Privacy-Preserving Online Ride-Hailing Matching System with an Untrusted Server

Hongcheng Xie[1](✉), Zizhuo Chen[1], Yu Guo[2], Qin Liu[3], and Xiaohua Jia[1]

[1] City University of Hong Kong, Hong Kong, China
{hongcheng.xie,zizhuo.chen}@my.cityu.edu.hk, csjia@cityu.edu.hk

[2] Beijing Normal University, Beijing, China
yuguo@bnu.edu.cn

[3] Wuhan University, Hubei, China
qinliu@whu.edu.cn

Abstract. With the popularity of Online Ride-Hailing (ORH) service, there are growing concerns about location privacy because the taxis and passengers need to upload their locations to the service provider. These locations can be used to infer the users' personal information. In this paper, we propose a privacy-preserving online ride-hailing matching system, which allows an untrusted service provider to calculate the distances between the taxis and one passenger and find the nearest taxi by itself while protecting the users' location privacy. To calculate the distances in road networks, we leverage Road Network Embedding (RNE) in our proposed system. We propose a secure distance calculation scheme to conduct RNE distance calculation securely. In this scheme, we redesign Property-Preserving Hash (PPH) with Pseudo-Random Functions (PRF) and use PRF-based PPH to calculate the distance between two RNE location vectors securely. To enhance security, we embed the partition ID and generation time in PRF-based PPH ciphertext to limit the ciphertext match-ability. Our security analysis and experimental evaluation show that our proposed system is secure and efficient.

1 Introduction

The vigorous development of Online Ride-Hailing (ORH) services has greatly facilitated people's daily travel. Unlike traditional taxi cabs that require passengers to hail a car on the street, ORH allows passengers to request a ride using their mobile phones. The service provider can then match them with the nearest taxi based on their location information. According to Uber's latest financial report [16], there are about 109 million monthly active platform consumers in Uber.

However, with the growing popularity of ORH services, our society has also raised strong concerns about the privacy and security of location data. Passengers and taxis must upload their locations to the untrusted service provider.

X. Yuan et al. (Eds.): NSS 2022, LNCS 13787, pp. 429–442, 2022.
https://doi.org/10.1007/978-3-031-23020-2_24

The service provider can infer their private information, such as driving paths or daily activities, based on their uploaded locations.

To address the security issue, a line of papers [3,9,11,12,18,20] have been proposed to provide ORH services while ensuring location privacy. The works [3,12] were proposed to protect data privacy by cloaking the locations. Since cloaking may result in a loss of matching accuracy, ORide [11] leveraged the cryptographic primitives to encrypt the precise locations and matched the nearest taxi based on Euclidean distances. However, Euclidean distance cannot accurately describe the distances in the road network. In [9], Luo *et al.* proposed a design that uses Road Network Embedding (RNE) [13] as the distance metric instead of Euclidean distance. The authors leveraged Homomorphic Encryption and Garbled Circuits to protect the RNE locations. However, it needs another non-colluded crypto provider to assist the service provider, which leads to high bandwidth overhead and multiple communication rounds. Recently, the authors [20] devised an ORH scheme that allows a service provider to perform privacy-preserving ride-hailing services without involving a third server. Nevertheless, recent work shows that the service provider can infer the underlying plaintext under some scenarios [17].

In this paper, we propose a privacy-preserving ORH scheme that allows an untrusted service provider to find the nearest taxi for a passenger by itself while protecting location privacy. Our design leverages RNE [13] to transform the locations in the road network and calculate the distance between two locations. To conduct the secure ride-matching, we redesign Privacy-Preserving Hash (PPH) with Pseudo-Random Functions (PRFs) and propose a secure difference calculation with PRF-based PPH. With PRF-based PPH, the component in RNE vectors will be divided and encrypted into several bit-block ciphertexts. Each block ciphertext corresponds to a mask weighted difference. Using the binary comparison between the PRF values, the service provider can calculate the RNE distances from the ciphertexts and find the nearest taxi for one passenger. To enhance security, we divide the map into several partitions. The partition ID and ciphertext generation time will be embedded into the block ciphertext. Given a passenger ciphertext, it ensures that only the taxi ciphertexts from the same partition and generated at the same time can be used to calculate the distances. It reduces the number of candidate taxi ciphertexts cryptographically to improve the system security. The efficient primitive PRF also improves the PPH's performance significantly. Besides, we design a tagging scheme to further reduce the computation overhead in PPH. The service provider can compare the PRF values with the same tag. Security analysis and experimental evaluation demonstrate that our proposed system is secure and efficient.

2 Related Works

Some studies proposed some non-cryptographic solutions to hide the exact locations. PrivateRide [12] hides the pick-up and drop-off locations via cloaking. In [22], the authors proposed a cloaking scheme to match the passenger and taxi based on their grid IDs. In [4], the authors proposed a cloaking algorithm based on Hilbert Curve. However, such solutions cannot provide accurate service as they use imprecise locations.

