This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2675864, IEEE Transactions on Information Forensics and Security

Two-Cloud Secure Database for Numeric-Related SQL Range Queries with Privacy Preserving

Kaiping Xue, *Senior Member, IEEE,* Shaohua Li, Jianan Hong, Yingjie Xue, Nenghai Yu, and Peilin Hong

***Abstract*—Industries and individuals outsource database to realize convenient and low-cost applications and services. In order to provide sufficient functionality for SQL queries, many secure database schemes have been proposed. However, such schemes are vulnerable to privacy leakage to cloud server. The main reason is that database is hosted and processed in cloud server, which is beyond the control of data owners. For the numerical range query (">", "<", etc.), those schemes cannot provide sufficient privacy protection against practical challenges, e.g., privacy leakage of statistical properties, access pattern. Furthermore, increased number of queries will inevitably leak more information to the cloud server. In this paper, we propose a two-cloud architecture for secure database, with a series of intersection protocols that provide privacy preservation to various numeric-related range queries. Security analysis shows that privacy of numerical information is strongly protected against cloud providers in our proposed scheme.**

***Index Terms*—database, range query, privacy preserving, cloud computing**

I. Introduction

THE growing industry of cloud has provide a service paradigm of storage/computation outsourcing helps to reduce users' burden of IT infrastructure maintenance, and reduce the cost for both the enterprises and individual users [1], [2], [3]. However, due to the privacy concerns that the cloud service provider is assumed semi-trust (honest-but-curious.), it becomes a critical issue to put sensitive service into the cloud, so encryption or obfuscation are needed before outsoucing sensitive data - such as database system - to cloud [4], [5], [6].

The typical scenario for outsouced database is described in Fig. 1 as that in CryptDB[7]: A cloud client, such as an IT enterprise, wants to outsource its database to the cloud, which contains valuable and sensitive information (e.g. transaction records, account information, disease information), and then access to the database (e.g. SELECT, UPDATE, etc.) [8], [9], [10], [11], [12]. Due to the assumption that cloud provider is honest-but-curious [13], [14], the cloud might try his/her best to obtain private information for his/her own benefits. Even worse, the cloud could forward such sensitive information to the business competitors for profit, which is an unacceptable operating risk.

The privacy challenge of outsouced database is two-hold. 1) Sensitive data is stored in cloud, the corresponding private information may be exposed to cloud servers; 2) Besides data privacy, clients' frequent queries will inevitably and gradually reveal some private information on data statistic properties. Thus, data and queries of the outsouced database should be protected against the cloud service provider.

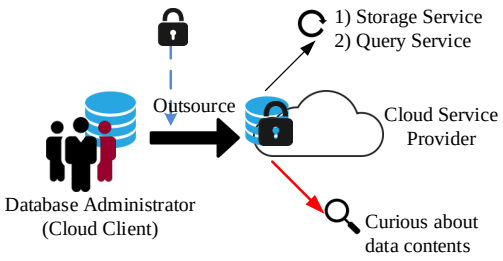


Fig. 1. Outsourced database, service and the privacy risk

One straightforward approach to mitigate the security risk of privacy leakage is to encrypt the private data and hide the query/access patterns. Unfortunately, as far as we know, few academia researches satisfy both properties so far. CryptDB [7] is the first attempt to provide a secure remote database application, which guarantees the basic confidentiality and privacy requirement, and provides diverse SQL queries over encrypted data as well. CryptDB uses a series of cryptographic tools to achieve these security functionality. Especially, *order-preserving encryption* [15] is utilized to realize numeric-related range query processes. From the perspective of query functionality, CryptDB supports most kinds of numerical SQL queries with such cryptology. However, such privacy leakage hasn't been well addressed thoroughly, since OPE is relatively weak to provide sufficient privacy assurance.

Some specific purpose cryptology like *order preserving encryption*(OPE) will expose some private information to the cloud service provider naturally: As it is designed to preserve the order on ciphertexts so that it can be used to conduct range queries, the order information of the data, the statistical properties derived therefrom, such as the data distribution, and the access pattern will be leaked. ***Can we design a new database system to provide range queries with stronger privacy guaranty?***

From the work in [16], the privacy can be preserved against the cloud, if the sensitive knowledge is partitioned into two parts, and distributed to two *non-colluding* clouds. In the literature [17], the authors also introduce a two-party system to design a secure knn query scheme, which enables the client to query k most similar records from the cloud securely. This divide-and-conquer mechanism can know any private

K. Xue, S. Li, J. Hong, Y. Xue, N. Yu and P. Hong are with Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China. (Email: kpxue@ustc.edu.cn (K. Xue)).

K. Xue and S. Li are also with State Key Laboratory of Information Security (Institute of Information Engineering), Chinese Academy of Sciences, Beijing 100093, China.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2675864, IEEE Transactions on Information Forensics and Security

information from one singe isolated part of the knowledge, and each of both clouds only knows its own part. In this paper, we introduce a secure two-cloud database service architecture, where the two clouds are *non-colluding* and both of them knows only part of knowledge. Based on this architecture, we further propose a series of interaction protocols for a client to conduct numeric-related query over encrypted data from remote cloud servers. The numeric-related query includes common query statements, such as *greater than*, *less than*, *between*, etc..

The main contribution of this paper can be summarized as follows: 1) We propose a two non-colluding cloud architecture to conduct a secure database service, in which the data is stored in one cloud, while the knowledge of query pattern is well partitioned into two parts, and knowing only one cannot reveal any private information; 2) We then present a series of intersection protocols to provide numeric-related SQL range query with privacy preservation, and especially, such protocols will not expose order-related information to any of the two non-colluding clouds.

The rest of this paper is organized as follows: Section II discusses the related work. In Section III, the two-cloud architecture is presented, following with security assumption and security requirements. In Section IV, some definitions are given, including the preliminaries of cryptographic techniques. We provide the detailed scheme in Section V, and discuss the privacy and performance properties of the proposed scheme in Section VI and Section VII, respectively. Section VIII concludes this paper.

II. Related Work

Fuzzy query over encrypted data is becoming a popular topic, since in practical scenarios, some query requests usually want to retrieve data with similar, rather than exactly same indexes[18], [19]. Fuzzy searchable encryption has been introduced for cloud computing in many literatures, such as [20], [21], [22], [23], [24], [25]. These schemes deal with the issue that search keywords allows small-scaled distinction in character/numeric level. Specifically for numerical keywords, the query predicate can get numerical records within a range. Some schemes targeted at spatial query, especially knn [17], [26], [27], [28], [29], which focus on the distance between the query vector and the data. They usually inquire about certain spatial objects (or several numerical attributes) related to the others within a certain distance. *Range query* [30], [31], [32] has been proposed for that purpose. However, such existing range query schemes are not suitable for practical secure database due to high storage overhead to maintain the corresponding ciphertext.

Subsequently, *order preserving encryption (OPE)* [15], [33], [34], [35], [36] has been introduced to provide numeric-related range query in structured database, such as CryptDB [7]. OPE preserves the order of values in encryption field, while hiding the actual values. Until now, OPE has been developed to increase both efficiency and security[15], [34], [36]. Popa et al. [15] proposed an ideal-security OPE scheme, in which, an adversary - even having the access privilege to a set of ciphertexts - still cannot learn the knowledge of data with non-negligible advantage. Although in Boldyreva et al. [34]'s definitions, such property has achieved the security boundary of OPE (IND-OCPA), that ideal-secure OPE still cannot satisfy the privacy requirement of secure database. ***OPE inherently exposes the order of data***, that can be utilized to reveal an amount of critical knowledge, although it is always expected to be private.

Bohli et al. [16] proposed a multi-cloud architecture, which can protect the private information of many outsourced services, including database. The main contribution is the introduction of four knowledge partition patterns among multiple cloud service providers: (1) Replication of applications, (2) Partition of application system into tiers, (3) Partition of application logic into fragments, and (4) Partition of application data into fragments. The knowledge is partitioned into two fragments, respectively stored in one cloud, who is assumed to be non-colluding to another cloud. Therefore, no cloud can get any private information in such multi-cloud architecture. However, Bohli et al. [16] have not provided a detailed scheme or realization for database.

In this paper, with the multi-cloud prototype in [16], [37], [38], we introduce a two non-colluding cloud database service architecture and propose a series of practical interaction protocols to conduct database range queries. In addition to securing the data contents, our scheme also well preserves the privacy of logical relationship among data contents, such as data order, the privacy of the statistical properties and query pattern.

