Privacy-Preserving Ride Sharing Organization Scheme for Autonomous Vehicles in Large Cities

Ahmed Sherif*, Ahmad Alsharif*, Jacob Moran*, Mohamed Mahmoud*
*Department of Electrical and Computer Engineering, Tennessee Tech. University, TN, USA 38505

***Abstract*—The autonomous vehicles will make ride sharing popular, and necessary. However, ride sharing organization requires the passengers to reveal sensitive information about their trips, which causes a serious privacy issue. In this paper, we propose a privacy-preserving ride sharing organization scheme using the kNN encryption scheme, Bloom filter, and group signature. Each user encrypts his trip's data and sends an encrypted ride-sharing request to a server that measures the similarity between users trips' to organize shared rides without revealing sensitive information. Comparing to our proposal in [1], this paper has three improvements. The proposed scheme is much more efficient because the trip data is much shorter. It is also more secure because each user has his own encryption key instead of using one shared key for all users. It can prevent linking the encryptions of the trip's data sent at different times because users frequently update their keys efficiently. Our privacy analysis demonstrates that the proposed scheme can preserve users' location privacy and trips' data privacy. Our experimental results on a real map demonstrate that the proposed scheme is much more efficient than the existing schemes, especially for large cities.**

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

I. Introduction

The automotive industry has made a great progress in bringing automation to vehicle driving [2]. Autonomous Vehicles (AVs) are equipped with various technologies to enable the vehicles to autonomously drive themselves without human involvement. For many people, AVs will not be an owned product but an on-demand service. Such a service will be efficient due to the elimination of human driver effort and the expected high cost of owning AVs. Ride-sharing reduces the cost of on-demand AV service by sharing the cost among different passengers. It also reduces vehicles number in streets by increasing vehicles' occupancy which facilitates traffic and reduces crashes. However, people need to reveal sensitive data about their trips to search for a ride-sharing partner.

Existing schemes for privacy preservation such as [3]–[6] can not be applied effectively in ride-sharing due to the unique problem and requirements. In [7], Ota et al. proposed a real-time, data-driven simulation framework that supports the efficient analysis of taxi ride sharing which serves unplanned trips. However, they did not consider privacy issues.

In this paper, we propose a privacy-preserving scheme to organize shared rides in large cities. In the proposed scheme, the city is divided into cells and each cell has a unique identifier. Each user should represent his trip's route using the cells' identifiers. A Bloom filter [8] is used to compactly store the data and also create a binary vector that is needed for the kNN similarity measurement technique over encrypted data [9]. Then the encrypted binary vectors and a group signature are sent to the server. The server can match the trip's data to organize shared rides without knowing any sensitive information. When the server finds the secondary user who can share the ride, it sends the user's signature to the primary user to trace the signature to the signer's identity. Our scheme considers different cases for ride sharing according to the users' constraints and requirements, such as the number of secondary users to share rides with the primary user and the drop-off location of the secondary user(s).

Comparing to our proposal in [1], which we call Complete City Representation-based Scheme (CCRS), the proposed scheme in this paper has several enhancements. CCRS is not scalable as all the city cells should be represented in the trip data, and thus as the city size increases as more communication and computation overheads are needed. In this scheme we represent only the routes' cells. Moreover, in CCRS all users share the same key. If an encryption of a user is intercepted, other users can decrypt it and know his location. In our scheme, each secondary user should use his own key to encrypt his trip's data. Finally, in CCRS, the server can link the same requests of a secondary user sent at different times. We prevent this linkability by updating the users' keys frequently in an efficient manner. Our analysis demonstrate that the proposed scheme is efficient and can organize shared rides without disclosing private information. We have implemented our scheme using Matlab and a real map. Our experimental results demonstrate that the communication and storage overheads are acceptable, and our scheme is much more efficient than CCRS, especially in case of large cities.

The remainder of the paper is organized as follows. Network and threat models are discussed in Section II. The proposed scheme is presented in Section III. The privacy analysis is given in Section IV. The performance evaluations are discussed in Section V. Finally, conclusions are drawn in Section VI.

