IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON SECURITY AND PRIVACY IN EMERGING DECENTRALIZED COMMUNICATION ENVIRONMENTS

Received April 29, 2019, accepted May 13, 2019, date of publication May 16, 2019, date of current version June 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2917227

An Effective Encrypted Scheme Over Outsourcing Data for Query on Cloud Platform

JIANCHAO TANG[1], SHAOJING FU[1,2], AND MING XU[1], (Member, IEEE)
[1]College of Computer, National University of Defense Technology, Changsha 410073, China
[2]State Key Laboratory of Cryptology, Beijing 100878, China

Corresponding author: Jianchao Tang (tangjianchao14@nudt.edu.cn)

This work was supported in part by the National Nature Science Foundation of China (NSFC) under Grant 61572026 and Grant 61672195, and in part by the Open Foundation of State Key Laboratory of Cryptology under Grant MMKFKT201617.

ABSTRACT With the proliferation of cloud platforms, users are inclined to outsource their encrypted data to cloud platforms. However, the existing encryption schemes only provide users with limited query types over encrypted data. Meanwhile, if users encrypt their data by deterministic encryption, their encrypted data will be easily subject to the frequency attack. Besides, users' data privacy is likely to be disclosed to the cloud server when their encrypted data is updated on cloud platforms. To address these problems, in this paper, we propose an effective encrypted query scheme over outsourcing data on cloud platforms. In our scheme, users' data is encrypted based on all possible queries to meet users' diverse query demands. Furthermore, a double AES encryption method is adopted to cope with the frequency attack existing in deterministic encryption. To protect users' privacy when their data is updated, a neighbor rows exchange method is designed in our scheme. The theoretical analysis and comparative experiments demonstrate the effectiveness of our scheme.

INDEX TERMS Cloud platform, encrypted data, possible queries encryption, double AES encryption, neighbor rows exchange.

I. INTRODUCTION

In the last decade, rapid development of cloud technology including cloud storage and cloud computing has been made over the world. More and more users tend to outsource their data to cloud platforms [2], [9], [34], [37], [38]. There are two motivations behind this phenomenon: One is that cloud platforms have sufficient resources to store enormous users' data and provide further services (e.g., cloud computing), which can greatly decrease the storage and computing burdens on users' side. The other is that cloud platforms provide the accessing interfaces for users, which allow users to access their data conveniently. However, in practice, cloud servers are not always trustworthy [1], [22], [32], [36]. If users directly submit their data to cloud platforms, the sensitive information in their data will be exposed to cloud servers, which results in the leakage of users' privacy. In this case, users will be reluctant or reject to outsource their data to cloud platforms.

To avoid the above situation, users usually encrypt their data before outsourcing it to cloud platforms. Since users' secret keys are private, cloud servers are unable to decrypt users' encrypted data, which protects users' privacy. Common encryption techniques adopted by users include deterministic encryption [4], [5], order-preserving encryption [6], [7] and homomorphic encryption [11], [26]. Concretely, deterministic encryption has deterministic algorithm which can always generate the same ciphertext for the same message. Based on this algorithm, users can realize equality check on encrypted data. Order-preserving encryption is another useful encryption technique, where ciphertexts can preserve the size order of plaintexts. This means that users can realize more complex operations on encrypted data by order-preserving encryption (e.g., range query). Besides, homomorphic encryption allows users to aggregate their encrypted data on cloud platforms. In this encryption technique, the decrypted calculation results on ciphertexts are same as the working directly on the raw data. Although users' privacy is protected under these encryption techniques, another important challenge for users is how to effectively query their encrypted data.

Most previous works on encrypted data query focus on user privacy protection but ignore the query efficiency. Several works [3], [12], [15], [31], [33] consider the query efficiency

The associate editor coordinating the review of this manuscript and approving it for publication was Zheli Liu.

but only focus on single query type (e.g., range query). Unlike them, an early scheme was proposed in [16] to support various SQL queries over encrypted data by performing approximate filtering at the server and performing final query processing at the client. It is further extended to handle aggregation queries in [17], [18]. However, to realize the approximate filtering in the encrypted domain, this scheme needs to consume a mass of hardware resources. Then, a query-based encryption scheme is proposed in [29] and [35]. In this scheme, users can efficiently perform multiple types of query (e.g., equality checks, range query and aggregation). However, query-based encryption requires users to provide their query sets in advance and it fails to defeat the frequency attack in deterministic encryption [25]. To address the second problem, a system named Seabed was proposed in [27]. Seabed adopts additively symmetric homomorphic encryption (ASHE) to reduce the overhead of encrypted data aggregation and the splayed ASHE to cope with the frequency attack. However, the splayed ASHE introduces multiple new columns for original databases to result in heavy storage overheads. Besides, users' privacy will be disclosed if their data is updated in this system.

To overcome the deficiencies in the existing schemes, in this paper, we propose an effective encrypted query scheme over outsourcing data on cloud platforms. In our scheme, users' data is encrypted based on all possible queries rather than users' query sets. This method takes full advantage of data characteristics and is more reasonable than the query-based encryption. Meanwhile, a double AES encryption method is proposed in our scheme, which leverages AES and row identifies of data to encrypt raw data twice to cope with the frequency attack in deterministic encryption. The cost of this method is much lower than that of the splayed ASHE. To implement the dynamic update of users' encrypted data, we design a neighbor rows exchange method. In this method, the updated data in neighbor rows will exchange their storage locations. Due to the location changes of updated data, cloud servers cannot infer sensitive information from the process of data update, which protects users' privacy. In summary, our contributions in this paper are listed as follows.

