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/JIOT.2016.2569090, IEEE Internet of Things Journal

Privacy-Preserving Ride Sharing Scheme for Autonomous Vehicles in Big Data Era

Ahmed B. T. Sherif, Khaled Rabieh, Mohamed Mahmoud, *Member, IEEE*, Xiaohui Liang *Member, IEEE*

Abstract—Ride sharing can reduce the number of vehicles in the streets by increasing the occupancy of vehicles, which can facilitate traffic and reduce crashes and the number of needed parking slots. Autonomous Vehicles (AVs) can make ride sharing convenient, popular, and also necessary because of the elimination of the driver effort and the expected high cost of the vehicles. However, the organization of ride sharing requires the users to disclose sensitive detailed information not only on the pick-up/drop-off locations but also on the trip time and route. In this paper, we propose a scheme to organize ride sharing and address the unique privacy issues. Our scheme uses a similarity measurement technique over encrypted data to preserve the privacy of trip data. The ride sharing region is divided into cells and each cell is represented by one bit in a binary vector. Each user should represent trip data as binary vectors and submit the encryptions of the vectors to a server. The server can measure the similarity of the users' trip data and find users who can share rides without knowing the data. Our analysis has demonstrated that the proposed scheme can organize ride sharing without disclosing private information. We have implemented our scheme using Visual C on a real map and the measurements have confirmed that our scheme is effective when ride sharing becomes popular and the server needs to organize a large number of rides in short time.

Index Terms—Privacy preservation, ride sharing, autonomous vehicles, and search over encrypted data.

1 Introduction

Over the past few years, the automobile industry has made significant leaps in bringing automation to car driving [1]. Autonomous Vehicles (AVs) are equipped with advanced sensing and communication capabilities, navigation devices, computer vision technology, etc., to enable the vehicles to autonomously drive themselves without any intervention from humans [2]. AVs have the potential to fundamentally advance transportation systems by reducing crashes, assisting traffic flows, and reducing travel time. However, the technology used in AVs is expensive and the AVs' price will be high [3].

Since AVs can drive themselves, they may not be personal devices anymore, but on-demand service. Currently, people used to own cars but in the future instead of owning an AV, many people can order an AV from a cab company when they need. This is an interesting and promising way to alleviate the high cost problem of the AVs. Such on-demand service will be efficient and popular in AVs due to the elimination of the human driver effort.

Ride-sharing (or carpooling) allows AVs to be shared by users, e.g., to share the cost of on-demand cab service. Since AVs can drive themselves, they will make ride sharing convenient, popular, and sometimes necessary. However, to organize ride sharing, users need to disclose not only their trips' pick-up/drop-off locations and times, but also their complete routes. The locations can include residences, places of employment, places of amusement, etc. [4]. Leaking location information will directly result in negative impacts, e.g., knowing the time a person will leave home is particularly useful information to the community of thieves.

In this paper, we propose a privacy-preserving scheme to organize ride sharing. To our knowledge, existing privacy-preserving techniques [5]–[7] cannot be applied effectively and efficiently in ride sharing due to the unique problems and requirements. Also, hiding the users' identities [8] is not enough because attackers can identify the users from their pick-up/drop-off locations.

We use a group signature scheme, such as our proposal in [9], to ensure users anonymity. We also use a similarity measurement technique over encrypted data, such as [10]–[12], to enable a server to measure the similarity of the users' trip data without knowing the data. Once the server finds a user who can share ride, it sends the user's signature to the AV user who can trace the signature to the signer's identity. Our scheme considers different cases for ride sharing and allows users to prescribe their preferences, such as the maximum distance between a trip's start/end locations and a user's pick-up/drop-off locations.

Our analysis has demonstrated that the proposed scheme can organize ride sharing without disclosing sensitive information. We have implemented the scheme using Visual C on a real map and measured performance metrics. The results have demonstrated that the communication and storage overhead is acceptable and the search time to organize a shared ride is very small.

The remainder of this paper is organized as follows. The network and threat models are discussed in Section 2. The used similarity measurement technique is ex-

- *Ahmed B. T. Sherif, Khaled Rabieh, and Mohamed Mahmoud are with the Center for Manufacturing Research (CMR), Tennessee Tech University, Cookeville, Tennessee, 38505, USA.*
  E-mail: absherif42@students.tntech.edu, kmrabieh42@students.tntech.edu, and mmahmoud@tntech.edu
- *Xiaohui Liang is with Department of Computer Science, University of Massachusetts at Boston, Boston, MA 02125, USA.*
  E-mail: Xiaohui.Liang@umb.edu

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/JIOT.2016.2569090, IEEE Internet of Things Journal

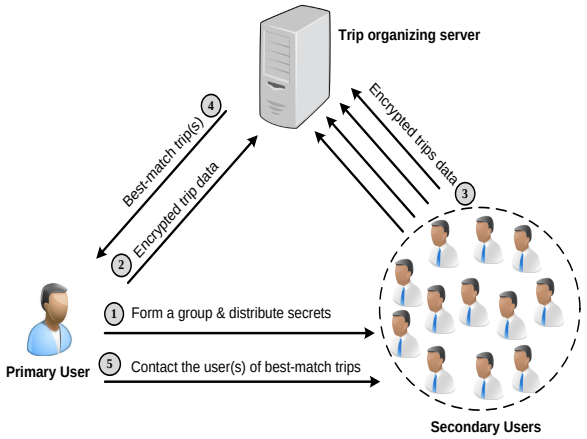


Fig. 1: The exchanged messages in our scheme.

plained in Section 3. The proposed scheme is presented in Section 4. The evaluation results are discussed in Section 5. The related works are summarized in Section 6. Finally, conclusions are drawn in Section 7.

