Efficient Privacy-Preserving Aggregation Scheme for Data Sets

Ahmed Sherif[§], Ahmad Alsharif[*], Mohamed Mahmoud[*], Mohamed Abdallah[**], Min Song[¶]
[§]Department of Electrical and Computer Engineering, Tennessee State University, TN, 37209.
[*]Department of Electrical and Computer Engineering, Tennessee Tech. University, TN, 38505.
[**] College of Science and Engineering, Hamad bin Khalifa University, Doha, Qatar.
[¶]Department of Electrical and Computer Engineering, Michigan Tech. University, MI, 49931.

***Abstract*—Many applications depend on privacy-preserving data aggregation schemes to preserve users' privacy. The main idea is that no entity should be able to access users' individual data to preserve privacy, but the aggregated data should be known for the application functionality. In these schemes, each user should encrypt a message and send it to an aggregator to compute and send the ciphertext of the aggregated messages to the decryptor without learning the individual messages. The decryptor should decrypt the ciphertext to obtain the aggregated message. However, the existing schemes are designed to aggregate one type/size of data and it is inefficient to modify them to aggregate messages that have data sets of different data types and sizes. In this paper, we propose an efficient privacy-preserving aggregation scheme for data sets. Unlike the existing schemes that do multi-bit number addition, the proposed scheme aggregates individual bits. Moreover, comparing to the existing schemes, our scheme has two new features. First, in some applications (such as those that need reporting location information), the aggregator can verify the encrypted messages to detect data pollution attacks without accessing the messages to preserve privacy. Second, our scheme has two types of decryptions; called full and partial. In full decryption, the decryptor can decrypt the whole data set, while in partial decryption, the decryptor can enable some entities to decrypt some data in the set. Our analysis demonstrates that the proposed scheme is secure and can preserve users' privacy. Extensive experimental results demonstrate that our scheme is more efficient than the existing schemes.**

I. Introduction

In privacy-preserving data aggregation schemes [1]–[3], users should encrypt their data and send the ciphertexts to an aggregator. The aggregator should be able to compute the ciphertext of the aggregated data of the users and send it to the decryptor without learning the individual data or even the aggregated data to preserve privacy. Finally, the decryptor can use its secret key to decrypt the aggregated data. The aggregation schemes have been extensively used in different applications to preserve user privacy [1]–[4]. An example to these applications is the collection of fine-grained power consumption readings in the smart grid advanced metering infrastructure (AMI) networks. In this application, the utility needs to access the aggregated reading of a number of smart meters for energy management. However, the individual readings should not be accessed because they can reveal sensitive information such as whether a consumer is at home and the appliances he/she uses.

The proposed schemes in the literature [1]–[3] are designed for doing multi-bit number addition and aggregating one type/size of data. However, several applications need collecting a set of data in one message, as will be discussed in Section IV. It is inefficient to modify the existing schemes to aggregate data sets [1]. Therfore, in this paper inspired by the k-Nearest Neighbor (kNN) encryption scheme [5], [6], we propose an efficient privacy-preserving aggregation scheme for data sets.

Unlike the existing schemes that do multi-bit number addition, the proposed scheme aggregates individual bits. After decrypting the aggregated data, the decryptor obtains the total number of ones in each bit. This aggregation method can efficiently aggregate messages that have data set of different types and sizes, and is more appropriate for some applications such as those that require location-based data aggregation [7]–[9], as will be explained later. Moreover, comparing to the existing schemes, our scheme has two new features. First, in some applications, the aggregator can verify the encrypted messages to detect data pollution attacks without accessing the messages to preserve privacy. Second, our scheme has two types of decryptions; called full and partial. In full decryption, the decryptor can decrypt the whole data set, while in partial decryption, the decryptor can enable some entities to decrypt some data in the set.

Our analysis demonstrates that the proposed scheme is secure and can preserve users' privacy. Extensive experimental results are given to evaluate the performance of the proposed aggregation scheme and compare it to the existing schemes. Specifically, the encryption/decryption times of our scheme are much lower than those of the homomorphic encryption based aggregation schemes.

The remainder of this paper is organized as follows. The network and threat models are discussed in section II. The proposed aggregation scheme is presented in section III. Applications that can use our aggregation scheme are discussed in section IV. The security analysis and performance evaluations are discussed in section V. Finally, conclusions are drawn in section VI.