- In this paper, we propose an effective encrypted query scheme on outsourcing data. In our scheme, the possible query encryption method provides users with all possible queries over encrypted data. Meanwhile, the double AES encryption method can defend against the frequency attack successfully. Besides, the neighbor rows exchange method can realize users' encrypted data update without disclosing users' privacy.
- We introduce detailed theoretical analysis to demonstrates the validity of the possible query encryption, the double AES encryption and the neighbor rows exchange method.
- The comparetive experiments in this paper indicate that our scheme has much better performance on users' data encryption cost and users' aggregation query cost compared with the existing schemes.

The remaining of this paper is organized as follows. Section 2 discusses the related work. Section 3 introduces the preliminaries of our scheme including system model, adversary model, query-based encryption and ASHE. Section 4 presents our scheme in detail. Section 5 provides the theoretical analysis for our scheme and section 6 shows the experimental results. Finally, section 7 concludes this paper.

II. RELATED WORK

Many works have addressed various security and privacy issues on data outsourcing (e.g., [13], [19]–[21], [23], [24], [28], [30]), but they do not consider the query efficiency. Meanwhile, there are a lot of existing works focus on encrypted data query, but they only provide a specific query for users and cannot meet user diverse query demands. Such as, range query over encrypted data is realized in [8] and [31]. Multi-keyword ranked search over encrypted cloud data is realized in [10].

A query scheme is proposed in [16], which can support various SQL queries over encrypted data. In this scheme, approximate filtering in the encrypted domain is performed at the server and the final query processing is performed at a trustworthy client. To support aggregation queries, this scheme is extended in [17] and [18]. However, approximate filtering in this scheme is at the level of rows, which results in the high bandwidth required to transfer intermediate results and heavy resources consumption to process the client-side query. In contrast, our scheme can provide more complete query operations over encrypted data at the server side (e.g., equality checks, sorting and aggregation), which can cut down the unnecessary consumption of bandwidth and other hardware resources (e.g., CPU).

A query-based query scheme over encrypted data is proposed in [29], which adopts specilized encryption techniques (e.g., deterministic encryption, order-preserving encryption and homomorphic encryption) to perform certain kinds of computations over encrypted data, such as equality, sorting, and aggregation. By this scheme, users can have rich queries on their encrypted data and avoid some unnecessary post-processes to their data. In [35], they further improve the query efficiency of this scheme by introducing several optimization techniques. However, query-based encryption requires users to provide their query sets in advance. In practice, this is difficult for users because they do not have clear plans for their data. Furthermore, this scheme fails to defeat the frequency attack in deterministic encryption [25]. In contrast, our scheme encryptes user data by possible query encryption method, which can provide all possible queries for users and allow users to change their query plans at any time. Meanwhile, the double AES in our scheme can effectively cope with the frequency attack.

A system named Seabed is implemented in [27], which adopts additively symmetric homomorphic encryption (ASHE) to implement data encryption and greatly reduce

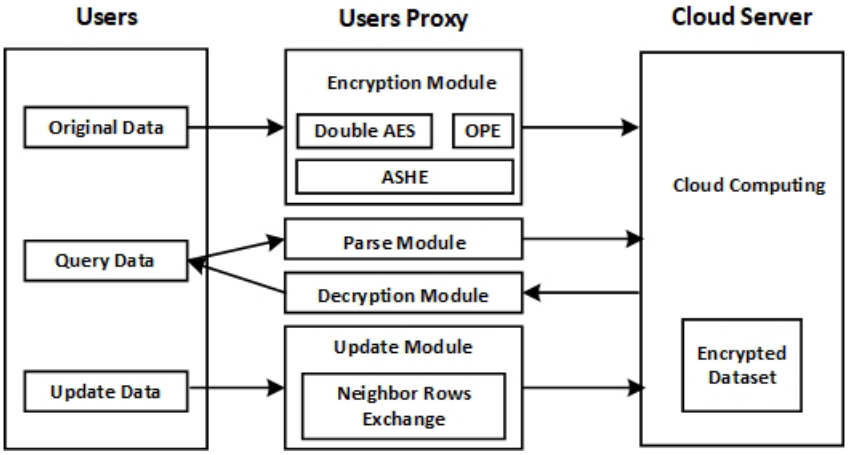

FIGURE 1. The framework of our scheme.

the overhead of data aggregation in the encrypted domain. Meanwhile, Seabed introduces a splayed ASHE to cope with the frequency attack by splaying sensitive columns to multiple columns. However, the splayed ASHE results in heavy storage overheads because multiple new columns are added in original databases. Besides, users' privacy will be leaked out if their data is updated in ASHE. In contrast, the double AES method used in our scheme can cope with frequency attack with much lower cost compared with the splayed ASHE. Besides, the neighbor rows exchange method designed in our scheme can successfully protect users privacy when their encypted data is updated.

III. PRELIMINARIES

A. SYSTEM MODEL AND ADVERSARY MODEL

Fig.1 shows our system model which invloves in three entities: users, the user proxy and the cloud server. Here, users are the owner of data and they generate their own secret keys for encryption. The user proxy is the middleman who is in charge of transmitting data between users and the cloud server. The cloud server is in charge of storing users' encrypted data. The workflow of our model is described as follows. First, users share their keys with the user proxy and submit their original data to the user proxy. Then, the user proxy encrypts user data according to the possible query encryption method and outsources the encrypted data to the cloud server. To query the encrypted data, users submit their original queries to the user proxy. Then, these queries are parsed and transformed into the queries used in the encrypted domain by the user proxy. Next, the user proxy submits the new version of queries to the cloud server. Finally, the cloud server returns the encrypted query results to the user proxy according to submitted queries. Since the user proxy has users' keys, he can decrypt these results and send them to users. Once users obtain the query results, the whole procedure is finished.