III. System Architecture, Security Assumption and Security Requirements

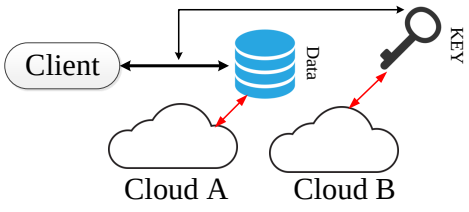
A. System Architecture

Our proposed secure database system includes a database administrator, and two non-colluding clouds. In this model, the database administrator can be implemented on a *client*'s side from the perspective of cloud service. The two clouds (refer to Cloud $\mathcal{A}$ and Cloud $\mathcal{B}$), as the server's side, provide the storage and the computation service. Fig. 2 briefly depicts the architecture of our outsourced secure database system in our scheme.

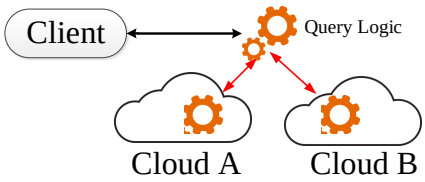
The two clouds work together to respond each query request from the client/authorized users (*availability*). For privacy concerns, these two clouds are assumed to be non-colluding with each other, and they will follow the intersection protocols to preserve privacy of data and queries (*privacy*).

In our scheme, the knowledge of stored database and queries is partitioned into two parts, respectively stored in one cloud. The mechanism guarantees that knowing either of these two parts cannot obtain any useful privacy information. As shown in Fig. 2(a), to conduct a secure database, data are encrypted and outsourced to be stored in one cloud (Cloud $\mathcal{A}$), and the private keys are stored in the other one (Cloud $\mathcal{B}$). For each query, the corresponding knowledge includes the data contents and the relative processing logic. We utilize a prototype of knowledge partition, *dividing application logic into two parts*, which is firstly proposed by Bohli et al. in [16]. The application logic, as a secret knowledge, is partitioned into

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2675864, IEEE Transactions on Information Forensics and Security



(a) Knowledge Partition of Stored Data



(b) Knowledge Partition of Query Logic

Fig. 2. Two-Cloud Database Architecture and Knowledge Partition Prototype

two parts, each of which is only known to one cloud. This prototype is shown in Fig. 2(b). Intuitively, this two-cloud architecture increases some complexity to some extent, and we will analyze and point out that this overhead is acceptable in Section VII-A.

B. Security Assumption

Following the general assumption of many related works in public cloud, we assume the clouds to be honest-but-curious:

On one hand, both of the two clouds will respond with correct information in the interactions of our proposed scheme (*honest*); on the other hand, the clouds try their best to obtain private information from the data that they process (*curious*). From the perspective of privacy assurance, here the data not only include permanently stored information (i.e., database), but also each temporary query request (i.e., queries).

Additionally and importantly, as the assumption in some existing works [16], [38], we assume that the two clouds $\mathcal{A}$ and $\mathcal{B}$ are non-colluding: Cloud $\mathcal{A}$ follows the protocol to add required obfuscation to protect privacy against cloud $\mathcal{B}$, so that cloud $\mathcal{B}$ cannot obtain additional private information in the interactions with Cloud $\mathcal{A}$. No private information is delivered beyond the scopes of protocols.

C. Potential Threats and Privacy Requirements

This section describes the potential threats and the privacy requirements when the database is outsourced to public cloud. The stored data contents and the query processes. Although there are many data encryption schemes, some fail to provide sufficient privacy preservation after statistical analysis: Repeated and large-amount query processes not only leak the access patterns, but also disclose the stored encrypted data progressively.

The privacy issues we consider in this paper mainly include *data contents*, *statistical properties*, and *query pattern* as follows:

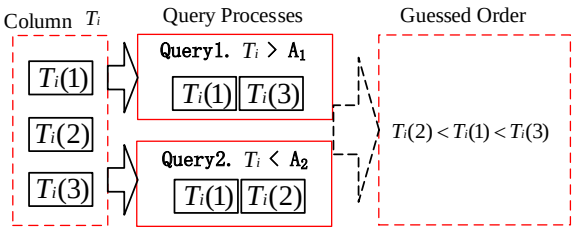


Fig. 3. Repeated Query Discloses Statistical Properties (For example, after two simple queries over one same column, the order relationship of some data in certain column can be determined.)

- ***Data contents***. The privacy of data contents includes (1) the definition and description of each column (column name) in the table of the stored database, and (2) the values of each record in the table. Some related works have mainly focused on this issue, in which the *column names* are blinded (such as CryptDB [7]) and meanwhile the values are encrypted with some other encryption techniques (such as *Order Preserving Encryption*[33]) and some *deterministic encryption* schemes[7], so that the adversaries cannot easily and directly guess the meaning of the column, or the values of the data. However, in an outsourced database, utilizing encryption alone, without other mechanisms, is far from being enough to preserve the privacy of the data contents. With the development of data analysis, by extracting features from data and queries, classification technique can help understand the definition of columns, and then breach of confidentiality of data contents. [39], [40].
- ***Statistical properties***. Besides the static properties can disclose the private information of data contents, such properties themselves are already sensitive and private for the client. *Order Preserving Encryption*(OPE), which is widely used in constructing the secure database, with support of range queries, directly exposes the statistical information in the encryption field. Furthermore, the leakage of statistic properties is part of the nature of outsourced cloud database service: the cloud can learn the statistical properties (like order) by repeated query requests. As an example, Fig. 3 describes such an attack: After two simple queries over one same column, the order relationship of some data in certain column can be determined. There are also some other direct and indirect scenarios to leak statistical properties. In this way, even though the order property is not exposed to the semi-trusted cloud at the beginning, the cloud can gradually find out the order information after many query requests.
- ***Query pattern***. The query pattern also contains privacy information, as they can reveal the client's purpose of the query. Even worse, such pattern can leak some statistical properties, as discussed above.

Based on the above discussion, we assert that an outsourced secure database providing numeric-related queries should prevent the following private information from being obtained by the honest-but-curious clouds:

- ***Data contents.*** The data contents includes item values and column names, which are the raw data that should be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2675864, IEEE Transactions on Information Forensics and Security

protected against any potential adversaries.

- ***Statistical properties.*** It includes the order of data and their probability distributions, some of which include "$>$", "$<$", "$=$", "*BETWEEN*", etc.
- ***Query pattern.*** Each query should be kept private against the honest-but-curious clouds and any unauthorized parties. The secrecy of such pattern should be well preserved even after many query processes.

IV. Preliminaries and Some Definitions

A. Paillier Cryptographic Algorithm

There are various cryptographic techniques to support numeric-related operations (e.g. addition, multiplication, X-OR) upon the encryption field. Paillier cryptosystem [41] is one of the most popular techniques that provides *addition homomorphic*, which means: if two integers a and b are encrypted with a same key k into two ciphertexts (be denoted as $E_k(a)$ and $E_k(b)$), there exists an operation (refer to as "$\otimes$"), such that

$$E_k(a) \otimes E_k(b) = E_k(a+b).$$

Paillier cryptographic algorithm is composed of the following phases: *key generation*, *encryption* and *decryption*.

- ***Key generation.*** Two large and independent prime numbers p and q are randomly selected. Then we compute $n = p \cdot q$ and $\mu = \lambda^{-1} \bmod n$, where λ is the least common multiple of p and q, and commonly $\lambda = lcm(p-1, q-1)$. The public key (*PK*) is n, and the private key (*SK*) is (λ, μ).
- ***Encryption.*** Let m be the integer to be encrypted. Firstly, we select a random number $r \in \mathbb{Z}_{n^2}^*$, and then the ciphertext of m can be computed as follows:

$$E(m; r) = (n+1)^m \cdot r^n \bmod n^2. \tag{1}$$

- ***Decryption.*** Let the ciphertext $c = E(m; r)$. The plaintext m can be recovered as follows:

$$m = \Big(\frac{(c^\lambda \bmod n^2) \ -1}{n} \Big) \cdot \mu \bmod n. \tag{2}$$

Paillier cryptosystem holds additive homomorphic in group $\mathbb{Z}_n^+$, which corresponds to the multiplication operation in the encryption field in $\mathbb{Z}_{n^2}$. The following equation illustrates the homomorphic property of Paillier cryptosystem.

$$\begin{aligned} E(m_1; r_1) \cdot E(m_2; r_2) =& (n+1)^{m_1} r_1^n \cdot (n+1)^{m_2} r_2^n \\ =& (n+1)^{m_1+m_2} (r_1 \cdot r_2)^n \\ =& E(m_1 + m_2; r_1 \cdot r_2) \end{aligned} \tag{3}$$

Another property can be be summarized as follows:

$$\begin{aligned} E^{m_2}(m_1; r_1) =& ((n+1)^{m_1} r_1^n)^{m_2} \\ =& (n+1)^{m_1 \cdot m_2} (r_1{}^{m_2})^n \\ =& E(m_1 \cdot m_2; r_1^{m_2}). \end{aligned} \tag{4}$$

As the random number r does not affect the result of decryption in Paillier encryption, Eq. (4) can be seen as the product of m_1 and m_2 in the encryption field.