II. Network and Threat Models

The considered network model has three main entities: primary user, secondary users, and trip organizing server. The primary user owns or rents an AV and he is motivated to share rides with secondary user(s) to reduce the cost of using the AV. The primary and secondary users are members in a group which can be for university students, neighborhood residents, etc. The primary user uses a group signature scheme such as [10] to compute and distribute the group signature's private keys to secondary users who use these keys to anonymously authenticate themselves to the server. Since the primary user is the group creator, he can trace the secondary users' signatures

978-1-5090-5935-5/17/$31.00 ©2017 IEEE

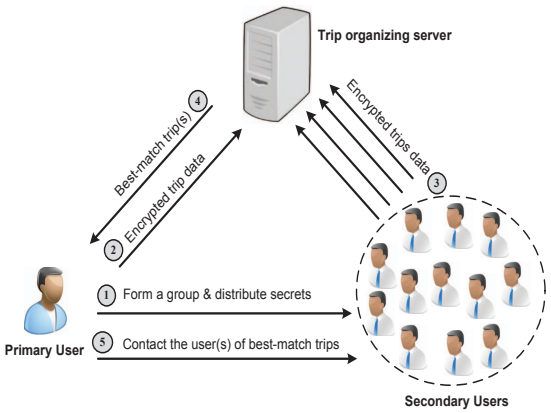

Fig. 1: The exchanged messages in our scheme [1].

to find their identities. Users use a smartphone application to report their trips' data to the trip organizing server which is a third party operated by a company known to all users. As shown in Fig. 1, each secondary user receives a unique secret key for kNN encryption from the primary user. The primary and secondary users should compute an encrypted request that has data about the trip's start time, start location (pick-up), end location (drop-off) and route. Then, they send the encrypted data to the server which matches the encrypted data to find the secondary user(s) who can share rides with the primary user without revealing any sensitive data.

For the threat model, the server and users are considered "honest-but-curious". They follow the scheme and do not act maliciously, but they are curious to learn private information about users' trips such as the pick-up/drop-off locations. Our scheme should achieve trip data privacy and users' anonymity. Also the server should not learn any information if it stores a lot of requests and offers and tries to link them to learn if same request is repeated, same primary/secondary pair shares same ride, or link different requests to the same user.

III. Proposed Scheme

In this section, we first explain the considered ride-sharing cases in our scheme and the Bloom filter concept. Then, we explain how the primary and secondary users compute their ride-sharing offers/requests. Finally, we illustrate the operations done by the server to organize shared rides.

A. Ride-Sharing Cases

We consider the same ride-sharing cases as in CCRS. As shown in Fig. 2, in all cases, the primary user chooses a pick-up area surrounding his trip's start location where he can pick-up secondary users. If the secondary user's start location lies outside the pick-up area, ride sharing is impossible. For the drop-off area, there are three possibilities as follows.

1) Matched Pick-up/Matched Drop-off (MP/MD)
   As shown in Fig. 2(a), the destination of the secondary user lies within the drop-off cell of the primary user.
2) Matched Pick-up/on-Route Drop-off (MP/RD)
   As shown in Fig. 2(b), the destination of the secondary user lies on the primary user's route. Hence, the AV drops the secondary user off, and then continues its main trip.

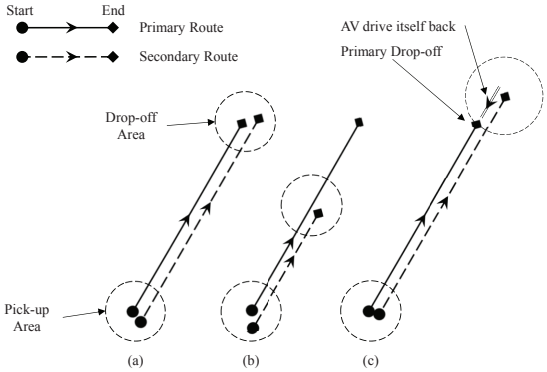

Fig. 2: Ride Sharing Cases.

3) Matched Pick-up/Extended Drop-off (MP/ED)
   As shown in Fig. 2(c), primary user destination lies on the secondary user's route. Therefore, the AV first drops the primary user off, then it continues to deliver the secondary user, and finally it drives itself back to the primary user.

B. Bloom Filter

Bloom filter is a space-efficient probabilistic data structure that reduces the overhead of searching a set of items [8]. A Bloom filter building algorithm takes N items and uses k-independent hash functions to compute a bit vector of length m that stores the items. The k hash functions are defined by $H_i : \{0,1\}^* \rightarrow n_i$ where $1 \leq i \leq k$ and $1 \leq n_i \leq m$. Initially, all the m bits are set to 0. To add an item C to the filter, C is hashed using the k hash functions and the bit locations in the filter corresponding to the hash values are set to 1. The algorithm must ensure that the k hash values are distinct. For example, If $H_1(C) = H_2(C)$, a collision resolution method such as linear probing is used to resolve this collision [11].