We assume that the cloud server in our model is semi-honest [14]. That is, he will strictly follow our scheme but he is also curious about each individual user's data. He may eavesdrop all the messages sent from or to every user. Meanwhile, the user proxy is assumed be trustworthy. This means that he will not expose users' data to the public or tamper the intermediate results. Besides, he will not collude with the cloud server to disclose users' privacy. For users, to protect their privacy, they will share the real keys with the user proxy.

B. QUERY-BASED ENCRYPTION

Query-based encryption in [29] and [35] aims to improve the efficiency of querying encrypted databases and it adopts multiple encryption techniques to achieve this goal, including randomization, deterministic encryption (DE), order-preserving encryption (OPE) and homomorphic encryption (HE). For randomization, two identical values are mapped to different ciphertexts. Thus, ciphertext operations are not allowed under this technique. For DE, two equal values are mapped to the same ciphertext and it allows the equality checks in the encrypted domain. In OPE, the size order of plaintexts is preserved after encryption and the range query is allowed. In HE, the ciphertexts are allowed to perform aggregation calculations and the decrypted results are still correct.

In query-based encryption, users first submit their data and query sets to the user proxy. Then, the user proxy selects the corresponding encryption techniques to encrypt users' data according to users' query sets. Finally, the user proxy uploads the encrypted data to the cloud server.

C. ASHE

ASHE in [27] assumes that there exists an additive group $\mathbb{Z}_n = \{0, 1, \cdots, n-2, n-1\}$ and a secret key k is shared between the encrypting entity and the decrypting entity. A message $m \in \mathbb{Z}_n$ is encrypted by ASHE as follow.

$$Enc_k(m, i) = ((m - F_k(i) + F_k(i-1)) \; mod \; n, \{i\}) \quad (1)$$

Here, i is an identifier from a set I. $F_k : I \rightarrow \mathbb{Z}_n$ is a pseudo-random function (PRF) that maps an identifier i in I to a value in $\mathbb{Z}_n$ and it is implemented by AES. For ease

of presentation, The ciphertexts in ASHE is also denoted as (c, S). Here, c is an element of $\mathbb{Z}_n$ and S is a multiset of identifiers. That is, the ciphertext $Enc_k(m, i)$ can be also denoted as $(m, \{i\})$. To create the additive homomorphism in ASHE, a special operation $\oplus$ is defined as follow.

$$(c_1, S_1) \oplus (c_2, S_2) = ((c_1 + c_2) \; mod \; n, S_1 \cup S_2) \quad (2)$$

That is, the elements are added togehter and the multisets of identifiers are combined in the operation $\oplus$. Besides, the ciphertext (c, S) is decrypted as follow.

$$Dec_k(c, S) = (c, \sum_{i \in S}(F_k(i-1) + F_k(i))) \; mod \; n \quad (3)$$

The additive result of two ciphertexts is decrpted by computing:

$$Dec_k(Enc_k(m_1, i_1) \oplus Enc_k(m_2, i_2)) = (m_1 + m_2) \; mod \; n \quad (4)$$

As shown in (1), the encryption function in ASHE is designed as $(m - F_k(i) + F_k(i-1))$ which has great advantages on data aggregation in the encrypted domain. For example, the ciphertexts of ASHE with consecutive identifiers $\{i, i+1, \cdots, n-1, n\}$ are added together. Due to the clever design of encryption function, the final result of these ciphertexts only contains $F_k(i) - F_k(n)$ and the other F_k is offset during the aggregation. Besides, $F_k(i)$ and $F_k(n)$ are easy to be worked out. Since the F_k is implemented by AES, the total computation overheads are low. Even if the identifiers of ciphertexts are consecutive partly, the overhead of data aggregation in ASHE is still much lower than that in the Paillier Homomorphic Encryption [26] adopted in [29] and [35].

IV. OUR SCHEME

In this section, we will introduce our scheme in detail, which includes three methods: possible query encryption, double AES encryption and neighbor rows exchange.

A. POSSIBLE QUERY ENCRYPTION

To meet users' diverse query demands for encrypted databases, the possible query encryption in our scheme adopts the same encryption techniques as the query-based encryption described in section 3.2. However, there are some obvious differences between them. First, DE is directly implemented by AES in the query-based encryption, which can not defend against the frequency attack. In contrary, in the possible query encryption, a double AES encryption is proposed to implement the DE. This encryption method can cope with the frequency attack effectively and will be discussed later. Second, in the possible query encryption, HE is implemented by ASHE rather than Paillier homomorphic encryption. From section 3.3, we can see that the ASHE is much more efficient than Paillier homomorphic encryption in terms of data aggregation in the encrypted domain. Last but not least, instead of users' query sets, all possible queries for users' data are taken into consideration in the possible query encryption. For one thing, users do not have clear understanding of their data so that they can not provide valid query sets. For another thing, users' query demands may change over time. This means that users' data should not be encrypted by a single encryption technique. Based on such considerations, the thought of all possible queries is adopted in the possible query encryption.

TABLE 1. The incomes of employees.

Gender	Age	Salary
male	31	7000
female	25	4800
female	37	10000
male	45	20000
···	···	···

Assume users' original data is presented as Table 1. The user proxy encrypts the data by columns according to the possible query encryption method. First, the user proxy picks out the column of gender and figures out the possible queries on it. Since the equality check is the only operation on this column, the user proxy adopts DE to encrypt it. Similarly, since the equality check and the range query are possible operations on the column of age, the user proxy encrypts it by DE and OPE. For the column of salary, data aggregation is the common operation on it. Therefore, apart from DE and OPE, HE is also used to encrypt it. Here, HE is implemented by ASHE. Besides, an identifier is introduced to each row of encrypted database due to the use of ASHE. By the possible query encryption, the encrypted version of users' data in Table 1 is shown as Table 2.