II. Network and Threat Models

Network Model. As shown in Fig. 1, the considered network model has three main entities: decryptor, encryptors (users), and an aggregator. The decryptor is the entity that computes the encryption and decryption keys. The decryption key is known only to the decryptor, whereas the encryption keys are distributed to the users, where each user receives a unique key. Each user encrypts its data, which is in the form of a binary

978-1-5386-2321-3/18/$31.00 ©2018 IEEE

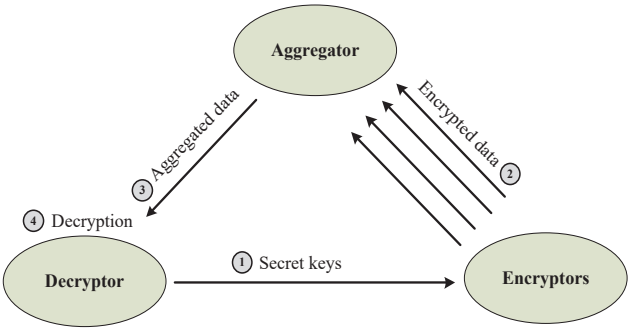


Fig. 1: The exchanged messages in our aggregation scheme.

vector, by using its encryption key, and sends the encrypted data to the aggregator. Using the encrypted users data, the aggregator can compute the ciphertext of the aggregated data and send it to the decryptor, without revealing the individual data of the encryptor or even the aggregated data. Finally, the decryptor uses its decryption key to decrypt this ciphertext to obtain the aggregated data.

Threat Model. The attackers can be the aggregator, encryptors (users), the decryptor, and external eavesdroppers. The attackers are considered "honest-but-curious". Specifically, they run the scheme honestly and do not aim to disrupt the proper operation of the scheme, but they are curious to learn the individual data reported by the users.

III. Proposed Data Aggregation Scheme

In this section, we explain the proposed privacy-preserving data set aggregation scheme in details.

A. Keys Generation and Distribution

First, the decryptor generates the decryption key set as $[S,\ M_1N_1,\ M_1N_2,\ M_2N_3,\ M_2N_4]$, where S is a binary vector of size n, $(M_1, M_2, N_1, N_2, N_3, N_4)$ are $n \times n$ invertible random matrices, and n is the number of bits in the data binary vector. Then, it generates a unique encryption key set for each user U_i as $[S,\ N_1^{-1}M_i^{'},\ N_2^{-1}M_i^{''},\ N_3^{-1}M_i^{'''},\ N_4^{-1}M_i^{''''}]$ such that $M_i{'} + M_i{''} = M_1^{-1}$, and $M_i{'''} + M_i^{''''} = M_2^{-1}$. Users have different encryption keys since the values of $M_i{'}$, $M_i{''}$, $M_i{'''}$ and $M_i^{''''}$ are different for each user, whereas they share only S. Finally, the encryption key sets should be sent to the users.

B. Encryption

First, each user U_i should use vector S to split his data vector q_i into two column vectors $q_i^{'}$ and $q_i^{''}$ of the same size as q_i. For each bit $S(j)$, if $S(j)$ is 1, $q_i'(j)$ and $q_i''(j)$ are set similar to $q_i(j)$, while if $S(j)$ is 0, $q_i'(j)$ and $q_i''(j)$ are set to two random numbers such that their summation is equal to $q_i(j)$, where $1 \leq j \leq n$. Then, U_i uses its encryption key set to encrypt the vector pair (q_i', q_i'') and generate the ciphertext I_i (called index) as

$$I_i = \begin{bmatrix} N_1^{-1}M_i^{'}q_i^{'} \\ N_2^{-1}M_i^{''}q_i^{'} \\ N_3^{-1}M_i^{'''}q_i^{''} \\ N_4^{-1}M_i^{''''}q_i^{''} \end{bmatrix} \tag{1}$$

Algorithm 1: Merging Algorithm

Procedure: MERGE
Input : $S, q_{agg}^{'}, q_{agg}^{''}$
Output : q_{agg}

```
for j = 1 to n do
    if S(j) == 1 then
        q_agg(j) ← q'_agg(j)
    else
        q_agg(j) ← q'_agg(j) + q''_agg(j)
return q_agg
```

where I_i is a column vector of size $4n$. Finally, U_i transmits I_i to the aggregator.