2 Network and Threat Models

As illustrated in Fig. 1, there are three main parties in the network model: primary user, secondary users, and a server. The primary user is the person who has the full right to use the AV. He should decide whether he will share rides or not, the number of users who can share the ride with him, and other preferences that will be explained later. The secondary users are the persons who want to share rides with the primary user. The server is an independent party that is operated by a third party. The primary and secondary users are members in a group that can be for university students, friends, or a company employees, etc. They use smart phones connected to the Internet to communicate to the server.

The server and the users are considered "honest-but-curious". Specifically, they do not aim to disrupt the proper operation of the scheme, but they are curious to learn information about the users' trips such as the pick-up/drop-off locations and times. In addition, the server is interested to know the identities of the users who share rides. Since the server is an independent party, it does not collude with the primary or secondary users.

3 Preliminaries

3.1 Similarity Measurement over Encrypted Data

In [11], Cao et. al. have adapted the dot product computation used in [12] to propose a k-Nearest Neighbors (kNN) secure algorithm to enable a server to extract the documents that have certain keywords without learning the keywords. The keywords are represented by a binary vector. If a document has a keyword, the corresponding bit in the vector is one, otherwise it is zero. Then, the vector is encrypted using secret keys to produce an index. A document owner uploads the document and its keywords index to the server. Then, when a user wants to download the documents that have

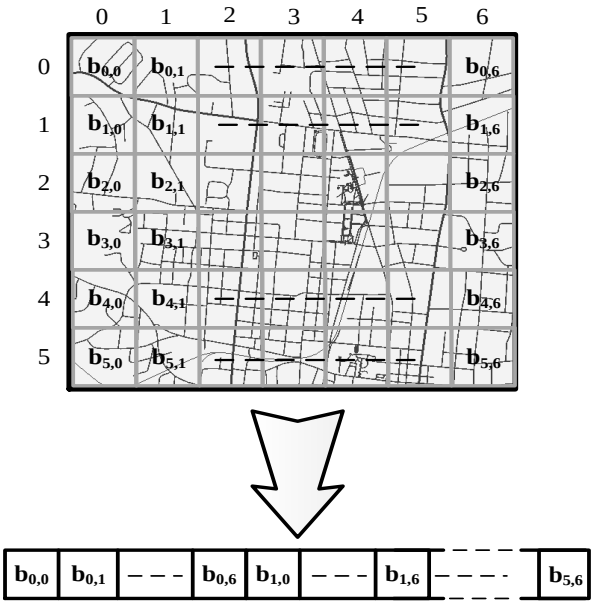


Fig. 2: Representing a ride sharing area as a binary vector.

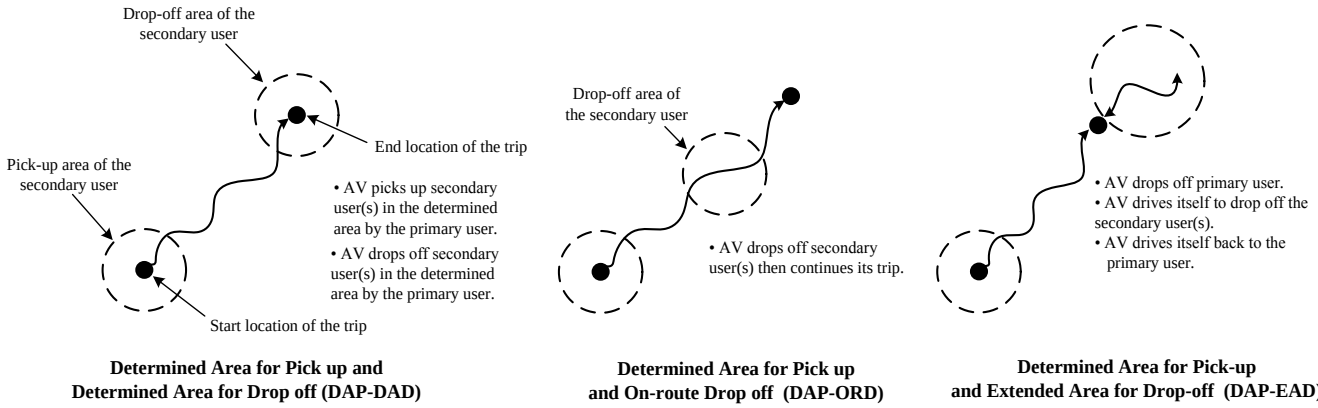
certain keywords, he sends an encrypted query index with the keywords to the server. The server applies an inner product operation to the query's index and each document's index to measure the similarity of the two indices without knowing the keywords. The result of the inner product gives the number of shared keywords. In the following, we explain this technique in details.