From the Table 2, we can find some extra columns are needed to store the encrypted data in possible queries encryption method. That is, this encryption method is likely to increase the storage burdens for users. But, the extra storage costs improves users' query efficiency by avoiding the post-process. Therefore, it is reasonable for users to improve their query efficiency at the sacrifice of their storage costs in possible queries encryption.

B. DOUBLE AES ENCRYPTION

For one thing, the frequency attack is a common attack mode in DE. Specifically, the attacker can obtain the occurrence frequency of plaintexts in advance. If these plaintexts are encrypted by the existing DE, then the attacker can infer the corresponding plaintexts according to the occurrence frequency of ciphertexts. This is because the same plaintexts have the same ciphertexts in DE. For another thing, the ASHE in the possible query encryption introduces an identifier for each row in the encrypted database. By these identifiers, the double AES encryption method in our scheme can defend against the frequency attack.

Double AES encryption method adopts two rounds of AES to encrypt users' data. During the first round of AES encryption, the user proxy encrypts users' data m by using the secret key k shared by users. The encrypted result $Enc_k(m)$ is calculated as follow.

$$Enc_k(m) = AES_k(m) \quad (5)$$

TABLE 2. The encrypted version of Table 1.

ID	DE(Gender)	DE(Age)	OPE(Age)	DE(Salary)	OPE(Salary)	ASHE(Salary)
1	DE(male)	DE(31)	OPE(31)	DE(7000)	OPE(7000)	ASHE(7000)
2	DE(female)	DE(25)	OPE(25)	DE(4800)	OPE(4800)	ASHE(4800)
3	DE(female)	DE(37)	OPE(37)	DE(10000)	OPE(10000)	ASHE(10000)
4	DE(male)	DE(45)	OPE(45)	DE(20000)	OPE(20000)	ASHE(20000)
...	...	...	...	...	...	...

Then, at the second round of AES encryption, the intermediate encrypted result $Enc_k(m)$ is viewed as the secret key of AES to encrypt the identifier i (i is the identifier of the row where m is). The final encrypted result $DE(m)$ is shown as follow.

$$DE(m) = AES_{Enc_k(m)}(i) = AES_{AES_k(m)}(i) \tag{6}$$

Since the identifier of each row is unique, the final encryption results of two identical data in different rows will be different by using double AES encryption. This means the double AES encryption in our scheme can defeat the frequency attack effectively. It is worth noting that we use $AES_k(m)$ to encrypt the identifier i rather than i to encrypt $AES_k(m)$. This is because the cloud server can directly access i and the final encryption result $DE(m)$. Since AES is a symmetric encryption technique, if i is the secret key to encrypt $AES_k(m)$, then the cloud server can directly decrypt $DE(m)$ and obtain $AES_k(m)$. In this case, the cloud server can still launch the frequency attack. In contrast, using $AES_k(m)$ as the key can avoid this because $AES_k(m)$ is unknown to the cloud server and AES is currently not vulnerable to known-plaintext attacks.

To support equality queries on encrypted databases implemented by double AES encryption, the user proxy should submit the intermediate encrypted result $Enc_k(m)$ to the cloud server. Then, the cloud server calculates the $DE(m)$ row by row according to (6) and performs the equality checks in the encrypted database. If the $DE(m)$ is equal to the data stored in the database, then the data meets users' query demands. Although the cloud server can know the counts of data being queried, he can not infer the corresponding plaintexts by the frequency attack because the occurrence frequency of other data is unknown to him under the double AES encryption. The double AES encryption version of Table 1 is shown as Table 3.

C. NEIGHBOR ROWS EXCHANGE

As mentioned before, in the possible query encryption, we adopt the ASHE to implement the HE. However, if users update their data encrypted by ASHE, their data privacy will be leaked out to the cloud server. Assume a user's original data is m_1. According to (1), it is encrypted by ASHE as follow.

$$Enc_k(m_1, i) = ((m_1 - F_k(i) + F_k(i-1)) \; mod \; n, \{i\}) \tag{7}$$

Then, the $Enc_k(m_1, i)$ is stored in the i-th row of database on the cloud server. Now, this user intends to change the m_1 to m_2. Then, m_2 is encrypted by ASHE as follow.

$$Enc_k(m_2, i) = ((m_2 - F_k(i) + F_k(i-1)) \; mod \; n, \{i\}) \tag{8}$$

The $Enc_k(m_2, i)$ is sent to the cloud server to update the content of the i-th row in database. However, the curious cloud server can disclose this user's data privacy by calculating:

$$\begin{aligned} \Delta m &= Enc_k(m_2) - Enc_k(m_1) \\ &= ((m_2 - F_k(i) + F_k(i-1)) \\ &\quad - (m_1 - F_k(i) + F_k(i-1))) \; mod \; n \\ &= m_2 - m_1 \end{aligned} \tag{9}$$

The Δm may indicate the changes in users' salaries or the personnel changes of a company. Anyway, the private information can be easily obtained by the cloud server, which results in the disclosure of users' privacy.

To address this problem, a neighbor rows exchange method is proposed in our scheme. Neighbor rows in this method are defined as two update rows which are adjacent to each other. Assume the data in the i-th, $(i+3)$-th, $(i+9)$-th and $(i+14)$-th row faces with update. Then, the i-th row and the $(i+3)$-th row are neighbor rows. Similar, the $(i+9)$-th row and the $(i+14)$-th row are also neighbor rows. In the case of multiple data updates at the same time, each pair of neighbor rows are divided into an exchange group and their updated value is stored in each other's locations. That is, for the data m_1 to be updated in i-th row and the m_2 to be updated in j-th row, assume i and j are neighbor rows. Then, in neighbor rows exchange method, m_1 and m_2 are divided into an exchange group. Meanwhile, the updated value of m_1 is stored in j-th row and the updated value of m_2 is stored in i-th row. Since the rows where users' data locates have changed after the update, the cloud server can not infer users' sensitive information anymore and users' privacy is protected.