Cryptography techniques, such as secure range query [2,7,19] and homomorphic encryption [6], can be used to address this issue. In [1], the authors proposed a privacy-preserving scheme that determines the meeting point based on Private Set Intersection. The authors in [14] leveraged secure kNN to calculate the trip similarity. Pham *et al.* [11] proposed a scheme that encrypts the exact locations of passengers and taxis and finds the nearest taxi based on homomorphic encryption and their Euclidean distances. However, Euclidean distances cannot represent the distances in road network. To better represent the distances in the road network, some works used different kinds of transformation methods to represent the road distances. Yu *et al.* [21] proposed a privacy-preserving ride matching system that utilized Hypercube embedding to transform the locations. They used Somewhat Homomorphic Encryption to calculate the Hamming distance between two transformed locations in the ciphertext domain. Luo *et al.* [9] utilized Road Network Embedding (RNE) to transform the locations. They leveraged homomorphic encryption and Garbled Circuit to calculate the distances and find the nearest taxi in a privacy-preserving manner. However, it needs another server to help the service provider in ride-matching. In [20], the authors proposed a PPH-based scheme that allows the cloud server to provide the privacy-preserving ORH service by itself. However, the server can infer the plaintexts under some scenarios in this system [17].

3 Preliminaries

3.1 Road Network Embedding

Road Network Embedding (RNE) [13] is a technique that transforms a road network into a high dimensional space to calculate the approximate distance between two points. In this scheme, every point in the road network can be assigned a vector. The distance between two points can be estimated by using their given vectors. The road network can be defined as a weighted graph $G = (V, E)$, where V is the set of road intersections, and E is the set of roads. We assume that G is an undirected graph. Let n denote the size of V, and $d(a, b)$ denote the length of minimum weighted path between a and b. A point u can be transformed into an $O(log^2 n)$-dimension vector as follows.

Let $\beta = O(logn)$ and $\kappa = O(logn)$. We define R as a set which consists of $\beta \cdot \kappa$ subsets of V, i.e., $R = \{S_{1,1}, ..., S_{1,\kappa}, ..., S_{\beta,1}, ..., S_{\beta,\kappa}\}$. Each subset $S_{i,j}$ is a random subset of V with 2^i nodes. For example, the subsets $S_{1,1}, ..., S_{\beta,1}$ have 2 nodes each. The subsets $S_{\beta,1}, ..., S_{\beta,\kappa}$ have 2^β nodes each. Let $D(u, S_{i,j})$ denote the minimum distance between u and the nodes in $S_{i,j}$, i.e., $D(u, S_{i,j}) = \min_{u' \in S_{i,j}} d(u, u')$. Thus, the embedded vector $E(u)$ of node u can be defined as:

$$E(u) = (E_{1,1}(u), ..., E_{1,\kappa}(u), ..., E_{\beta,1}(u), ..., E_{\beta,\kappa}(u)) \quad (1)$$

where $E_{i,j}(u) = D(u, S_{i,j})$.

Now we consider a moving object o which is moving on the edge between node u and node v. The component $E_{i,j}(o)$ of its embedded vector $E(o)$ can be defined as Eq. 2.

$$E_{i,j}(o) = \min\{d(o,u) + D(u, S_{i,j}), d(o,v) + D(v, S_{i,j})\} \tag{2}$$

Given the embedded vectors $E(u)$ and $E(v)$ of two points u and v, the shortest distance $\delta(u,v)$ between u and v can be estimated by calculating the chessboard distance between $E(u)$ and $E(v)$, as shown in Eq. 3.

$$\delta(u,v) = \max_{i,j}(|E_{i,j}(u) - E_{i,j}(v)|) \tag{3}$$

3.2 Pseudo-Random Function

A Pseudo-Random Function (PRF) is a function $f : \{0,1\}^n \times \{0,1\}^s \rightarrow \{0,1\}^m$ where its output is computationally indistinguishable from the output of a random oracle. Formally, a function $f : \{0,1\}^n \times \{0,1\}^s \rightarrow \{0,1\}^m$ is a (t, ϵ, q)-PRF if given $k \in \{0,1\}^s$ and $x \in \{0,1\}^n$, it is efficient to calculate $f_k(x) = f(x,k)$, and for any t-time oracle algorithm A, we have $|Pr_{\in\{0,1\}^s}[A^{f_k}] - Pr_{f\in\mathcal{F}}[A^f]| < \epsilon$ where $\mathcal{F} = \{f : \{0,1\}^n \rightarrow \{0,1\}^m\}$ and A makes at most q queries to the oracle.

3.3 Property-Preserving Hash

Property-Preserving Hash (PPH) [2] allows an entity to reveal the order of ciphertexts. Given the PPH ciphertexts $\hat{x}$ and $\hat{x'}$ of two bits x and x', we define the property P as Eq. 4.

$$P(\hat{x}, \hat{x'}) = \begin{cases} 1 & x = x' + 1 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

In particular, the PPH ciphertext $\hat{x}$ includes two matchable ciphertexts of x and $x+1$. Thus, given $\hat{x}$ and $\hat{x'}$, $P(\hat{x}, \hat{x'}) = 1$ if the matchable ciphertext of x in $\hat{x}$ matches the ciphertext of $x'+1$ in $\hat{x'}$. With property P, the order between two ciphertexts can be determined.

