the effects of different partition granularities, such as 4×4 that partitions the whole city into 16 zones of equal size.

6.2 Accuracy

As the core of ORH service, ride-matching operation is to find the nearest driver for every incoming rider over the road network. Since our proposed scheme estimates road distances between riders and drivers through approximation, matching errors may exist. However, we claim that the ride-matching results of our scheme are much more accurate than previous schemes. To demonstrate the high accuracy, we compare our schemes with *ORide* [8] and *GREC* [9] in terms of the ride-matching accuracy rate, which indicates the matching degree of ride-matching results comes from these schemes with the ground truths in plaintext. Specifically, we randomly generate 100 riders and counts the number of riders whose matched drivers are the same as the ground truths. As shown in Fig. 5a, our schemes return true nearest drivers for almost all riders, which significantly outperforms the other two schemes under all partition granularities. Even the contrast setting *pRide*$_3$ that contains no refinement phase can achieve an accuracy rate of above 85%. Besides, due to the existing of the refinement phase, we can see that *pRide*$_2$ produces the same accuracy as *pRide*$_1$, which matches riders and drivers without partitioning. Also, we test the ride-matching accuracy of our schemes on different sketch sizes. Fig. 5b shows that larger sketch size will guarantee a higher ride-matching accuracy rate. However, larger sketch size will result in more computation and communication overhead. Thus, we set the sketch size to 24, which produces almost 100% accuracy rate with ***Refinement*** as shown in Fig. 5b.

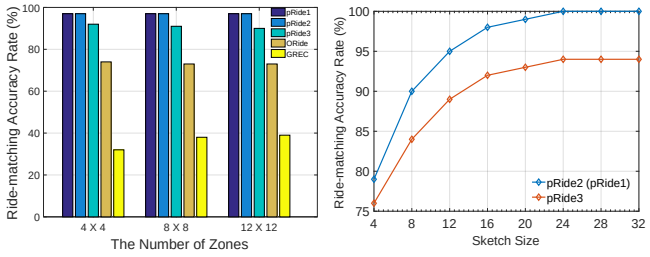
1. http://gas.dia.unisa.it/projects/jpbc/
2. https://github.com/kunerd/jpaillier

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security

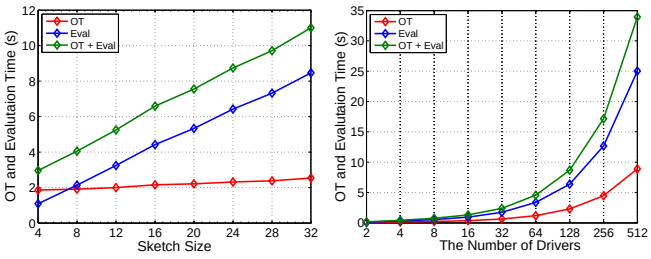
Sketch Size	The Basic Construction $pRide_1$						The Optimized Construction $pRide_2$					
	Rider		Driver		Server		Rider		Driver		Server	
	Comm (KBs)	Comp. (secs)	Comm (KBs)	Comp. (secs)	Comm (MBs)	Comp. (secs)	Comm (bytes)	Comp. (ms)	Comm (bytes)	Comp. (ms)	Comm (MBs)	Comp. (secs)
4	2.0	3.4	2.0	3.4	8.7	130.7	256	5.0	256	5.1	4.5	2.9
8	4.1	6.9	4.1	6.9	17.2	297.3	256	4.9	256	5.0	7.3	4.2
12	6.1	10.3	6.1	10.2	25.8	380.3	256	5.1	256	5.2	10.2	5.4
16	8.2	13.8	8.2	13.8	34.3	549.4	256	5.2	256	5.3	13.1	6.6
20	10.3	17.0	10.3	17.0	42.9	729.1	256	5.3	256	5.2	16.0	7.7
24	12.3	20.6	12.3	20.6	51.4	801.4	256	5.1	256	5.0	18.9	9.1
28	14.4	24.0	14.4	24.1	59.9	1048.3	256	5.0	256	5.1	21.8	10.0
32	16.5	27.5	16.5	27.5	68.5	1118.7	256	5.2	256	5.3	24.6	11.1

TABLE 1: Per-ride performance summary of the construction $pRide_1$ and the optimized construction $pRide_2$ for different sketch size. The number of available drivers for every ride request is set to 128 and the partition granularity is set to 8×8.