In the rest of this paper, we use $E(m, PK)$ to denote the encryption result of the plaintext m with PK, and $D(X, SK)$ to denote the decryption result. We use capital letters like "X" to denote encrypted results (ciphertext), and lowercase letters like "x" to denote unencrypted results (plaintext). The random number $r \in \mathbb{Z}_n^*$ is omitted in the discussion of our scheme.

For number comparison, the sign of an plaintext number in Paillier cryptosystem is defined as follows: each participated plaintext integer x is assumed to be $x < n/2$. Then for clarity, the sign of x is defined to be positive if $0 < x < n/2$, and the sign is defined to be negative if $x > n/2$. As a result, the arithmetic subtraction of arbitrary two integers $(x_i - x_j)$ will not exceed the threshold $n/2$ if $x_i > x_j$, and the subtraction will exceed $n/2$ if $x_i < x_j$.

B. Numeric-Related SQL Queries

The Structured Query Language (SQL) is a specified-purpose programming language, which is used to manage data in a relational database system, which has became a standard of the ANSI and ISO in 1986[42] and 1987[43] respectively.

A query operation can request arbitrary data with a statement to describe the desired data. The requested data can be several columns of one or more tables in the database, and it can also be aggregated results from the original data (such as *sum*, *average*, *count* of the data.). To obtain the desired data, the query contains some statements to describe the requirement, e.g. some numeric-related ("$>$", "$<$", "$=$", "*BETWEEN*", etc.). For clarity, we refer to those query requests as *numeric-related SQL queries* in the rest of the paper.

Based on the introduced two-cloud architecture, we further propose a series of interaction protocols between the *client* and the two *clouds*, which can realize numeric-related SQL queries, and satisfy *privacy requirements*. It should be noted that, apart from the query operation, there are other SQL operations (e.g. update, insert) which modify the data. The privacy issue for such cases can be relolved with other existing approaches, such as ORAM (*Oblivious RAM*) [38], [44], [45], which is beyond the scope of our paper. In this paper, we focus on implementing query operation with privacy preserving.

V. Our Proposed Two-Cloud Scheme

In this section, we firstly give an overview of our proposed two-cloud scheme, and then present the detailed interaction protocols to realize range query with privacy preservation on outsourced encrypted database.

A. Overview

In our scheme, two clouds (refer to Cloud $\mathcal{A}$ and Cloud $\mathcal{B}$, respectively) have been assigned distinct tasks in the database system: Cloud $\mathcal{A}$ provides the main storage service and stores the encrypted database. Meanwhile, Cloud $\mathcal{B}$ executes the main computation task, to figure out whether each numerical record satisfies the client's query request with its own security key. With the assumption of *no collusion* between two clouds, the knowledge of application logic can be partitioned into two parts in our proposed scheme, where each one part is only known to one cloud. As we will analyze in this paper, one single part of knowledge cannot reveal privacy of the data and the query.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2675864, IEEE Transactions on Information Forensics and Security

Based on the two-cloud architecture, our scheme provides an approach to query numeric-related data with privacy preservation. The client can retrieve the desired data from the cloud, when the query predicates contain operators like "$>$", "$<$" and "BETWEEN" for one column, or even diverse condition combinations over one or more columns. For example, the client wants to retrieve items from the table, whose column $\mathcal{T}_i$ should be greater than a constant a (i.e., *SELECT* $*$ *FROM table WHERE* $\mathcal{T}_i > a$). In our scheme, it is resolved by figuring out the *sign* of each value of $(\mathcal{T}_i(j) - a)$, in which j traverses all rows of the whole table. If the result is greater than 0, the relevant item satisfies the query predicate. These procedures are executed in the encryption field, so that the privacy is strongly preserved. Meanwhile, each column name $\mathcal{T}_i$ must be encrypted.

Accordingly, if the operator is reversed, i.e., the predicate becomes "$\mathcal{T}_i < a$", the corresponding operation is $(a - \mathcal{T}_i(j))$. The remaining phases are similar as the above mentioned case. Meanwhile, if the predicate is "*BETWEEN a and b*"(*SELECT * FROM table WHERE* $\mathcal{T}_i$ *BETWEEN* a *AND* b), the result is the intersection of $\mathcal{T}_i > a$ and $\mathcal{T}_i < b$. For the predicate "=a", it is treated as a special case of the operator "*BETWEEN*", where the retrieved items are intersection set $\mathcal{T}_i > a - 1$ and $\mathcal{T}_i < a + 1$. Additionally, the operator of *COMBINATION* is another one that combines predicates with boolean expression with $\vee$ and $\wedge$.

In Section V-B, we first present the intersection procedure of the first case("$>$"). Then in Section V-C we give some necessary introductions about "$<$", "*BETWEEN*", "$=$" and *COMBINATION*.

The proposed mechanism can preserve the privacy of data and query requests against each of the two clouds. Specifically, Cloud $\mathcal{A}$ only knows the query request type and the final indexes, but due to dummy items appending, Cloud $\mathcal{A}$ cannot accurately understand the finally satisfied index set for each single request. Meanwhile, in order to prevent Cloud $\mathcal{A}$ from launching multiple specific-purpose query requests to deliberately to seek more knowledge about the data, we introduce a token based scheme, which can restrict the number of items and the range of columns that Cloud $\mathcal{A}$ can only process. For Cloud $\mathcal{B}$, it knows the satisfied indexes of each single request, but after the proposed operations, it does not know the relationship of the corresponding items. Moreover, Cloud $\mathcal{B}$ can hardly distinguish whether two received columns are generated from one or more columns in the original database.

B. The Basic Scheme for Operator "$>$"

As mentioned above, in our scheme, Cloud $\mathcal{A}$ permanently stores the client's encrypted database, and it also keeps the public keys related to the encrypted items in the database. Cloud $\mathcal{B}$ keeps the relevant private keys and undertakes the main task of computation.

Our proposed scheme is composed of *Table Creation* and *Query Protocol*. The intersection procedure of *Query Protocol* consists of four parts: *Query Request*, *Item Send*, *Index Send*, and *Query Response*, along with necessary computation operations as depicted in Fig. 4.

1) Table Creation: After the client rents the cloud service, he/she will outsource the database application to the cloud. To protect the private information, the following procedure is implemented before uploading to the cloud:

- For each column of the table (column in the table), the client randomly selects a symmetric key K, and then use it to encrypt each column name $\mathcal{T}_i$ ($1 \leq i \leq m$, where m is the total column number of the table). The encrypted name is denoted as $E(\mathcal{T}_i)$, assumed with equal length. The symmetric key K should be securely kept by the client.
- For each item (row in the table), its values in multiple columns should also be encrypted. In this paper we only take into consideration the numeric-related data. The client generates a public/private key pair for *Pallier cryptosystem*, denoted as PK and SK. For each numeric-related value x, the client uses PK to encrypt it as follows:

$$X = E(x,\ PK),$$

and the client should record the total item number of the table N.

Then, the encrypted table is uploaded to Cloud $\mathcal{A}$, as well as the public key PK. Meanwhile, the private key SK will be securely sent to Cloud $\mathcal{B}$. Without loss of generality, we take only one table for example in this paper. For multiple tables in a database, table names can be encrypted in the same way that column names are encrypted.

2) Query Request: When the client wants to retrieve some data from the outsourced database, he/she firstly generates a SQL query (e.g. "*SELECT* $*$ *FROM table WHERE* $\mathcal{T}_i > a$"). After the plaintext query request is generated, it will be modified to an encrypted query following these steps:

- ***Encrypt the column name.*** The client computes the column name $E(\mathcal{T}_i)$ with the symmetric key K.
- ***Encrypt the range boundary value.*** The client encrypts the range boundary value a with the public key ***PK*** in Pallier cryptosystem. The encrypted boundary value is denoted as "A", as shown in Fig. 4.
- ***Generate the token.*** The client analyzes the query request and figures out how many columns are involved. Then, the client generates the corresponding token $Sign(TNO||CN||N||T)$, where *TNO* is the token serial number, and *CN* is the number of involved columns, N is the total item number in the table, and T is the current timestamp. All these data are signed by the client's private key SK.
- ***Send the query request.***Then the client sends the encrypted query request to Cloud $\mathcal{A}$ as follows:

$$SELECT\ *\ FROM\ table\ WHERE\ E(T_i) > A,$$

together with the signed token. Here, for the above SQL query, the specific column number CN is "1".