The secret keys used to create the indices include two matrices and a row vector. The matrices M_1 and M_2 and a vector S are used by the document owner, while the matrices $(M_1^{-1})^T$ and $(M_2^{-1})^T$ with the same row vector S are used by users. First, the document owner creates n-dimensional bit vector for the document keywords, where n is the number of keywords. Each element in the vector represents one keyword and it is one if the document has the keyword, otherwise it is zero. Then, p is split into two random vectors p' and p''. The vector S is used as a splitting function indicator. If the j-th bit of S is zero, $p'[j]$ and $p''[j]$ are set similar to $p[j]$, and if the j-th bit of S is one, $p'[j]$ and $p''[j]$ are set to two random numbers so that their summation is equal to $p[j]$. Finally, the document owner's index is $(p' \cdot M_1, p'' \cdot M_2)$.

To create a query index, users first generate n-dimensional bit vector s for the keywords. Then, s is split into two random vectors s' and s''. If the j-th bit of S is one, $s'[j]$ and $s''[j]$ are set similar to $s[j]$. If the j-th bit of S is zero, $s'[j]$ and $s''[j]$ are set to two random numbers so that their summation is $s[j]$. Finally, the query's index is $(s' \cdot (M_1^{-1})^T, s'' \cdot (M_2^{-1})^T)$.

To calculate the number of shared keywords in the indices of the document owner and the user, the two indices are multiplied using inner product.

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/JIOT.2016.2569090, IEEE Internet of Things Journal



Determined Area for Pick up and Determined Area for Drop off (DAP-DAD)

Determined Area for Pick up and On-route Drop off (DAP-ORD)

Determined Area for Pick-up and Extended Area for Drop-off (DAP-EAD)

Fig. 3: Ride sharing cases.

4 The Proposed Scheme

4.1 Overview

Fig. 1 illustrates the required steps in our scheme. The primary user creates a group of secondary users who can share rides with him. It should calculate a group signature credentials and distribute them to the group members. The group can be used for a long time to organize many shared rides. Since the primary user is the group creator, he can trace the secondary users' signatures to know the identity of the signer. The primary user should also compute the key set $\{S, (M_1^{-1})^T, (M_2^{-1})^T\}$ and send it to the group members. It should also send the public parameters of the group signature to the server to enable it to verify the secondary users' signatures to make sure that they are members in the group.

As indicated in Fig. 1, if a primary user wants to share a trip with secondary users, it uses the technique discussed in Section 3.1 to create indices for the the trip data. Then, it sends a packet to the server having the indices along with a signature and other information like the number of secondary users he needs to share the trip with and the case(s) of ride sharing he prefers. Similarly, the secondary users submit packets having indices for their trip data and group signatures. The server uses the indices to find a secondary user who can share the ride with the primary user. Once such a user is found, the server sends the secondary user's signature to the primary user to trace the signature and identify the user.

4.2 Trip Data Creation

The similarity measurement technique explained in Section 3.1 needs to represent data as a binary vector. To be able to use this technique to organize ride sharing, we propose dividing the ride sharing region into cells as indicated in Fig. 2. Then, in reporting trip data, each cell is represented by a bit in a vector. In the figure, the ride sharing area is divided into 6 rows $\times$ 7 columns (42 cells). $b_{i,j}$ is one bit in the trip data vector and represents cell (i, j), where $b_{i,j} \in \{0, 1\}$ and i and j are the row and column numbers, respectively. The figure shows how each cell is mapped to an element in a bit vector. The cells of the first row are represented in the vector first then the cells of the second row are next, and so on.

4.3 Ride Sharing Cases

Our scheme considers multiple cases for ride sharing. These cases are illustrated in Fig. 3. In *determined area for pick up and determined area for drop off ($DAP-DAD$)*, the primary user decides the pick-up and drop-off areas of the secondary user. These areas are in the surrounding of the trip's start and end locations. In human-driven cars, ride sharing is not convenient if the pick-up and drop-off locations are not too close to the trip's start and end locations, but in AVs, they do not need to be too close because cars can drive themselves.

The second case of ride sharing is called *determined area for pick-up and on route drop-off ($DAP-ORD$)*. In this case, the primary user decides the pick-up area like $DAP-DAD$ case. For the drop off, the primary user can share a ride with a user whose drop-off location is different from his trip's end location, but it lies on the surrounding of his route. In this case, the AV diverts from its route to drop off the secondary user(s) and then it drives back to the trip's route.

The third case of ride sharing is called *determined area for pick-up and extended area for drop-off ($DAP-EAD$)*. In this case, the primary user decides the pick-up area like $DAP-DAD$ and $DAP-ORD$ cases, but the drop-off location of the secondary user does not lie on the surrounding area of his trip's route or end location. The AV drops off the primary user first, then it delivers the secondary user and then drives back to the primary user. This case is not possible in human-driven cars unless there is a human driver who can drive the car back to the primary user after delivering the secondary user.