In the case of a single data update, after receiving the i-th user's update data m, the user proxy stores this update data locally and does not modify the corresponding data on the cloud platform for the time being. Once some other users submit their update data, the user proxy will take out m from the local and combines it with other update data. The next steps will the same as those in multiple data updates. It is worth noting that the i-th user can query his update data at any time and the user proxy can ensure the correctness of query result. The above procedure can protect the i-th user's privacy.

In neighbor rows exchange method, the neighbor rows rather than two random rows exchange their stored data is

TABLE 3. The double AES encryption version of Table 1.

ID	Gender	Age	Salary
1	$AES_{AES(male)}(1)$	$AES_{AES(31)}(1)$	$AES_{AES(7000)}(1)$
2	$AES_{AES(female)}(2)$	$AES_{AES(25)}(2)$	$AES_{AES(4800)}(2)$
3	$AES_{AES(female)}(3)$	$AES_{AES(37)}(3)$	$AES_{AES(10000)}(3)$
4	$AES_{AES(male)}(4)$	$AES_{AES(45)}(4)$	$AES_{AES(20000)}(4)$
...	...	...	...

to minimize the location changes of update data, which can take full advantage of the homomorphic property in ASHE. In addition, the validity of neighbor rows exchange method will be demonstrated in the next section. The neighbor rows exchange method is summarized as Algorithm 1.

Algorithm 1 Neighbor Rows Exchange

Input: Users' update data sets $M = \{m_1, \cdots, m_i, \cdots, m_n\}$;
Output: The updated database on the cloud platform;
1: The user proxy confirms the number of update data n in M;
2: **if** $n > 1$ **then**
3: The user proxy devides the update data into multiple neighbor rows;
4: The user proxy encrypts the update data according to ASHE where the identifiers of their neighbor row are used;
5: The cloud server stores the encrypted data in neighbor row;
6: **else**
7: The user proxy stores the single update data m_o in the local;
8: **if** other update data is submitted from users **then**
9: m_o is combined with these update data;
10: repeat the step 3, 4 and 5;
11: **end if**
12: **end if**
13: **return** the updated version of dataset;

At the end of this section, we compare our scheme with other existing outsourcing data encryption schemes as shown in Table 4.

V. THEORETICAL ANALYSIS

In this section, we will present the theoretical analysis of our scheme. Specifically, we will respectively analyze the effectiveness of possible query encryption, double AES encryption and neighbor rows exchange.

Theorem 1: Possible query encryption in our scheme can provide users with richer queries on encrypted data.

Proof 1: In the possible query encryption, users' data is encrypted according to all possible queries on their data. Each type of data has its own characteristics: some are suitable for equality checks but range queries and data aggregations are meaningless to them (e.g., gender). DE is enough for this type of data. Some are not only suitable for equality checks but also for range queries and even data aggregations (e.g., salary). This type of data should be encrypted by DE, OPE and HE simultaneously. Considering these cases, possible query encryption encrypts users' data according to the characteristics of the data. This can avoid a lot of unnecessary encryption for users' data.

Compared with query-based encryption, possible query encryption fully excavates the potential characteristics of users' data and provides a more complete query view for users. For one thing, this method does not depend on users' query sets, which can avoid users' subjective limitations. For another thing, users can change their query plans at any time. In this case, their queries will not become invalid. Therefore, possible query encryption is more reasonable than the query-based encryption.

Theorem 2: Double AES encryption in our scheme can defend against the frequency attack.

Proof 2: In the double AES encryption, users' data has a unique identifier and the different identifiers ensure that the equal data has different ciphertexts. Assume m_1 is equal to m_2 and their identifiers are i and j (Here, $i \neq j$). According to (5) and (6), m_1 and m_2 is encrypted by double AES encryption as follow.

$$DE(m_1) = AES_{Enc_k(m_1)}(i) = AES_{AES_k(m_1)}(i) \quad (10)$$

$$DE(m_2) = AES_{Enc_k(m_2)}(j) = AES_{AES_k(m_2)}(j) \quad (11)$$

Since m_1 is equal to m_2, then $AES_k(m_1)$ is equal to $AES_k(m_2)$. But i is not equal to j, then $AES_{AES_k(m_1)}(i)$ is not equal to $AES_{AES_k(m_2)}(j)$. That is, $DE(m_1)$ is not equal to $DE(m_2)$.

From the above discussion, we can find two equal data are mapped to different ciphertexts by double AES encryption. This can prevent the attacker from inferring the occurrence frequency of plaintexts from the occurrence frequency of ciphertexts. Therefore, double AES encryption can defend against the frequency attack effectively.

Theorem 3: Neighbor rows exchange method in our scheme can protect users' privacy when their data is updated.

Proof 3: In the neighbor rows exchange method, in the case of multiple data update, the update data of neighbor rows exchanges their storage locations. Suppose m_1 in the i-th row and m_2 in the j-th row are facing updates and their update value are m_1' and m_2' respectively. Meanwhile, the i-th row and the j-th row are the neighbor rows. According to (1), m_1 and m_2 are encrypted by ASHE as follow.

$$Enc_k(m_1, i) = ((m_1 - F_k(i) + F_k(i-1)) \bmod n, \{i\}) \quad (12)$$

$$Enc_k(m_2, j) = ((m_2 - F_k(j) + F_k(j-1)) \bmod n, \{j\}) \quad (13)$$

TABLE 4. Different outsourcing data encryption schemes.