3) Item Send: Cloud $\mathcal{A}$ finds the column named $E(\mathcal{T}_i)$. Before sending the items to Cloud $\mathcal{B}$, it implements the following three phases:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2675864, IEEE Transactions on Information Forensics and Security

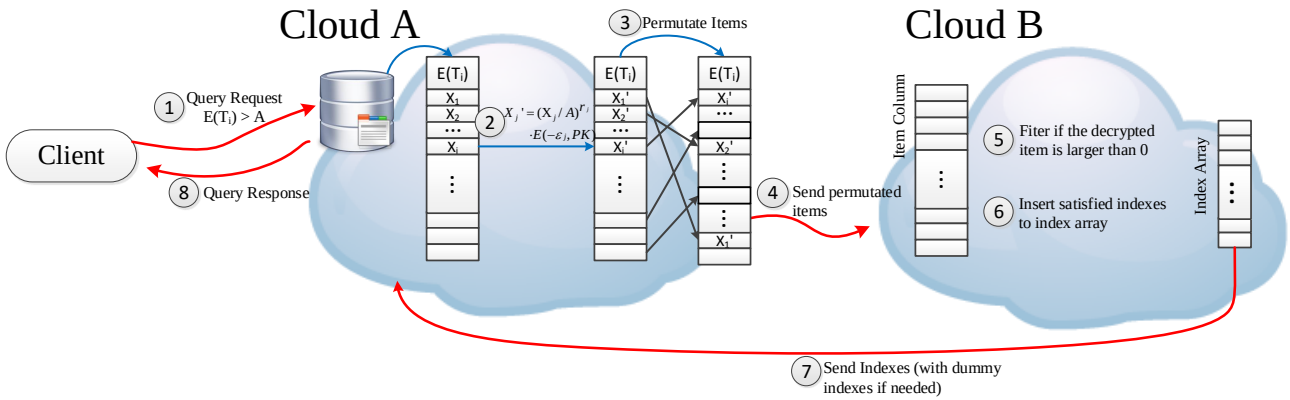


Fig. 4. The Query Protocol. The actions are performed in our designate orderd, which is marked up with circled number, like ①, ②,..., ⑨.

◇ *Number Comparison* (② in Fig. 4). For each item $X_j = \mathcal{T}_{ij}$ in the column, Cloud $\mathcal{A}$ selects a random positive integer r_j and ϵ_j individually, where $0 \leq \epsilon_j < r_j$, and then computes:

$$X'_j = \left(\frac{X_j}{A}\right)^{r_j} \cdot E(-\epsilon_j, PK). \tag{5}$$

With the additive homomorphic property of *Paillier cryptosystem*, the decryption result of Eq. (5) is equal to $(x_j - a) \cdot r_j - \epsilon_j$. As the integer r_j is positive and not too large, the values of $(x_j - a) \cdot r_j - \epsilon_j$ and $x_j - a$ have the same sign. All X'_j ($j \in$ indexes of items in the column.) are stored in another temporary column (named $\mathcal{L}$).

◇ *Items shuffling* (③ in Fig. 4). Cloud $\mathcal{A}$ further makes a random item shuffling in the column $\mathcal{L}$ to generate a new column $\mathcal{L}'$. To be noted, Cloud $\mathcal{A}$ should securely store the mapping of the items between the shuffled column $\mathcal{L}'$ and the original column $\mathcal{L}$ in a new column $\mathcal{M}$.

Finally, Cloud $\mathcal{A}$ removes the column name $E(\mathcal{T}_i)$ from the column $\mathcal{L}'$, and sends it to Cloud $\mathcal{B}$ together with the token received from the client.

4) Index Send: After receiving the column and the token from Cloud $\mathcal{A}$, Cloud $\mathcal{B}$ firstly verifies the legitimacy of the received token to make sure it hasn't expired and hasn't reused in a specific time interval. Then Cloud $\mathcal{B}$ checks the column from $\mathcal{A}$ to make sure that the column number and the item number are consistent with these corresponding values in the token. If the request is authorized, then Cloud $\mathcal{B}$ decrypts each item as follows:

$$x'_{j'} = D(X'_{j'},\ SK), \tag{6}$$

where j belongs to the item indexes in the column. For each decrypted item $x'_{j'}$, if $x'_{j'} > 0$, the index j' is inserted into a new index array $\mathcal{L}''$. Additionally, from the aspect of privacy preservation, then Cloud $\mathcal{B}$ appends a certain number of dummy indexes and inserts them to the random positions of the new index array $\mathcal{L}''$. Finally, Cloud $\mathcal{B}$ returns the final index array $\mathcal{L}''$ to Cloud $\mathcal{A}$.

5) Query Response: For each item j' in the received index column $\mathcal{L}''$, Cloud $\mathcal{A}$ looks up the index mapping information column $\mathcal{M}$, and gets its corresponding index j in the original column. According to the mapped index j, Cloud $\mathcal{A}$ sends the corresponding rows in the table, as the query response, to the client.

After receiving the response, the client can decrypt the items with SK to obtain the required data, and removes dummy items that does not satisfy the query predicate.

C. Variant Schemes for Operator "<", "BETWEEN", "=" and COMBINATION

In Section V-B, we have introduced the query procedure for operator ">". Here we extend that procedure to other operators "<", "*BETWEEN*", "=" and *COMBINATION*.

1) Operator "<": When the operator in the query is "<", the operation of *query request* and *item send* are slightly modified based on the scheme for the operator ">".

In the operation of *query request*, the form of the encrypted query is modified as follows:

$$SELECT\ *\ FROM\ table\ WHERE\ E(\mathcal{T}_i) < A.$$

In the operation of *item send*, the difference lies in the phase of *Number Comparison*. In order to implement the subtraction $(a - \mathcal{T}_i(j))$, the corresponding operation in the encryption field is:

$$X'_j = \left(\frac{A}{X_j}\right)^{r_j} \cdot E(-\epsilon_j, PK). \tag{7}$$

It should be noted that Cloud $\mathcal{B}$ cannot learn whether the operator in the query request is ">" or "<", since in the operation of *index send*, Cloud $\mathcal{B}$ only needs to filter the items greater than 0 in the operated column, where it is the same for ">" and "<".

2) Operator "BETWEEN" and "=": When the operator in the query is "*BETWEEN*" (*SELECT * FROM table WHERE* $\mathcal{T}_i$ *BETWEEN* a *AND* b), it is equivalent to an "*AND*" logic as follows:

$$(\mathcal{T}_i > a) \wedge (\mathcal{T}_i < b). \tag{8}$$

The operator "=" can be treated as a special case of "*BETWEEN*": the predicate "$\mathcal{T}_i = a$" can be translated to: "$\mathcal{T}_i$ *BETWEEN* $a - 1$ *AND* $a + 1$", so it is also equivalent to an AND logic:

$$(\mathcal{T}_i > a - 1) \wedge (\mathcal{T}_i < a + 1). \tag{9}$$

Therefore, the operator "*BETWEEN*" and "=" can be treated as the combination of operator ">" and operator "<" over one column. However, it should be noted that, from the view

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2675864, IEEE Transactions on Information Forensics and Security

of Cloud $\mathcal{B}$, it is still one combined query request over two independent columns originating from the same values but with different processing.

Cloud $\mathcal{A}$ generates two new columns based on the same original column in the operation of *item send*. There are some key points to be worthy mentioned: 1) The random positive integers to generate the two items of these two different columns with the same index j are chosen randomly and independently; 2) The specific column number (CN) in the token is set to "2"; 2) These two new generated columns should be shuffled with the same mapping, which will result in only one mapping information column both for these two columns.

These two restrictions are introduced to keep the privacy protection beyond each cloud's knowledge. The detailed scheme can be treated as a special case of that in operator *COMBINATION*, which will be discussed in Section V-C3.

3) Operator COMBINATION: More complex numeric-related query requests can be regarded as a combination of multiple simple requests, where the predicate is over multiple columns. For this scenario, the predicate can be concatenation of several simple conditions with logic gates, "$\vee$" and/or "$\wedge$", e.g., "$((\mathcal{T}_{i_1} > a_1) \wedge (\mathcal{T}_{i_2} > a_2)) \vee (\mathcal{T}_{i_3} < a_3)$".