Since users have different preferences, when the primary user uploads its trip information, it should prescribe the ride sharing case(s) he prefers.

4.3.1 Organizing DAP-DAD Shared Rides

In this case, the primary user and each secondary user should first create binary vectors that have data about the pick-up time, pick-up location, and drop-off location.

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/JIOT.2016.2569090, IEEE Internet of Things Journal

Then, they use the vectors and the key sets to create indices, as explained in Section 3.1. The primary user uses the key set $\{S, M_1, M_2\}$ but the secondary user uses the key set $\{S, (M_1^{-1})^T, (M_2^{-1})^T\}$. The primary user's indices $\{I_P^{(p)}, I_P^{(d)}, I_P^{(t)}\}$ and the secondary user's indices $\{I_S^{(p)}, I_S^{(d)}, I_S^{(t)}\}$ are for the pick-up location, drop-off location, and pick-up time, respectively. The time index is created using the same way used to create the location indices, where each bit in the binary vector corresponds to one time slot. The users should submit to the server the three indices and a group signature.

For the pick-up vector, as indicated in Fig. 4, the primary user should put one in the element that corresponds to the trip's start cell. It should also put ones in the cells of the pick-up area (the gray cells in the figure). The primary user can pick up secondary users from anywhere in the pick-up area. In the simulations, when the pick-up area is greater than the trip's start cell, we call this case "with flexibility". Some users may want to pick up secondary users only from the trip's start cell. In this case the pick-up vector has only one element that has one corresponding to the trip's start cell. In the simulations, we call this case "without flexibility". The secondary users' vectors have one in only one element that corresponds to the pick-up location. Similarly, the primary user and each secondary user should create vectors for the drop-off locations.

As explained in Section 3.1, the server can measure the similarity of two indices by doing inner product operation. Since the secondary users' indices have only one element having one, the result of the inner product is one if the primary user's index has one in the same location, i.e., the same cell. Therefore, the server can know that the primary user can pick up the secondary user which is a necessary information to organize shared rides, but it cannot learn the pick-up area or cell.

To organize ride sharing in $DAP - DAD$ case, the server first measures the similarity of the time indices of the primary and secondary users. If the result is one, it proceeds to the next check, otherwise ride sharing is not possible and it checks other users. Then, the server measures the similarity of the pick-up location indices. If the result is one, it proceeds to the final check, otherwise ride sharing is not possible and it checks other users. Finally, the server measures the similarity of the drop-off location indices. If the result is one, ride sharing with the secondary user is possible and it sends the secondary user's signature to the primary user. $DAP - DAD$ ride sharing is infeasible if the servers searches all the users and does not find a user who can share ride.

4.3.2 Organizing DAP-ORD Shared Rides

Matching only the pick-up and drop-off areas in $DAP - DAD$ may be very restrictive. In order to increase the chance of finding a secondary user who can share ride, in $DAP - ORD$ case, the secondary user's drop-off location is in the surrounding area of the trip route.

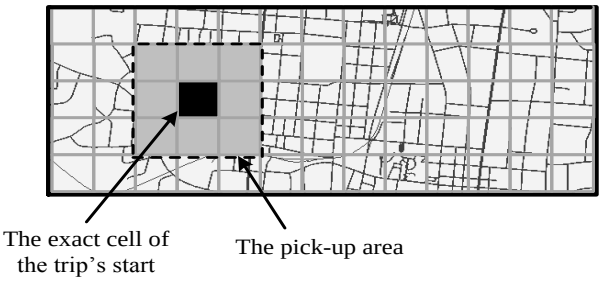


Fig. 4: Primary user's pick-up vector.

In this case, the primary user decides the pick-up area like $DAP - DAD$ case, but he can divert from its route to drop off the secondary user(s) and return back to its trip's route. The primary user should submit indices for the pick-up area, pick-up time, and route, i.e., $\{I_P^{(p)}, I_P^{(t)}, I_P^{(r)}\}$ and the secondary user should submit indices for the pick-up location, pick-up time, and drop-off location, i.e., $\{I_S^{(p)}, I_S^{(t)}, I_S^{(d)}\}$.

For the primary user's route index ($I_P^{(r)}$), as indicated in Fig. 5, the index's binary vector should have ones in the cells of the trip's route (black cells in the figure). Also, the primary user can determine the possible drop-off area in the vicinity of its route by putting ones in these areas (the gray cells in the figure). The drop-off area is totally determined by the primary user. It does not need to have the complete route, e.g., the primary user may not prefer to drop off users in areas close to the pick-up area. This is particularly useful when the AV is rented and the primary user wants to ensure that the secondary user will share a ceratin part of the route, and thus pay a ceratin share in the trip cost. Also, the primary user may not include some cells in the vicinity of the route because it does not want to drop off secondary users in slow-traffic areas like downtown and school zones.

To organize $DAP - ORD$ rides, the server should first match the time and pick-up location vectors of the primary and secondary users. If they are matched, the server should measure the similarity of the primary user's route index ($I_P^{(r)}$) and the secondary user's drop-off location index ($I_S^{(d)}$). The result should be one if the drop-off cell of the secondary user lies on the route of the primary user, otherwise it is zero.