Scheme	Defeat Frequence Attack	Update	Special
our scheme	√	√	possible query encryption
scheme in [29]	×	√	query-based encryption
scheme in [35]	×	√	query-based encryption
scheme in [27]	√	×	ASHE

According to the neighbor rows exchange method, m'_1 is encrypted by ASHE with the identifier j and m'_2 is encrypted by ASHE with the identifier i:

$$Enc_k(m'_1, j) = ((m'_1 - F_k(j) + F_k(j-1)) \; mod \; n, \{j\}) \quad (14)$$

$$Enc_k(m'_2, i) = ((m'_2 - F_k(i) + F_k(i-1)) \; mod \; n, \{i\}) \quad (15)$$

To disclose users' privacy, the cloud server will try to obtain Δm_1 by calculating:

$$\begin{aligned} \Delta m_1 &= Enc_k(m'_1) - Enc_k(m_1) \\ &= ((m'_1 - F_k(j) + F_k(j-1)) \\ &\quad - (m_1 - F_k(i) + F_k(i-1))) \; mod \; n \\ &= m'_1 - m_1 + (F_k(i) + F_k(j-1) \\ &\quad - F_k(j) - F_k(i-1)) \end{aligned} \quad (16)$$

Since the cloud server does not know the key k, he can not work out the $F_k(i)+F_k(j-1)-F_k(j)-F_k(i-1)$ and obtain the Δm_1 according to the (16). Similar, the cloud server can not obtain Δm_2 either. In the case of a single data update, the only update data m_o is stored in the user proxy temporarily. At this stage, the cloud server can not disclose users' privacy because the encrypted database on the cloud platform has no change. When other update data is submitted, m_o is combined with them and they are updated by following the method of multiple update data. Therefore, at this stage, users' privacy is also protected. In summary, the neighbor rows exchange method in our scheme can protect users' privacy when their data is updated.

VI. EXPERIMENT

A. EXPERIMENT CONFIGURE

In this section, we will run some simulated experiments to evaluate the performance of our scheme. The hardware used in our experiments is a laptop with Intel i5-5200U CPU @ 2.20GHz and 4GB RAM. Meanwhile, the operating system is Windows 10 and the programming language is Java 1.8.0. To implement OPE and Pallier Homomorphic encryption, the opetoolbox [1] and the pailliertoolbox [2] are used in our experiments. Meanwhile, AES and AHSE are also implemented in our experiments. In addition, the experimental data is synthetic which includes 21 users. Each user has gender data, age data and salary data, as shown in Table 1. We will evaluate the performance of our scheme from two aspects: the encryption cost of the user proxy and the aggregation query cost of users.

[1] https://github.com/ssavvides/jope
[2] http://cs.utdallas.edu/dspl/cgi-bin/pailliertoolbox

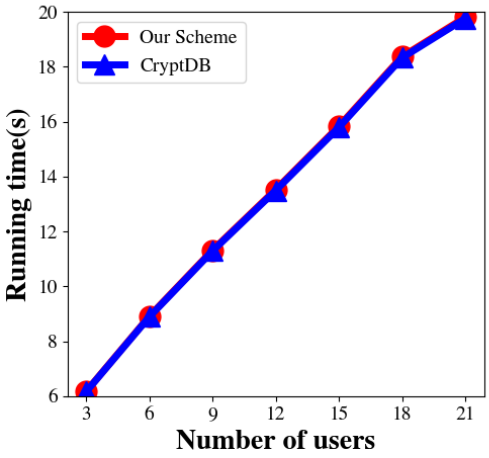

FIGURE 2. The user proxy's encryption cost.

B. THE ENCRYPTION COST OF THE USER PROXY

In this experiment, we will compare the encryption cost of the user proxy in our scheme with that in CryptDB [29] which is a mature encrypted query scheme and has been implemeted by Google. Concretely, in our scheme, the user proxy needs to encrypt users' data by double AES encryption, OPE and AHSE, as shown in Table 2. While In CryptDB, the user proxy adopts DE, OPE and Pallier Homomorphic encryption to encrypt users' data. Meanwhile, in this experiment, we assume users' query sets in CryptDB includes all possible queries. To observe the encryption cost of the user proxy, we measure the encryption time of the user proxy under the different number of users which varies from 3 to 21. Repeat 10 times for each experiment and calculate the averages. The experimental result is shown as Fig. 2.

From Fig. 2, we can find the user proxy in the two schemes has similar encryption cost. Through our analysis, we find OPE is most time-consuming in all of the encryption techniques mentioned in this paper. OPE in our scheme has the same implementation as OPE in CryptDB, which results in the similar encryption cost of the user proxy in the two schemes. To further compare the DE cost and HE cost of the user proxy in the two schemes, we use double AES encryption, AES, ASHE and Paillier homomorphic encryption to encrypt users' salary data. Similarly, in these experiments, the number of users is varied from 3 to 21. Each experiment is repeated 10 times and the averages are calculated. The experimental results are shown as Fig. 3.

From Fig. 3(a), we can find the deterministic encryption cost of the user proxy in our scheme is higher than that in CryptDB. This is because double AES encryption in our scheme is implemented by two rounds of AES while