To check whether an item C belongs to the set, a query algorithm computes the k hash values of C to get k locations in the filter. If any bit at these locations is 0, C is definitely not in the set while if all the bits are 1s, C belongs to the set with high probability. It is possible that an item does not belong to the set and the corresponding bit locations are set to 1s by other items in the set which is called false positive. The false-positive probability of a Bloom filter is $(1-(1-1/m)^{kN})^k$ [8]. Given the k and N, the Bloom filter size m is chosen such that the false positive probability is less than 0.01.

C. Location and Trip Data Representation

1) Location Representation: As shown in Fig. 3, the ride-sharing area is divided into 80 cells where each cell has a unique identifier (ID). Each user should represent his pick-up, drop-off locations and route by a list of cell IDs. For example, the figure shows a user's route can be represented as $\{C7, C17, C27, C26, C25, C35, C45, C55\}$.

2) Primary User's Offer: The primary user creates a ride-sharing offer that has four encrypted vectors, called indices, which are pick-up index $I_P^{(p)}$, drop-off index $I_P^{(d)}$, trip route index $I_P^{(r)}$, and pick-up time index $I_P^{(t)}$.

For example, to generate $I_P^{(r)}$, route cells are passed to the Bloom filter building algorithm to generate a binary

Fig. 3: Dividing a ride sharing area into cells

vector (B_r) of size m. Then, B_r is encrypted using the kNN encryption scheme [9]. The key set of the primary user is $K_{pri} = [S, M_1N_1, M_1N_2, M_2N_3, M_2N_4]$, where S is a binary vector of size m, and $(M_1, M_2, N_1, N_2, N_3, N_4)$ are $m \times m$ invertible matrices. This encryption is done as follows. The primary user uses S as splitting indicator to split the Bloom filter (B_r) into two random binary row vectors p' and p'' of the same size as B_r. If the j^{th} bit of S is 0, $p'[j]$ and $p''[j]$ are set similar to $B_r[j]$, while if it is 1, $p'[j]$ and $p''[j]$ are set to two random numbers such that their summation is equal to $B_r[j]$. It should be noted that, every time the same vector is split, the result is different due to the randomness in creating p' and p''. After splitting B_r, the encrypted index $I_P^{(r)}$ can be computed as $I_P^{(r)} = (p'M_1N_1, p'M_1N_2, p''M_2N_3, p''M_2N_4)$ where $I_p^{(r)}$ is a row vector of the size $4m$. Using the same process, the indices $I_P^{(p)}$, $I_P^{(d)}$, can be computed by passing the pick-up and drop-off cells to the algorithm respectively.

To generate the time index, we use a binary vector where each bit represents a time slot of the day such that all the bits' values are 0s except the bit corresponding to the trip start time. Then, this binary vector is encrypted to obtain $I_P^{(t)}$. Beside the indices, the primary user informs the server with the maximum number of secondary users with whom he can share rides. Moreover, the primary user can select some of the preferred ride-sharing cases. For example, he might prefer only MP/MD so that the AV never stops during the trip, or he can select more than one ride-sharing case.

3) Secondary Users' Requests: Each user should compute a ride-sharing request that has the pick-up time index $I_S^{(t)}$, the pick-up cell index $I_S^{(p)}$, the drop-off cell index $I_S^{(d)}$, and the route index $I_S^{(r)}$. Each user has a unique key set $K_i = \left[S, N_1^{-1}M_i^{'}, N_2^{-1}M_i^{''}, N_3^{-1}M_i^{'''}, N_4^{-1}M_i^{''''}\right]$, where $M_i^{'}, M_i^{''}, M_i^{'''}, M_i^{''''}$ are random matrices such that $M_i^{'} + M_i^{''} = M_1^{-1}$ and $M_i^{'''} + M_i^{''''} = M_2^{-1}$.