C. Aggregation

For m users, the aggregator receives m indices. The ciphertext of the aggregated data q_{agg}, i.e., the aggregated index I_{agg}, can be computed as follows.

$$I_{agg} = \sum_{i=1}^{m} I_i = \begin{bmatrix} N_1^{-1}\sum_{i=1}^{m} M_i^{'}q_i^{'} \\ N_2^{-1}\sum_{i=1}^{m} M_i^{''}q_i^{'} \\ N_3^{-1}\sum_{i=1}^{m} M_i^{'''}q_i^{''} \\ N_4^{-1}\sum_{i=1}^{m} M_i^{''''}q_i^{''} \end{bmatrix} \equiv \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix} \tag{2}$$

The value of an element $q_{agg}(j)$ is the addition of all the j^{th} bits in the original binary data vectors q_i, where, $1 \leq i \leq m$, i.e., $q_{agg}(j) = \sum_{i=1}^{m} q_i(j)\ \forall\ 1 \leq j \leq n$.

D. Decryption

In this section, we explain two types of decryption. In full decryption, all the elements of the aggregated vector can be recovered, whereas in partial decryption, only a number of elements in the aggregated vector can be recovered.

1) Full Decryption: Full decryption can be done only by the decryptor as follows. First, the decryptor uses its decryption key to recover $q_{agg}^{'} = \sum_{i=1}^{m} q_i^{'}$ as follows,

$$q_{agg}^{'} = M_1N_1\ A_1 + M_1N_2\ A_2 = \sum_{i=1}^{m} q_i^{'} \tag{3}$$

Then, with a similar process, the decryptor uses its key to recover $q_{agg}^{''} = \sum_{i=1}^{m} q_i^{''}$ as follows.

$$q_{agg}^{''} = M_2N_3\ A_3 + M_2N_3\ A_4 = \sum_{i=1}^{m} q_i^{''} \tag{4}$$

Finally, the decryptor can reconstruct q_{agg} by merging $q_i^{'}$ and $q_i^{''}$ using algorithm 1.

2) Partial Decryption: To enable an aggregator to decrypt the k^{th} element in the aggregated index ($q_{agg}(k)$) and the users' indices as well, the decryptor should use its keys to compute the secret D_k and send it to the aggregator. To compute D_k, the decryptor should first create a binary vector p where the k^{th} bit in p is set to 1 while all other bits are set to 0. Then, it splits p into two vectors $p^{'}$ and $p^{''}$ in an opposite way to subsection III-B. To elaborate, for each bit $S(j)$ if $S(j)$ is 0,

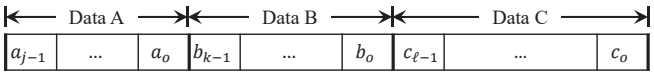


Fig. 2: Power consumption data set representation in a single binary vector.

then $p^{'}(j)$ and $p^{''}(j)$ are set similar to $p(j)$, while if $S(j)$ is 1, then $p^{'}(j)$ and $p^{''}(j)$ are set to two random numbers such that their summation is equal to $p(j)$, where $1 \leq j \leq n$. Then, the decryptor uses its key set to encrypt the vector pair $(p^{'}, p^{''})$ and generate row vector D_k of size $4n$ as follows.

$$D_k = \left[p^{'}M_1N_1 \quad p^{'}M_1N_2 \quad p^{''}M_2N_3 \quad p^{''}M_2N_4\right] \tag{5}$$

Using D_k, the aggregator can compute $q_{agg}(k)$ as follows.

$$q_{agg}(k) = D_k I_{agg} \tag{6}$$

The product pq_{agg} result in $q_{agg}(k)$ since all the bits in p are set to 0 except only $p(k)$ that is set to 1. Applying the same procedure on the users' indices, the aggregator can decrypt the k^{th} element in the users' vector.

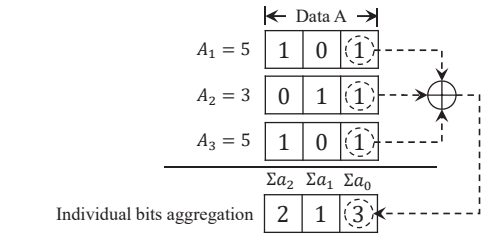
IV. Applications

In this section, we discuss how the proposed aggregation scheme can be used in different applications.