The basic idea to realize this type of complex query request is intuitive: Firstly, we run separately and independently the procedures in Section V-B and Section V-C1 to generate a index set for each simple condition(e.g. $\mathcal{T}_{i_1} > a_1$ or $\mathcal{T}_{i_2} < a_2$). Then, if the logic gate is "$\vee$", we can compute the *union* of two index sets; Otherwise, for "$\wedge$", we can compute the *intersection* of them. For instance, if S_1, S_2 and S_3 are the index sets respectively for three simple conditions $\mathcal{T}_{i_1} > a_1$, $\mathcal{T}_{i_2} > a_2$, and $\mathcal{T}_{i_3} < a_3$. Then the final index S of the combination query request above is as follows:

$$S = (S_1 \bigcap S_2) \bigcup S_3 \tag{10}$$

To realize that in our two-cloud architecture with privacy preservation, there are some modifications from our schemes for operator "$>$" and operator "$<$". The modified procedures for operator *COMBINATION* is as follows:

- *Query Request.* The request with operator *COMBINATION* can be transformed to the combination of several simple condition requests. For the query predicate "$((\mathcal{T}_{i_1} > a_1) \wedge (\mathcal{T}_{i_2} > a_2)) \vee (\mathcal{T}_{i_3} < a_3)$", the corresponding predicate uploaded to Cloud $\mathcal{A}$ is as follows:

$$((E(\mathcal{T}_{i_1}) > A_1) \wedge (E(\mathcal{T}_{i_2}) > A_2)) \vee (E(\mathcal{T}_{i_3}) < A_3), \tag{11}$$

  where A_1, A_2 and A_3 are the ciphertexts of a_1, a_2, and a_3, respectively encrypted with the public key PK. Therefore, the complex query request can be split into 3 simple query requests (with only one condition) as follows:

$$\textit{SELECT } * \textit{ FROM table WHERE } E(\mathcal{T}_{i1}) > A_1,$$
$$\textit{SELECT } * \textit{ FROM table WHERE } E(\mathcal{T}_{i2}) > A_2,$$
$$\textit{SELECT } * \textit{ FROM table WHERE } E(\mathcal{T}_{i3}) < A_3.$$

- *Item Send.* Following the operations in Section V-B and Section V-C1, Cloud $\mathcal{A}$ independently deals with each simple request, and stores the corresponding new generated columns. Here the involved column number equals to the number of the simple requests. To be noted, the random item shuffling from $\mathcal{L}$ to $\mathcal{L}'$ is identical for different new generated columns in one operator *COMBINATION*. Finally, Cloud $\mathcal{A}$ sends the shuffled item columns without column names, as well as the logic relationship (composed of *Union* and/or *Intersection*) and the client's signed token to Cloud $\mathcal{B}$.
- *Index Send.* After the token verification, for each received column, Cloud $\mathcal{B}$ goes through each decrypted items to obtain the individual satisfied indexes as the operation in Section V-B4. Then Cloud $\mathcal{B}$ computes the final index array following the received logic relationships, like the instance shown in Eq. (10). In addition, from the aspect of privacy preservation, Cloud $\mathcal{B}$ appends a certain number of dummy indexes into the final index array. The final index array is sent back to Cloud $\mathcal{A}$.
- *Query Response.* It is the same as that in the basic scheme for operator "$>$" with no additional processing needed.

VI. Security Analysis

In this section, we will focus on the privacy preservation in the outsourced query processes against two honest-but-curious clouds. We first prove two theorems to illustrate the privacy-preserving of Cloud $\mathcal{A}$ and $\mathcal{B}$. And we further analyze the order privacy of item values in Cloud$\mathcal{A}$ and possible security issues due to repeated queries.

A. Security Proof

Theorem 1. *Cloud $\mathcal{A}$ cannot obtain any information from the user's query and the stored encrypted database as long as Paillier cryptosystem is semantically secure, and Cloud $\mathcal{A}$ and $\mathcal{B}$ are non-colluding.*

Proof. We can prove this theorem using the composition theorem [46] under the semi-honest model by analyzing the security of step 1)-3). Note that, in these steps, since all the data received by Cloud $\mathcal{A}$ is encrypted and the computation steps are all performed in the ciphertext domain, and because of the semantic security of Paillier cryptosystem [41], Cloud $\mathcal{A}$ cannot deduce any private information from these three steps unless Cloud $\mathcal{B}$ colludes with it. □

Theorem 2. *Cloud $\mathcal{B}$ cannot infer any private information from Cloud $\mathcal{A}$'s input as long as blinding factors are properly generated, and Cloud $\mathcal{A}$ and $\mathcal{B}$ are non-colluding.*

Proof. Similarly, this theorem can be proved using the composition theorem [46] under the semi-honest model by analyzing the security of step 4). After receiving the Cloud $\mathcal{A}$'s input, Cloud $\mathcal{B}$ will decrypt it with private key and obtain the plaintext of $\mathcal{L}'$. Since $\mathcal{L}'$ is generated from the original column $\mathcal{T}_i$, as long as knowing $\mathcal{L}'$ gives Cloud $\mathcal{B}$ negligible advantage in distinguishing $\mathcal{T}_i$ compared with random guesses [47], the privacy of $\mathcal{T}_i$ will be well-preserved. Blinding factors r_j and ϵ_j obfuscate the true value of $x_i - a$, so this requirement can be fulfilled by generating them properly. Literature [48] gives a security proof of such blinding factors in Appendix 1, we

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2675864, IEEE Transactions on Information Forensics and Security

now briefly prove it. Our goal is to let Cloud $\mathcal{B}$ not be able to derive x_i from $r_i \cdot (x_i - a) - \epsilon_i$. In the following, we assume a equals to 0, $x_i > 0$ and $y_i = r_i \cdot x_i - \epsilon_i$. This goal can be achieved if given that one knows y_i, the corresponding x_i that satisfied $y_i = r_i \cdot x_i - \epsilon_i$ has ρ different uniformly distributed values, and ρ is large enough. Let us define $S(y_i)$ be the set of possible x_i values, where $|S(y_i)| = \rho$, and prove the following theorem. If theorem 3 holds, we can express $|S(y_i)|$ as follows:

$$|S(y_i)| = \begin{cases} 2a - 1 & \text{if } a(a-1) + 1 \le y_i \le a^2 \\ 2a & \text{if } a^2 + 1 \le y_i \le a(a+1) \end{cases} \quad (12)$$

So, as long as a is large enough, which is equivalent to r_j is large enough, the possibility of guessing correct x_i is negligible.

Theorem 3. *If $y_i = a^2$ or $y_i = a^2 \pm a$, for some $a \ge 2, a \in \mathbb{Z}$, $|S(y_i)| + 1 = |S(y_i + 1)|$. Otherwise, $|S(y_i)| = |S(y_i + 1)|$.*

To prove the theorem, 1) we first consider such a case: $\epsilon_i \neq 0$. As $y_i = r_i \cdot x_i - \epsilon_i$, we get $y_i + 1 = r_i \cdot x_i - (\epsilon_i - 1)$, since $r_i > \epsilon_i - 1 \ge 0$, so $x_i \in S(y_i + 1)$. 2) When $\epsilon = 0$ and $x_i - r_i < -1$, $y_i = r_i \cdot x_i = x_i \cdot r_i$, it is obvious that $r_i \in S(y_i)$. Rearrange $y_i = r_i \cdot x_i$ as $y_i + 1 = (r_i + 1) \cdot x_i - (x_i - 1)$. Since $r_i + 1 > x_i - 1$, so $x_i \in S(y_i + 1)$. Since $x_i > x_i - 1$, so $r_i + 1 \in S(y_i + 1)$. We can further prove $r_i \notin S(y_i + 1)$ and $r_i + 1 \notin S(y_i + 1)$ based on contradiction, to illustrate that $x_i \in S(y_i)$ and $r_i \in S(y_i)$ uniquely maps to $x_i \in S(y_i + 1)$ and $r_i + 1 \in S(y_1 + 1)$. 3) Using similar proof technique, we can prove that when $\epsilon_i = 0$ and $x_i - r_i > 1$, $x_i \in S(y_i)$ and $r_i \in S(y_i)$ uniquely maps to $x_i + 1 \in S(y_i + 1)$ and $r_i \in S(y_1 + 1)$. When $\epsilon_i = 0$ and $|x_i - r_i| \le 1$, which means $y_i = x_i^2$ or $y_i = x_8 \cdot (x_i - 1)$, it can be proved that $x_i \in S(y_i)$ and $r_i \in S(y_i)$ uniquely maps to $x_i \in S(y_i + 1)$, $x_i + 1 \in S(y_i + 1)$ and $r_i \in S(y_1 + 1)$. Thus, the theorem holds. The detailed proof can be found in [48]. □

B. Order Privacy of Item Values

An ideal scheme is required to make nothing of the statistical properties be leaked to the curious clouds. However, the privacy leakage of statistical properties in a practical outsourced database system is inevitable, as returning subset of data rather than universe requires knowledge for filtering. For instance, if the client wants to retrieve $\mathcal{T}_i > a$ from the outsourced database, a cloud server without any knowledge of the order can only return all items of the database to the client, which is not usable.

Therefore, there should be a trade-off between the order privacy and availability of range queries. In the outsourced range query application, there is a range boundary value (e.g., the parameters a and b in the description of our scheme). We consider order privacy from two aspects: 1) the preservation of data order privacy in the same direction with the operator "$>$" or "$<$"; 2) the preservation of data order privacy in opposite directions to the boundary value. The following paragraphs discuss these two aspects, respectively.