4.3.3 Organizing DAP-EAD Shared Rides

In this case, the primary user decides the pick-up area like $DAP - DAD$ and $DAP - ORD$, but he can share a ride with a user whose drop-off location is not in the surrounding area of its route or end location. After the AV drops off the primary user, it drives itself to drop off the secondary user and then drives back to the primary user. In this case, the primary user should submit indices for the pick-up area, pick-up time, end location, and route, i.e., $\{I_P^{(p)}, I_P^{(t)}, I_P^{(d)}, I_P^{(r)}\}$ and the secondary user should submit indices for the pick-up location, pick up time, and route, i.e., $\{I_S^{(p)}, I_S^{(t)}, I_S^{(r)}\}$.

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/JIOT.2016.2569090, IEEE Internet of Things Journal

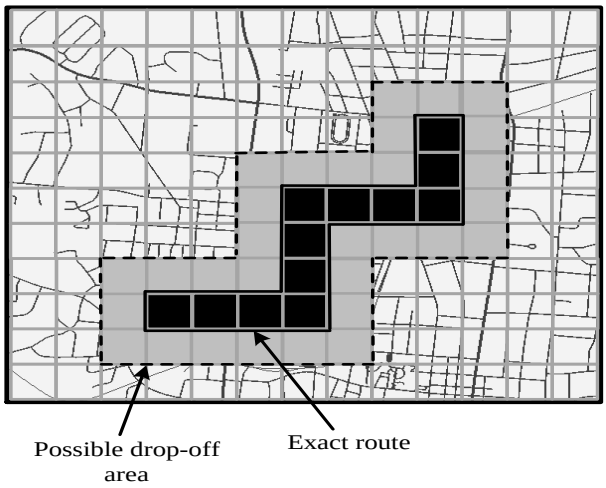


Fig. 5: Primary user's drop-off area in $DAP - ORD$.

The server should first match the pick-up and time indices of the primary and secondary users. If they are matched, it matches the primary user's drop-off location and the secondary user's route. If the primary user's drop-off location lies on the secondary user's route, then the users can share the ride.

In case of rented AV cab, the primary user does not need to put a limit on the distance the cab will drive after dropping off him because the secondary user will pay for this distance. In this case, the AV can drop off the primary user first and then delivers the secondary user. The AV cab does not need to drive back to the primary user but he can request a new cab if needed. In this case, the primary user does not need to submit its route index. However, if the primary user owns the AV, he may like to limit the distance the AV drives to drop off the secondary user. To do that, each secondary user should send its route length represented by the number of cells. The server should match the indices of the primary and secondary users' routes to obtain the number of shared cells between the two routes. The distance the AV will drive after dropping off the primary user can be estimated by subtracting the number of shared cells from the secondary user's route length. If this distance is below a threshold determined by the primary user, the ride can be shared with the secondary user.

4.3.4 Search Time Enhancements

In our scheme, location information is represented by cells and users have the same location if they share the same cell. On one hand, if the cell size is large, the distances between users can be large but the scheme considers them in the same location. On the other hand, if the cell size is small, the indices size increases because each cell is represented by one bit in the binary vector and small cells increase the the number of cells in the ride sharing area. This will definitely increase the search time because the number of inner products increases with increasing the indices size.

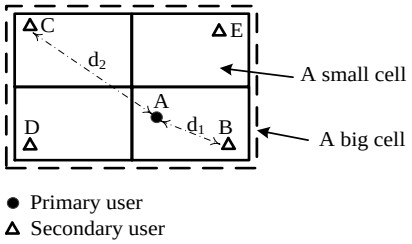


Fig. 6: Effect of cell size on precision.

To elaborate, Fig. 6 shows the positions of a primary user A and secondary users B, C, D, and E. If the cell size is large (the dashed square), it can include the positions of the five users. In this case, our scheme considers that the five users are in the same location. If all the secondary users can share the ride with the primary user, our scheme returns the secondary user it finds first in the search. Although, user B is a better choice because it is closer to the primary user, the search scheme can not figure out such information, and it may return user C who is far from A. In this case, the AV needs to drive more distance to pick up or drop off the secondary user. The figure can also show that this problem can be alleviated by making the cells smaller. For instance, if the big cell is split into four smaller cells (the solid squares), our scheme considers that users $\{C, D, E\}$ are not in the same location of the primary user A, and it returns the secondary user B.

In Section 5, we will measure the effect of the cell size on the precision. We will measure the distance between the closest secondary user to the primary user, d_1 in the figure, and the distance between the primary user and the secondary user selected by the scheme, e.g., d_2 if user C is selected. The scheme is more precise when d_2 is closer to d_1. However, reducing the cell size increases the search time, which is problematic in big data era when many rides should be organized in short time.