(a) (b)

Fig. 5: The ride-matching accuracy. (a) Comparisons of accuracy rates of our schemes and *ORide*, *GREC* under different partition granularities with a sketch size of 24; (b) The accuracy rates of our schemes under different sketch sizes with a partition granularity of 8×8.



(a) (b)

Fig. 6: The per-matching execution time of the garbled circuits in $pRide_2$, which includes two parts, i.e., the oblivious transfer part (OT) and the circuit evaluation part (Eval). (a) The execution time under different sketch sizes; (b) The execution time under different numbers of drivers.

6.3 Performance

In this section, we evaluate the performance of our schemes for every side in terms of communication (comm) overhead and computation (comp.) cost.

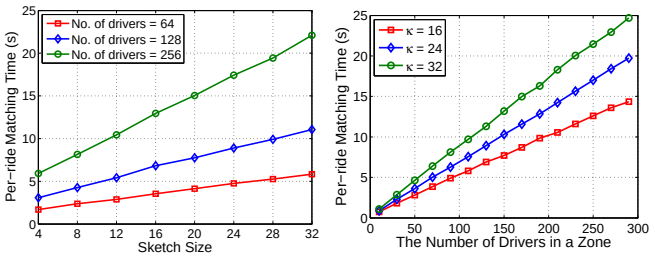
The performance of riders. For a rider, its overhead depends only on the sketch size. As shown in Table 1, the per-request communication overhead and computation cost for a ride are both linear to the sketch size in the first construction. This is mainly because that a rider in the first construction has to encrypt every dimension of its sketch vector separately and send all the ciphertexts to the server. With data packing technique, the overhead of the rider side is reduced significantly in the optimized construction. By packing all the dimensions of a sketch into one ciphertext, the communication cost is only about 4.2% of that in the first construction. Thanks to the novel usage of partial homomorphic encryptions scheme, the computation cost is also significantly reduced, which is about three orders of magnitude less than that in the first construction.

The performance of drivers. The performance of a driver also depends on the sketch size. Similar to a rider, the per-update communication overhead and computation cost of a driver can be significantly reduced through data packing as shown in Table 1 of the experiment results for the driver side. Note that, the drivers is able to update their locations at their own paces. For example, updating when their they are available or their locations change.

The performance of the ORH server. Referring to the design of our schemes, the communication overhead of the ORH server comes from three aspects, i.e., the location updates from drivers, the requests from riders and the interactions with the CP during the ride-matching. The communication overhead from the drivers and riders depends on the size of per-update or per-request and the number of them, among which the former can be significantly reduced through our data packing in the optimized construction and the later depends on the market. The communication cost of interactions with the CP includes two parts, namely, the ciphertexts sent to the CP and the communication cost caused by the garbled circuits. Although the communication cost of interactions with the CP is relatively large, the server side infrastructures usually allow network communication with large bandwidth (from 1,000Mbps to 14,000Mbps), and therefore the interactions can be conducted efficiently in practice. Also, we can see that the per-matching communication overhead of our optimized construction is generally 50% less than that of the first construction from Table 1. This mainly owes to our partitioning and data packing. The computation cost of the ORH server depends on the sketch size and has a linear relation to the number of drivers. As shown in Fig. 7a, the per-ride matching time increases with the sketch size. However, we can seen from Table 1 that the computation cost of the ORH server has been significantly reduced through our partitioning and data packing techniques in the optimized construction. The overall computation cost of the ORH server in the optimized construction is generally two orders of magnitude less than that in the first construction. To capture the performance of our optimized construction on different user densities, we vary the number of drivers within a zone. When the partition system is fixed, the number of drivers within a zone indicates the user density. As shown in Fig. 7b, the

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security



(a) (b)

Fig. 7: Per-ride matching time in $pRide_2$. (a) Per-ride matching time on different sketch sizes with respective 64, 128, 256 drivers in a zone; (b) Per-ride matching time on different user densities (number of users within a zone) with sketch size $\kappa = 16, 24, 32$.

per-ride matching time is linear to the density of users. In practice, we could use dynamic partitions to ensure an acceptable ride-matching efficiency. With optimized per-ride matching efficiency, parallel computing can be further used to handle multiple rides simultaneously, which is a problem of computation resources.