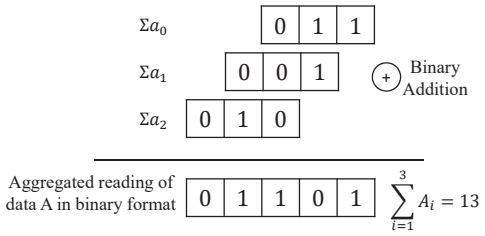
A. Privacy-preserving Collection of Fine-grained Power Consumption Readings

In smart grid AMI networks, smart meters should send fine-grained power consumption readings to the utility. For improved fine-grained control and grid monitoring, power consumption readings should be a set of data [1]. For example, power consumption data set can be constructed based on the consumption purposes such as air conditioning/heater, washer/dryer, and other appliances. Since these readings can reveal sensitive information about consumers, privacy-presreving aggregation schemes can be used to preserve consumers' privacy. In this subsection, we show how our aggregation scheme can be used for reporting the power consumption data set to the utility efficiently.

The AMI entities are mapped to our scheme entities as follows. The utility is the decryptor, smart meters are the encryptors and, a local collector is the aggregator. As shown in Figure 2, each smart meter should represent its power consumption data set in a binary vector. The figure shows that the power consumption data set consists of three data elements A, B, and C with sizes of j, k, and ℓ bits, respectively. Then, users should encrypt their data set using the procedure explained in subsection III-B and send the encrypted indices to the aggregator. The aggregator should aggregate the meters' indices as explained in subsection III-C and send aggregated index to the utility. The utility decrypts the aggregated index using the full decryption scheme explained in subsubsection III-D1 to obtain the aggregation of the individual bits. As an example, Figure 3(a) shows the individual bits aggregation results of one data element of three meters. The utility needs



(a) Individual bits aggregation for data A



(b) Calculating total reading for data A in binary format

Fig. 3: Recovery of aggregated reading for data element A.

to perform binary addition and shifting operations, as shown in Figure 3(b), to obtain the aggregated reading of a data element.

B. Secure and Privacy-Preserving Location Reporting

Our aggregation scheme can also be used in the applications that need reporting location information. An example of the applications that need reporting location information is the service management of the Autonomous Cab (AC). Each user will have an account in an AC company and will use a smartphone to request an AC when needed. To be able to provide high-quality service with acceptable operating costs, the AC companies need to learn the geographic distribution of the potential service requests, and the type of service that may be needed, e.g., regular van (RV), regular sedan (RS), handicapped accessible van (HV), and handicapped accessible Sedan (HS), etc

However, privacy concerns might make people reluctant to report their locations. Our aggregation scheme can be used to aggregate the location information of the users, so that the AC company can only learn the number of potential requests in each geographic area. In addition to privacy issues, some users may pollute the data by reporting multiple locations instead of only one location. Using our scheme, polluted data can be detected without leaking private information.

Our scheme can be used in this application, as follows. The AC company is the decryptor, the users are the encryptors, and the aggregator can be users who represent heads of clusters of users. The ACs service area is divided into cells where each cell is represented by one bit in a binary vector. As shown in Figure 4, the area is divided into 42 cells (6 rows and 7 columns). In addition to the location information, the data set has some bits for the type of service requested by the users.

When a user reports his/her location, he/she first composes a binary vector using the format given in Figure 4. In the location bits, only one bit is set to "1" which represents the location of the user and only one bit in the service type

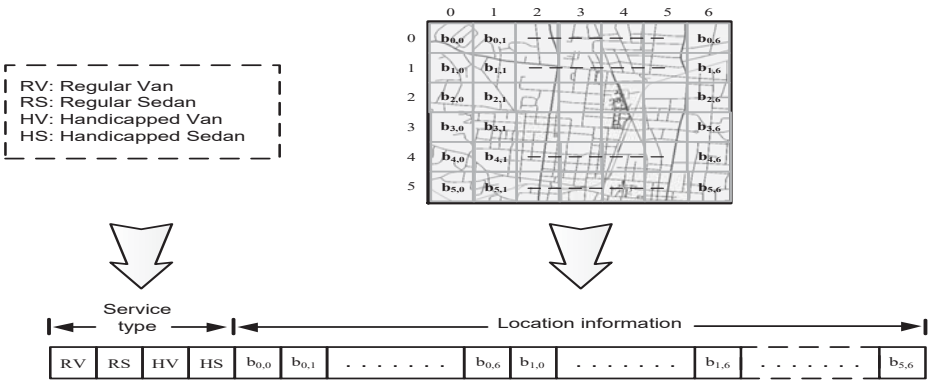


Fig. 4: Representing the city cells and services as binary vector.

is set to "1" to represent the requested service. Then, the user uses his/her secret keys to encrypt this vector using the procedure illustrated in subsection III-B and sends an index to the aggregator. The aggregator should first classify the requests based on the requested service and verify the requests against pollution attacks. Then, the aggregator should use the aggregation procedure discussed in subsection III-C to aggregate the indices that request the same service and send the aggregated index to the AC company. The company should use its secret keys to decrypt the aggregated index, using the procedure discussed in subsubsection III-D1, to learn the number of requests in each cell.