4 Problem Statements

4.1 System Model

Our system model considers a ride-hailing system that helps a passenger find the nearest taxi in the road network. As shown in Fig. 1, our system consists of three entities, i.e., *service provider*, *taxis*, and *passengers*. *Service provider* is the entity that performs taxi-passenger matching based on their encrypted locations. It calculates the approximate distances between each taxi and an incoming passenger based on their ciphertexts and matches them by selecting the nearest taxi. *Taxis* are entities waiting for passengers. They encrypt their locations and upload the ciphertexts to Service Provider periodically to find the matched passenger. *Passengers* are the entities who want to hail the nearest taxi to start their trips. They encrypt their locations and upload the ciphertexts to Service Provider to find the nearest taxi.

According to Eq. 3, to calculate the distance and select the nearest taxi securely, our design should be able to calculate the comparable difference from the vector ciphertexts.

4.2 Threat Model

We assume that the service provider is *honest-but-curious*. It honestly follows our proposed design, but is curious to learn the private information about the locations of the taxis and passengers from the ciphertexts and results. In practice, ride-hailing service provider are usually large reputable companies. Active attacks can be detected easily. We also assume that the passengers and the taxis are *fully trusted* so that they can keep their secret keys secure. The road data that is used to generate the location vectors is assumed to be public.

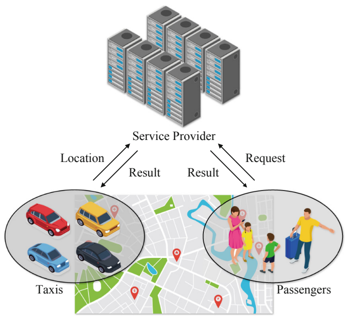


Fig. 1. System Structure.

5 Proposed System

5.1 Secure Distance Calculation from PRF-based PPH

In this section, we discuss about how to encrypt the RNE vectors and calculate the distances between two vectors. Let $E_i(u)$ denote the i-th component of the RNE vector $E(u)$, and z denote the ID of the partition in the map. Suppose that there is one passenger u_p and one taxi u_t in the road network. Their RNE vectors are $E(u_p)$ and $E(u_t)$. They are in the partition z and the vectors are generated at time slot s. To securely calculate the distance between u_p and u_t in the RNE context, we need to securely calculate the difference between the i-th pair of components first, i.e., $E_i(u_p) - E_i(u_t)$.

To encrypt the components $E_i(u_p)$ from the passenger, the passenger first divides the binary representation of $E_i(u_p)$ into m bit-blocks with the same block size l. We denote by $[E_i(u_p)]_j$ the j-th bit-block of $E_i(u_p)$, where j counts from 0 and is indexed from the least significant bit. For instance, we suppose that $E_i(u_p)$ is 41 ("101001" in binary) and it will be divided into 3 bit-blocks with block size 2. $[E_i(u_p)]_2$ is "10", $[E_i(u_p)]_1$ is "10" and $[E_i(u_p)]_0$ is "01". The component $E_i(u_t)$ from the taxi is also divided into m bit-blocks as above.

We discuss about how to encrypt $[E_i(u_p)]_j$ first. As the block size is l, there are 2^l possible values for one block, i.e., from 0 to $2^l - 1$. Let q denote the possible value, i.e., $q \in [0..2^l - 1]$. For each $q \in [0..2^l - 1]$, we calculate the difference between it and $[E_i(u_p)]_j$. Note that we need to multiply the difference by a block weight instead of calculating it directly. Each block $[E_i(u_p)]_j$ has its position-related block weight, similar to bit weight. We denote by w_j the block weight for the j-th block and define $w_j = (2^l)^j$. In the aforementioned example, the weight of $[E_i(u_p)]_2$ is $(2^2)^2 = 16$. The weight represents the contribution of the difference in one block. Let $[E_i(u_p)]_{j,q}$ denote the tuple of the possible value q and the weighted difference between $[E_i(u_p)]_j$ and q. To encrypt one block $[E_i(u_p)]_j$, we encrypt all the possible values q to the matchable ciphertexts in PPH together with their corresponding weighted differences. To protect the weighted difference, it should be masked by a token so that it can be unmasked if and only if there is a correct token from the block ciphertext of the taxi. We will discuss the details later.

The set that needs to be encrypted for $[E_i(u_p)]_j$ is defined as shown in Eq. 5, including all possible values with their weighted differences.

$$\{[E_i(u_p)]_{j,q} = (q, (q - [E_i(u_p)]_j) * w_j) | q \in [0..2^l - 1]\} \tag{5}$$

To encrypt the block $[E_i(u_t)]_j$ from the taxi, we generate the matchable ciphertext for $[E_i(u_t)]_j$ itself instead of all the possible values on the passenger's side. Our basic idea is to ensure that the matchable ciphertext of $[E_i(u_t)]_j$ matches the ciphertext of q if and only if $[E_i(u_t)]_j = q$. Thus, the weighted difference between q and $[E_i(u_p)]_j$ is that between $[E_i(u_t)]_j$ and $[E_i(u_p)]_j$ so that the difference is revealed correctly.