1) Order privacy preservation of data in the same direction to the boundary value: Data in the same direction is all greater or all less than some given boundary value (e.g., for query "*SELECT $*$ FROM table WHERE $\mathcal{T}_i > 100$*", "100" is the boundary value, and data "102, 114" or "40, 65" are in the same direction to it). For Cloud $\mathcal{A}$, the order privacy of data items can be guaranteed, since the data is stored in Cloud $\mathcal{A}$ in the encrypted form, and Cloud $\mathcal{A}$ cannot get any knowledge of the order information, unless it has the private key SK following Paillier cryptosystem's security properties, while in our proposed system, only the client and Cloud $\mathcal{B}$ have the private key.

In our proposed two non-colluding cloud scheme, Cloud $\mathcal{B}$ has the private key SK for data filtering as shown in Fig. 2(a). To minimize the privacy leakage against Cloud $\mathcal{B}$, we make an obfuscation in the encryption field, as described in Section V-B3. As a result, the order information presented to Cloud $\mathcal{B}$ is strongly confused. In this paper we quantifies the obfuscation of the order privacy with the concept of *Order Correlation Coefficient (OCC)* as follows:

Definition 1. ***Order Correlation Coefficient (OCC).*** *Given an n-item integer sequence $\mathcal{F}$, for each item $f_i \in \mathcal{F}$, its original order is denoted as s_i, and s_i' is its order after implementing random multiplication for all items. Then OCC of the two orders is defined as the following formulation:*

$$OCC = \frac{2 \cdot \sum_{f_i \in \mathcal{F}} s_i \cdot s_i' - (max + min)}{max - min}, \quad (13)$$

where $max = \sum_{i=1}^{n} i^2$ and $min = \sum_{i=1}^{n} i \cdot (n - i + 1)$.

The parameter *OCC* of two orders of any sequence lies in the range from -1 to 1, based on Eq. (13). If the order privacy is not preserved (e.g., adopting order preserving encryption), we can get $s_i = s_i'$ for each item, then *OCC* will return "1". *OCC* equals to "-1" if the obfuscated order is exactly the inverse of the original one. A perfect privacy preservation mechanism will make *OCC* be close to "0", which means that the new order is entirely independent to the original one.

We measure *OCC* of our scheme in an experiment scenario. In this measurement, each data item (after homomorphic subtraction) is independently, identically and uniformly distributed in the range of $(1, 10^4)$. We measure *OCC* with different data scale and different range size of selected random integers (r_i in ② for obfuscation in Fig. 4). In our experiment, each case is measured 1000 times, and the average value in Fig. 5 shows the measured result of *OCC*.

From Fig. 5, it can be figured out that a larger data scale brings about a bigger *OCC*, closer to 1, which shows the order privacy preservation is easier for smaller data scale, and requires smaller random integer r_i. On the other hand, if the data scale increases to 10000, *OCC* decreases slowly with the larger range of random integer selection for multiplication. The curves for data scale larger than 10000 are not given because the experimental results are similar to that for the scale of 10000, shows that *OCC* remains stable for data scale that is large enough.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2675864, IEEE Transactions on Information Forensics and Security

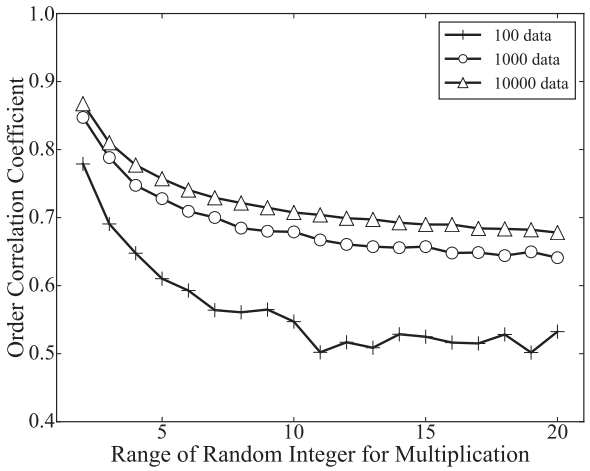


Fig. 5. Order Correlation Coefficient Analysis

As random integers for multiplication are from 1 to 10, *OCC* decreases rapidly with increasing range. Then, the decrease rate slows down as the range further increases. For the range larger than 20, the variation of *OCC* for different range values (even changes from 20 to 10^5) is quite slight.

From the measurement, we can assert that the order privacy leaked to Cloud $\mathcal{B}$ can be reduced to a specific level when *OCC* reaches to be less than the threshold, such as 0.7, by a practical and not-too-large random integer range. In addition, Cloud $\mathcal{B}$ cannot increase the accuracy of the order guess after numerous processing, as we will analyze in Section VI-C2.

2) Order privacy preservation of data in opposite directions to the boundary value: Data in opposite directions are not all greater or all less than some given boundary value (e.g., for query "*SELECT* $*$ *FROM table WHERE* $\mathcal{T}_i > 100$", "100" is the boundary value, data "40" and "155" are in opposite directions to it). In order to correctly respond to the query request, the order information that two different data set belong to opposite directions to the boundary value is inevitable known to Cloud $\mathcal{B}$, i.e., only if Cloud $\mathcal{B}$ can distinguish the items less than the boundary value a from those greater than it, as well as Cloud $\mathcal{B}$ must know which direction the data item is in, can it make an accurate response of query request such as "*SELECT* $*$ *FROM table WHERE* $\mathcal{T}_i > a$". Our scheme can mislead Cloud $\mathcal{A}$ and Cloud $\mathcal{B}$ into learning a wrong order as shown in Section V-B. The data privacy preservation in opposite directions to the boundary value against the two non-colluding clouds is as follows:

- For Cloud $\mathcal{A}$, more complex query operation combination generally exposes less order information of one column, as Cloud $\mathcal{B}$ only sends the final combined result, which has much less information than processing individually. Section VI-C1 will analyze this type of privacy preservation in details. Also, Cloud $\mathcal{B}$ appends a certain number of dummy indexes and respectively inserts them to randomly chosen positions of the index array. This further obfuscates Cloud $\mathcal{A}$ and preserves the order privacy of data in opposite directions to the boundary value;
- It seems that Cloud $\mathcal{B}$ has certain privacy information for executing the query predicate. However, in our proposed scheme, each column name is encrypted and unknown to Cloud $\mathcal{B}$. Different columns generated and shuffled by Cloud $\mathcal{A}$ makes Cloud $\mathcal{B}$ difficult to distinguish whether they are from a same original column. Therefore previous query requests will not help Cloud $\mathcal{B}$ to learn the privacy information. Moreover, our scheme makes the directional query operators (">" and "<") be indistinguishable to Cloud $\mathcal{B}$ (see in Section V-C1), which makes Cloud $\mathcal{B}$ obfuscated on whether order relationship is inverse or not.

C. Privacy Preservation in repeated queries

The clouds could collect more and more statistical information after receiving repeated query requests and generating the corresponding responses towards the database (e.g. Fig. 3). However, we will demonstrate that our scheme can reduce the privacy leakage greatly in this scenario.

1) For Cloud $\mathcal{A}$: Repeated query requests will make Cloud $\mathcal{A}$ learn more and more about the privacy information, while in our scheme, this ability is restricted as follows.

On one hand, many query requests are crossing over multiple columns, and simple query requests are just a part of usual database query requests. In such a situation, Cloud $\mathcal{A}$ only receives the final index result (with dummies, optionally) from Cloud $\mathcal{B}$ filtered with multiple conditions, it cannot get the original comparison result of each one column.

On the other hand, Cloud $\mathcal{B}$ responds Cloud $\mathcal{A}$ based on the token obtained from the client, and there have two ways to guarantee the security. 1) Each token contains the specific column number (CN) and the total item number in the table(N), which Cloud $\mathcal{A}$ must operate on exactly. Cloud $\mathcal{A}$ must send the result to Cloud $\mathcal{B}$ exactly with these two numbers without modification: If Cloud $\mathcal{A}$ increases or decreases CN or N, Cloud $\mathcal{B}$ will find that unmatched with the token, and if Cloud $\mathcal{A}$ replaces any item in these CN columns, it will take the risk of responding wrong result to client, which can be assumed not happening based on the assumption that semi-trusted clouds are honest. 2) Each token has been signed with SK by client, Cloud $\mathcal{A}$ cannot modify any tokens or generate a new one, and every token contains a different serial number and timestamp, so Cloud $\mathcal{A}$ cannot conduct the replay attack.

2) For Cloud $\mathcal{B}$: The name of each involved column is removed before sending to Cloud $\mathcal{B}$, and meanwhile, different random integers are selected for each item in each query request by Cloud $\mathcal{A}$. As a result, Cloud $\mathcal{B}$ cannot distinguish whether two previous query requests are on the same column, hence repeated queries cannot be utilized to increasing the accuracy of order guessing. Moreover, based on item shuffled, Cloud $\mathcal{B}$ cannot distinguish one same item from two previous queries, even though the plaintext SQL queries are identical.