The AC company should provide the aggregators with secret indices to be able to classify the reports based on the requested service (i.e., to know partial information) and also to detect pollution attacks without knowing sensitive location information. Specifically, the AC company should send an index encrypted by its keys for each service to enable the aggregator to classify the requests. Each service is represented by one vector with all elements "0s" except the corresponding bit of this service. The aggregator should use the partial decryption procedure discussed in subsubsection III-D2 to decrypt the service bits to know the requested service without being able to know the sensitive location information. This is done by computing the dot product operation of each service index and the user index. The user requests the service of the used index if the result is "1". Moreover, to enable the aggregator to detect polluted reports without inferring any sensitive information about the locations of the users, the AC company should provide the aggregator with an index encrypted with its key for a binary vector that has ones in all the location bits. By using this index and applying the partial decryption procedure discussed in subsubsection III-D2 on the users' indices, the result is $p \cdot q_i$ which should equal to the number of ones in the location bits. If the result is more than one, this indicates that the user stores one in more than one location to pollute the data.

V. Evaluations

A. Security/Privacy Analysis

Data privacy. Our scheme can preserve the users' privacy as the users' data are encrypted by their keys and without knowing the keys it is infeasible to decrypt the indices. The aggregator can not use the indices reported by one user at different times to know any information about the user's data. This is because the users use different numbers each time they encrypt their binary vectors. Also, our scheme can preserve the users' privacy from the decryptor by reporting only the aggregated index to it.

The aggregator can not decrypt the indices of the users. The index of each user is encrypted by using the kNN encryption scheme, which is proven to be secure as long as the keys are unknown. As the aggregator does not have these secret keys, it can not decrypt the indices to know the users' data.

Each user can not compute other users' keys and the decryptor's key. As the key of each user has [S, ($N_1^{-1}M_i^{'}$, $N_2^{-1}M_i^{''}$), ($N_3^{-1}M_i^{'''}$, $N_4^{-1}M_i^{''''}$)], the attackers do not know the values of N_1, N_2, N_3, and N_4 to extract $M_i^{'}$ and $M_i^{''}$ to compute M_1. They also cannot extract $M_i^{'''}$ and $M_i^{''''}$ to compute M_2. As a result, the users can not compute the decryptor's key from their keys. In addition, as the decryptor uses different $M_i^{'}, M_i^{''}, M_i^{'''}$, and $M_i^{''''}$ for each user in creating its key, it is impossible to compute a user's keys.

Data pollution attack. The attacker may pollute the data by store "1" in more than one location. Our scheme can detect this attack without revealing the user's information. As explained earlier, the aggregator needs to measure the similarity of an index provided by the decryptor and each user's index.

B. Performance Evaluations

In this section, we evaluate the performance of our scheme under the afformentioned two applications;

1) Evaluation of power consumption data set aggregation: In this subsection, we compare our scheme to EPPA [1]. We implemented both schemes using Python and a server with an Intel® Xeon® Processor E5-2420 @2.2GHZ (2 processors) and 32 GB RAM. We used *Charm* cryptographic library [10] to implement the cryptographic operations used in EPPA. We assumed that the size of each element in the data set before encryption is one byte. All simulations were run 1000 times and average results are given in the following figures.

Figure 5 gives the encryption time needed by each meter to encrypt the reading data set versus the data set size. As shown in the figure, although both schemes increases linearly as the number data set elements increases, however EPPA exhibits higher rate. This because for each data element to

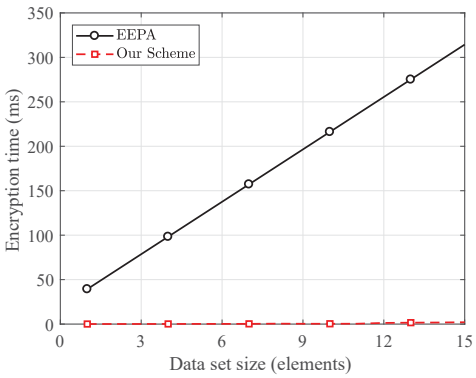


Fig. 5: Encryption time comparison.