In order to obtain good precision with reduced search time, we propose an enhancement that is based on two levels of cell sizes. Fig. 7 shows that the ride sharing region is divided into two different cell sizes; one is fine grained (the solid gray squares) and the other is coarse grained (the dashed black squares). The users should represent their location data with two indices: fine-grained index and coarse-grained index. The server first matches the coarse-grained indices and then matches the fine-grained indices of the users whose coarse-grained indices are matched. In Fig. 7, the users $\{D, E, F\}$ can be excluded after efficient coarse-grained index matching operations. In this way, the high precision of the fine-grained indices can be obtained with less search time.

5 Evaluations

5.1 Security/Privacy Analysis

Trip data privacy: The server cannot infer the trip information, such as the trip's time, start/end locations, and

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/JIOT.2016.2569090, IEEE Internet of Things Journal

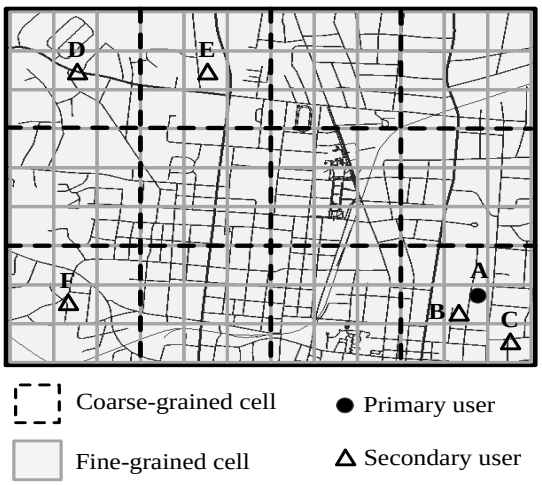


Fig. 7: An enhancement technique.

Fig. 8: The ride sharing region used in the experiment.

routes, of any user from the indices because it does not know the secret keys used to compute them.

Primary-secondary users unlinkability: If a pair of primary and secondary users share a ride, our scheme does not leak any information that can help the server to identify the same pair when they share rides in other occasions. This is because the use of one-time random numbers to compute the indices makes them look different even if they have the same data or are sent from the same user. For that reason, our scheme can achieve *requests-users unlinkability* and *requests unlinkability*. For the former, the server cannot link the ride sharing requests (and offers) sent from the same user at different occasions, and for the latter, the server cannot link the ride sharing requests (and offers) of the same trip at different times.

Identity anonymity and authentication: with using group signatures, each user can prove that he is a member in the group anonymously.

Access control: The way the group signature is used in our scheme can implicity achieve access control, where the server can only organize shared rides for users who are members in the primary user's group.

5.2 Performance Evaluations

5.2.1 Experiment Setup

In order to evaluate the performance of our scheme, we implemented it using visual C and a server with an Intel Core i7-4765T Processor at 2.0 GHZ and 8 GB RAM. First, we used the real map shown in Fig. 8, OpenStreetMap (OSM) project, and SUMO program [13] to create real routes for the users. The map is for the Cookeville city located in the Tennessee state in the USA. The size of the ride sharing area is 21Km $\times$ 13Km. For the coarse-grained cells, we divided the area to 273 cells, where the size of each cell is 1Km $\times$ 1Km. For the fine-grained cells, we divided the area to around 3,010 cells, where the size of each cell is 300m $\times$ 300m.

We assume that the primary user prefers *DAP-DAD* ride, but if it is not possible, the second preference is *DAP-ORD* and the last preference is *DAP-EAD*. First, the server matches the pick up indices of the primary and secondary users. If they are matched, it matches the drop-off indices. If they are matched, then *DAP-DAD* shared ride is possible and the search stops, otherwise it stores the secondary user's data in the list of users who will be searched for *DAP-ORD* and *DAP-EAD* rides, if *DAP-DAD* ride is not possible. After searching all the users, if the server cannot organize *DAP-DAD* ride, it starts searching for *DAP-ORD* ride. by checking whether a secondary user's drop-off location lies on the route of the primary user. If such a user is found, *DAP-ORD* ride can be organized and the search stops, otherwise, the server starts searching for *DAP-EAD* ride by checking whether the primary user's drop-off location lies on the route of a secondary user. If such a user is found, the search stops and *DAP-EAD* ride can be organized, otherwise it is impossible to organize a shared ride.

We are interested to measure the following metrics and the reported results are averaged over 30 different runs.

- *Search time* is the time taken by the server to search the secondary users' trip data to organize a trip.
- *Success probability* is the probability of finding a secondary user who can share a ride.
- *Failure probability* is the probability of failing to find a user who can share a ride.
- *Precision* is measured by comparing the distance between the primary user and the closest secondary user, and the distance between the primary user and the secondary user selected by the server.
- *Computation overhead* is the computation time required from all parties to run our scheme.
- *Communication overhead* is the amount of data that should be sent from the users to the server.