Now we focus on how to encrypt one possible tuple $[E_i(u_p)]_{j,q}$ based on PPH. We leverage PRF as the matching cryptographic primitive. Let $[E_i(\hat{u_p})]_{j,q}$ denote the ciphertext of $[E_i(u_p)]_{j,q}$. It is defined as

$$\begin{aligned} &(tag_{i,j,q}, F(H(k_1, q||i||j||z||s), \gamma_j), \\ &\quad F(H(k_2, q||i||j||z||s), \gamma_j) \oplus ((q - [E_i(u_p)]_j) * w_j)) \end{aligned} \tag{6}$$

where $||$ is the string concatenation operator, $\oplus$ is the XOR operator, H and F are PRFs, γ_j is a random number shared among all the ciphertexts for block $[E_i(u_p)]_j$, and k_1 and k_2 are two secret keys shared among the passengers and taxis. They are distributed by a key manager, an independent admittance control entity that is not involved in taxi matching. $tag_{i,j,q}$ is a tag used to improve the query efficiency, which is defined as

$$tag_{i,j,q} = F(H(k_1, q||i||j||z||s), \gamma_j) \& (2^\theta - 1) \tag{7}$$

where & is bit-wise AND operator, and θ is a pre-defined value which is smaller than l. That means we use the last θ bits as the tag of this ciphertext. The ciphertexts with the same tag can be grouped together. The ciphertext order in one group can be shuffled. We will discuss about how to use it to improve the query efficiency later.

We denote by $[E_i(\hat{u_p})])_j$ the ciphertext of block $[E_i(u_p)]_j$. $[E_i(\hat{u_p})]_j$ includes the ciphertexts of all possible tuples $[E_i(u_p)]_{j,q}$ and the random nunce γ_j, as shown in Eq. 8.

$$[E_i(\hat{u_p})]_j = \{\gamma_j, [E_i(\hat{u_p})]_{j,q} | q \in [0..2^l - 1]\} \tag{8}$$

In the above discussion, we have discussed that we generate the matchable ciphertext for $[E_i(u_t)]_j$ itself for the taxi block. In particular, the ciphertext $[E_i(\hat{u_t})]_j$ is defined as Eq. 9.

$$(H(k_1, [E_i(u_t)]_j || i || j || z || s), H(k_2, [E_i(u_t)]_j || i || j || z || s)) \tag{9}$$

The ciphertexts of $E_i(u_p)$ and $E_i(u_t)$ consists of the ciphertexts of all the blocks respectively, as shown in Eq. 10 and Eq. 11.

$$E_i(\hat{u_p}) = \{[E_i(\hat{u_p})]_j | j \in [0..2^l - 1]\} \tag{10}$$

$$E_i(\hat{u_t}) = \{[E_i(\hat{u_t})]_j | j \in [0..2^l - 1]\} \tag{11}$$

Thus, given two block ciphertexts $[E_i(\hat{u_p})]_j$ and $[E_i(\hat{u_t})]_j$, the service provider can scan the possible ciphertexts in $[E_i(\hat{u_p})]_j$. The block ciphertext $[E_i(\hat{u_t})]_j$ matches one possible ciphertext $[E_i(\hat{u_p})]_{j,q}$ if and only if Eq. 12 holds, which the left side is from $[E_i(\hat{u_p})]_{j,q}$ and the right side is the PRF value with the ciphertext from $[E_i(\hat{u_t})]_j$ and γ_j from $[E_i(\hat{u_p})]_j$.

$$F(H(k_1, q || i || j || z || s), \gamma_j) \overset{?}{=} F(H(k_1, [E_i(u_t)]_j || i || j || z || s), \gamma_j) \tag{12}$$

According to Eq. 12, we can find that Eq. 12 holds if and only if $q = [E_i(u_t)]$. The other parameters in PRF also ensures that they are from the j-th block in the i-th components, from the same zone z, and generated at the same time slot s.

To improve query efficiency, we can extract the last θ bits of the right side in Eq. 12, as shown in Eq. 13. We can only test the possible ciphertexts $[E_i(\hat{u_p})]_{j,q}$ with the same tag, i.e., $tag' = tag_{i,j,q}$, to reduce the number of matching tests.

$$tag' = F(H(k_1, [E_i(u_t)]_j || i || j || z || s), \gamma_j) \& (2^\theta - 1) \tag{13}$$

Once $[E_i(\hat{u_t})]_j$ matches one $[E_i(\hat{u_p})]_{j,q}$, we can generate the mask $mask_j$ with $H(k_2, [E_i(u_t)]_j || i || j || z || s)$ from $[E_i(\hat{u_t})]_j$ and γ_j from $[E_i(\hat{u_p})]_j$, as shown in Eq. 14. According to Eq. 6, we can find that $mask_j$ is equal to the mask of weighted difference $(q - [E_i(u_p)]_j) * w_j$ if $[E_i(u_t)]_j = q$, i.e., they are matched.