The performance of the CP. As described in our scheme, the main work of the CP can be done offline, including security parameters and circuits generations. The online work of the CP is to decrypt the received randomized distances and compute garbled input values for them. As shown in Fig. 6, the sketch size has significant effect on the garbled circuits evaluation, while the number of drivers affects both the oblivious transfer process and garbled circuits evaluation. However, we can precompute some random values and obtain their garbled input values in advance to accelerate the online process.

7 Related Work

Privacy in online riding services has received extensive attention due to its increasing popularity. Many privacy-enhancing solutions have been proposed for ride-sharing, secure shortest path (distance) computation and other problems in online riding services.

7.1 Privacy-preserving Ridesharing

Friginal *et al.* [25] introduced the problem of privacy leakage in ridesharing (i.e., carpooling) and proposed a privacy-preserving dynamic ridesharing framework following the privacy-by-design principle. Goel *et al.* [4] proposed a privacy-aware ridesharing scheme based on the obfuscation technique, which preserves the privacy of riders by submitting obfuscated locations. Other non-cryptographic solutions such as pseudonyms [26], cloaking [27] and anonymity [28] are used to preserve the location privacy in location-based service (LBS). However, most of these schemes rely on imprecise querying locations to hide riders' exact locations that will produce bad matching accuracy. To provide accurate ride-matching, some cryptographic approaches have been proposed based on different cryptographic primitives. He *et al.* [29] proposed a privacy-preserving partner selection scheme for ridesharing, which takes into consideration the travel time saving for all parties and hence is more practical than previous schemes using Euclidean distance. However, it's not clear how to obtain the travel time privately in the scheme. Besides, the rider and drivers are involved in multiple rounds interactions with the server during the computation, which incurs much communication and computation cost for the user side. A privacy-preserving meeting points determination scheme was proposed in [2], which integrated Private Set Intersection (PSI) [30] technique to calculate the optimal meeting points without sacrificing the location privacy of passengers. Sherif *et al.* [5] proposed a privacy-preserving ridesharing scheme, which utilizes the secure kNN computation technique [31] to calculate the similarity of trips in a privacy-preserving manner. All these schemes do not consider privacy leakage in the online ride-hailing service, where the riders hail a ride on-demand.

7.2 Secure Shortest Path (Distance) Computation

The core of ride-matching is to calculate and compare the shortest path (distance) between riders and drivers. Mouratidis *et al.* [32] proposed a secure shortest path compute scheme, which allows the user to compute the shortest path locally while obtaining data needed from the LBS server without leaking the users' privacy. However, the multiple interactions during the computation will incur heavy computation and communication burden for the user side. Using the same framework, Xi *et al.* [33] proposed a scheme that allows the user computes the shortest path routing locally by retrieving needed data from the server using PIR technique, Samanthula *et al.* [34] proposed a scheme that allows a user to find the shortest path locally while retrieving specific information from the server using the homomorhpic property of Paillier cryptosystem [14]. Xie *et al.* [35] constructs a practical shortest path computation scheme by using Oblivious Storage [36]. However, all these schemes requires the user to retrieve needed data interactively from the remote server during the shortest path computation, which results in heavy computation and communication overhead for the user side. Besides, it not clear how to efficiently conduct the ride-matching that needs to find the nearest driver to the rider among a large number of drivers in this local-computation framework.

Meng *et al.* [10] integrated Somewhat Homomorphic Encryption (SHE) [15] and Distance Oracle [11], and proposed a graph encryption scheme for approximate shortest distance queries. However, the result of this scheme is not accurate enough for ride-hailing service. Wang *et al.* [9] proposed a graph encryption scheme for exact shortest distance based on Paillier cryptosystem [14] and Garbled Circuits [21]. However, the efficiency of this scheme is too low for practical use.