5.2.2 Experiment Results

For the similarity measurement technique, the indices' sizes are 6.02 K bytes and 546 bytes in cases of fine-grained and coarse-grained cells, respectively. The sizes of the secret keys are 149.3 KB and 18.2 MB in the cases of coarse-grained and fine-grained cells, assuming that

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/JIOT.2016.2569090, IEEE Internet of Things Journal

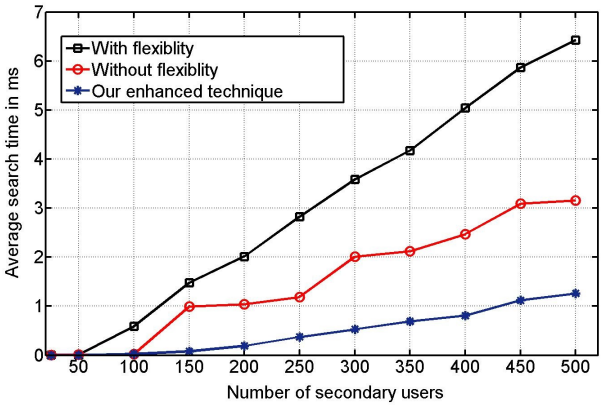


Fig. 9: search time versus n.

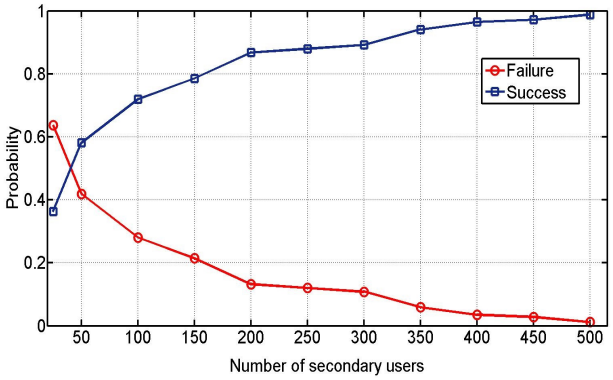


Fig. 10: The success and failure probabilities in flexibility case.

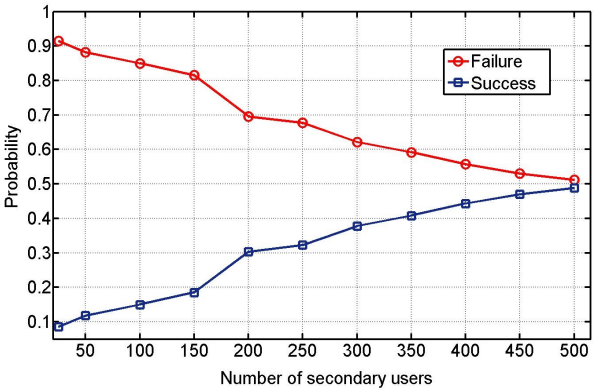


Fig. 11: The success and failure probabilities in the non-flexibility case.

each element in M_1 and M_2 takes one byte. Using our proposed group signature in [9], the signature size is 320 bytes, assuming the prime number q has 256 bits.

For the similarity measurement technique, the computation of the secret key takes 846.5 ms and 18.8 minutes in cases of coarse-grained and fined-grained cells, respectively. This is the time required to select random numbers for $\{S, M_1, M_2\}$ and calculate $(M_1^{-1})^T$ and $(M_2^{-1})^T$. Although this time is relatively long, the key computation is done only one time but used for a long time. The computation of an index takes 1.57 ms and 417 ms in cases of coarse-grained and fined-grained cells, respectively. For the group signature, using our measurements presented in [14], signature computation needs 31.6 ms and signature verification needs 51.8 ms.

Fig. 9 gives the search time in milliseconds (ms) versus the number of secondary users for three cases: with flexibility, without flexibility, and our enhanced technique. In flexibility case, the pick-up area of the primary user includes all the cells around the trip's start cell. We did the same for the primary user's drop-off area and routes. In non-flexibility case, the primary user's pick-up area is only the trip's start cell and the drop-off area is the trips's end cell. The figure shows that the search time increases with the increase of the number of secondary users because the server does more inner product operations. Overall, the search time is very short even at a large number of users, which is very desirable in AV and big data era when ride sharing becomes popular and many shared rides should be organized in a short time.

The search time in the flexibility case is more than that of the non-flexibility case because a larger number of secondary users matches the pick-up area of the primary user, and thus the server has to do more inner product operations to match the route and drop-off indices. Comparing to the flexibility case, the enhanced technique can reduce the search time. This is because the inner product operations of the coarse-grained indices take much less time than the those of the fine-grained indices, and the technique can filter some users which can reduce the number of fine-grained inner product operations.

Figs. 10 and 11 give the success and failure probabilities versus the number of users for flexibility and non-flexibility cases, respectively. The figures demonstrate that the increase of the secondary users' number increases the success probability and decreases the failure probability. The success rate is much higher in flexibility case because increasing the pick-up and drop-off areas increases the chance of finding secondary users who can share rides. Also, comparing the two figures to Fig. 9, it can be seen that although the non-flexibility case requires low search time, its failure probability is much higher.

We found that the precision for the optimum case that gives the average distance between the primary user and the closest secondary user is 49 meter, while the precision for the fine-grained and coarse-grained cases that give the average distances between the primary user and the secondary users selected by the scheme using fine-grained and coarse-grained cells are 145 and 510 meters. These results are for pick-up area and assuming no flexibility. The results demonstrate how our scheme can select closer users by reducing the cell size, but this necessitates more search time as indicated in Fig. 9. However, with using the enhanced technique, better precision can be achieved with short search time.