Thus, by using XOR operation, we can reveal the weighted difference $([E_i(u_t)]_j - [E_i(u_p)]_j) * w_j$ from the block ciphertext.

$$mask_j = F(H(k_2, [E_i(u_t)]_j||i||j||z||s), \gamma_j) \tag{14}$$

After revealing $([E_i(u_t)]_j - [E_i(u_p)]_j) * w_j$ for all $j \in [0..m]$, the absolute difference between the i-th components $|E_i(u_t) - E_i(u_p)|$ can be calculated as Eq. 15.

$$|E_i(u_t) - E_i(u_p)| = |\sum_{j \in [0..m]} (([E_i(u_t)]_j - [E_i(u_p)]_j) * w_j)| \tag{15}$$

Let $E(\hat{u_t})$ and $E(\hat{u_p})$ denote the ciphertexts of RNE vectors $E(u_t)$ and $E(u_p)$ respectively. The ciphertext $E(\hat{u_t})$ for the taxi is defined as Eq. 16. It consists of the ciphertexts of all the components, the time slot and the partition ID.

$$E(\hat{u_t}) = \{s, z, E_i(\hat{u_t}) | i \in [0..\beta \cdot \kappa]\} \tag{16}$$

The ciphertext $E(\hat{u_p})$ for the passenger is defined as Eq. 17.

$$E(\hat{u_p}) = \{s, z, E_i(\hat{u_p}) | i \in [0..\beta \cdot \kappa]\} \tag{17}$$

As we can calculate the absolute differences for one pair of components, given the ciphertexts of two RNE vectors, the distance between one passenger and one taxi can be calculated according to Eq. 3.

In summary, the service provider can calculate the RNE distance between a passenger and a taxi securely if they are from the same partition and their ciphertexts are generated in the same time slot.

5.2 Passenger-Driver Matching

In Sect. 5.1, we discussed about how to calculate the distance between a passenger and a taxi. Let U_t denote the set of all taxis, and $E(\hat{U_t})_{z,s}$ denote the ciphertexts of the taxis with partition ID z and time slot s. Given a passenger's ciphertext $E(\hat{u_p})$ with partition ID z and time slot s, as the partition ID and time slot are included explicitly, the service provider can select the taxis with the same partition ID and time slot to calculate the distances, i.e., between $E(\hat{u_p})$ and the ciphertexts in $E(\hat{U_t})_{z,s}$. It can reduce the number of candidate taxis to improve the query efficiency. These two values that are embedded in PRFs also ensures that the service provider cannot calculate the distance between two ciphertexts if their partition IDs or time slots are different. After calculating all the distances between $E(\hat{u_p})$ and the ciphertexts in $E(\hat{U_t})_{z,s}$, the service provider selects the taxi with minimum distance as the nearest taxi. The distance does not leak the location information of both passenger and taxi.

In the above design, only the passenger and taxi in the same partition can be matched. However, the nearest taxi in one partition may not be the nearest one in the global map. To improve the accuracy, we let the taxi in the partition z generate the another extra 8 ciphertexts of its current RNE vectors with z', which is the adjacent partition of z. Thus, a taxi will participate the matching procedure for its partition and its neighbor partitions. Relatively, one hailing request from a passenger will be served by the taxis from its partition and its neighbor partitions. This design can improve the query accuracy of our proposed system.

5.3 Early Stopping in Distance Calculation

In the above discussion, we calculate the absolute differences for all the components and select the maximum as the distance between the passenger and one taxi. The taxi with smallest distance will be selected as the nearest taxi. As the service provider needs to calculate the distances between the passenger and taxis one by one, we denote by min as the current smallest distance when the service provider is calculating the distance between the passenger u_p and one taxi u_t. We have one observation.

Observation: We suppose that the service provider is calculating the absolute difference between the i-th pair of components, i.e., $|E_i(u_t) - E_i(u_p)|$. u_t is not the nearest taxi if $|E_i(u_t) - E_i(u_p)| > min$, as the distance between u_p and u_t is impossible to be smaller than $|E_i(u_t) - E_i(u_p)|$ according to Eq. 3. The distance must be larger than the current smallest distance min if $|E_i(u_t) - E_i(u_p)| > min$.

According to the above *Observation*, the distance calculation between u_p and u_t can be terminated if the service provider finds that $|E_i(u_t) - E_i(u_p)| > min$. Thus, the unneccessary calculation can be avoided.

6 Security Analysis

6.1 Leakage Definition

As the difference between two components are calculated from the revealed weighted differences in plaintext, our security analysis will focus on the secure block weighted difference calculation. First we define the following leakage functions:

Leakage Function $\mathcal{L}_1(u)$: Let $(tag_{i,j,q}, CT_1, CT_2)$ denote the ciphertext in Eq. 6. Given a block value u from one passenger, the leakage function for passengers is defined as $\mathcal{L}_1(u) = (\langle |tag_{i,j,q}|, |CT_1|, |CT_2| \rangle, l, |\gamma|)$, where l is the block size, and $|tag_{i,j,q}|$,$|CT_1|$,$|CT_2|$, and $|\gamma|$ are the bit lengths.