7.3 Solutions for Other Problems

K-anonymity techniques are well investigated location privacy-preserving methods. However, they are not suitable for the scenario of ORH service. Existing k-anonymity based schemes protect location privacy by either altering the locations to be reported [37], [38], [39] or reporting real locations along or interleaved with fake ones [40], [41], [42].

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security

It is obvious that altering the locations may result in mismatches due to the reported inaccurate locations. Moreover, reporting locations with fake ones will incur additional ride-matchings among fake locations and match drivers with fake riders or riders with fake drivers, which compromises the usability of ORH service.

Location-based recommendation systems [43], [44], [45] are widely studied in LBS. They concern about the nearby facilities of the user, and these facilities and locations are static and usually stored in spatial database, while in our ride-matching service, both the rider and drivers are dynamic objects in road networks and the road distance between the rider and drivers is the main selection criteria.

Besides, Cao *et al.* [46] focused on the privacy leakage during payment in ride-hailing and proposed new e-cash system for ride-hailing service that achieves e-cash unforgeability and passenger privacy. Wang *et al.* [47] proposed a privacy-preserving ridesharing recommendation scheme which first models the destination distributions of taxi trips and then utilizes searchable encryption to query the potential of ridesharing.

8 Conclusions

In this paper, we proposed a privacy-preserving ride-matching scheme *pRide* for ORH service. *pRide* is the first scheme that supports privacy-preserving ride-matching over road networks. By using *pRide*, the ORH server can match riders with drivers according to their road distances in the road network without learning the locations of them. To achieve this, *pRide* converts the expensive shortest distance computation in road network into chessboard distance evaluation among sketch points in a higher embedding space. Based on that, an efficient partial homomorphic encryption and garbled circuits are used to preserve the privacy during the matching. We proved that *pRide* is adaptively semantically-secure in the random oracle model. The experimental results over real dataset have shown that *pRide* achieves both high ride-matching accuracy and practical ride-matching efficiency.

Acknowledgments

This work was supported in part by the National Science and Technology Major Project Grants (Project No. 2016YFB0800804), NSF China Grants (Project No. 61379144, 61572026 and 61732022) and the Research Grants Council of Hong Kong (Project No. CityU C1008-16G).

References

[1] H. Caitlin, "Ride-hailing market scale prediction," http://www.marketwatch.com/story/ride-hailing-industry-expected-to-grow-eightfold-to-285-billion-by-2030-2017-05-24, Accessed May 30, 2017.

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

[3] U. M. Aïvodji, "Privacy enhancing technologies for ridesharing," in *Student Forum of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 2016.

[4] P. Goel, L. Kulik, and K. Ramamohanarao, "Privacy-aware dynamic ride sharing," *ACM Transactions on Spatial Algorithms and Systems*, vol. 2, no. 1, p. 4, 2016.

[5] A. B. Sherif, K. Rabieh, M. M. Mahmoud, and X. Liang, "Privacy-preserving ride sharing scheme for autonomous vehicles in big data era," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 611–618, 2017.

[6] W. Tong, J. Hua, and S. Zhong, "A jointly differentially private scheduling protocol for ridesharing services," *IEEE Transactions on Information Forensics and Security*, 2017.

[7] A. Pham, I. Dacosta, B. Jacot-Guillarmod, K. Huguenin, T. Hajar, F. Tramèr, V. Gligor, and J.-P. Hubaux, "PrivateRide: A privacy-enhanced ride-hailing service," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 2, pp. 38–56, 2017.

[8] T. V. A. Pham, I. I. Dacosta Petrocelli, G. F. M. Endignoux, J. R. Troncoso-Pastoriza, K. Huguenin, and J.-P. Hubaux, "ORide: A privacy-preserving yet accountable ride-hailing service," in *Proceedings of the 26th USENIX Security Symposium*, no. EPFL-CONF-228219, 2017.

[9] Q. Wang, K. Ren, M. Du, Q. Li, and A. Mohaisen, "SecGDB: Graph encryption for exact shortest distance queries with efficient updates," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017.

[10] X. Meng, S. Kamara, K. Nissim, and G. Kollios, "GRECS: graph encryption for approximate shortest distance queries," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 504–517.