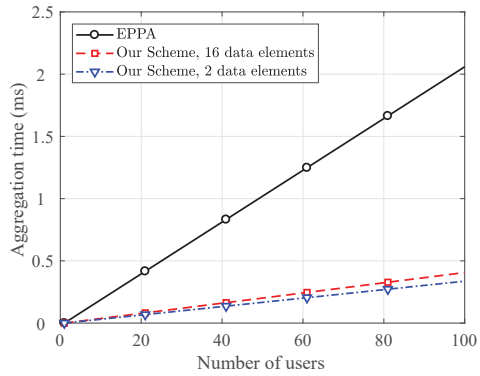


Fig. 6: Aggregation time comparison.

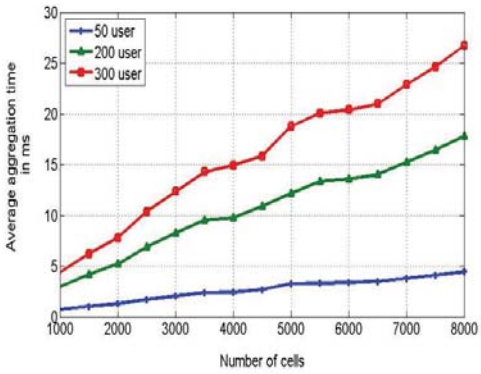


Fig. 7: Average Aggregation time.

be encrypted in EPPA, a modular exponentiation operation is required while in our scheme encryption is done efficiently by product operations as explained in subsection III-B

Figure 6 gives the time needed by the aggregator to aggregate ciphertexts versus the number of meters. As shown in the figure, EPPA requires more aggregation time than our scheme. This because in the aggregation in EPPA is done by multiplying Paillier ciphertexts while in our scheme aggregation is done by efficient addition operations as explained in subsection III-C. The figure also shows that, although in EPPA the aggregation time is independant on the data set size while it affects our scheme, our scheme is still more efficient than EPPA when the data set size increases from 2 to 16 elements.

2) Evaluation of our scheme used for autonomous cabs location-reporting: Figure 7 gives the average aggregation time for different numbers of cells and users. It can be seen that the aggregation time increases linearly as the number of cells increases because the indices sizes increases and thus more addition operations are needed. Also, more aggregation time is needed to aggregate indices for more users. However, the figure shows that, the time required to aggregate users' requests is very short.

VI. Conclusions

In this paper, we have proposed an efficient privacy-preserving aggregation scheme for data sets. Unlike the existing schemes that do multi-bit number addition, our scheme can efficiently aggregate data sets of different data types and sizes because it aggregates individual bits. Moreover, our scheme can enable the aggregators to verify the encrypted messages to detect data pollution attacks without decrypting the messages to preserve privacy. These new features are very useful specially in the applications that need reporting location information. Our analysis have demonstrated that the proposed scheme is secure, and the experimental results have demonstrated that the scheme can aggregate different types of data efficiently.

VII. Acknowledgment

This work is supported in part by US National Science Foundation under the grant number 1618549.

References

[1] R. Lu, X. Liang, X. Li, X. Lin, and X. Shen, "EPPA: An Efficient and Privacy-Preserving Aggregation Scheme for Secure Smart Grid Communications," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1621–1631, Sept 2012.

[2] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," in *Mobile and Ubiquitous Systems: Networking and Services, 2005. MobiQuitous 2005. The Second Annual International Conference on*. IEEE, 2005, pp. 109–117.

[3] M. A. Mustafa, N. Zhang, G. Kalogridis, and Z. Fan, "DEP2SA: A Decentralized Efficient Privacy-Preserving and Selective Aggregation Scheme in Advanced Metering Infrastructure," *IEEE Access*, pp. 2828–2846, 2015.

[4] K. Rabieh, M. Mahmoud, and M. Younis, "Privacy-preserving route reporting schemes for traffic management systems," *IEEE Transactions on Vehicular Technology (TVT)*, vol. 66, no. 3, pp. 2703–2713, 2017.

[5] 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, pp. 74–80, 2015.

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

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

[8] A. Sherif, A. Alsharif, J. Moran, and M. Mahmoud, "Privacy-preserving ride sharing organization scheme for autonomous vehicles in large cities," in *Proceedings of IEEE 86th Vehicular Technology Conference: VTC2017-Fall*. IEEE, 2017.

[9] A. Sherif, A. Alsharif, M. Mahmoud, and J. Moran, "Privacy-preserving autonomous cab service management scheme," in *Proceedings of 3rd Africa and Middle East Conference on Software Engineering (AMECSE)*. ACM, 2017.

[10] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.