Leakage Function $\mathcal{L}_2(u^*)$: Let (CT_1', CT_2') denote the ciphertext in Eq. 9. Given a block value u^* from one taxi, the leakage function for taxis is defined as $\mathcal{L}_2(u^*) = (|CT_1'|, |CT_2'|)$.

Leakage Function $\mathcal{L}_3(\hat{u}, \hat{u^*})$: Given two block ciphertexts from a passenger and a taxi, the leakage function for the comparison is defined as $\mathcal{L}_3(\hat{u}, \hat{u^*}) = (MP, dif, N_{t\times t})$, where MP is the matched possible ciphertext, dif is the revealed weighted difference, and $N_{t\times t}$ is a symmetric binary matrix that records the repeated comparisons.

Based on the simulation-based security definition in [5], the security definition of our system is defined as follows.

Definition 1. *Let* $\Omega = \{\text{EncPassBlock, EncTaxiBlock, BlockDiff}\}$ *be the secure block weighted difference calculation scheme,* $\mathcal{A}$ *be a probabilistic polynomial time (PPT) adversary,* $\mathcal{S}$ *be a PPT simulator, and* λ *be the security parameter. The probabilistic experiments* $\boldsymbol{Real}_{\Omega,\mathcal{A}}(1^\lambda)$ *and* $\boldsymbol{Ideal}_{\Omega,\mathcal{A},\mathcal{S}}(1^\lambda)$ *are defined as:*

$\boldsymbol{Real}_{\Omega,\mathcal{A}}(1^\lambda)$*:* $\mathcal{A}$ *selects a block value* u *and lets a passenger generate a ciphertext via EncPassBlock. Then* $\mathcal{A}$ *launches a polynomial number of comparisons, which lets a taxi generate the ciphertext via EncTaxiBlock, and gets the result via BlockDiff. Finally,* $\mathcal{A}$ *outputs a bit as the output.*

$\boldsymbol{Ideal}_{\Omega,\mathcal{A},\mathcal{S}}(1^\lambda)$*:* $\mathcal{A}$ *selects the block value* u*.* $\mathcal{S}$ *simulates the ciphertext for* $\mathcal{A}$ *based on* $\mathcal{L}_1$*. Then* $\mathcal{A}$ *launches a polynomial number of comparisons adaptively, which lets* $\mathcal{S}$ *simulate the ciphertexts based on* $\mathcal{L}_2$ *and the comparisons based on* $\mathcal{L}_3$*. Finally,* $\mathcal{A}$ *outputs a bit as the output.*

Definition 2. Ω *is* $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3)$*-secure if for any PPT adversary* $\mathcal{A}$*, there exists a PPT simulator* $\mathcal{S}$ *such that* $|Pr[\boldsymbol{Real}_{\Omega,\mathcal{A}}(1^\lambda) = 1] - Pr[\boldsymbol{Ideal}_{\Omega,\mathcal{A},\mathcal{S}}(1^\lambda) = 1]| \leq neg(\lambda)$*, where* $neg(\lambda)$ *is a negligible function for* λ*.*

6.2 Analysis

Theorem 1. Ω *is* $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3)$*-secure if* F *and* H *are PRFs.*

Proof. We first define a sequence of hybrid experiments.

$\mathcal{H}_0$: It is the same as the experiment $\mathbf{Real}_{\Omega,\mathcal{A}}(1^\lambda)$.

$\mathcal{H}_1$: It is the same as $\mathcal{H}_0$, except that $\mathcal{S}$ picks random strings as the passenger's ciphertext using $\mathcal{L}_1$ instead of calling F.

$\mathcal{H}_2$: It is the same as $\mathcal{H}_1$, except that $\mathcal{S}$ picks random strings as the taxi's ciphertext using $\mathcal{L}_2$ instead of calling H.

$\mathcal{H}_3$: it is the same as $\mathcal{H}_2$, except that $\mathcal{S}$ simulates the comparison result using $\mathcal{L}_3$ adaptively, which is $\mathbf{Ideal}_{\Omega,\mathcal{A},\mathcal{S}}(1^\lambda)$.

First, due to the pseudo-randomness of PRF, $\mathcal{A}$ cannot distinguish between a PRF string and a random string computationally. Thus, $\mathcal{H}_0$ and $\mathcal{H}_1$ is computationally indistinguishable to $\mathcal{A}$. Due to the same reason, $\mathcal{H}_1$ and $\mathcal{H}_2$ is also computationally indistinguishable to $\mathcal{A}$. In $\mathcal{H}_3$, $\mathcal{S}$ generates the random strings if one taxi's block value is not requested before. Otherwise, it returns the same ciphertext. As discussed before, $\mathcal{A}$ cannot distinguish between the random string and the ciphertexts generated by PRF computationally. Thus, $\mathcal{H}_2$ and $\mathcal{H}_3$ are computationally indistinguishable. In summary, $\mathbf{Real}_{\Omega,\mathcal{A}}(1^\lambda)$ and $\mathbf{Ideal}_{\Omega,\mathcal{A},\mathcal{S}}(1^\lambda)$ are computationally indistinguishable to $\mathcal{A}$. This completes the proof.