For example, to compute the index $I_S^{(r)}$, the route's cells are passed to the Bloom filter building algorithm to generate a binary vector (Q). Then, S is used to split Q into two random binary column vectors q' and q'' of the same size as Q. If the j^{th} bit of S is 1, $q'[j]$ and $q''[j]$ are set similar to $Q[j]$, while if it is 0 $q'[j]$ and $q''[j]$ are set to two random numbers such that their summation is equal to $Q[j]$. Finally, the encrypted index $I_S^{(r)}$ can be computed as $I_S^{(r)} = (N_1^{-1}M_i^{'}q'; N_2^{-1}M_i^{''}q'; N_3^{-1}M_i^{'''}q''; N_4^{-1}M_i^{''''}q'')$ and $I_S^{(r)}$ is a column vector of size $4m$. Beside these indices, each secondary user should submit to the server a group signature.

D. Organizing Shared Rides

All secondary users' requests are stored in a list called active requests. When the server finds that a request can not match the primary user's offer, the request is deleted from the list for efficient searching. We also assume that the primary user requests to share rides with n secondary users and he prefers first MP/MD then MP/RD and finally MP/ED. When the server finds a secondary user who matches the primary user, the server appends his signature and the matched case into another list called matched users' list. Once the matched users' list's size becomes n, the server stops searching and forwards this list to the primary user. The following subsections explain the matching process for each ride-sharing case.

1) Organizing MP/MD shared rides: The server first measures the similarity between the time indices by computing a dot product operation between $I_P^{(t)}$ and $I_S^{(t)}$. Based on the kNN similarity measurement scheme [9], the dot product result between two indices gives the number of common 1's between the binary vectors used to create these indices. Since only one bit in the time vectors was set to 1, if the similarity result is 0, the primary and secondary users start their trips at different times and ride sharing is impossible. Therefore, the server removes this request from the active requests' list and proceeds to the next user. If the measured similarity is 1, the users start their trips at the same time and the server proceeds to the pick-up check by computing another dot product operation between $I_P^{(p)}$ and $I_S^{(p)}$. Since the secondary pick-up location index $I_S^{(p)}$ represents only one cell, its Bloom filter contains only k bits set to 1. Therefore, if the measured similarity between $I_P^{(p)}$ and $I_S^{(p)}$ is less than k, they have different trip start locations and ride sharing is impossible. The server removes this request from the active requests' list and proceeds to the next user. If the measured pick-up similarity is exactly k, their start locations are the same and the server proceeds to the drop-off check. In the drop-off check, the server must ensure that their destinations are the same. This can be done by measuring the similarity between $I_P^{(d)}$ and $I_S^{(d)}$. If the similarity is less than k, the MP/MD ride sharing case is not possible and the server proceeds to the next user while if the measured similarity is exactly k, then MP/MD ride sharing is possible. Consequently, the secondary user's signature and the matched ride-sharing case are added to the matched user's list and the server proceeds to the next user. If all users have been checked for MP/MD and the matched users' list size is less than n, the server proceeds to check the next ride-sharing case. It should be noted that the active users' list now contains only requests that matched the primary user in time and pick-up.

2) Organizing MP/RD shared rides: Since the active requests' list contains only requests that matched the primary user in time and pick-up, the server only needs to check if the secondary user's destination cell lies around the primary route or not by measuring the similarity between the primary route index $I_P^{(r)}$ and the secondary user's drop-off $I_S^{(d)}$. If the

similarity is less than k, the MP/RD ride sharing case is not possible and the server proceeds to the next user while if it is exactly k, then MP/RD ride sharing is possible and the secondary user's signature and the matched ride-sharing case are added to the matched user's list.

3) Organizing MP/ED shared rides: Similar to the MP/RD case, the server needs to perform only one check. In MP/ED case, the server checks whether primary user's destination lies on the secondary's route or not by measuring the similarity between the secondary user's route index $I_S^{(r)}$ and the primary user's destination cell $I_P^{(d)}$. If the similarity is less than k, MP/ED ride sharing case is not possible while if it is exactly k, then MP/ED ride sharing is possible and the secondary user's signature and the matched ride-sharing case are added to the matched user's list. Once the server finds the matched secondary user(s), it forwards the matched users' list to the primary user so that he can trace signers' identities.

IV. Privacy Analysis

Trip data privacy. Using the kNN similarity measurement technique, the server can compute the number of shared cells in the primary and secondary users' indices to organize shared rides without learning the trip information, such as the trip's time, and start/end locations because it does not have the primary key set that is necessary to decrypt the requests. Moreover, unlike CCRS, a user can not know the trip information of other users in the same group because in CCRS all the users share the same key while in our scheme each user has his own key and the decryption of users' indices can be done only by the primary key set.