[11] A. Das Sarma, S. Gollapudi, M. Najork, and R. Panigrahy, "A sketch-based distance oracle for web-scale graphs," in *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 2010, pp. 401–410.

[12] C. Shahabi, M. R. Kolahdouzan, and M. Sharifzadeh, "A road network embedding technique for k-nearest neighbor search in moving object databases," *Proceedings of the tenth ACM international symposium on Advances in geographic information systems - GIS '02*, p. 94, 2002.

[13] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*. IEEE, 1995, pp. 41–50.

[14] P. Paillier *et al.*, "Public-key cryptosystems based on composite degree residuosity classes," in *Eurocrypt*, vol. 99. Springer, 1999, pp. 223–238.

[15] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *TCC*, vol. 3378. Springer, 2005, pp. 325–341.

[16] C. Gentry *et al.*, "Fully homomorphic encryption using ideal lattices." in *STOC*, vol. 9, no. 2009, 2009, pp. 169–178.

[17] A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*. IEEE, 1982, pp. 160–164.

[18] M. O. Rabin, "How to exchange secrets with oblivious transfer," *IACR Cryptology ePrint Archive*, vol. 2005, p. 187, 2005.

[19] N. Linial, E. London, and Y. Rabinovich, "The geometry of graphs and some of its algorithmic applications," in *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*. IEEE, 1994, pp. 577–591.

[20] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 895–934, 2011.

[21] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits." in *USENIX Security Symposium*, vol. 201, no. 1, 2011, pp. 331–335.

[22] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free xor gates and applications," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2008, pp. 486–498.

[23] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 220–250.

[24] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng, "On trip planning queries in spatial databases," in *SSTD*, vol. 5. Springer, 2005, pp. 273–290.

[25] J. Friginal, S. Gambs, J. Guiochet, and M.-O. Killijian, "Towards privacy-driven design of a dynamic carpooling system," *Pervasive and mobile computing*, vol. 14, pp. 71–82, 2014.

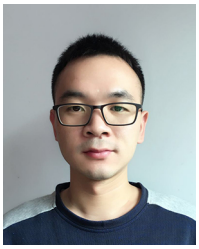
[26] C. Bettini, S. Mascetti, X. S. Wang, D. Freni, and S. Jajodia, "Anonymity and historical-anonymity in location-based services," in *Privacy in location-based applications*. Springer, 2009, pp. 1–30.

[27] C.-Y. Chow, M. F. Mokbel, and X. Liu, "A peer-to-peer spatial cloaking algorithm for anonymous location-based service," in *Pro-*

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2885282, IEEE Transactions on Information Forensics and Security