6.3 Discussion About the Difference Leakage

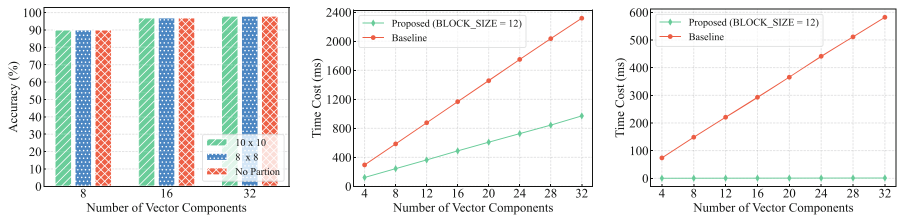
In the comparison, weighted differences for matched possible ciphertexts will be leaked to the service provider. One ciphertext for a passenger's block includes

2^l possible ciphertexts with different weighted differences. An attack against PPH shows that a passenger's block value can be revealed if the service provider collects block ciphertexts from 2^l taxis, each of which leaks a different weighted difference from this passenger's block ciphertext [17]. However, in our proposed system, the attack is difficult to be achieved under real-world conditions. For example, according to the statistics [15] from Uber, there are about 14,000 taxis in Hong Kong. We suppose that taxis are distributed uniformly. If we set the partition scheme to 20×20 and set the block size l to 12, there are about 315 taxis in a 3×3 area, which can participate in one passenger's comparison, but the number of needed taxis is at least 4096. It is difficult to be achieved.

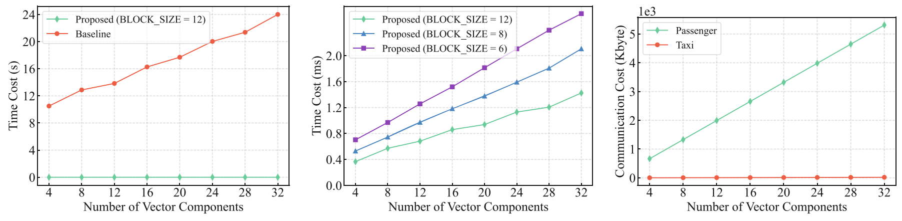
7 Experimental Evaluation

7.1 Experiment Setup

The prototypes of our proposed system and the baseline [20] are implemented by C. The evaluations were conducted on a computer equipped with Intel i7-10700 and 64 GB memory. The binary length of one component in a vector is set as 24. Our proposed system uses HMAC-SHA256 as the PRF and uses the first 128 bits. We compare with the baseline [20], a privacy-preserving ride-hailing system based on RNE and PPH with bilinear maps. It is implemented by Pairing-Based Cryptography library [10] with Type A, where r is 160 bits and q is 512 bits. The block size l in the baseline is set as 2. We use a real-world dataset [8] which includes road information of 21048 nodes and 21693 edges. The passengers and taxis are generated randomly on the edges. The results is the mean of 10 trials.



(a) Matching Accuracy (b) Passenger's Generation (c) Taxi's Generation



(d) On Different Schemes (e) On Different Block Sizes (f) Communication Cost

Fig. 2. System evaluation

7.2 Performance

Accuracy: Since RNE estimates the distances in the road network, it may incur inaccurate result. Moreover, we use partition to reduce the number of candidate taxis. We evaluate the accuracy of our system and the influence of partition granularity on accuracy. We generate 100 passengers randomly and count the number of passengers whose matched taxis from RNE are the same as the ground truths. From Fig. 2a, the accuracy raises as the component raises. The accuracy is 98% when the component is 32. Our partition scheme is not influence on the accuracy.

The Performance of Generation: Figure 2b shows that the passenger's time cost of ciphertext generation increases when the number of components increases. Figure 2c illustrates that taxi's also increases as the number of components increases. The taxi's time cost is smaller than the passenger's, although the taxi needs to encrypt 9 vector ciphertexts for different partition IDs. That is because the passenger needs to encrypt several possible values for each block encryption, while the taxi only needs to encrypt one value. According to Fig. 2f, as the same reason, the taxi's cost is smaller than the passenger's. It is reasonable because taxis need to upload their tokens frequently to wait for the passenger.

The Performance of Taxi-Passenger Matching: Our evaluation is conducted under 800 taxis and the partition scheme 10×10. According to Fig. 2d, we can find that the time cost of our scheme is significantly better than the baseline. From Fig. 2e, the time costs of all the three block settings raise linearly as the number of components increases. We can find that the time cost decreases as the block size increases, as the increase of the block size leads to the decrease of the number of blocks in one vector component. Although the number of the ciphertexts for the possible values is exponential to the block size in one block, the service provider needs to compare only a small part of them by using our tag optimization scheme. The comparison costs in one block with different block size are close. Thus, less blocks result in less time cost. PRFs provide efficient ciphertext matching, and our partition design and tagging scheme reduce the number of ciphertext matching operations.