Identity anonymity and anonymous authentication. by using group signatures, the server can learn that a received ride-sharing request is coming from a legitimate group member without learning the identity of the user.

Linkability Prevention. The encryption and request-signing processes include the following features. First, the one-time random numbers used to compute the group signatures will make all the signatures look different even if they are generated by the same user. Second, encrypted requests of the same trip look different because the random numbers used to encrypt the requests make them look different even if they have the same trip data. Third, the keys used for encryption are modified every short time interval. For example, the primary and secondary users can share a set of numbers such that the primary user changes a part from his key from M_1N_1 to $M_1(N_1/a)$ and all the secondary users modifies the corresponding part from $(N_1^{-1}M_i')$ to $(aN_1^{-1}M_i')$. Therefore, if the server tries to match new requests to old offers, they can not be matched. Modifying keys does not affect the kNN similarity measurement results of the vectors encrypted during the same interval while the matching technique does not give the correct result when matching an offer sent in an interval to a request sent in another interval because of using different keys. These features can prevent the server from doing the following linkability cases.

- *Requests-Users unlinkability.* Given different ride sharing requests sent from one user at different occasions, the server can not learn if these requests are sent from the same user or not. This is because the signatures associated with the requests look different.
- *Primary-secondary users pair unlinkability.* If a pair of primary and secondary users share a ride, the server can not identify the same pair when they share future rides.
- *Same request unlinkability.* The server cannot link the ride sharing offers/requests of the same trip at different times.

V. Performance Evaluations

A. Performance Metrics and Experiment Setup

Three performance metrics are used for comparison and assessment of our scheme and CCRS.

1) *Search time.* The time needed by the server to organize shared rides.
2) *Computation overhead.* The time needed by the primary/secondary users and the server to run our scheme.
3) *Communication overhead.* The amount of data transmitted during the communication between users and the server.

We implemented our scheme and CCRS using MATLAB and conducted experiments on a server with an Intel Core i7-4765T Processor @ 2.0 GHZ and 8.00 GB RAM. We used a real map for the city of Nashville, TN, USA by using OpenStreetMap project [12], and random routes for 450 users have been created by SUMO program [13]. The ride-sharing area is 75.5 km $\times$ 33 km, and was divided into 15,687 cells with a cell size of 400 m $\times$ 400 m. For the Bloom filter size, it was computed to be 2048 bits such that the false positive probability is less than 0.01.

B. Experiment Results

For the kNN technique, secret key generation takes 198.4 sec for CCRS and 2.7 sec per user for our scheme considering the city size of 15,687 cells. Note that all users in CCRS use the same shared key but in our scheme each user has a unique key. That means the key generation time in our scheme is less than that of CCRS when the number of secondary users is below 75. Although our scheme needs more time when the number of users is greater than 75, keys are generated one time and used for a long time. For encryption, the computation of an encrypted request takes 1.256 sec for CCRS and 0.145 sec for our scheme. Our scheme needs less time because in CCRS all the ride-sharing cells are represented in the encryption of the trip data while in our scheme only the route's cells are added to the Bloom filter to be encrypted and the size of the Bloom filter is much less than the vector size in CCRS.

Fig. 4 gives the search time versus the number of requests when the primary user wants to share rides with only one secondary user or with three secondary users. We also assumed that the primary user requirements for ride-sharing are MP/MD, MP/RD, and MP/ED in order. We run the experiment 30 times and average results are reported. The figure shows that the search time in our scheme is only 50% of the search time of CCRS for 50 requests. For 150 and 300 requests,