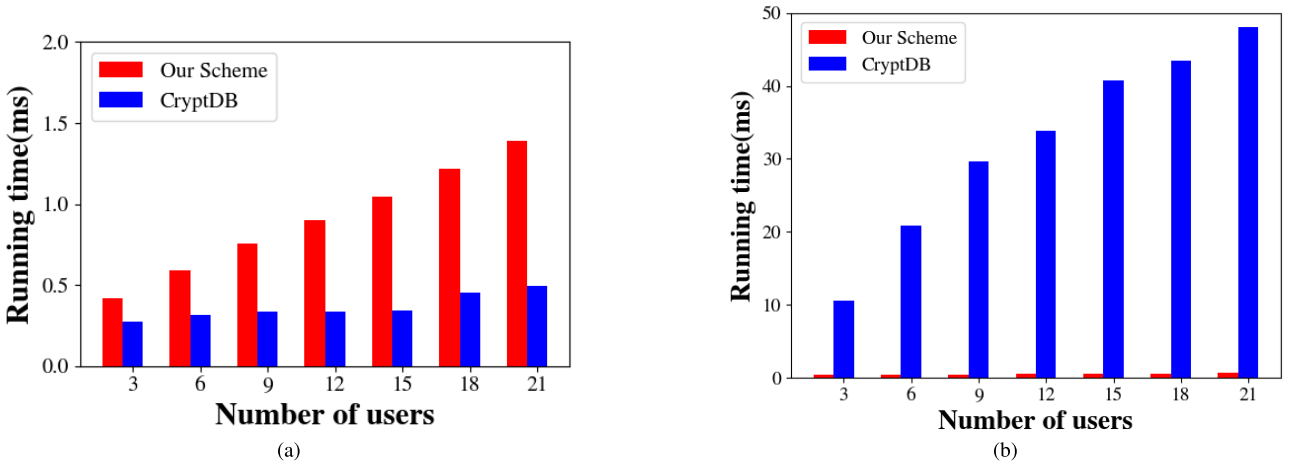

FIGURE 3. The encryption cost of the user proxy: (a) Deterministic Encryption, (b) Homomorphic Encryption.

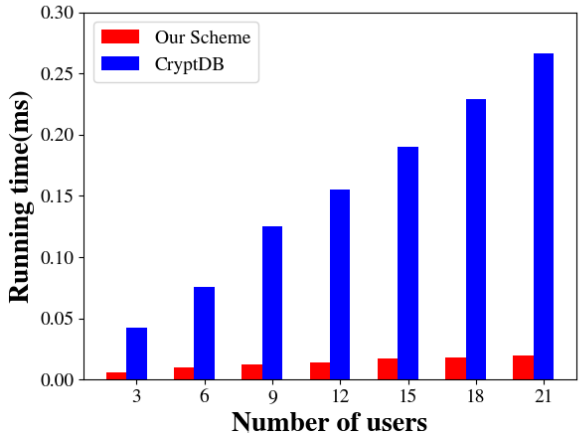

FIGURE 4. The users' aggregation query cost.

the deterministic encryption in CryptDB is implemented by one round of AES. To defend against the frequency attack, the extra encryption cost for the user proxy in our scheme is acceptable. From Fig. 3(b), we can find the homomorphic encryption cost of the user proxy in our scheme is much lower than that in CryptDB. This is because ASHE used in our scheme is implemented by the symmetric encryption AES. Compared with the asymmetric encryption (i.e., Paillier homomorphic encryption) used in CryptDB, ASHE is obviously much more efficient.

C. USERS' AGGREGATION QUERY COST

In this experiment, we assume users intend to query the sum of their salaries. This is a typical aggregation query in the encrypted domain, which is supported by our scheme and CryptDB. To compare the users' aggregation query cost in the two schemes, we measure users' query time under the different number of users which varies from 3 to 21. Each experiment is repeated 10 times and the averages are calculated. The experimental result is shown as Fig. 4.

From Fig. 4, we can find that users' aggregation query cost in our scheme is much lower than that in CryptDB. This is because users' data is encrypted by ASHE in our scheme. When aggregating the encrypted data, many calculation items are automatically cancelled out in our scheme, as discussed in section 3.3. In contrast, in CryptDB, since users' data is encrypted by Paillier homomorphic encryption, many exponent operations are executed when aggregating the encrypted data, which results in huge time consumption. Therefore, users' aggregation query in our scheme is much more efficient than that in CryptDB.

VII. CONCLUSION

In this paper, we propose an effective scheme to support for query over encrypted databases on cloud platforms. In our scheme, the possible query encryption provides a complete query view for users, which guarantees the efficiency of users' queries. Meanwhile, the double AES encryption can defend against the frequency attack in deterministic encryption successfully. Besides, the neighbor rows exchange method can protect users' privacy when their data is updated on encrypted databases. The theoretical analysis in this paper demonstrates the effectiveness of the three methods in our scheme. Meanwhile, The comparative experiments show that our scheme has good performance on the encryption cost of the user proxy and users' aggregation query cost.

REFERENCES

[1] M. Almorsy, J. Grundy, and I. Müller, "An analysis of the cloud computing security problem," 2016, *arXiv:1609.01107*. [Online]. Available: https://arxiv.org/abs/1609.01107

[2] P. K. Baldwin, "Cloud services," in *Encyclopedia Big Data*. Berlin, Germany: Springer, 2017, pp. 1–4.

[3] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Proc. Int. Conf. Inf. Secur. Pract. Exper.* Berlin, Germany: Springer, 2008, pp. 71–85.

[4] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2007, pp. 535–552.

[5] M. Bellare, M. Fischlin, A. O'Neill, and T. Ristenpart, "Deterministic encryption: Definitional equivalences and constructions without random oracles," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, 2008, pp. 360–378.