VII. Performance analysis

In this section, we will first analyze the complexity of our proposed scheme, and then evaluate the computation and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2675864, IEEE Transactions on Information Forensics and Security

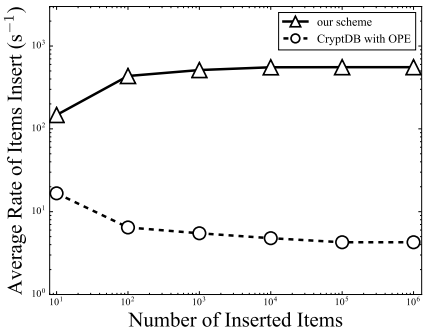


Fig. 6. Efficiency Comparison for Item Insert

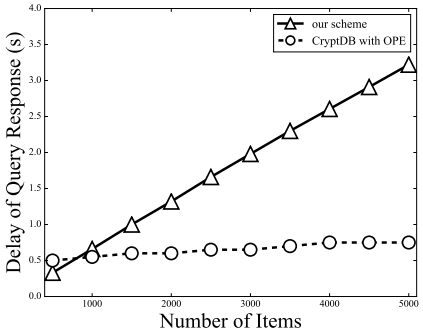


Fig. 7. Efficiency for Item Select in Single Process

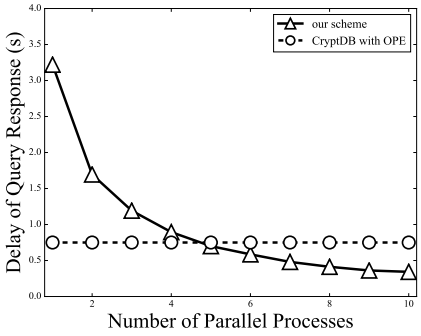


Fig. 8. Efficiency for Item Select in Parallel Processes

communication overhead within our constructed experiment platform.

The experiment platform is implemented by C based on GMP library, with 1024-bit length public key n for Paillier homomorphic cryptosystem. All data are stored in the MySQL database. For comparison, we utilize AVL tree[49] to simulate Popa's *order preserving encryption* (OPE)[15], and embed it into CryptDB[7]. We will refer this comparison scheme as CryptDB with OPE in the rest of this paper.

We provide evaluations measured in a computer with Intel i3-4130 CPU @ 3.40GHz and 16G memory. According to the simulation assumption in [38], unless otherwise specialized, we simulate a 50ms (round-trip) latency between the client and each cloud by asynchronously delaying the client's requests and responses. Although our two clouds are practically two different ones, they are commonly connected with high-speed network, we do not simulate additional latency between them, which is assumed to be 0.

A. Complexity of Our Proposed Scheme

In our proposed scheme, both stored data and query logic are partitioned into two parts. This improves the privacy preservation of range query, while the complexity increases, too. In fact, the complexity of client is no significant increase compared with common OPE schemes, such as [33], [34], [35]. For a query, the client in these schemes needs to send a query request, and then receive and decrypt the response to get the results. The client in our scenario also only needs a round-trip communication to perform a query. As for the clouds, the communication overhead between two clouds does not exist in single cloud schemes. However, as mentioned in Section 5.1 in [38], the two clouds are in fact two different clouds (e.g. Amazon and Azure), the communication latency between the clouds is relative low. What is more, during a query, only one interaction is required for both clouds in our scheme. In total, our system does increase complexity to some extent, but it is acceptable, as the increase in overhead is small and the security has been greatly improved.

B. Efficiency of Item Insert

We first evaluate the efficiency of item insert with only one column, as it can be easy to expand one column to multiple ones. The cost for multiple columns is linear to the number of columns for both the proposed scheme and the compared one (CryptDB with OPE). Fig. 6 shows the average rate of *item insert* with the increase of inserted items number. From Fig. 6, due to the cost of initializing database table, the first point of our scheme is not as better as the other points in the curve. But overall, our scheme's insertion rate remains stable by the number of inserted items. On the contrary, the average insertion rate of CryptDB with OPE decreases as the number of inserted items goes up.

The cost of inserting items to the database are different between these two schemes: In our scheme, Paillier's homomorphic encryption makes up a large proportion of the cost. While in CryptDB with OPE, the encryption cost with strong symmetric cryptographic algorithms, such as AES-128, is negligible. However, in CryptDB with OPE, inserting one item requires a number of round-trip communications between the client and the cloud, where the number of communications is equal to the depth of the tree in average - approximately the logarithm of the total number of inserted items.

Simulation result shown in Fig. 6 depicts the average insertion time of two schemes. Although Paillier's homomorphic encryption of our scheme is relatively inefficient than the symmetric cryptographic algorithm used in CryptDB with OPE, our scheme requires only one round trip communication. Therefore, the insertion rate is stable and the efficiency will not decrease when the item number becomes large in our scheme. By contrast, the depth of tree in CryptDB with OPE increases obviously with a larger number of data records. As a result, the efficiency decreases when the data scale increases. As shown in Fig. 6, when the inserted items increase to 10^4, the communication cost of item insertion is unbearable, which brings in about 13 to 14 (as $2^{13} < 10^4 < 2^{14}$) round trip communications to insert one item.

C. Efficiency of Range Query

This section evaluates the efficiency in executing the range query condition. Fig. 7 and Fig. 8 show the delay of a query and the corresponding response. Especially, Fig. 7 shows the result when the query is executed in one single process, and Fig. 8 shows the result when implementing the procedure in parallel computing with multi-process.

When only one single process conducts the query response on the cloud side, CryptDB with OPE shows a great advantage over our proposed scheme, as shown in Fig. 7, the delay of CryptDB with OPE increases slowly, while our scheme's delay is almost linear to the number of items. The reason is as follows:

In CryptDB with OPE, the cloud should find several middle nodes in the tree according to the boundary value of the range. This procedure will go through the tree from the root to the leaf node until reaching a node associated with the boundary value. After that, as a result, the cloud can pick up all the required items in that subtree without additional cost. As the depth of the tree increases with a logarithmic growth of the item number, the increase of the query and response delay of CryptDB with OPE is also in logarithmic growth. By contrast, in our scheme, an subtraction, multiplication and addition are required for each item in Cloud $\mathcal{A}$, and a decryption is required in Cloud $\mathcal{B}$, therefore the delay is linear to the number of the items. From the evaluation result to compare the two schemes shown in Fig. 7, the efficiency of CryptDB with OPE exceeds ours when the number of involved items increased to over 1000 for single process.

However, the above comparison results are based on one process, which is too conservative in Cloud $\mathcal{B}$: cloud computing is a service platform built upon numerous servers with multi-kernel CPUs, which has parallel property to jointly complete a task. Our main evaluation will analyze the performance of two schemes in the parallel scenario as follows:

The delay in our proposed scheme is mainly caused by the computation cost (e.g., encryption and decryption for each item), but without communication cost. The processing of different items is independent and can be implemented in parallel. Compared with CryptDB with OPE, which needs to have multiple round-trip communications for each item and has non-ignorable communication delay, our scheme does not need to wait for any other part's response.

As shown in Fig. 8, for 5000 items, the efficiency increases linearly to the number of parallel processes, and the query and response delay decreases rapidly. By contrary, the delay in CryptDB with OPE is mainly caused by multiple round-trip communications between the client and the cloud, spent in searching for the internal node associating with the range boundary value. Each round trip cannot begin until the former one is completed. As a result, their scheme cannot enjoy the advantage of parallel computing in cloud.

For practical datasets in real-world, our scheme can achieve higher efficiency with the advantage of parallel property of the cloud computing compared with CryptDB with OPE.

D. Storage Overhead

For the storage overhead, in our proposed scheme, the client and Cloud $\mathcal{B}$ only keeps private key, and Cloud $\mathcal{A}$ stores all the encrypted data as well as public key. And in CryptDB with OPE, the client keeps a symmetric key, and the cloud stores both the whole encrypted data and the corresponding tree. The mainly storage overhead of both schemes is on the encrypted data, which is consistent with the actual situation.

VIII. Conclusion

In this paper, we presented a two-cloud architecture with a series of interaction protocols for outsourced database service, which ensures the privacy preservation of data contents, statistical properties and query pattern. At the same time, with the support of range queries, it not only protects the confidentiality of static data, but also addresses potential privacy leakage in statistical properties or after large number of query processes. Security analysis shows that our scheme can meet the privacy-preservation requirements. Furthermore, performance evaluation result shows that our proposed scheme is efficient.

In our future work, we will consider to further enhance the security while ensuring practicality, and we will extend our proposed scheme to support more operations, such as "*SUM/AVG*".

Acknowledgment

The authors sincerely thank the anonymous referees for their invaluable suggestions that have led to the present improved version of the original manuscript. This work is supported by the National Natural Science Foundation of China under Grant No. 61379129 and No. 61671420, Youth Innovation Promotion Association CAS, and the Fundamental Research Funds for the Central Universities.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *IEEE Transactions on Services Computing*, vol. 5, no. 2, pp. 220–232, 2012.