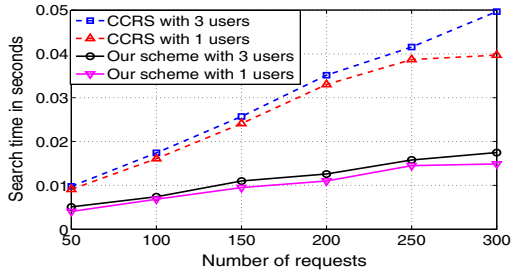

Fig. 4: Search time versus number of secondary users

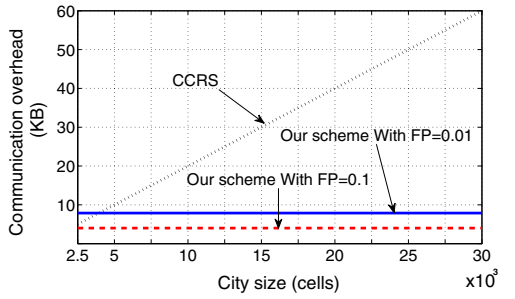

Fig. 5: Communication overhead versus city size

our scheme can reduce the search time by 60% and 65%, respectively. The reason for the reduction is that in CCRS all the 15,687 cells must be represented in the pick-up, drop-off and route vectors with 15,687 bits for each vector while in our scheme only the cells of pick-up/drop-off and routes are represented in Bloom filters with a small size, 2048 bits, compared to the total number of cells. Consequently, CCRS requires more inner product operations to match users' encryptions to organize rides. The figure also shows that the search time for CCRS increases at a rate higher than that of our scheme resulting in more search time improvement while increasing the secondary users' request. Furthermore, given the same number of secondary users and the same scheme, the search time to find three users is very close to that of finding one user. This is because when searching for matches in the first ride-sharing case, we also create a short list for requests that match the pick-up time and location of the primary user to be used for future search. Thus, we search only this short list to find users who match the remaining ride sharing cases.

Fig. 5 gives the communication overhead versus the city size for CCRS, our scheme with 0.1 and 0.01 false positive probabilities. As shown in the figure, for small cities (≈4000 cell/city), both schemes have close communication overhead. However, the communication overhead of CCRS increases linearly with the city size while our scheme maintains almost a fixed overhead. This is because in CCRS all the city cells are represented in binary vectors before encryption even if most of theses cells are not pick-up, drop-off or route cells whereas increasing the city size almost has no impact on the communication overhead of our scheme since only pick-up, drop-off, and route cells are injected to the Bloom filter. The figure also shows that less false positives rates require more overhead because the size of the Bloom filter is increased to obtain lower false positive probabilities. However, the communication overhead required in our scheme is still much less than that of CCRS. These results suggest that for small ride-sharing areas, both scheme can be used as the difference between the total number of cells and the Bloom filter size is very small which makes both schemes perform almost similar, but for large ride sharing areas, our scheme outperforms CCRS.

ACKNOWLEDGEMENT

This work is supported by US National Science Foundation under the grants numbers 1618549 and 1560434. The statements made herein are solely the responsibility of the authors.

VI. CONCLUSION

In this paper, we proposed a privacy-preserving ride-sharing organization scheme for AVs in large cities. Unlike CCRS scheme, the proposed scheme is much more efficient because the trip's data is represented in a shorter number of bits. It is also more secure as each secondary user uses his own key to encrypt his trip's data instead of using one key shared with all users. The proposed scheme can also prevent linking of the encryptions of the trip's data sent at different times because users frequently update their keys efficiently. Our privacy analysis demonstrated that the proposed scheme can achieve users trips' data privacy. Our experiments results demonstrated that our scheme much outperforms CCRS regarding the computation and communication overheads, e.g., it can reduce the search time by more than half compared to CCRS.

REFERENCES

[1] A. Sherif, K. Rabieh, 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, April 2017.

[2] 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.

[3] K. Rabieh, M. Mahmoud, and M. Younis, "Privacy-preserving route reporting scheme for traffic management in VANETs," in *IEEE International Conference on Communications (ICC)*, June 2015, pp. 7286–7291.

[4] M. Mahmoud, S. Taha, J. Misic, and X. Shen, "Lightweight privacy-preserving and secure communication protocol for hybrid Ad Hoc wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2077–2090, Aug 2014.

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

[6] S. Gunukula, A. Sherif, M. Pazos-Revilla, B. Ausbyy, M. Mahmoud, and X. Shen, "Efficient scheme for secure and privacy-preserving electric vehicle dynamic charging system," *Proceedings of 2017 IEEE International Conference on Communications (ICC)*, May 2017.

[7] M. Ota, H. Vo, C. Silva, and J. Freire, "A scalable approach for data-driven taxi ride-sharing simulation," *Proceedings of IEEE International Conference on Big Data (Big Data)*, pp. 888–897, October 2015.

[8] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[9] H. Li and D. Liu and 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.

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

[11] W. W. Peterson, "Addressing for random-access storage," *IBM Journal of Research and Development*, vol. 1, no. 2, pp. 130–146, April 1957.

[12] OpenStreetMap contributors, *OpenStreetMap*, 2017[accessed 06-May-2017]. [Online]. Available: https://www.openstreetmap.org

[13] *SUMO - Simulation of Urban MObility*, [accessed May 2017]. [Online]. Available: http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883