6 Related Works

Ride sharing has received extensive attention due to its importance [15]. Lam et. al. [16] studied a public transportation system for AVs offering point-to-point services with ride sharing. Due to the unmanned nature, AVs are operated by following the routes decided by

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/JIOT.2016.2569090, IEEE Internet of Things Journal

the control center of the system. The control center assigns the transportation requests to appropriate AVs to minimize the total operational cost.

Several schemes have been proposed to enable the cloud to search stored encrypted documents for certain keywords and outsource the documents that have the keywords without knowing the keywords [10], [17], [18]. In [10], Xia et. al. propose an efficient multi-keyword search scheme meeting a strict privacy requirements. The efficiency is improved by modifying the proposed scheme in [11] to use the product of three matrix pairs to create keyword indices, and using bloom filter to update the keyword indices. In [17], the authors present a secure multi-keyword ranked search scheme over encrypted cloud data, which can simultaneously support dynamic update operations like deletion and insertion of documents. Like these schemes, we use similarity measurement technique, but organizing ride sharing in AVs is a completely different application, e.g., the indices should have time/location data, and different cases of ride sharing should be considered.

7 Conclusions

In this paper, we have proposed a privacy-preserving ride sharing scheme for AVs. A similarity measurement technique over encrypted data is used to enable the server to organize rides without learning the trips' data. Different cases of ride sharing can be organized by our scheme with considering the users' preferences. Our analysis has demonstrated that our scheme can achieve desirable privacy features. We have implemented the proposed scheme using data extracted from a real map. The experiment measurements have demonstrated that the search time is short and it can be used in the AV and big data era when ride sharing is very popular and a large number of rides should be organized in a short time. Also, with using the enhanced technique, better precision can be achieved with short search time.

References

[1] J. Petit and S. E. Shladover, "Potential cyberattacks on automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 546–556, 2015.

[2] T. Lassa, *The Beginning of the End of Driving*, 2014[Online; accessed 21-January-2016]. [Online]. Available: http://www.motortrend.com/news/the-beginning-of-the-end-of-driving/

[3] N. Shchetko, "Laser eyes pose price hurdle for driverless cars," *The Wall Street Journal*, vol. 21, 2014.

[4] K. Rabieh, M. Mahmoud, A. Seraj, and J. Misic, "Efficient privacy-preserving chatting scheme with degree of interest verification for vehicular social networks," *Proc. of IEEE Global Communications Conference*, San Diego, USA, 2015.

[5] H. Li, Y. Yang, T. Luan, X. Liang, L. Zhou, and X. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. PP, no. 99, pp. 1–1, 2015.

[6] H. Li, X. Lin, H. Yang, X. Liang, R. Lu, and X. Shen, "Eppdr: An efficient privacy-preserving demand response scheme with adaptive key evolution in smart grid," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2053–2064, Aug 2014.

[7] M. Mahmoud and X. Shen, "Cloud-based scheme for protecting source location privacy against hotspot-locating attack in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems (IEEE TPDS)*, vol. 23, no. 10, pp. 1805–1818, 2012.

[8] J. Shao, X. Lin, R. Lu, and C. Zuo, "A threshold anonymous authentication protocol for vanets," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2015.

[9] X. Liang,Z. Cao, J. Shao, and H. Lin, "Short group signature without random oracles," *Proc. of 9th International Conference Information and Communications Security (ICICS), Zhengzhou, China*, pp. 69–82, Dec. 2007.

[10] C. Yang, W. Zhang, J. Xu, J. Xu, and N. Yu, "A fast privacy-preserving multi-keyword search scheme on cloud data," *Proc. of the International Conference on Cloud and Service Computing (CSC)*, pp. 104–110, Nov 2012.

[11] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, Jan 2014.

[12] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. of the ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 139–152.

[13] SUMO - Simulation of Urban MObility, 2015. [Online]. Available: http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/

[14] E. Oriero , K. Rabieh,, M. Mahmoud, M. Ismail, K. Akkaya, E. Serpedin, and K. Qaraqe, "Trust-based and privacy-preserving fine-grained data retrieval scheme for msns," April 2016.

[15] Y. Huang, F. Bastani, R. Jin, and X. S. Wang, "Large scale real-time ridesharing with service guarantee on road networks," *Proc. of VLDB Endow.*, vol. 7, no. 14, pp. 2017–2028, Oct. 2014.

[16] Y. C. X. Lam, AYS; Leung, "Autonomous vehicle public transportation system," *Proc. of the 3rd International Conference on Connected Vehicles and Expo (ICCVE 2014)*.

[17] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 2, pp. 340–352, Feb 2016.

[18] H. Li, D. Liu, Y. Dai, and T. H. Luan, "Engineering searchable encryption of mobile cloud networks: when qoe meets qop," *IEEE Wireless Communications*, vol. 22, no. 4, pp. 74–80, August 2015.