ceedings of the 14th annual ACM international symposium on Advances in geographic information systems. ACM, 2006, pp. 171–178.
[28] R. Chen, B. C. Fung, N. Mohammed, B. C. Desai, and K. Wang, "Privacy-preserving trajectory data publishing by local suppression," *Information Sciences*, vol. 231, pp. 83–97, 2013.
[29] Y. He, J. Ni, X. Wang, B. Niu, F. Li, and X. S. Shen, "Privacy-preserving partner selection for ride-sharing services," *IEEE Transactions on Vehicular Technology*, 2018.
[30] E. De Cristofaro and G. Tsudik, "Practical private set intersection protocols with linear complexity," in *International Conference on Financial Cryptography and Data Security*. Springer, 2010, pp. 143–159.
[31] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*. ACM, 2009, pp. 139–152.
[32] K. Mouratidis and M. L. Yiu, "Shortest path computation with no information leakage," *Proceedings of the VLDB Endowment*, vol. 5, no. 8, pp. 692–703, 2012.
[33] Y. Xi, L. Schwiebert, and W. Shi, "Privacy preserving shortest path routing with an application to navigation," *Pervasive and Mobile Computing*, vol. 13, pp. 142–149, 2014.
[34] B. K. Samanthula, F.-Y. Rao, E. Bertino, and X. Yi, "Privacy-preserving protocols for shortest path discovery over outsourced encrypted graph data," in *Information Reuse and Integration (IRI), 2015 IEEE International Conference on*. IEEE, 2015, pp. 427–434.
[35] D. Xie, G. Li, B. Yao, X. Wei, X. Xiao, Y. Gao, and M. Guo, "Practical private shortest path computation based on oblivious storage," in *Data Engineering (ICDE), 2016 IEEE 32nd International Conference on*. IEEE, 2016, pp. 361–372.
[36] M. T. Goodrich, M. Mitzenmacher, O. Ohrimenko, and R. Tamassia, "Practical oblivious storage," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*. ACM, 2012, pp. 13–24.
[37] C. A. Ardagna, M. Cremonini, S. D. C. di Vimercati, and P. Samarati, "An obfuscation-based approach for protecting location privacy," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 1, pp. 13–27, 2011.
[38] R. Shokri, "Privacy games: Optimal user-centric data obfuscation," *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 299–315, 2015.
[39] L. Wang, D. Yang, X. Han, T. Wang, D. Zhang, and X. Ma, "Location privacy-preserving task allocation for mobile crowdsensing with differential geo-obfuscation," in *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2017, pp. 627–636.
[40] T.-H. You, W.-C. Peng, and W.-C. Lee, "Protecting moving trajectories with dummies," in *Mobile Data Management, 2007 International Conference on*. IEEE, 2007, pp. 278–282.
[41] R. Kato, M. Iwata, T. Hara, A. Suzuki, X. Xie, Y. Arase, and S. Nishio, "A dummy-based anonymization method based on user trajectory with pauses," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. ACM, 2012, pp. 249–258.
[42] V. Bindschaedler and R. Shokri, "Synthesizing plausible privacy-preserving location traces," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 546–563.
[43] M.-H. Park, J.-H. Hong, and S.-B. Cho, "Location-based recommendation system using bayesian users preference model in mobile devices," in *International Conference on Ubiquitous Intelligence and Computing*. Springer, 2007, pp. 1130–1139.
[44] D. Yang, D. Zhang, Z. Yu, and Z. Wang, "A sentiment-enhanced personalized location recommendation system," in *Proceedings of the 24th ACM Conference on Hypertext and Social Media*. ACM, 2013, pp. 119–128.
[45] J. Bao and Y. Zheng, "Location-based recommendation systems," in *Encyclopedia of GIS*. Springer, 2017, pp. 1145–1153.
[46] C. Cao and X. Zhu, "Practical secure transaction for privacy-preserving ride-hailing services," *Security and Communication Networks*, vol. 2018, 2018.
[47] C. Dai, X. Yuan, and C. Wang, "Privacy-preserving ridesharing recommendation in geosocial networks," in *International Conference on Computational Social Networks*. Springer, 2016, pp. 193–205.



Yuchuan Luo received his BS and MS degrees from the National University of Defense Technology (NUDT) in 2013 and 2015, respectively, both in Computer Science and Technology. He is currently working towards the PhD degree in Computer Science in NUDT. His research interests focus on security and privacy in cloud computing and crowdsourcing.

Xiaohua Jia received his BS and MS degrees from University of Science and Technology of China in 1984 and 1987, respectively, and PhD degree in Information Science from University of Tokyo in 1991. He is currently a Chair Professor with the Department of Computer Science, City University of Hong Kong. His research interests include cloud computing and distributed systems, computer networks and mobile computing. He is an editor of IEEE Internet of Things, IEEE Transactions on Parallel and Distribution Systems (2006-2009), Wireless Networks, Journal of World Wide Web, Journal of Combinatorial Optimization, etc. He is also the General Chair of ACM MobiHoc 2008, TPC Co-Chair of IEEE GlobeCom 2010, Area-Chair of IEEE INFOCOM 2010 and 2015. He is a Fellow of IEEE.

Shaojing Fu Fu received his PhD degree in applied cryptography from National University of Defense Technology in 2012. During his doctoral studies, he also spent a year as a joint doctoral student at University of Tokyo for one year. He is currently an associate professor in the College of Computer, National University of Defense Technology. His research interests include cryptography theory and application, security in cloud and mobile computing.

Ming Xu is currently a professor of the computer science department in the College of Computer, National University of Defense Technology, China. Prof. Xu is a senior member of CCF and member of IEEE and ACM. His major research interests include mobile computing, wireless network, cloud computing and network security.

1556-6013 (c) 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.