[3] K. Xue and P. Hong, "A dynamic secure group sharing framework in public cloud computing," *IEEE Transactions on Cloud Computing*, vol. 2, no. 4, pp. 459–470, 2014.

[4] J. W. Rittinghouse and J. F. Ransome, *Cloud computing: implementation, management, and security*. CRC press, 2016.

[5] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, 2012.

[6] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[7] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*. ACM, 2011, pp. 85–100.

[8] C. Curino, E. P. Jones, R. A. Popa, N. Malviya *et al.*, "Relational cloud: A database-as-a-service for the cloud," 2011, http://hdl.handle.net/1721.1/62241.

[9] D. Boneh, D. Gupta, I. Mironov, and A. Sahai, "Hosting services on an untrusted cloud," in *Advances in Cryptology-EUROCRYPT 2015*. Springer, 2015, pp. 404–436.

[10] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," *IEEE Transactions on Computers*, vol. 65, no. 10, pp. 3184–3195, 2016.

[11] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, "New publicly verifiable databases with efficient updates," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 546–556, 2015.

[12] S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Annual Cryptology Conference*. Springer, 2011, pp. 111–131.

[13] W. Li, K. Xue, Y. Xue, and J. Hong, "TMACS: A robust and verifiable threshold multi-authority access control system in public cloud storage," *IEEE Transactions on Parallel & Distributed Systems*, vol. 27, no. 5, pp. 1484–1496, 2016.

[14] K. Xue, Y. Xue, J. Hong, W. Li, H. Yue, D. S. Wei, and P. Hong, "RAAC: Robust and auditable access control with multiple attribute authorities for public cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 953–967, 2017.

[15] R. A. Popa, F. H. Li, and N. Zeldovich, "An ideal-security protocol for order-preserving encoding," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP'13)*. IEEE, 2013, pp. 463–477.

[16] J.-M. Bohli, N. Gruschka, M. Jensen, L. L. Iacono, and N. Marnau, "Security and privacy-enhancing multicloud architectures," *IEEE Transactions on Dependable and Secure Computing*, vol. 10, no. 4, pp. 212–224, 2013.

[17] Y. Elmehdwi, B. K. Samanthula, and W. Jiang, "Secure k-nearest neighbor query over encrypted data in outsourced environments," in *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 2014, pp. 664–675.

[18] F. Hao, J. Daugman, and P. Zielinski, "A fast search algorithm for a large fuzzy database," *IEEE Transactions on Information Forensics and Security*, vol. 3, no. 2, pp. 203–212, 2008.

[19] A. Castelltort and A. Laurent, "Fuzzy queries over NoSQL graph databases: perspectives for extending the cypher language," in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, 2014, pp. 384–395.

[20] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM2010)*. IEEE, 2010, pp. 1–5.

[21] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS2010)*. IEEE, 2010, pp. 253–262.

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

[23] B. Wang, S. Yu, W. Lou, and Y. T. Hou, "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud," in *Proceedings of the 33rd Annual IEEE International Conference on Computer Communications (INFOCOM2014)*. IEEE, 2014, pp. 2112–2120.

[24] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2266–2277, 2013.

[25] Z. Fu, X. Wu, C. Guan, X. Sun, and K. Ren, "Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2706–2716, 2016.

[26] Z. Yu, Y. Liu, X. Yu, and K. Q. Pu, "Scalable distributed processing of k nearest neighbor queries over moving objects," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1383–1396, 2015.

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

[28] X. Yi, R. Paulet, E. Bertino, and V. Varadharajan, "Practical approximate k nearest neighbor queries with location and query privacy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 6, pp. 1546–1559, 2016.

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

[30] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Theory of Cryptography*. Springer, 2007, pp. 535–554.

[31] E. Shi, J. Bethencourt, T. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *IEEE Symposium on Security and Privacy (SP'07)*. IEEE, 2007, pp. 350–364.

[32] Y. Yang, H. Li, M. Wen, H. Luo, and R. Lu, "Achieving ranked range query in smart grid auction market," in *2014 IEEE International Conference on Communications (ICC2014)*. IEEE, 2014, pp. 951–956.

[33] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. ACM, 2004, pp. 563–574.

[34] A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill, "Order-preserving symmetric encryption," in *Advances in Cryptology–EUROCRYPT 2009*. Springer, 2009, pp. 224–241.

[35] H. Kadhem, T. Amagasa, and H. Kitagawa, "MV-OPES: Multivalued-order preserving encryption scheme: A novel scheme for encrypting integer value to many different values," *IEICE Transactions on Information and Systems*, vol. 93, no. 9, pp. 2520–2533, 2010.

[36] Z. Liu, X. Chen, J. Yang, C. Jia, and I. You, "New order preserving encryption model for outsourced databases in cloud environments," *Journal of Network and Computer Applications*, vol. 59, pp. 198–207, 2016.

[37] M. A. AlZain, E. Pardede, B. Soh, and J. A. Thom, "Cloud computing security: from single to multi-clouds," in *Proceedings of the 45th Hawaii International Conference on System Science (HICSS2012)*. IEEE, 2012, pp. 5490–5499.

[38] E. Stefanov and E. Shi, "Multi-cloud oblivious storage," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. ACM, 2013, pp. 247–258.

[39] X. Wu, X. Zhu, G.-Q. Wu, and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 97–107, 2014.

[40] S. R. Ganta, S. P. Kasiviswanathan, and A. Smith, "Composition attacks and auxiliary information in data privacy," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2008, pp. 265–273.

[41] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology-EUROCRYPT'99*. Springer, 1999, pp. 223–238.

[42] ANSI, X3-135, "American national standard for information systems: Database language SQL," American National Standards Institute, NY, 1986.

[43] ISO9075, "Information processing systems. database language SQL," 1987.

[44] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *Journal of the ACM*, vol. 43, no. 3, pp. 431–473, 1996.

[45] E. Stefanov and E. Shi, "Oblivistore: High performance oblivious cloud storage," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP'13)*. IEEE, 2013, pp. 253–267.

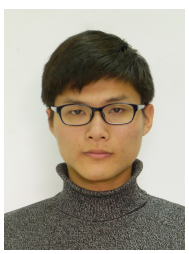
[46] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[47] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2014.

[48] Y. Dou, K. C. Zeng, H. Li, Y. Yang, B. Gao, K. Ren, and S. Li, "P^2-SAS: Privacy-preserving centralized dynamic spectrum access system," *IEEE Journal on Selected Areas in Communications*, 2016.

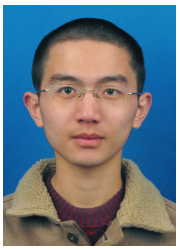
[49] M. AdelsonVelskii and E. M. Landis, "An algorithm for the organization of information," DTIC Document, Tech. Rep., 1963.

Kaiping Xue (M'09-SM'15) received his B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received his Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. Currently, he is an Associate Professor in the Department of Information Security and Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks and network security.

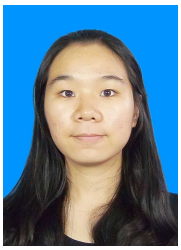


Shaohua Li received the B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2016. He is currently a graduated student in Communication and Information System from the Department of Electronic Engineering and Information Science (EEIS), USTC. His research interests include network security and system security.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2017.2675864, IEEE Transactions on Information Forensics and Security



Jianan Hong received the B.S. degree from the department of Information Security, University of Science and Technology of China (USTC), in 2012. He is currently working toward the Ph.D. degree in Information Security from the Department of Electronic Engineering and Information Science (EEIS), USTC. His research interests include secure cloud computing and mobile network security.



Yingjie Xue received her B.S. degree from the department of Information Security, University of Science and Technology of China (USTC) in July, 2015. She is currently a graduated student in Communication and Information System from the Department of Electronic Engineering and Information Science (EEIS), USTC. Her research interests include Network security and Cryptography.

Nenghai Yu received the B.S. degree from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 1987, the M.E. degree from Tsinghua University, Beijing, China, in 1992, and the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2004. Since 1992, he has been a Faculty in the Department of Electronic Engineering and Information Science, USTC, where he is currently a Professor. He is the Executive Director of the Department of Electronic Engineering and Information Science, USTC, and the Director of the Information Processing Center, USTC. He has authored or co-authored more than 130 papers in journals and international conferences. His research interests include multimedia security, multimedia information retrieval, video processing, and information hiding.

Peilin Hong received her B.S. and M.S. degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1983 and 1986. Currently, she is a Professor and Advisor for Ph.D. candidates in the Department of EEIS, USTC. Her research interests include next-generation Internet, policy control, IP QoS, and information security. She has published 2 books and over 150 academic papers in several journals and conference proceedings.