8 Conclusion

In this work, we proposed a privacy-preserving online ride-hailing system with an untrusted server. The service provider can find the nearest taxi for one passenger by itself. We used RNE to transform the road location to a vector. To calculate the distance securely, we redesigned PPH using PRF as its matching primitive, and used PRF-based PPH to propose a secure distance calculation scheme. To enhance security, we divided the map into several partitions and embedded the information about partition and generation time into PPH ciphertexts. The efficient PRFs and our tag scheme improve the efficiency. We provided the security analysis, and evaluation results showed that our system is accurate and efficient.

Acknowledgement. This work is financially supported by NSFC under Grant No. 62102035, and HKRGC under Grant No. CityU 11213920.

References

1. Aïvodji, U.M., Gambs, S., Huguet, M.J., Killijian, M.O.: Meeting points in ridesharing: a privacy-preserving approach. Transp. Res Part C: Emerging Technol. **72**, 239–253 (2016)
2. Cash, D., Liu, F.-H., O'Neill, A., Zhandry, M., Zhang, C.: Parameter-hiding order revealing encryption. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11272, pp. 181–210. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03326-2_7
3. Chow, C.Y., Mokbel, M.F., Liu, X.: A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In: Proceedings of the 14th ACM GIS, pp. 171–178 (2006)
4. Cui, N., Yang, X., Wang, B.: A novel spatial cloaking scheme using hierarchical hilbert curve for location-based services. In: Cui, B., Zhang, N., Xu, J., Lian, X., Liu, D. (eds.) WAIM 2016. LNCS, vol. 9659, pp. 15–27. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39958-4_2
5. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. J. Comput. Secur. **19**(5), 895–934 (2011)
6. Fan, J., Vercauteren, F.: Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive (2012)
7. Guo, Y., Xie, H., Wang, M., Jia, X.: Privacy-preserving multi-range queries for secure data outsourcing services. IEEE TCC (2022)
8. Li, F., Cheng, D., Hadjieleftheriou, M., Kollios, G., Teng, S.-H.: On trip planning queries in spatial databases. In: Bauzer Medeiros, C., Egenhofer, M.J., Bertino, E. (eds.) SSTD 2005. LNCS, vol. 3633, pp. 273–290. Springer, Heidelberg (2005). https://doi.org/10.1007/11535331_16
9. Luo, Y., Jia, X., Fu, S., Xu, M.: pride: privacy-preserving ride matching over road networks for online ride-hailing service. IEEE TIFS **14**(7), 1791–1802 (2018)
10. Lynn, B.: On the implementation of pairing-based cryptosystems. Ph.D. thesis, Stanford University Stanford, California (2007)
11. 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: 26th {USENIX} Security 17), pp. 1235–1252 (2017)
12. Pham, A., et al.: Privateride: a privacy-enhanced ride-hailing service. PoPETs **2017**(2), 38–56 (2017)
13. Shahabi, C., Kolahdouzan, M.R., Sharifzadeh, M.: A road network embedding technique for k-nearest neighbor search in moving object databases. GeoInformatica **7**(3), 255–273 (2003)
14. Sherif, A.B., Rabieh, K., Mahmoud, M.M., Liang, X.: Privacy-preserving ride sharing scheme for autonomous vehicles in big data era. IEEE IoTJ **4**(2), 611–618 (2016)
15. Uber: Uber marks 6 years in Hong Kong (2020). https://www.uber.com/en-HK/newsroom/uber-marks-6-years-in-hong-kong/
16. Uber: Uber announces results for third quarter 2021 (2021). https://investor.uber.com/news-events/news/press-release-details/2021/Uber-Announces-Results-for-Third-Quarter-2021/

17. Vivek, S.: Comments on" a privacy-preserving online ride-hailing system without involving a third trusted server. arXiv preprint arXiv:2112.06449 (2021)
18. Wang, F., et al.: Efficient and privacy-preserving dynamic spatial query scheme for ride-hailing services. IEEE TVT **67**(11), 11084–11097 (2018)
19. Wong, W.K., Cheung, D.W.l., Kao, B., Mamoulis, N.: Secure KNN computation on encrypted databases. In: Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data, pp. 139–152 (2009)
20. Xie, H., Guo, Y., Jia, X.: A privacy-preserving online ride-hailing system without involving a third trusted server. IEEE TIFS **16**, 3068–3081 (2021)
21. Yu, H., Jia, X., Zhang, H., Shu, J.: Efficient and privacy-preserving ride matching using exact road distance in online ride hailing services. IEEE TSC **15**, 1841–1854 (2020)
22. Zhu, L., Li, M., Zhang, Z., Qin, Z.: Asap: an anonymous smart-parking and payment scheme in vehicular networks. IEEE TDSC **17**(4), 703–715 (2018)