[6] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, 2009, pp. 224–241.
[7] A. Boldyreva, N. Chenette, and A. O'Neill, "Order-preserving encryption revisited: Improved security analysis and alternative solutions," in *Proc. Annu. Cryptol. Conf.* Berlin, Germany: Springer, 2011, pp. 578–595.
[8] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory Cryptography Conf.* Berlin, Germany: Springer, 2007, pp. 535–554.
[9] B. Calder *et al.*, "Windows azure storage: A highly available cloud storage service with strong consistency," in *Proc. 23rd ACM Symp. Operating Syst. Principles*, Oct. 2011, pp. 143–157.
[10] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.
[11] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. IT-31, no. 4, pp. 469–472, Jul. 1985.
[12] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *Proc. IEEE 30th Int. Conf. Data Eng. (ICDE)*, Mar./Apr. 2014, pp. 664–675.
[13] A. Friedman and A. Schuster, "Data mining with differential privacy," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2010, pp. 493–502.
[14] O. Goldreich, "Foundations of cryptography: Basic applications," *J. Acm*, vol. 10, no. 509, pp. 359–364, 2004.
[15] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, Oct. 2006, pp. 89–98.
[16] H. Hacigümüs H, B. Iyer, C. Li, and S. Mehrotra, "Executing sql over encrypted data in the database-service-provider model," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, Jun. 2002, pp. 216–227.
[17] H. Hacigümüş, B. Iyer, and S. Mehrotra, "Efficient execution of aggregation queries over encrypted relational databases," in *Proc. Int. Conf. Database Syst. Adv. Appl.* Berlin, Germany: Springer, 2004, pp. 125–136.
[18] H. Hacigümüş, B. Iyer, and S. Mehrotra, "Query optimization in encrypted database systems," in *Proc. Int. Conf. Database Syst. Adv. Appl.* Berlin, Germany: Springer, 2005, pp. 43–55.
[19] T. Jung, X. Mao, X.-Y. Li, S.-J. Tang, W. Gong, and L. Zhang, "Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation," in *Proc. INFOCOM*, Apr. 2013, pp. 2634–2642.
[20] N. J. King and V. J. Raja, "Protecting the privacy and security of sensitive customer data in the cloud," *Comput. Law Secur. Rev.*, vol. 28, no. 3, pp. 308–319, Jun. 2012.
[21] J. Li *et al.*, "Searchable symmetric encryption with forward search privacy," *IEEE Trans. Dependable Secure Comput.*, to be published.
[22] J. Li, Z. Liu, X. Chen, F. Xhafa, X. Tan, and D. S. Wong, "L-encdb: A lightweight framework for privacy-preserving data queries in cloud computing," *Knowl.-Based Syst.*, vol. 79, pp. 18–26, May 2015.
[23] Z. Liu, Y. Huang, J. Li, X. Cheng, and C. Shen, "Divoram: Towards a practical oblivious ram with variable block size," *Inf. Sci.*, vol. 447, pp. 1–11, Jun. 2018.
[24] Z. , B. Li, Y. Huang, J. Li, Y. Xiang, and W. Pedrycz, "Newmcos: Towards a practical multi-cloud oblivious storage scheme," *IEEE Trans. Knowl. Data Eng.*, to be published.
[25] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 644–655.
[26] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, Apr. 1999, pp. 223–238.
[27] A. Papadimitriou *et al.*, "Big data analytics over encrypted datasets with seabed," in *Proc. OSDI*, 2016, pp. 587–602.
[28] S. K. Pasupuleti, S. Ramalingam, and R. Buyya, "An efficient and secure privacy-preserving approach for outsourced data of resource constrained mobile devices in cloud computing," *J. Netw. Comput. Appl.*, vol. 64, pp. 12–22, Apr. 2016.
[29] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: Protecting confidentiality with encrypted query processing," in *Proc. 23nd ACM Symp. Operating Syst. Principles*, Oct. 2011, pp. 85–100.
[30] L. Qiu, Y. Li, and X. Wu, "Protecting business intelligence and customer privacy while outsourcing data mining tasks," *Knowl. Inf. Syst.*, vol. 17, no. 1, pp. 99–120, Oct. 2008.
[31] E. Shi, J. Bethencourt, T.-H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 350–364.
[32] A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *J. Netw. Comput. Appl.*, vol. 79, pp. 88–115, Feb. 2017.
[33] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, May 2000, pp. 44–55.
[34] J. Stanek, A. Sorniotti, E. Androulaki, and L. Kencl, "A secure data deduplication scheme for cloud storage," in *Proc. Int. Conf. Financial Cryptography Data Secur.* Berlin, Germany: Springer, Nov. 2014, pp. 99–118.
[35] S. Tu, M. F. Kaashoek, S. Madden, and N. Zeldovich, "Processing analytical queries over encrypted data," *Proc. VLDB Endowment*, vol. 6, no. 5, pp. 289–300, Mar. 2013.
[36] C. Wang, K. Ren, S. Yu, and K. M. R. Urs, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *Proc. INFOCOM*, Mar. 2012, pp. 451–459.
[37] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud storage as the infrastructure of cloud computing," in *Proc. Int. Conf. Intell. Comput. Cogn. Inform. (ICICCI)*, Jun. 2010, pp. 380–383.
[38] J. Yang, S. He, Y. Lin, and Z. Lv, "Multimedia cloud transmission and storage system based on Internet of things," *Multimedia Tools Appl.*, vol. 76, no. 17, pp. 17735–17750, Sep. 2017.

JIANCHAO TANG received the B.Sc. degree from the Beijing Institute of Technology, in 2014, and the master's degree from the Department of Compute Science, National University of Defense Technology, in 2016, where he is currently pursuing the Ph.D. degree. He achieved best paper awards in The 11th Chinese Conference on Trusted Computing and Information Security. His current research interests include privacy protection in big data, cloud computing, and truth discovery.

SHAOJING FU received the B.Sc. and Ph.D. degrees in applied mathematics from the National University of Defense Technology, China, in 2005 and 2010, respectively, where he has been an Associate Professor with the College of Computer, since 2015. His research interests include applied cryptography, and security and privacy issues for cloud computing.

MING XU is currently a Professor with the College of Computer, National University of Defense Technology. He edited three volumes of international conference proceedings of Springer. His current research interests include mobile computing, wireless networks, cloud computing, and network security. He is a Senior Member of CCF and a member of ACM. He was invited to serve as a program committee of more than 30 international academic conference proceedings and serve as the Chairman or the Chairman of the procedural committee four times. He is also an international journal Editorial Board of IASTED International Journal of Computers and Applications and the Journal of Communication.